Visual Task Outcome Verification Using Deep Learning

Özgür Erkent[1], Dadhichi Shukla[1], Justus Piater[1]

***Abstract*— Manipulation tasks requiring high precision are difficult for reasons such as imprecise calibration and perceptual inaccuracies. We present a method for visual task outcome verification that provides an assessment of the task status as well as information for the robot to improve this status. The final status of the task is assessed as success, failure or in progress. We propose a deep learning strategy to learn the task with a small number of training episodes and without requiring the robot. A probabilistic, appearance-based pose estimation method is used to learn the demonstrated task. For real-data efficiency, synthetic training images are created around the trajectory of the demonstrated task. We show that our method can estimate the task status with high accuracy in several instances of different tasks, and demonstrate the accuracy of a high-precision task on a real robot.**

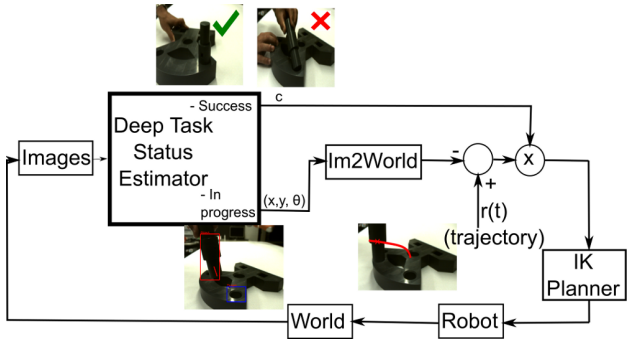


Fig. 1: A general overview of the proposed method. The images show a peg-in-hole task.

I. Introduction

Daily tasks like insertion, placement or cutting a strip are simple to perform by humans whereas they are challenging for robots. Paolini et al. [1] found a success probability in the range of $[0.20, 0.55]$ for an insertion task and a range of $[0.60, 1.00]$ for the task of dropping an object into a container in their experiments. The difficulty can usually be attributed to the precision required of the task rather than the complexity. Humans can use visual feedback to guide their movements and to assess the outcome to a large extent. In this paper, we follow a similar strategy; we gather data from visual sensors and use it to guide the motions and assess the outcome within the same process.

We perform three tasks to evaluate the proposed framework. An overview of the system is as shown in Fig. 1. The components of a simple task include the object manipulated by the robot, the target location of this object and the trajectory to be followed by the robot. The status of the task is one of *successful*, *failure* or *in progress*. To be able to define the status of the task, both the target and the object should at least be partially observable. The task outcome is generally described as the assessment of the final state of the task as *success* or *failure* [2], [3], [4], [5]. When the task status is *in progress*, the coarse states of the manipulated object and the target are provided so that the controller can plan the next motion of the robot. The controller is assumed to be able to plan the motion of the robot given the poses of the objects, and the relation between the image plane and the object poses is assumed to be known. There are two important aspects to this problem: First, the trained system should generalize from the training task to similar tasks involving distinct objects. Secondly, the approach that is used to solve this problem should be used to learn more than one task.

We propose a method that includes regression using deep learning to estimate the pose of the manipulated object and a classifier based on deep learning to classify the task status. A network that combines a Faster Region-based Convolutional Neural Network (R-CNN) [6] with a neural network architecture developed for image classification [7] is designed. We modify the image classification network to be used as a regression network. The trajectory is learned from a few samples representative of the task. A probabilistic appearance-based pose estimation method [8] is used to estimate the trajectory during training. After the trajectory is obtained, synthetic images are created around this trajectory to be used for regression with deep learning. Since the tremendous amount of data required by deep learning is created synthetically, the required amount of time to capture training data is also reduced significantly. To estimate the task status, successful tasks are demonstrated and an estimator is trained.

The contributions can be summarized as follows:

- The task status and type can be estimated by regression with deep learning during execution.
- The task can be executed with novel objects.
- Using the trajectory of the manipulated object reduces the necessity of using a tremendous amount of data.

In the next section we review previous work related to verification, task status feedback and deep learning for regression and robotics. Then we introduce the details of our approach, and in the subsequent section we evaluate our method under different conditions. We conclude with a brief summary.

[1] Özgür Erkent, Dadhichi Shukla and Justus Piater are with the Intelligent and Interactive Systems Lab, Institute of Computer Science, University of Innsbruck, Austria. ozgur.erkent@uibk.ac.at

A. Related Work

Studies on task status verification in robotics mainly focus on outcome prediction. Outcome prediction can be used for several practical purposes. For example, Pile et al. [3] use force sensing to detect the folding in a cochlear implant surgery and stops the process in the case of a failure. In another study, Pastor et. al. [2] predict the outcome to learn a complicated skill using a combination of different sensors, including a laser sensor and the proprioceptive sensors. We are focused on works related to visual perception in this study. Nguyen et al. [9] learn locations where the manipulation will succeed; however, the detections are simple and hand-crafted, e.g. detection of lights on or off. Paolini et al. [1] propose a framework where they compute the probabilities of success after the execution of a task and select the action that has the highest probability. They obtain the probabilities from a data-driven statistical framework and use the sensors on the gripper. Task outcome has also been studied in industry with the name visual inspection. However, usually these studies are very specific to the task, for example Ong et al. [5] detect failed solders using a special camera and a lightning. Interested readers are referred to the work of Malamas et al. [4] for a survey on visual inspection in industry.

To learn the status of a task, regression seems to be a natural candidate since it can give an accurate estimate of the task status. Visual servoing methods estimate the position of the features in the images, which are usually determined by hand; as a result, they are not usually generalizable over a range of tasks and they cannot learn from real world visual data [10]. Studies have been made in deep learning where the regression have been used to find the state of a new sample. For example Kendall et al. [11] use regression in place recognition to find the pose of the camera in the streets in an area of $50000m^2$ which shows that regression can be used to localize accurately in a dynamically changing environment. Miao et al. [12] use regression to estimate the location of an object precisely using patches around the object in an X-ray image. Convolutional neural networks (CNNs) are also used to estimate the depth estimates from monocular images with regression, although the problem is hard, they achieve sufficient results [13], [14] which show that regression technique can be used in various different domains. Another usage area of deep learning with regression is human pose estimation, where the poses of multiple human body parts are estimated [15]. Hand pose estimation can also be made with CNN regression [16]. This indicates that the regression with CNNs can generalize over parts with different appearances. One of the candidates to estimate the pose of the objects in 2D images is the spatial transform networks (STN) [17], which can find the affine transform of an object in the image. Multiple views of the object are necessary in different affine transforms to train the network, which would be costly for a robot to obtain for each task and object type. We combine faster R-CNN [6] to find the bounding box of the object and the target in the image with a modified version of a CNN architecture [7] which is originally developed for classifying 1.2 million images in the ImageNet dataset [18] to find the precise location of the object in the image. Availability of pre-trained network weights allow to train the networks with a smaller number of samples.

It should be noted that there are also studies that try to learn the robot motion from the image pixels and the current robot state [19]. The main difference from our study is that we try to learn the task status robustly from visual perception only. First, we learn the trajectory while the human is performing the action, and then find the appropriate robot actions to execute the trajectory after observing the state of the task. Various studies show that the trajectories for the task can be learned from human demonstrations. Zadeh et al. [20] propose a method to learn potential functions from human demonstrations. In this approach, the demonstration should be performed on the robot to learn the parameters necessary for the task. We use probabilistic motor primitives (ProMP) [21] to learn the trajectories from human demonstration without the robot. Then the robot mimics this trajectory using its inverse kinematics to convert ProMPs into robot motion.

II. Method

In this section, we describe our approach to estimating the task status. As shown in Fig. 1, the process involves three main components: the deep task status estimator (regressor), a trajectory generator, and a controller to follow the trajectory required of the manipulated object (inverse-kinematics planner).

A. Task Status Estimator

We phrase the task status estimation as a two-step problem and propose a combined network that can be trained in an end-to-end manner (Fig. 2). A Successful outcome of the task is also considered an object class. Therefore, a classifier is not only required to detect the objects of the task, but also the successful outcome of a task (successful outcomes of three sample tasks can be seen in Fig. 3). At the first step, a Faster R-CNN [6] is used to detect the successful outcome of the task, the objects and their bounding boxes in the scene. If there is more than one object of interest, all of them are detected. A Faster R-CNN consists of two modules: a region proposal network (RPN), which outputs a set of rectangular object proposals, and an object detection CNN to utilize these. A Zeiler-Fergus network [22] with five shared convolution layers are used in both of these modules. The details for the RPN can be found in [6]. A Fast R-CNN is utilized for the object detection module [23]. A Faster R-CNN is trained end-to-end with back-propagation and stochastic gradient descent (SGD) [24]. The weights of the layers are obtained from a pre-trained Faster R-CNN for 20 objects [6] with a ZF net architecture. The remaining weight layers are initialized by drawing from a zero-mean Gaussian distribution with a standard deviation of 0.01. The learning rate is selected to be 0.001 for the first 10k mini-batches, and 0.0001 for the next 10k mini-batches on our

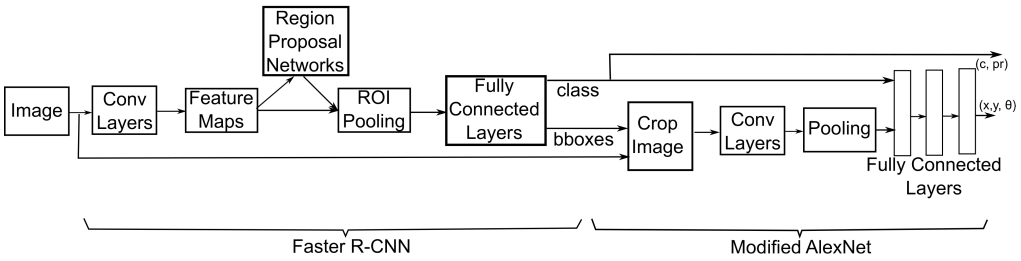


Fig. 2: The architecture of the combined networks. The outputs: location of the objects, the class and the estimated accuracy.

task dataset that will be explained. A momentum of 0.9 and a weight decay of 0.0005 are selected and implemented in Caffe [25].

At the second step, the pose of the objects in the 2D image plane, denoted by (x, y, θ), is estimated if the output class of the first step is not a "success". This signals that estimation continues when the task is in progress. The bounding boxes from the first step are used to crop the image and resize them to 227×227 RGB pixels. The weights from the convolution layers of the first step are not used since the task is to find a regression for the detected object. We use an architecture similar to the CNN trained on the ImageNet dataset [18] by Krizhevsky et al. [7] that consists of five convolutional and three fully-connected layers, totaling around 60 million parameters. We modify the network by concatenating the class output from the first step network with the output from the last pooling layer of the second step network to obtain the first fully connected layer of the second step network. The class corresponds to one of the object classes in one of the tasks. We apply regression to estimate the pose of the manipulated object. We train the CNN with an Euclidean loss with stochastic gradient descent using the loss function

$$loss(I) = \|\hat{\mathrm{x}} - \mathrm{x}\|_2 + \lambda \left\|\hat{\theta} - \theta\right\|_2 \qquad (1)$$

where x is the location in the image plane, θ is the in-plane orientation, and λ is a normalization factor to equalize the errors. The model is trained with a batch size of 64, momentum of 0.9, and weight decay of 0.0005. The learning rate is selected to be 0.00001 and decreases at every 10k mini-batches. We modified the parameters depending on the process power of computers, number of available training images and number of classes.

We obtain the training images with a few instances of the same task with different object types. First, we show the successful result of the task with different instances and move the completed task to different locations as shown in Fig. 3. The completed task is considered as an object and the location of the completed task is found by a probabilistic, appearance-based pose estimation (PAPE) [8]. The reason will be explained shortly.

Next, the tasks are executed once with each task instance. The main purpose of this execution is to obtain the trajectory of the task. These images are not directly used in training. We learn the object using (PAPE) [8] with RGB images only, but

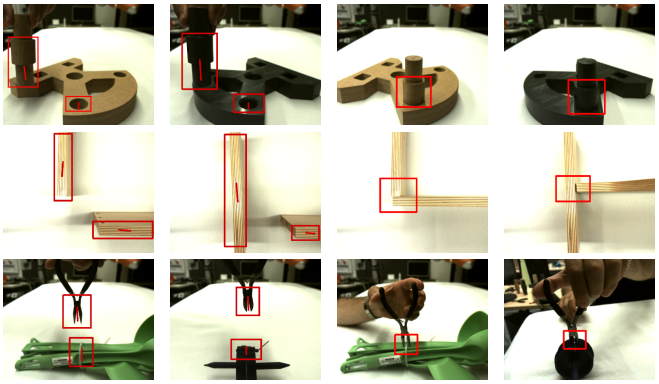


Fig. 3: Training images for different instances of different tasks. Top: Task 1 (Insertion), Middle: Task 2 (Tool Box), Bottom: Task 3 (Plier-Strip). Left two columns: bounding boxes, center and in-plane rotation for object and target, Right two columns: bounding box for successful outcomes.

any suitable pose estimation method that can work with RGB images can be used. We used PAPE due to its ease of learning the object models from appearance only and providing the pose of the object in the image plane accurately. We use one viewpoint of the object. After the trajectory of the object with respect to the target is found, to reduce the effect of different scales and positions of the object and the target, we produce images with different backgrounds at varying scales and positions around the trajectory. Since the trajectory extends over a limited region of the image, the training images are restricted to this area, therefore the number of training images is significantly reduced. The procedure for training the regression network is shown in Fig. 4. Here, the insertion task is shown as an example. It should be noted that the robot is not necessary at this stage. Each image is labeled with the image location, orientation, object type and bounding box. For each task there are at most three object types: manipulated object, target and the successful outcome. However, this can be less, as shown for the toolbox. In this case, both the target and the manipulated object are the same.

B. Trajectory Generator and Controller

In the previous section, we explained how to obtain the trajectory to train the networks for task status estimation. Now, we will describe how to obtain a trajectory from multiple instances of the task to be used by the robot to

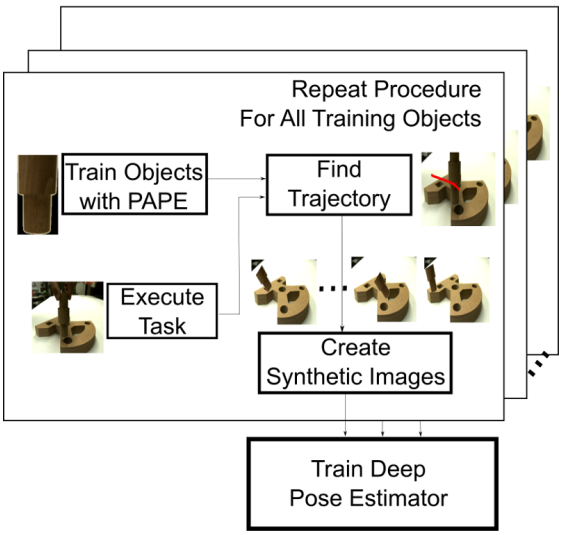
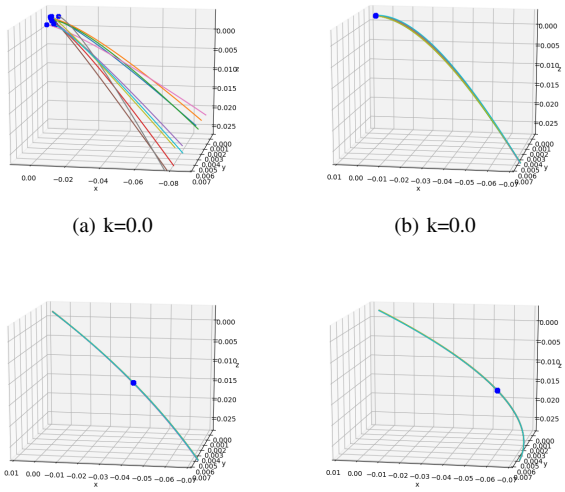


Fig. 4: Training regression network with CNNs



(a) k=0.0 (b) k=0.0

(c) k=0.5 (d) k=0.6

Fig. 5: Trajectories for the insertion skill created with ProMPs. The current pose of the manipulated object is shown with a "+" sign. The trajectory direction is from top to bottom. **a**) All trajectories from all the insertion task instances are shown; **b**) The trajectories that can be chosen after the object and the target pose are observed at the start of the motion (k=0.0); **c**) Updated trajectory after the robot moves and the object is observed at $k = 0.5$; **d**) Updated trajectory after the robot moves and the object is observed at $k = 0.6$.

update its motion if necessary.

We use ProMPs to generate trajectories in Cartesian coordinates [21]. Instead of observing the human user's action [21], we observe the task status by multiple cameras and update the current state of the robot accordingly. We assume that there exists a controller with inverse dynamics feedback to convert the output of the ProMPs into robot motion.

A sample trajectory for an insertion task is shown in Fig. 5. It should be noted that these are the trajectories obtained from pose estimation of objects in 6D using the PAPE method. The object that does not move in the training (e.g. the hole in the insertion task) is selected as the target. First, the task instances are collected as shown in Fig. 5a for each task. Then, after the observation is made, the task type is detected, and the object and the target poses are found by the task status estimation process. Since this is the first observation, these are considered to be the start and end points. The initial trajectory is shown in Fig. 5b, which is obtained by conditioning ProMPs on the initial trajectory points. There can always be errors in the observation; therefore, we assume that the true trajectory is corrupted with a Gaussian noise with variance $\sigma^2 = 0.0001$. We consider the total trajectory length as $K = 1$, and the current location on this trajectory is $k \in [0, 1]$ which is obtained from the task status estimation. After the initial trajectory, the robot moves along the trajectory with a step length of δk. After m steps, the location of the observed object $k = 0.5$ is shown in Fig. 5c. As can be seen, the trajectory is updated depending on the current location of the object. The current location of the object can be different from the planned location for various reasons, including errors in sensor calibration, errors in estimation algorithms, or any perturbation to the robot. For a step size of $\delta k = 0.1$, the object is observed at $k = 0.6$ as shown in Fig. 5d in the next step. When the output is estimated as *success*, if the accuracy of the task is higher than the threshold, then the task is completed. If the probability of the verified task is low, then the robot does small motions (i.e. in range $[-0.01, 0.01]$ cm) randomly until the accuracy of the task is high as observed with the task estimation module. If the task does not succeed after N steps, the task is classified as a failure.

III. Experiments

In this section, we evaluate our proposed method in three different ways. First, we evaluate the accuracy in outcome classification, then the regression evaluation is made, and finally the approach is tested on the robot. The experimental setup for taking training images is shown in Fig. 6. Although the camera is placed on the robot, the robot is not necessary during the training phase. We use three different tasks, insertion, mating of toolbox parts and bringing a plier close to a strip. We will denote them as Insertion, Tool Box and Plier-Strip respectively in the rest of the paper. A single object is manipulated in all of the tasks. We collect approximately 600 images from five instances of Insertion Task, three instances of Plier-Strip Task, and two instances of Tool Box Task. First, we collect images of the successful outcome for each task instance. The distances of the objects to the camera may vary, but their scales in the images do not change drastically. Then, we obtain the trajectories of the object during the execution of the task instance. Approximately 11000 synthetic images are created using these trajectories. We train the combined deep neural network as explained in Sec. II-A with defined

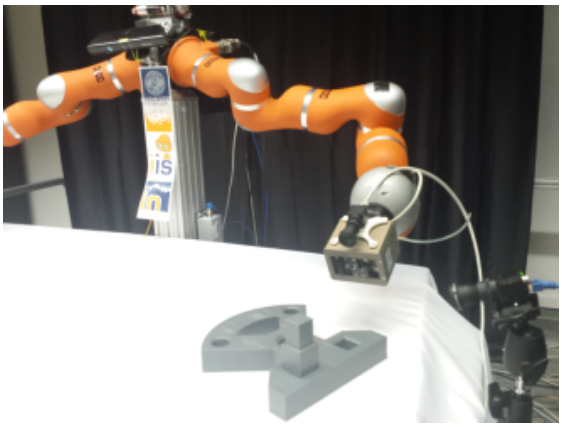


Fig. 6: Training setup for the experiments.

TABLE I: F1-Scores for Success Outcome Estimation

F1 scores	Insertion	Tool Box	Plier-Strip
Learned	0.722	0.708	0.871
Novel	0.707	0.643	0.671
All	0.712	0.700	0.772

parameters. We use this network throughout all experiments.

A. Task Outcome

In the first set of experiments, we evaluate the performance of our method on the estimation of the task outcome. The tasks are executed by a human for this experiment. Each task instance contains 1700 images of a task. In addition to the images of the learned objects, we collect images from two instances with novel objects for Insertion, one instance with novel objects for Plier-Strip and one instance with novel objects for Tool Box. For groundtruth, the images are labeled as successful or not successful by a human.

We show the F1-scores for the estimation of the success outcome in Table.I. We use F1-scores so that we consider both false negative and false positive results. It can be seen that for novel objects, which were not observed during training, the accuracy decreases as expected. The accuracy is highest for plier-strip, probably due to its distinct properties in its appearance. However, the accuracy decreases more than other objects when a novel plier is used. This is probably due to the change in the appearance of the strip. One of the reasons of failure in estimation is that we do not restrict the scene; therefore, in some of the scenes, the objects are very close to the camera, and the estimation is false. Some of the samples for correct estimations of successful task outcome and failures can be seen in Fig.7.

B. Task Status Estimator

In the second set of experiments, we evaluated the accuracy of regression for estimating the pose of the manipulated object in the image. We used two tasks for evaluation: Insertion and Plier-Strip. We skipped the Toolbox, since it was not possible to create a setup where the toolbox is manipulated by the robot and observed by a camera at the same time

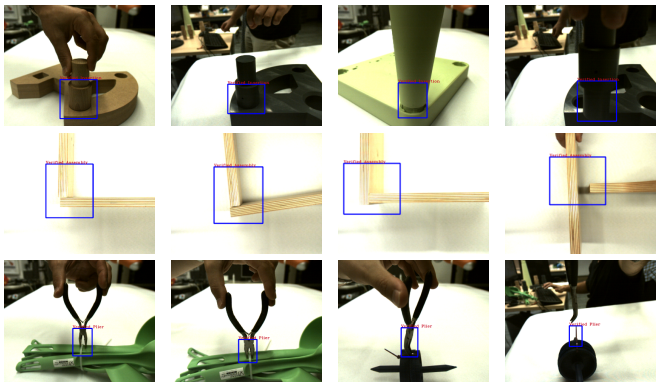


Fig. 7: Training images for different instances of different tasks. Top: Insertion, Middle: Tool Box, Bottom: Plier-Strip. Left three columns: Correctly estimated as verified, Third column: Novel task instances with novel objects, Right column: Some of the failure cases in detection.

TABLE II: Average Error for Object Position

	Insertion		Plier-Strip	
Avg Err (px)	**fRCNN**	**fRCNN + R**	**fRCNN**	**fRCNN + R**
Learned	17.6	14.6	16.3	14.6
Novel	20.9	18.7	15.3	14.7
All	18.5	15.8	16.1	14.7

due to configuration of the robot. The experimental setup is again similar to the previous experiment and the objects are shown by a human. The correct position and orientation of the object in the images are labeled by a human. To be able to evaluate the improvement introduced by the regression network proposed by our method, we compare two results. First, we take the center of the bounding box as the center of the object that is provided by the faster R-CNN network that exists in the first step of our approach denoted as fRCNN in Table.II. Then, we use the values computed by our regression network as the center position of the objects denoted as fRCNN+R.

The results are shown in Table. II. Learned task instances are the ones with which the network is trained, novel task instances are the ones that were not observed previously and all task instances are the combination of both learned and novel task instances. None of the test images were used to train the network. “Learned task” is used to indicate that the task was learned with the same object. 1700 images were used to compare the accuracy of two methods. Average error is the mean of the Euclidean distance in image plane between the center of the object found by the method and the groundtruth over all the test samples. The location error is reduced by 15% for the Insertion task, and 9% for the Plier-Strip task. As can be seen in Table.II, the error decreases in the Plier-Strip task for novel objects with fRCNN, which is probably due to an appearance property of the plier/strip combination that affects the network. The width and height of the images are 400x320 pixels accordingly. Adding the regression network increases the accuracy for all objects and

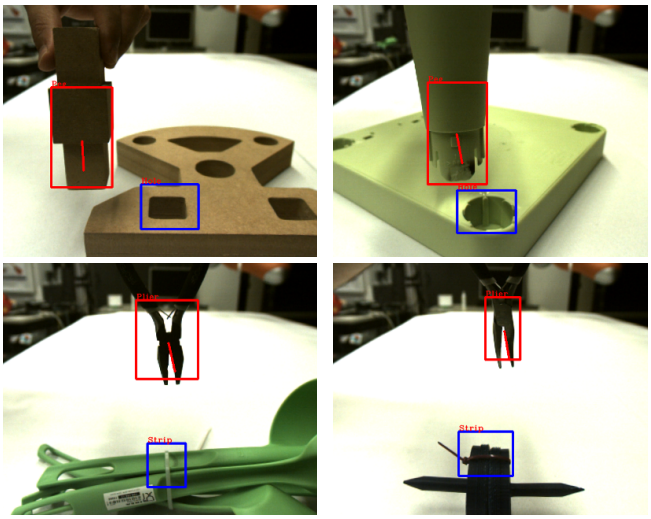


Fig. 8: Sample results for detection the center of the objects. Top: Insertion, Bottom: Plier-Strip, Left: Learned task instance, Right: Novel task instance.

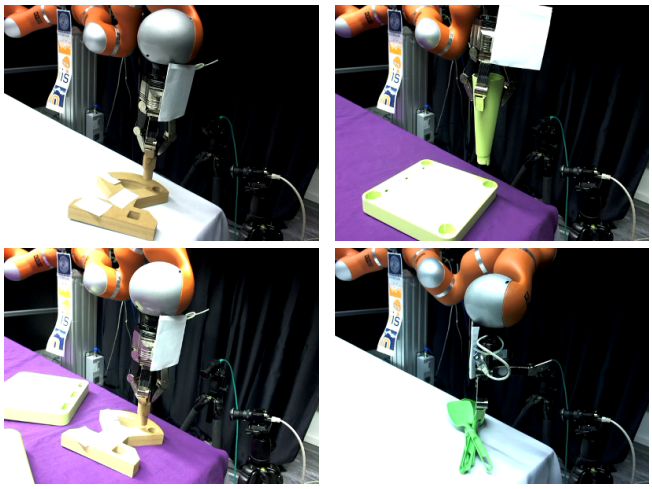


Fig. 9: Top Left: Setup is similar to the scenes where the tasks were learned. Top Right: The background color is different. Bottom Left: There is clutter and the background color is different. Bottom Right: The plier is shown to the cameras at a different angle.

TABLE III: Average Error for Object Orientation

Avg Err (rad)	Insertion	Plier-Strip
Learned	0.19	0.15
Novel	0.23	0.16
All	0.22	0.15

tasks. It should be noted that when the novel object is used, even if its appearance is different from other objects, the error does not increase drastically. Some of the results are shown in Fig. 8.

The results regarding the orientation error are shown in Table. III. As it can be observed, the error increases slightly when a novel task instance is estimated. This precision is sufficient for the tasks implemented in this paper. A meaningful comparison with fRCNN is not possible since it does not provide the orientation of the objects.

C. Robot Guidance

In this Section, we use our approach to test whether this will help the robot to achieve a high precision task by training on a different instance of the same task. We perform two tasks: Insertion and Plier-Strip. The tests are made with the learned task instance objects and novel task instance objects. The background is changed and clutter is added to test the performance of the approach to test if the method is capable of generalizing to new instances even with clutter in the scene. KUKA Light-Weight-Robot arm is used to manipulate the object. The robot setups are shown in Fig. 9.

The performance of the robot is affected by the changes in the background. When the learned instance of Insertion is used with the background similar to the one in training, the task is guided and completed successfully in 100 % of the 3 trials. When the background is changed by adding a different colored table cloth or clutter, the task is guided and completed successfully in 50 % of the 4 trials. When a new instance of the task is tested (green chair leg as shown in Fig. 9-Top Right) with different colored table cloth and clutter, the task is guided and completed successfully in 80 % of the 5 trials. For the Plier-Strip, when the angle of the manipulator alter slightly and a different task instance is used, the task is guided and completed successfully in 66 % of the 6 trials. The results show that if the task is trained with the same task instance as in the test case, even high precision tasks can be achieved with a high rate. When the environment conditions change, the performance starts to degrade. It should be noted that although the novel task instance chair leg insertion is visually different from the learned Insertion task instances, its performance is comparable to that of the learned task instance.

IV. Conclusion

We proposed a method to detect the outcome of the task and to guide the task with feedback spontaneously. The interaction required with the robot is not necessary during training. After a few task instances are shown to the system, the method is capable of generating a general model of the task and provide a feedback about the current status of the task. The improvement introduced by the regression network proposed by our method is analyzed in the experimental results. Furthermore, the experiments show that the accuracy of the method with novel objects is sufficient to guide the robot for a task that needs precision similar to the ones presented in our results.

Acknowledgement

The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 (Specific Programme Cooperation, Theme 3, Information and Communication Technologies) under grant agreement no. 610878, 3rd HAND.

REFERENCES

[1] R. Paolini, A. Rodriguez, S. S. Srinivasa, and M. T. Mason, "A data-driven statistical framework for post-grasp manipulation," *The International Journal of Robotics Research*, vol. 33, no. 4, pp. 600–615, 2014.

[2] P. Pastor, M. Kalakrishnan, S. Chitta, E. Theodorou, and S. Schaal, "Skill learning and task outcome prediction for manipulation," in *IEEE International Conference on Robotics and Automation*, 2011, pp. 3828–3834.

[3] J. Pile, G. B. Wanna, and N. Simaan, "Robot-assisted perception augmentation for online detection of insertion failure during cochlear implant surgery," *Robotica*, pp. 1–18, 2016.

[4] E. N. Malamas, E. G. M. Petrakis, M. Zervakis, L. Petit, and J. D. Legat, "A survey on industrial vision systems, applications and tools," *Image and Vision Computing*, vol. 21, no. 2, pp. 171–188, 2003.

[5] T. Y. Ong, Z. Samad, and M. M. Ratnam, "Solder joint inspection with multi-angle imaging and an artificial neural network," *International Journal of Advanced Manufacturing Technology*, vol. 38, no. 5-6, pp. 455–462, 2008.

[6] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances In Neural Information Processing Systems*, 2012, pp. 1–9.

[8] O. Erkent, D. Shukla, and J. Piater, "Integration of Probabilistic Pose Estimates From Multiple Views," in *European Conference on Computer Vision*, 10 2016, amsterdam, Netherlands.

[9] H. Nguyen and C. C. Kemp, "Autonomously learning to visually detect where manipulation will succeed," *Autonomous Robots*, vol. 36, no. 1-2, pp. 137–152, 2014.

[10] M. Hussein, "A review on vision-based control of flexible manipulators," *Advanced Robotics*, vol. 29, no. 24, pp. 1575–1585, 2015.

[11] A. Kendall, M. Grimes, and R. Cipolla, "PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization," in *Computer Vision, IEEE Int. Conf. on International Conference on*, 2015, p. 9.

[12] S. Miao, Z. J. Wang, and R. Liao, "A CNN Regression Approach for Real-time 2D/3D Registration," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1–1, 2016.

[13] B. Li, C. Shen, Y. Dai, A. Van Den Hengel, and M. He, "Depth and surface normal estimation from monocular images using regression on deep features and hierarchical CRFs," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1119–1127.

[14] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Advances in neural information processing systems*, 2014, pp. 1–9.

[15] A. Toshev and C. Szegedy, "Deeppose: Human pose estimation via deep neural networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 1653—-1660.

[16] X. Sun, Y. Wei, S. Liang, X. Tang, and J. Sun, "Cascaded hand pose regression," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 824–832, 2015.

[17] M. Jaderberg, K. Simonyan, A. Zisserman, and Others, "Spatial transformer networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 2017–2025.

[18] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[19] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-End Training of Deep Visuomotor Policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016.

[20] S. M. Khansari-Zadeh and O. Khatib, "Learning potential functions from human demonstrations with encapsulated dynamic and compliant behaviors," *Autonomous Robots*, vol. 41, no. 1, pp. 1–25, 2017.

[21] G. Maeda, M. Ewerton, R. Lioutikov, H. B. Amor, J. Peters, and G. Neumann, "Learning interaction for collaborative tasks with probabilistic movement primitives," in *Humanoid Robots (Humanoids), 14th IEEE-RAS International Conference on*. IEEE, 2014, pp. 527–534.

[22] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[23] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1440–1448.

[24] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[25] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.