Towards a Programming-Free Robotic System for Assembly Tasks Using Intuitive Interactions

Nicolas Gauthier(✉), Wenyu Liang(✉), Qianli Xu(✉), Fen Fang(✉), Liyuan Li(✉), Ruihan Gao, Yan Wu(✉), and Joo Hwee Lim(✉)

Institute for Infocomm Research, A*STAR, Singapore 138632, Singapore
{nicolas_gauthier,liang_wenyu,qxu,fang_fen,lyli,gao_ruihan,wuy,joohwee}@i2r.a-star.edu.sg

Abstract. Although industrial robots are successfully deployed in many assembly processes, high-mix, low-volume applications are still difficult to automate, as they involve small batches of frequently changing parts. Setting up a robotic system for these tasks requires repeated reprogramming by expert users, incurring extra time and costs. In this paper, we present a solution which enables a robot to learn new objects and new tasks from non-expert users without the need for programming. The use case presented here is the assembly of a gearbox mechanism. In the proposed solution, first, the robot can autonomously register new objects using a visual exploration routine, and train a deep learning model for object detection accordingly. Secondly, the user can teach new tasks to the system via visual demonstration in a natural manner. Finally, using multimodal perception from RGB-D (color and depth) cameras and a tactile sensor, the robot can execute the taught tasks with adaptation to changing configurations. Depending on the task requirements, it can also activate human-robot collaboration capabilities. In summary, these three main modules enable any non-expert user to configure a robot for new applications in a fast and intuitive way.

Keywords: Robotic manipulation · Multimodal perception · Object and task teaching · Grasping and insertion · Human-robot collaboration

1 Introduction

With the intensified development of robotic technologies, robots are expected to work alongside and together with humans to complete tasks and improve productivity. This requires Human-Robot Collaboration (HRC) capabilities [1,2], to adapt to changing configurations and human behaviours. The current COVID-19 pandemic's social distancing requirements force companies to operate at a reduced capacity, while production demand remains strong and requires adaptability. Collaborative robots (cobots) can help to address this challenge, provided

This research is supported by the Agency for Science, Technology and Research (A*STAR) under its AME Programmatic Funding Scheme (Project #A18A2b0046).

H. Li et al. (Eds.): ICSR 2021, LNAI 13086, pp. 203–215, 2021.
https://doi.org/10.1007/978-3-030-90525-5_18

that they are safe, flexible, robust, easy and fast to deploy. Notably, deployment time is critical, to ensure little downtime and fast return on investment.

However, such systems are currently highly constrained in their ability to collaborate with humans: (i) by their inability to interact naturally through common modalities like vision, speech and touch; and (ii) by their limited ability to adapt in task performance (re-programming is needed even for small changes in procedure) [3]. In this paper, we seek to develop a suite of capabilities that would allow cobots to adapt to humans and their work environments through natural interactions and robust learning, to remove the need for specialized infrastructure, expert programming and human training, thus reducing deployment costs.

Fast object learning is a desirable feature, to quickly adapt to new objects, but is still underdeveloped for many real-world problems. Legacy deep learning-based methods have achieved remarkable performance in object recognition but are not directly usable for domain-specific problems: off the shelf models (i.e., pre-trained on large-scale, generic dataset) perform rather poorly on the recognition of novel industrial objects [4]. To achieve good performance, sufficient high-quality training data is required, which is costly and time-consuming. Some works deal with the generation of high-quality training data by (i) simplifying the data collection and annotation process using specific devices or user interfaces [5], (ii) automating the generation of data samples through augmentation and synthesis [6]. For the former, it still involves a lengthy process of data sampling and manual annotation. For the latter, the characteristics of synthesized data need not match with those of actual data from domain-specific operating environments, leading to unstable and deteriorated performance. A few works resort to interactive data collection and annotation [7–9]. The agent is equipped with the ability to register object instances with human guidance. However, they are restricted by the limited functionalities of hardware (e.g., robot mobility) and software, thus only addressing small-scale toy problems. It is paramount to fill this gap with an effective interaction protocol for agents to learn from human, and further to perform self-learning similar to human learners.

Additionally, setting-up a cobot without programming requires real-time parsing of human inputs into recognized task activities and human actions, to then relate them to a task representation for task understanding and learning. This research will notably develop capabilities for modelling task structure and understanding a human worker's roles during task execution and collaboration, adapting to uncertainties and exceptions over time. There is a rich and growing literature of robotic learning from demonstration (LfD) due to the importance and advantages of scaling up robots for new tasks without hand-crafted programming, as mentioned above. A full review of the topic is beyond the scope of this paper; readers may refer to recent surveys [10–12]. Existing LfD solutions focus on capturing signals directly related to the robot operations, from kinaesthetic, motion-sensor, or tele-operated demonstrations on the end-effector's pose, or force and motion trajectory. They focus on learning a policy of task execution using models like Hidden Markov Models (HMMs), Gaussian Mixture Models (GMMs), Dynamic Movement Primitives (DMPs), and Task-Parameterised

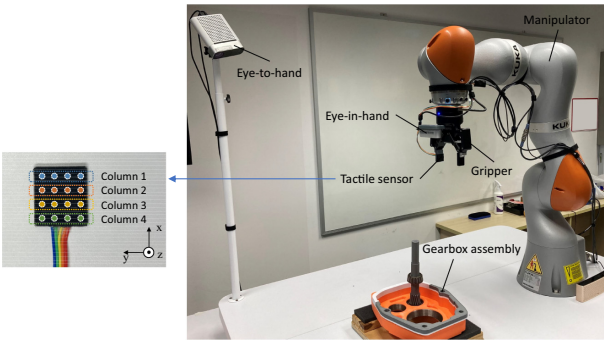

Fig. 1. Robotic system setup

Gaussian Mixture Model (TP-GMM), as well as Deep Reinforcement Learning (DRL) recently. Our method instead focuses on learning a general and conceptual task representation based on visual observation of human hand actions which is beyond the state and action space of the robot's pose, force or motion.

2 Robotic System

The robotic system used in this work is based on a robot manipulator, coupled to two cameras and a 2-finger gripper with a tactile sensor.

Figure 1 shows the overall robotic system setup. More specifically, it consists of a 7-degree-of-freedom (7-DoF) *KUKA LBR iiwa* robot manipulator, which enables a wide range of motions, as well as safe HRC with its compliant mode and collision detection capability. To identify and locate the objects to interact with, two RGB-D cameras are used: a fixed eye-to-hand one (*Microsoft Kinect Azure*) detects the objects in the robot workspace, and estimates their 3D positions with respect to the robot. Based on this, the robot can move closer to a desired object, and then better estimate its location by using an eye-in-hand camera (*Intel RealSense D435*). Eventually, it is anticipated that these vision outputs have errors due to calibration imperfections, occlusions, or limited resolution for instance, and that tasks like grasping will happen blindly at some point (when the robot gets too close to the object to see it). To overcome these limitations, the gripper (*Robotiq 2F-85*) is equipped with a *XELA uSkin XR1944* tactile sensor, which is used to estimate with how much offset an object has been grasped. This main suite of sensors is used to bring down the perceptual error in order to interact with the gearbox components, in this use case.

Finally, the whole system also comes with an intuitive Graphical User Interface (GUI), available on a touch monitor, allowing easy control and set-up of the robot. The user also gets audio feedback on the machine's status and actions. The different components of the system, both hardware and software, are brought together with The Robot Operating System (ROS) [13]. The overall software is made of modules coded in C++ and Python.

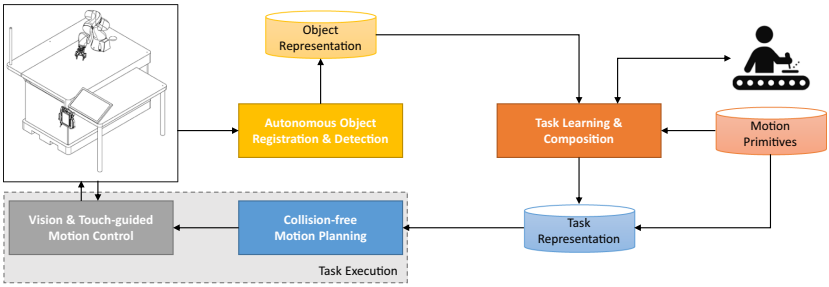

Fig. 2. Block diagram of the proposed framework

3 Framework for Programming-Free System

To achieve programming-free task execution, the robotic system should be capable of detecting objects, understanding tasks, and precisely executing them. Therefore, the solution proposed in this paper mainly consists of three modules: object teaching, task teaching, and task execution. The corresponding block diagram is depicted in Fig. 2. The object teaching module is for autonomous object registration and detection, the task teaching module is for understanding tasks from visual human demonstrations, and the task execution module is for executing the learned and composed tasks, using multimodal perception.

3.1 Object Teaching

There are three key steps for object teaching as explained below. The entire process is summarised in Fig. 3.

Canonical View Selection. This first step is used to actively capture an object's informative views, by exploring various viewpoints around it and and select the most informative ones based on a criteria termed goodness of view (GOV).

- **Object Segmentation** At a viewpoint, the RGB-D camera provides a color image and point cloud of the scene. First, we need to get a regional mask of the object. For this, the Point Cloud Library (PCL) is used, to estimate the dominant plane and separate it from the object to register. Alternatively, for small or flat objects, which are difficult to be isolated, we use color information to perform segmentation (provided that there is adequate contrast between the object and table top). The mask is then used to extract the object from the RGB image.
- **Goodness of view** The *goodness* of a viewpoint is defined based on the informativeness of the object, which is computed from a set of basic visual features [14,15]. In this work, silhouette length, depth distribution, curvature entropy, and color entropy are considered in the computation [16]. Canonical views are registered as those with higher combined GOV.

– **Viewpoint exploration** Canonical viewpoints are sampled by evaluating the aggregated GOV of the visited viewpoints on-the-fly. It is assumed that the RGB-D camera is calibrated against a set of viewpoints on a spherical surface and the target object is located at the sphere center (cf Fig. 3). The robot then follows the OnLIne Viewpoint Exploration (OLIVE) method [16] to visit the viewpoints. The RGB-D data captured at each viewpoint is used to extract object features and compute the GOV. Given any viewpoint, the robot searches the local maxima of GOV, where the GOV is computed as the weighted sum of individual GOV metrics. Once the local maxima is found, the next view is chosen as one with the largest geographical distance to the current view.

Data Augmentation. Contemporary object detectors require a reasonably large amount of training images to reach good accuracy. The image samples from the previous step may be too few to reach the desired detection performance. To enhance the dataset, we perform a series of image manipulation techniques to augment the data. In essence, we superimpose the object's image (mask) onto various backgrounds to generate synthesized data. The techniques adopted in this study include (i) 2D variation and 3D transformation to add variations of viewing perspective [17], (ii) blending to remove boundary artifacts [6], and (iii) object scaling to allow optimal object sizes [18].

Training Object Detectors. Once the training images of one or a few novel objects have been generated, the system performs on-site object detector training. Ideally, the object learning is conducted incrementally, whereby new object classes are added gradually to existing classes. In other words, the detector learns new object classes without catastrophic forgetting of old ones. In this study, we explored several incremental learning algorithms, such as the methods presented in [19] and [20]. Notably, these incremental learning techniques still require lengthy training time due to the distillation operation. Hence, in practice, we implemented the training protocol of aggregated training (i.e., to include all training images of known and unknown classes) on the conventional framework of Faster-RCNN with ResNet101 [21].

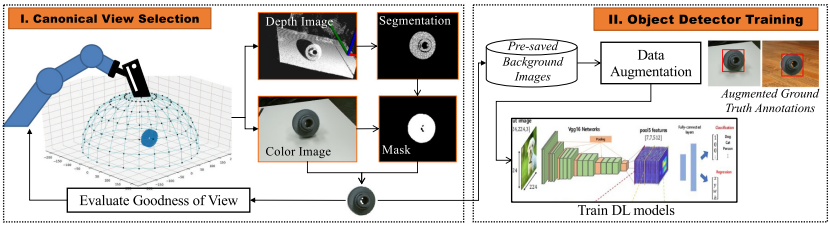

Fig. 3. Object teaching process

3.2 Task Teaching

The task teaching module has been developed to allow any non-expert user to teach new tasks to the manipulator, simply based on visual demonstration. For this, we target at establishing a structured semantic representation of robotic task which matches human common knowledge of task representation. To this end, we propose a novel graph-based task representation: Compounded Task Graph (CTG). The CTG task model provides a concept-level hierarchical representation of industrial tasks. There are three layers of nodes in CTG:

- **Atomic action** (A_h): this node represents an atomic hand action in an industrial task, such as *pick_up*, *move_to*, *release*, etc. The atomic action nodes form the bottom layer of CTG.
- **Primary task** (T_p): this node represents a sequence of atomic hand actions to complete a basic task. It is a parent node of a sequence of atomic actions.
- **Compounded task** (T_c): this node represents a complex task which involves several primary tasks, and/or other compounded tasks as sub-tasks of it. It is a parent node which has a few primary task child nodes in the lower layer, and/or other compounded tasks child nodes (sub-tasks) in the same layer.

Task Teaching on Vision Perception. As mentioned previously, the aim here is to reduce the need for human expertise and coding efforts to produce the CTG of a new task when deploying a cobot. To this end, we propose a novel task teaching approach based on learning from visual human demonstration, to automatically generate the CTG of a new task.

First, the vision system runs a previously trained Faster R-CNN to detect task-related objects and hands in each incoming frame. A tracking algorithm relying on visual working memory is designed to track the detected objects and hands, and estimate their inter-frame movements. Based on these visual observations, a two-level probabilistic logic reasoning is designed to recognize task-related hand actions in real-time.

The first level, visual reasoning, tries to capture four kinds of hand movements, expressed as probabilistic logic rules:

- $move\_to(h_i, o_j)$:- $dist\_reduct(h_i, o_j)$ & $orient\_consistency(v_i, v_{ij})$;
- $hold(h_i, o_j)$:- $hand\_static(h_i)$ & $hand\_touch(h_i, o_j)$ & $dist\_const(h_i, o_j)$;
- $move\_together(h_i, o_j)$:- $hand\_move(h_i)$ & $hand\_touch(h_i, o_j)$ & $dist\_const(h_i, o_j)$ & $orient\_consistency(v_i, v_j)$;
- $move\_away(h_i, o_j)$:- $hand\_move(h_i)$ & $dist\_increase(h_i, o_j)$ & $orient\_consistency(v_i, v_{ji})$;

where, for example, for $move\_to(h_i, o_j)$, if the hand h_i is moving towards object o_j, the distance between h_i and o_j should be reduced constantly, and the motion vector of h_i should be aligned with the orientation vector from h_i to o_j. Hence, if the probabilities of $dist\_reduct(h_i, o_j)$ and $orient\_consistency(v_i, v_{ij})$ are high based on visual observations, the probability of $move\_to(h_i, o_j)$ is high.

The second level, probabilistic reasoning, tries to capture atomic hand actions. During a short duration of recent observations, the algorithm predicts

the probabilities of atomic actions based on accumulated evidence of basic hand movements. The probabilistic logic rules are expressed as:

- $pick\_up(h_i, o_j)$:- $move\_to(h_i, o_j)$ & $hold(h_i, o_j)$ & $move\_together(h_i, o_j)$;
- $move\_a2b(h_i, o_j, o_k)$:- $move\_together(h_i, o_j)$ & $move\_to(h_i, o_k)$;
- $release\_a2b(h_i, o_j, o_k)$:- $move\_a2b(h_i, o_j, o_k)$ & $hold(h_i, o_j)$ & $move\_away(h_i, o_j)$ & $move\_away(h_i, o_k)$;

where, for $pick\_up(h_i, o_j)$, if the hand h_i has moved to object o_j, has held the object statically for a while, and then moves together with it, it means that an atomic action of $pick\_up(h_i, o_j)$ has happened.

During the demonstration, the vision system tracks the events and produces a list of atomic actions gradually. Once the demonstration is completed, the system performs Bayesian inference on the list of atomic actions to extract the task's related actions and associated objects. For example, for a demonstrated task of $insert(a, b)$, Bayesian inference tries to find the maximum joint probability:

$$P_{task}^{insert(a,b)} = \max_{i,j,k} \left[P_r(i,j,k|a,b) P_m(i,j,k|a,b) P_p(i,j|a) P_t(m,r) P_t(p,m) \right], \quad (1)$$

where $P_r(i,j,k|a,b)$=$P(release\_a2b(h_i, o_j, o_k)|o_j = a, o_k = b)$, $P_m(i,j,k|a,b)$= $P(move\_a2b(h_i, o_j, o_k)|o_j = a, o_k = b)$, $P_p(i,j|a)$=$P(pick\_up(h_i, o_j)|o_j = a)$, and P_t represents temporal constraint: $P_t(m,r)$= $P(t_m < t_r)$. Once the maximum joint probability is larger than a threshold, the observed task is confirmed, and a CTG is generated with corresponding atomic actions and associated objects.

Task Composition with Graphical User Interface. With the capabilities presented above, any non-expert user is able to both teach new objects and demonstrate new tasks to the robot without manually programming anything. Furthermore, one can also leverage on the previously taught tasks, using the GUI to compose more complex or repetitive ones, rather than demonstrating them again. We term this approach "Programming-Free" system. Additionally, for tasks requiring HRC, the user can chose to activate the robot's impedance mode to make it compliant. This ensures a safe execution, as well as ease for joint manipulation, as the user can manually move the end-effector.

3.3 Task Execution

Once a task has been taught (e.g. accurate, adaptable grasping and insertion of a shaft), multimodal perception is used to execute it, without having to perform extensive calibration and risk erroneous and unsafe execution.

Locating Objects Using Visual Inputs. In order to locate objects in the robot workspace and interact with them, the system is equipped with two RGB-D cameras. The first one is fixed and has a bird's eye view on the scene. Its extrinsic parameters have been calibrated with respect to the robot frame, using [22], meaning that its position and orientation are known in this frame. In order to

estimate the objects' 3D positions, the system first runs the previously mentioned object detector on the RGB image. Then, to estimate the 3D position of a specific object, it takes the center of the object's bounding box and gets the depth of the corresponding pixel (RGB and depth image have been registered).

Then, the following formula is used to get the 3D position of the mentioned pixel, in the camera frame: $z = d/\alpha$; $x = (n - c_x) * z/f_x$; $y = (m - c_y) * z/f_y$, where d is the depth read at the pixel of coordinates (n, m), $\alpha = 1$ or 1000 depending on whether depth is given in m or mm. f_x, f_y, c_x and c_y come from the intrinsic matrix $K = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$.

Finally, the system converts the obtained coordinates into the robot frame, using the following calculation:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}_{robot} = \begin{bmatrix} 1 - 2(q_y^2 + q_z^2) & 2(q_x.q_y - q_z.q_w) & 2(q_x.q_z + q_y.q_w) & t_x \\ 2(q_x.q_y + q_z.q_w) & 1 - 2(q_x^2 + q_z^2) & 2(q_y.q_z - q_x.q_w) & t_y \\ 2(q_x.q_z - q_y.q_w) & 2(q_y.q_z + q_x.q_w) & 1 - 2(q_x^2 + q_y^2) & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}_{camera} \quad (2)$$

where q_x, q_y, q_z, q_w and t_x, t_y, t_z, represent the camera's extrinsic parameters.

A similar process has been used for the in-hand camera: its extrinsic parameters have been calibrated with respect to the robot's end-effector. Then during run-time, this information is combined to the end-effector's actual position, to compute the camera's position and orientation with respect to the robot's fixed frame. (2) is then applied to compute the 3D positions of objects perceived through it.

A critical step in the studied use-case (gearbox assembly), is the insertion of a shaft into a circular bearing cup. There are three of them and the user can choose which one to insert in, via the GUI or demonstration. To perform this, visual perception inputs are used to locate this insertion target, based on a circle detection and tracking algorithm we developed: once it locates the target object, it crops the image around it and generates a list of potential circles for the insertion. Then, using a custom-trained support vector machine (SVM), it will output the true candidates, which are the three insertion possibilities. This solution is able to track the circles as well as keep in memory their position with respect to each other (output example is shown on Fig. 6).

Grasping and Inserting Objects with Tactile Perception. Once the cameras detects the object of interest and locate it, the robot manipulator approaches and grasps it with the gripper. However, due to internal and external uncertainties, such as lighting change, viewing angle, calibration error, resolution and occlusion, the vision accuracy is compromised. Therefore, we propose to attach a tactile sensor to one of the gripper fingertips. This device detects objects as well as the contact surfaces profiles, and provides fine-scale location and shape information. The *XELA uSkin XR1944* sensor, as shown in the left side of Fig. 1, is used in this research. It has 16 taxels arranged in a 4-by-4 array that detect forces in Cartesian coordinates [23].

In this work, the Naive Bayes algorithm is employed to train a classifier to detect the position offset of the object (e.g., a shaft) being grasped with respect to the gripper's fingertip. Assuming that the gripper applies a constant force to grasp the shaft statically, the algorithm takes the normal forces, i.e., forces in z direction (cf. Figure 1), as input features and assumes them to follow Gaussian distribution. Given the regular shape of the shaft, the normal forces are averaged along four columns (y direction) of the taxel array and the position offset in the x direction is estimated following the Bayes' Theorem $p(y_k|\boldsymbol{x}) = \frac{p(y_k)*p(\boldsymbol{x}|y_k)}{p(\boldsymbol{x})}$, where y_k represents label of class k, $\boldsymbol{x}$ is $\mathbb{R}^n$ feature vector and $n = 4$. Gaussian kernel is used to approximate the likelihood of features. The feature plot shown in Fig. 4 demonstrates the distribution of four extracted features. Each feature corresponds to the average reading of one sensor's column. Each subplot represents one feature, the horizontal axis represents the position offset in x direction in mm, and the vertical axis represents the force readings.

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}}\exp(-\frac{(x_i - \mu_y)^2}{2\sigma_y^2}) \tag{3}$$

where μ_y and σ_y are estimated class mean and standard deviation.

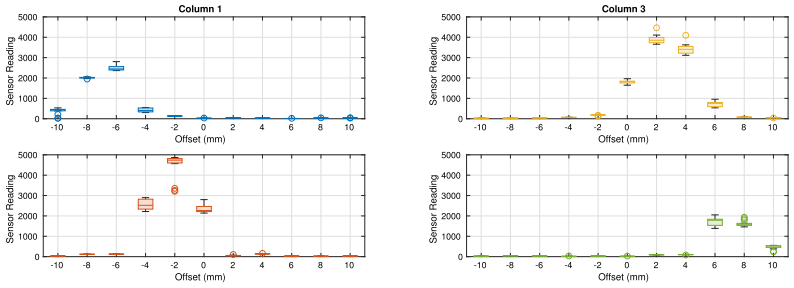

Fig. 4. Boxplot of four extracted features

To validate this approach, a test dataset has been collected: the shaft has been grasped with different offsets, equally spaced. We then compared ground truth values to the predicted ones. Overlapping between two sets of values demonstrated that the offset position can be estimated with an accuracy of less than 2 mm thanks to the proposed algorithm.

4 Experimental Results and Discussions

To validate the effectiveness and performance of the proposed system, experiments have been conducted with the robotic system setup shown in Fig. 1. As mentioned previously, a critical step to perform is the insertion of a shaft into one of the casing's bearing cup. We use this task for validation.

4.1 Object Teaching

The object teaching module (using four pieces of Geforce RTX 2080 in the training, one for inference) can achieve the mAP of more than 0.86 after 10 epochs. More detailed results and discussions can be found in [24].

4.2 Task Teaching

To validate the effectiveness of the task teaching, three different users are asked to demonstrate to the robotic programming-free system the task of *inserting a specific shaft into a casing hole*. It takes 18.7 s on average for the robot to learn the task using the proposed method while it takes roughly twice as much time for a user to program the task for the robot using our designed GUI. Additionally, while we have not documented the task learning success rate, experiments have shown that the algorithm is robust and reliably extract the relevant task structure. Existing benchmark datasets and evaluation metrics might not be applicable to this specific branch of task learning, yet the results obtain show that the proposed task teaching method is effective, efficient and intuitive.

4.3 Task Reproduction

For comparison purposes, an ablation study is performed, with the following configurations, for the shaft insertion task: (i) global (eye-to-hand) camera only; (ii) global camera + tactile sensor; (iii) global camera + eye-in-hand camera; and (iv) global camera + eye-in-hand camera + tactile sensor. 20 tests are carried out for each configuration, respectively, with similar objects positions and difficulty. The success rates of using different configurations are shown in Table 1.

Table 1. Success rate of object execution with different configurations

Configuration	Grasping	Insertion
Global camera only	80%	5%
Global camera + tactile sensor	80%	15%
Global + in-hand camera	100%	60%
Both cameras + tactile sensor	100%	85%

It can be observed that relying only on the global camera gives decent grasping results, but even in successful cases, experiments showed that it would be done either with an offset or the object is not picked up straight. As for the insertion, this sensor alone fails to properly estimate the insertion target (hole)

location, mainly due to viewing angle restrictions. Adding the in-hand camera to the system considerably enhances the results for both grasping and insertion. It helps to properly center the gripper on the object to grasp, and prevents the gripper from colliding with the object. It also helps to more accurately locate the target hole for insertion. Yet, the visual perception inputs are not perfect, which results in a slight offset between the object axis and the gripper center. It can be observed that the tactile sensor combined to the proposed algorithm greatly helps to compensate for this and improves the insertion success rate.

4.4 Collaborative Task

The user can also request the system to help him/her in collaborative tasks. A perfect use-case for this is the insertion of three shafts simultaneously, as these components need to be inserted into the casing at the same time. It is impossible for a worker or the robot to single-handedly complete this task reliably. Thus, a new collaborative insertion task can be composed via the GUI, using the previously taught insertion action. The step of insertion in the learnt task will be set to be done in collaborative mode, i.e., the robot compliant mode will be enabled for safer and easier execution, while waiting for human trigger.

Fig. 5. Robot operations during collaborative insertion task

The robot operations for this collaborative task are shown in Fig. 5. In practice, the system will stop and inform the user after it grasps the shaft and moves it above the casing. It will then trigger the insertion once it senses a pulling-down force/motion from the co-worker. Eventually, the collaborative function allows easy HRC for a challenging task.

4.5 GUI

Most of the interactions with the system are done via a custom GUI, available on a touch monitor. Figure 6 depicts the different use cases.

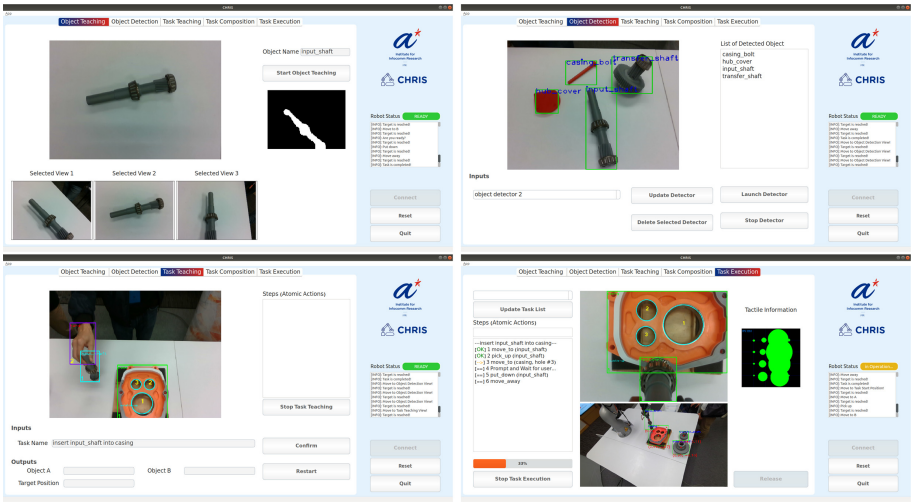

Fig. 6. GUI for object teaching, object detection, task teaching and task execution

5 Conclusion

In this paper, a framework with intuitive interactions is proposed towards programming-free robotic system setup. The framework consists of object teaching, vision-based task teaching and multimodal-perception-based task execution. Several experiments are carried out with the use case of an assembly task (shaft insertion) to validate the effectiveness and performance of the proposed framework. The experimental results show that the proposed method can not only help the robot to register objects and learn tasks in a fast way but also improve the task execution precision and robustness through multimodal perception. Compared to other simpler configurations, the ablation study showed that combining two cameras and a tactile sensor brings considerable improvement for grasping and insertion tasks.

References

1. Bicchi, A., Peshkin, M.A., Colgate, J.E.: Safety for physical human–robot interaction. In: Siciliano, B. (ed.) Springer Handbook of Robotics, vol. Springe, pp. 1335–1348. (2008). https://doi.org/10.1007/978-3-540-30301-5_58
2. Wang, L., Liu, S., Liu, H., Wang, X.V.: Overview of human-robot collaboration in manufacturing. In: Wang, L., Majstorovic, V.D., Mourtzis, D., Carpanzano, E., Moroni, G., Galantucci, L.M. (eds.) Proceedings of 5th International Conference on the Industry 4.0 Model for Advanced Manufacturing. LNME, pp. 15–58. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-46212-3_2
3. El Zaatari, S., Marei, M., Li, W., Usman, Z.: Cobot programming for collaborative industrial tasks: an overview. Robot. Auton. Syst. **116**, 162–180 (2019)

4. Pasquale, G., Ciliberto, C., Odone, F., Rosasco, L., Natale, L.: Are we done with object recognition? The iCub robot's perspective. Robot. Auton. Syst. **112**, 260–281 (2019)
5. Label fusion: a pipeline for generating ground truth labels for real RGBD data of cluttered scenes. In: ICRA2018, pp. 1–8 (2018)
6. Dwibedi, D., Misra, I., Hebert, M.: Cut, paste and learn: Surprisingly easy synthesis for instance detection. In: ICCV2017, pp. 1310–1319 (2017)
7. Dehghan, M., et al.: Online object and task learning via human robot interaction. In: ICRA2019, pp. 2132–2138 (2019)
8. Kasaei, S., et al.: Interactive open-ended learning for 3D object recognition: an approach and experiments. J. Intell. Robot. Syst. **80**, 537–553 (2015). https://doi.org/10.1007/s10846-015-0189-z
9. Kasaei, A., et al.: Perceiving, learning, and recognizing 3D objects: an approach to cognitive service robots. In: AAAI-2018 (2018)
10. Argall, B.D., Chernova, S., Veloso, M., Browning, B.: A survey of robot learning from demonstration. Robot. Auton. Syst. **57**, 469–483 (2009)
11. Zhu, Z., Huosheng, H.: Robot learning from demonstration in robotic assembly: a survey. Robotics **7**, 17 (2018)
12. Jangwon Lee. A survey of robot learning from demonstrations for Human-Robot Collaboration. arXiv e-prints, page arXiv:1710.08789, October 2017
13. Stanford Artificial Intelligence Laboratory et al. Robotic operating system. ROS Melodic Morenia. https://www.ros.org. Accessed 23 May 2018
14. Dutagaci, H., Cheung, C.P., Godil, A.: A benchmark for best view selection of 3D objects. In: 3DOR 2010, pp. 45–50 (2010)
15. Polonsky, O., et al.: What's in an image? Towards the computation of the "best" view of an object. Vis. Comput. **21**, 840–847 (2005)
16. Q. Xu et al. Active image sampling on canonical views for novel object detection. In: ICIP 2020, pp. 2241–2245. IEEE (2020)
17. Fang, F., et al.: Self-teaching strategy for learning to recognize novel objects in collaborative robots. In: ICRAI 2019, pp. 18–23 (2019)
18. Fang, F., et al.: Detecting objects with high object region percentage. In: ICPR 2020, pp. 7173–7180 (2020)
19. Shmelkov, K., Schmid, C., Alahari, K.: Incremental learning of object detectors without catastrophic forgetting. In: ICCV 2017, pp. 3400–3409 (2017)
20. Peng, C., Zhao, K., Lovell, B.C.: Faster ILOD: incremental learning for object detectors based on faster RCNN. Pattern Recognit. Lett. **140**, 109–115 (2020)
21. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. Adv. Neural Inf. Process. Syst. **28**, 91–99 (2015)
22. easy_handeye. https://github.com/IFL-CAMP/easy_handeye
23. uSkin sensors. https://xelarobotics.com/en/uskin-sensor
24. Xu, Q., et al.: TAILOR: teaching with active and incremental learning for object registration. In: AAAI 2021, vol. 35, pp. 16120–16123 (2021)