Robotics and Autonomous Systems 99 (2018) 1–16

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Automated assembly skill acquisition and implementation through human demonstration

Ye Gu [a], Weihua Sheng [b,*], Christopher Crick [c], Yongsheng Ou [d]

[a] Shenzhen Academy of Robotics, Shenzhen, Guangdong, 518000, China
[b] School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK 74078, USA
[c] Computer Science Department, Oklahoma State University, Stillwater, OK 74078, USA
[d] Shenzhen Institutes of Advanced Technology. Chinese Academy of Sciences, Shenzhen, Guangdong 518055, China

HIGHLIGHTS

- This paper develops an overall framework for robot skill learning through human demonstration.
- A Portable Assembly Demonstration (PAD) system is developed as the learning platform.
- Both human motion and object information are considered for action recognition.
- Assembly states are estimated based on the 3D part models created by a 3D scanner.
- The overall framework is evaluated on a Baxter robot.

ARTICLE INFO

Article history:
Received 14 July 2016
Received in revised form 8 September 2017
Accepted 2 October 2017
Available online 18 October 2017

Keywords:
Object recognition
Action recognition
Assembly state estimation
Learning by demonstration

ABSTRACT

Acquiring robot assembly skills through human demonstration is an important research problem and can be used to quickly program robots in future manufacturing industries. To teach robots complex assembly skills, the robots should be able to recognize the objects (parts and tools) involved, the actions applied, and the effect of the actions on the parts. It is non-trivial to recognize the subtle assembly actions. To estimate the effect of the actions on the assembly part is also challenging due to the small part sizes. In this paper, using a RGB-D camera, we build a Portable Assembly Demonstration (PAD) system which can automatically recognize the objects (parts/tools) involved, the actions conducted and the assembly states characterizing the spatial relationship among the parts. The experiment results proved that this PAD system can generate a high level assembly script with decent accuracy in object and action recognition as well as assembly state estimation. The assembly script is successfully implemented on a Baxter robot.

1. Introduction

1.1. Motivation

With the emergence of advanced robots, the demand for teaching robot complex skills increases [1,2]. If a robot can replace or assist a human worker doing complex assembly tasks, the labor cost can be greatly saved. Compared to the heavy-duty industrial robot manipulators employed in automotive industry, light weight arm (LWA) robots and dual-arm robots have emerged recently for small batch manufacturing. For example, Foxconn Technology Group has already deployed their own assembly robot called "Foxbot" in their factories [3]. Dual-arm robots have been developed by several companies that mainly target small part assembly. These robots include Rethink Robotics' Baxter robot [4] and Kawada's Nextage robot [5], among others.

Currently these robots can only handle simple tasks. Most complicated assembly processes still require human labors. If delicate assembly skills are taught through human demonstration, lengthy robot programming can be avoided, meanwhile no robotic expertise is required for the operation. However, a complex assembly task usually involves many parts and tools while being conducted in many steps, which leads to several challenges.

First, it is still a difficult task to recognize small parts and tools using computer vision based methods [6]. Second, it is not reliable to recognize fine assembly actions purely based on human motion features [7]. Third, obtaining the relative position and orientation between parts using 2D vision is challenging, since occlusion frequently occurs between assembled parts [8]. Finally, traditional

* Corresponding author.
E-mail addresses: yegu@szarobots.com (Y. Gu), weihua.sheng@okstate.edu (W. Sheng), chriscrick@cs.okstate.edu (C. Crick).

https://doi.org/10.1016/j.robot.2017.10.002
0921-8890/© 2017 Elsevier B.V. All rights reserved.

approaches use multiple sophisticated sensors to capture the information of both the objects (parts and tools) and human movement, which are costly and not easy to use.

Existing human skill demonstration systems emphasize extracting either human motion information or object information from the demonstration. In [9], object information is extracted from the demonstration to create chains of two-object relationships. In [10], the demonstration is segmented into a sequence of primitives that describe the user actions. In [11], humans are tracked by several calibrated stereo cameras for human motion imitation. However, both the objects and human motion during the demonstration are important clues for assembly task learning. Dillmann [12] built a human skill demonstration platform using multiple cameras for object recognition and data gloves to capture human motion. Multiple delicate sensors are used for collecting data from the demonstration. A rule-based system is established for elementary operators interpretation. In this paper, in contrast to previous works, we adopt a single RGB-D camera to build a Portable Assembly Demonstration (PAD) system. This PAD system can generate skill scripts by capturing the tools being used, the parts being manipulated and the information about human motion during human demonstration of a complex assembly task. The skill scripts are presented in a format that is compatible with an existing planning language. Based on the skill scripts, the robot does the planning accordingly, then executes the task. The PAD system has the following features. First, it is able to recognize small assembly parts and tools used based on both color and depth information. Second, with the parts and tools serving as prior information, the accuracy of assembly action recognition can be improved. Third, the final state of the assembled parts can be reliably estimated, which represents the effect of the assembly action. The state describes the relative pose between the parts in the final assembly. With this capability, for example, it is possible to estimate how deep a bolt is hammered into a hole, or by what angle a screw is rotated, in the demonstration. This is useful information for the robotic assembly process. Furthermore, the skill script can represent general assembly knowledge independent of the robots used in the assembly. The contributions of this paper are as follows: 1. This paper develops a new action recognition algorithm that improves the recognition accuracy by fusing both human motion and object information. 2. This paper develops a new algorithm to estimate the assembly state based on the 3D part models created by our customized 3D scanner, which significantly enhances the robustness to partial occlusions between parts. 3. For assembly state estimation, this paper proposes a novel point cloud registration method which does not need initial alignment. 4. The system teaches a robot to learn multi-stage assembly skills by using a low-cost RGB-D camera.

1.2. Related work

Learning from demonstration has drawn much attention in the robotics community recently [12]. The approaches used to demonstrate skills to robots can be divided into two categories: robot-based demonstration and human-based demonstration. Robot-based demonstration uses robots as the demonstration platform. This method mainly includes teleoperation [13] and kinesthetic teaching [14]. This method is suitable for primitive action learning [15,16].

Compared to robot-based demonstration, human-based demonstration is more convenient for the human operators since they can focus on the task itself and do not need to operate the robot. Two types of sensors are usually used for collecting human demonstration data: wearable sensors and vision sensors. Wearable sensors include inertial measurement units (IMUs) [17] and data gloves [12,18]. In robot learning, it is also called the "sensor on teacher" approach. Wearable sensors are usually working with vision sensors that extract the information of the involved objects. Lee et al. [19] proposed a skill inference and learning framework for a Skilligent robot. Data gloves are used during the demonstration. Based on the framework, the robot learns and infers situation-adequate and goal-oriented skills to handle uncertainties and human perturbations. Kunze et al. [20] performed a manipulation task in an interactive simulation. By performing manipulation tasks in a virtual environment using a data glove, task-related information of the demonstrated actions can be collected from the simulator. By using simulation, they avoid the practical problem of object and pose estimation. Fang et al. [21] proposed a framework that combines a flexible constraint-based motion control approach with a learning algorithm using random forest regression. The task is demonstrated in simulation and implemented on a PR2 robot. On the other hand, vision-based human motion tracking systems are regarded as a natural way to capture demonstration since no sensors need to be attached to the demonstrator. Ahmadzadeh et al. [22] proposed a visuospatial learning approach which adopts a simple algorithm and minimum prior knowledge to learn a sequence of operations from a single demonstration. Later, Ahmadzadeh et al. [23] proposed a complete system which integrates visuospatial skill learning, imitation learning, and conventional planning methods. In their work, only object information is adopted to interpret the demonstration. In contrast, we take both object information and human motion into consideration for action recognition. The task we handle is more challenging which requires high-accuracy object orientation estimation. Takano et al. [24] proposed a novel approach to describe motion–object in terms of natural language. An action language model stochastically combines the motion categories, object categories, and words in the descriptive sentences manually. Then the order of words are learned based on the relations between these elements as the sentence structures. The descriptions are further processed for robot use. Matuszek et al. [25] proposed a method to interact with a robot through voice and gestures. This method is suitable for simple tasks that can be clearly described through verbal communication. Ramirez-Amaro et al. [26] presented a method to extract semantic rules of human activities. Low-level information is extracted first, then high-level knowledge is inferred by reasoning about the human behavior. The robot is able to understand the semantic meaning of the human behavior. However, the demonstrator has to put augmented reality (AR) markers on the head for tracking purpose.

Song et al. [27] developed a platform for learning by demonstration from natural languages and a Kinect sensor, with the initial domain of kitchen activities. In their work, only a single large object is involved, which can be recognized easily. Additionally, there are research works specifically on vision-based assembly skill learning from human demonstration. Hovland et al. [28] proposed an approach to representing an assembly skill by a hybrid dynamic system where a discrete event controller models the skill. The controller is represented as a Hidden Markov Model (HMM) [29]. The problem with this approach is that as the task becomes complex, the HMM will demand a tremendous training dataset. Dantam et al. [30] developed a method to interpret human demonstration to robots. The demonstration is decomposed into a sequence of object connection symbols which can be further transformed into the task language. The assembly task they handled is simple and only the location of the part is required. Takamatsu et al. [9] aimed to recognize assembly tasks through human demonstration. A conventional 6 degree-of-freedom (DOF) object-tracking system recognizes two rigid polyhedral objects from noisy data. Assembly tasks are expressed as sequences of two-object relationships. However, they ignored the problem of occlusion between the parts. Aleotti et al. [10] demonstrated the assembly task through simulation,

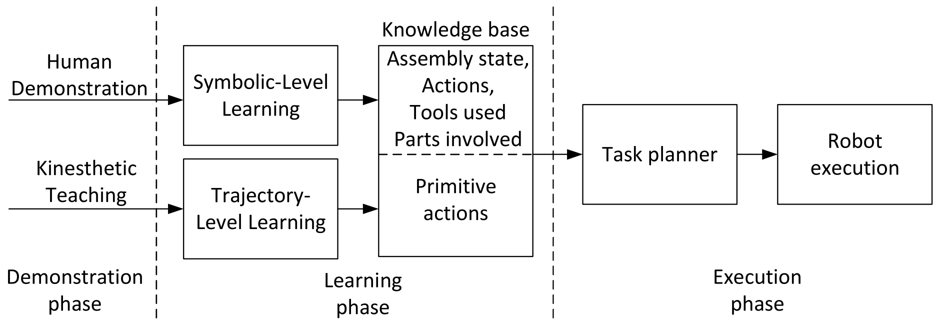


Fig. 1. The overall learning framework.

in which a task planner is designed to analyze the demonstration and segment it into a sequence of primitive actions. However, they did not address the action recognition and object recognition problem. Wang et al. [31] present a graph-based representation of knowledge called assembly graph (AG) to describe the knowledge on parts assembly. The assembly relations are detected from images and represented by AG. It is solved by taking the property of parts as well as the robot into consideration to obtain the precise pose of each part.

In this paper, the goal is to develop a human skill demonstration system with minimum hardware. It can recognize the parts, the tools, the assembly actions and the assembly state in a coherent way. Such a capability is facilitated by the new Portable Assembly Demonstration (PAD) system and the associated theoretical framework. The rest of the paper is organized as follows: Section 2 introduces the PAD system setup and formulates the problem to be solved. Section 3 presents the proposed methodology. Section 4 describes the experimental procedures and gives the experimental results. In Section 5, the skill scripts generated are implemented by a dual-arm robot. In Section 6, conclusions are drawn and future work is proposed.

2. System overview

2.1. PAD system setup

The overall skill learning framework is shown in Fig. 1. The framework consists of three phases: demonstration, learning and execution. In the demonstration phase, the operator first demonstrates primitive actions through Kinesthetic teaching, then the assembly tasks. In the learning phase, the robot first learns the primitive actions through task generalization. Then task demonstrations are transformed into high-level skill scripts using the PAD system. The skill scripts are compatible with planning language PDDL 3.0 [32]. In the execution phase, the robot implements the task based on the solution of the task planner.

The proposed Portable Assembly Demonstration (PAD) system is shown in Fig. 2, which is responsible for demonstration capture and interpretation. It consists of a Kinect sensor on a tripod, a motor-driven turntable, and a computer for data processing (not in the picture). The Kinect sensor captures the object information and the human motion, based on which the actions, parts and tools are recognized. Pre- and Post-conditions as well as action effects are identified. Altogether, these information are converted into a symbolic representation. The PAD system has three modes: (1) object recognition and modeling mode, (2) action recognition mode and (3) assembly state estimation mode. The Kinect faces downward 45 degree at the objects on the turntable while modeling and recognizing them. It looks horizontally at the demonstrator for action recognition. The assembly state which represents the

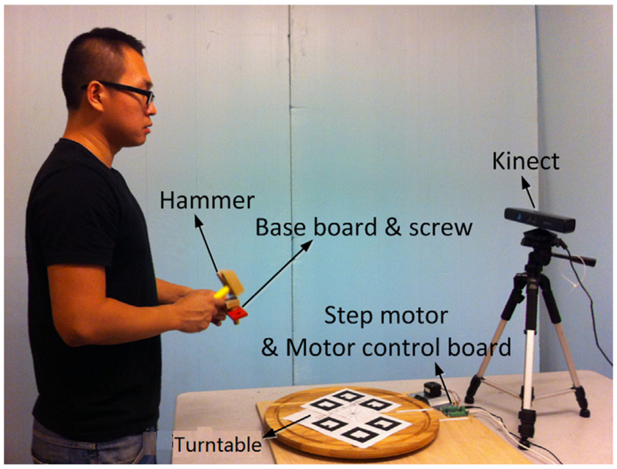


Fig. 2. The hardware setup of the PAD system.

spatial relationship between assembly parts is estimated based on 3D models. The turntable and the Kinect also work together as a 3D modeling subsystem, or 3D scanner. The RoboEarth package [33] is used to create the 3D models. Meanwhile, a point cloud segmentation algorithm is developed to extract the object from the scene. For 3D modeling, we attach the marker template to the turntable and use a step motor to control its rotation. As the turntable rotates, the 3D templates of the object will be extracted. After the turntable rotates a full circle at a proper speed, a 3D model is created.

All the programs run in ROS (Robot Operating System) [34] in Linux. There are seven ROS nodes in total: a step motor control node, an object modeling node, an object template extraction node, an object recognition node, a human skeleton tracking node, an action recognition node and an assembly state estimation node. The object modeling node and the object template extraction node start to work when the motor starts to rotate. They stop once the motor stops. The object recognition node uses the prebuilt templates to recognize parts and tools. The action recognition node integrates the outputs of the human skeleton tracking node and the outputs of the object recognition node. The assembly state estimation node uses two kinds of 3D models: the individual 3D models of the involved parts and the 3D model of the assembled part created after the assembly action.

2.2. Formulation of the problem

Generally, a complex assembly task which involves n parts can be represented as a sequence of subtasks as shown in Eqs. (1) and

(2):

$$\zeta = \underbrace{\underbrace{\underbrace{p_1}_{M_1{}^1} \oplus \underbrace{p_2}_{M_2{}^1}}_{M_1{}^2} \oplus \underbrace{p_3}_{M_2{}^2} \oplus \cdots}_{M_1{}^{n-1}} \oplus \underbrace{p_n}_{M_2^{n-1}} \tag{1}$$

$$\zeta = \underbrace{\underbrace{\underbrace{p_1}_{M_1{}^1} \oplus \underbrace{p_2}_{M_2{}^1}}_{M_1{}^3} \oplus \underbrace{\underbrace{p_3}_{M_1{}^2} \oplus \underbrace{p_4}_{M_2{}^2}}_{M_2{}^3} \oplus \cdots}_{M_1{}^{n-1}} \oplus \underbrace{p_n}_{M_2^{n-1}} \tag{2}$$

where ζ is the final product, p_n is the nth part, $\oplus$ is the assembly action on the two parts. The complex assembly task can be treated as a sequence of two-part assembly subtasks. As shown in Eq. (1), in a two-part assembly task, the models for each part are defined as $M_1{}^1$ and $M_2{}^1$. First, take p_1's model as $M_1{}^1$, p_2's model as $M_2{}^1$, The superscript denotes the index of the subtask. Then, $M_1{}^2$ and $M_2{}^2$ will be used to denote $p_1 \oplus p_2$ and p_3 respectively. This process continues until the last part is assembled. On the other hand, the implementation of a complex task can also follow the sequence show in Eq. (2) using the same idea.

For the case when there are multiple actions applied to the parts p_1 and p_2, we have

$$\begin{aligned} M_1{}^{m+1} &= p_1 \oplus_1 \oplus_2 \cdots \oplus_m p_2 \\ &= \underbrace{p_1}_{M_1{}^1} \oplus_1 \underbrace{p_2}_{M_2{}^1} => \cdots => \underbrace{\underbrace{p_1}_{M_1{}^m} \oplus_m \underbrace{p_2}_{M_2{}^m}}_{M_1{}^{m+1}} \end{aligned} \tag{3}$$

$M_1{}^{m+1}$ is the model of $p_1 \oplus_m p_2$. $\oplus_m$ is the mth action on part p_1 and p_2. It will be used as the M_1 for the $(m+1)$th subtask. => denotes the action sequence flow.

Given the above setup, the problems to be solved are action and part recognition and assembly state estimation. The input to the action recognition system consists of two elements: the observation of the objects and the observation of the human action. The observation of the objects has two parts, ψ_p is the decision of the part recognition which is one of the parts in the assembly sets. ψ_t is the decision of the tool recognition system which is one of the tools in the tool set. If the action does not involve any tool then $\psi_t = null$. The observation of the action ψ_a is a sequence of skeleton data of the human subject $\{\Gamma_1, \Gamma_2 \dots \Gamma_t\}$, where $\Gamma_t = \{\Theta_1, \Theta_2 \dots \Theta_k\}$, Θ_i is the angle of joint i and k is the total number of joints. We assume the part set $P = \{P_1, P_2 \dots P_m\}$, the tool set $T = \{T_1, T_2 \dots T_n\}$, and the action set $A = \{A_1, A_2 \dots A_q\}$. The action recognition is to develop an algorithm to decide on $A_i \in A$ given the above observation. In other words, the goal of the action recognition is to find a mapping function f_a,

$$A_i = f_a(\psi_a, \psi_p, \psi_t) \tag{4}$$

The assembly state describes the spatial relationship between individual parts in the final assembly. To estimate the assembly state, the input includes the 3D models of each individual part and the 3D model of the assembled part after each action. The goal of assembly state estimation is to find a mapping function f_s,

$$S = f_s(M_1, M_2, M_1 \oplus M_2) \tag{5}$$

Here, S is the assembly state. M_1 and M_2 are the models of the two parts involved. $M_1 \oplus M_2$ is the 3D model of the assembled part which is a joint of part M_1 and M_2. To estimate the assembly state, M_2 is treated as the reference part. Then the state S is the pose of the non-reference part with respect to the reference part. Based on the solutions to the two problems mentioned, we can generate a skill script Σ as follows

$$\Sigma = (\Sigma_1, \Sigma_2, \Sigma_3 \dots \Sigma_n) \tag{6}$$

where each assembly skill Σ_i is a 5-tuple symbolic description defined as

$$\Sigma_i = \langle A_i, T_i, P_{1i}, P_{2i}, S_i \rangle \tag{7}$$

where P_{1i} and P_{2i} are the two parts involved in the ith step.

3. Methodology

The robot learns the primitive actions through Kinesthetic teaching. The learned actions are kept in the knowledge base. On the other hand, after the assembly task demonstration, the sequence of actions, involved parts and tools, as well as assembly state are learned. The skill scripts are then generated which are described in a PDDL compatible way. The following sections explain each component of the system.

3.1. Primitive action learning

The primitive actions are learned through Kinesthetic teaching. Fig. 3 shows the teaching of hammering action. The trajectory of the robot's left gripper with respect to the reference part is recorded. The reference part is tracked using an Kinect sensor [35]. Therefore, the relationship between the Kinect frame and the robot frame has to be calibrated. The trajectories extracted from multiple demonstrations are generalized. The generalized trajectory will be reproduced according to the pose of the reference part. For each action, the robot arm is guided to the desired pose multiple times from different initial poses. To extract the task and joint space constraints from multiple demonstrations, the Gaussian Mixture Model (GMM) and Gaussian Mixture Regression (GMR) methods [14] are adopted. The data is first normalized before applying GMM/GMR. The normalization ensures that each demonstration has the same data length. The normalized dataset is first modeled by a Gaussian Mixture Model (GMM) which has three components. These parameters are experimentally decided using a trial-and-error method. Based on the GMM, a generalized version of the trajectories is computed by applying Gaussian Mixture Regression (GMR). When a new pose of the part is obtained by the Kinect sensor, it has to be transformed into the pose with respect to the robot base frame. Then the reproduced pose trajectory can be derived. Ultimately it is converted from the task space to the joint space using the given Inverse Kinematic function.

3.2. Object recognition

To recognize the objects, the tools and parts are put on the turntable one by one before demonstration for recognition. Therefore, the system knows what parts and tools (if any) will be involved before the demonstration. The 3D templates for each object are created beforehand. For each object, we capture multiple templates (2 for symmetrical objects and 4 for asymmetrical objects) from different camera views. To extract the point cloud of the object on the turntable, the distant points will be filtered out first. Then the top surface of the turntable will be extracted and all the points above that surface are considered as the points on the object.

Object recognition in the PAD system is based on the recognition module in Point Cloud Library (PCL) [36]. The recognition process is as follows. First, from the point cloud of the whole

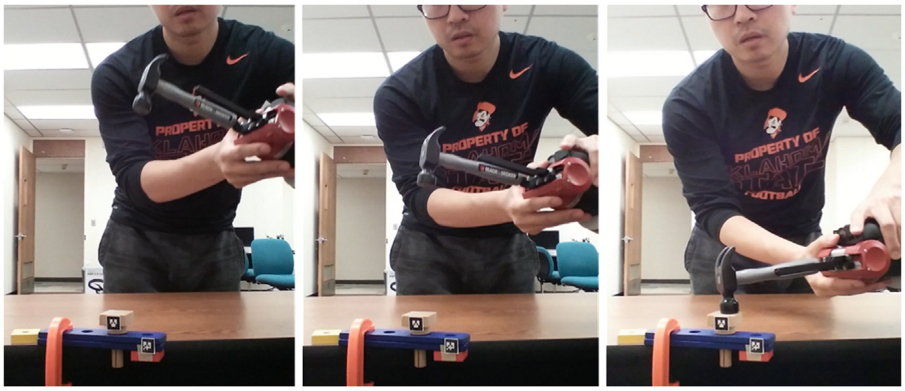


Fig. 3. Kinesthetic teaching of the hammering action.

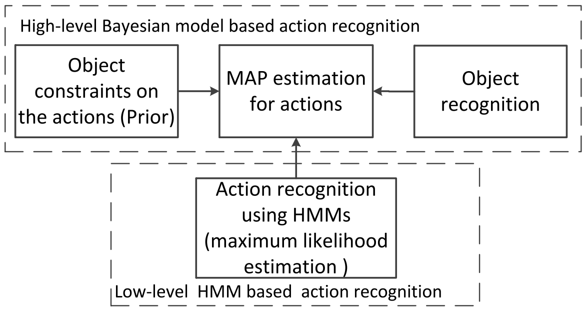

High-level Bayesian model based action recognition
Object constraints on the actions (Prior)
MAP estimation for actions
Object recognition
Action recognition using HMMs (maximum likelihood estimation)
Low-level HMM based action recognition

Fig. 4. The action recognition framework.

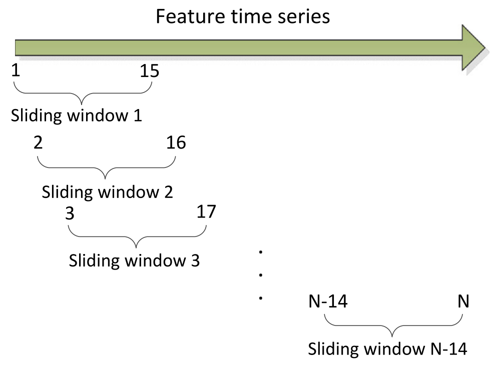


Fig. 5. Data segmentation in recognition phase.

scene, only points above the turntable are extracted, which greatly decreases the number of points to be processed. This process can also help improve the recognition accuracy since the outliers can be removed effectively. Second, the normals of each point in both the model point cloud and the scene point cloud are computed using the 10 nearest neighbors. Third, each point cloud is down-sampled uniformly to find a small number of keypoints, which are associated with a 3D descriptor in order to perform keypoint matching and determine point-to-point correspondences. According to [37], the SHOTCOLOR [38] descriptor shows competitive performance among the 3D descriptors which is a color version of SHOT (Unique Signatures of Histograms) [39]. Therefore, we choose the SHOTCOLOR descriptor. The measure for correspondence matching is squared descriptor distance. The density of the keypoints is adjusted according to the size of the object. To extract enough features, higher sampling densities are required for small objects. Fourth, once correspondences between the model and the scene are obtained, each correspondence can cast a vote for the position of the reference point in the scene. Evidence for the presence of a particular object can then be evaluated by thresholding the peaks of the Hough space. Seamlessly, multiple peaks in the Hough space indicate the presence of multiple instances of the object [40].

3.3. *Action recognition*

We aim to develop an accurate action recognition algorithm by considering the object/action correlation. A two-level probabilistic approach is proposed. At the low level, HMMs model the dynamics of the actions. At the high level, a Bayesian model captures action–object dependencies. The action recognition framework is shown in Fig. 4.

3.3.1. *HMMs for low-level action recognition*

We propose an HMM based approach to performing real-time action spotting and classification from continuous user motion. Each action is characterized by an HMM. Each model is trained with 20 sets of training data at a sampling rate of 20 Hz. The training data is a sequence of right arm joint angles which can be accessed through the Openni [41] driver and the skeleton tracker function as shown in Fig. 6. Four joint angles from two right arm joints are considered: the roll and yaw angles of the right elbow, the roll and pitch angles of the right shoulder. The number of state in each HMM is 10. The number of observation symbol is 8. These values are determined through experiments.

In the training phase, there are four steps:

- Step 1: Segment the training set.
- Step 2: Quantify the vectors into observation symbols using the K-means clustering.
- Step 3: Set up the initial HMM parameters.
- Step 4: Parameter estimation using the EM method [42].

In the recognition phase, as shown in Fig. 5, the testing data is segmented using a fixed sliding window of size 15. The step size for the sliding window is 1. The probability of having the observation sequence given the model $P(\psi_a|\lambda_j)$ is computed for each sliding window. ψ_a is a sequence of feature vector. λ_j is the HMM model for action A_j. Solving this problem allows us to choose the model which best matches the observations. More details of the low-level HMM can be found in [43].

3.3.2. *Bayesian network for modeling object–action dependencies*

The objects include tools and parts. They both have correlation with the action. The correlation is modeled in a Bayesian model

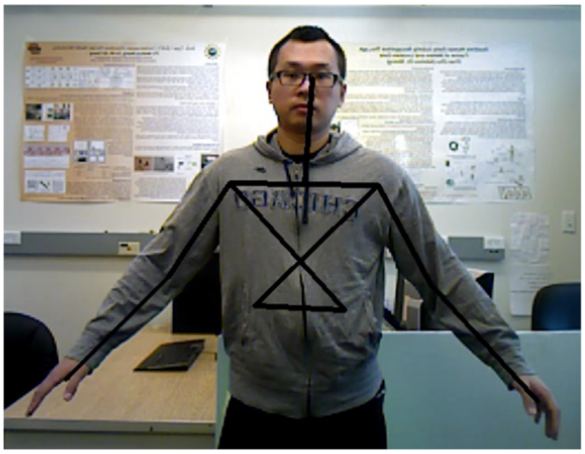


Fig. 6. Human skeleton tracking by the RGB-D sensor.

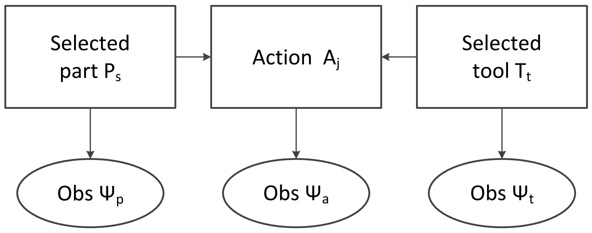


Fig. 7. The Bayesian model for action recognition.

shown in Fig. 7. The manipulative action is denoted by A_j. ψ_p is the decision of part recognition. ψ_t is the decision of tool recognition where $\psi_p \in \Psi_p$, $\psi_t \in \Psi_t$ and $\psi_a \in \Psi_a$. At this level, the goal is to find the maximum posterior likelihood (MAP) estimation,

$$A_j^* = arg\ \max_{A_j} P(A_j|\psi_a, \psi_p, \psi_t) \tag{8}$$

According to the Bayesian rule,

$$P(A_j|\psi_a, \psi_p, \psi_t) \propto P(\psi_a|A_j, \psi_p, \psi_t) \cdot P(A_j|\psi_p, \psi_t) \tag{9}$$

As shown in the Bayesian model, we made the assumption that ψ_a is independent of ψ_p and ψ_t given A_j. Therefore, $P(\psi_a|A_j, \psi_p, \psi_t) = P(\psi_a|A_j)$. $P(\psi_a|A_j)$ is interpreted as $P(\psi_a|\lambda_j)$ which is the recognition output of the low-level HMMs. Applying the total probability theorem, we have

$$P(A_j|\psi_p, \psi_t) = \sum_m \sum_n P(A_j|P_m, T_n, \psi_p, \psi_t) \cdot P(P_m, T_n|\psi_p, \psi_t) \tag{10}$$

where P_m is part m and T_n is tool n. A_j is independent of ψ_p, ψ_t given P_m, T_n. Therefore,

$$P(A_j|P_m, T_n, \psi_p, \psi_t) = P(A_j|P_m, T_n) \tag{11}$$

$P(A_j|P_m, T_n)$ is the prior probability characterizing the action that occurs on part P_m using tool T_n, which can be calculated based on the occurrence of A_j given P_m and T_n in the training set. On the other hand, we have

$$P(P_m, T_n|\psi_p, \psi_t) = P(P_m|\psi_p) \cdot P(T_n|\psi_t) \tag{12}$$

$P(P_m|\psi_p)$ is the part classification accuracy and $P(T_n|\psi_t)$ is the tool classification accuracy.

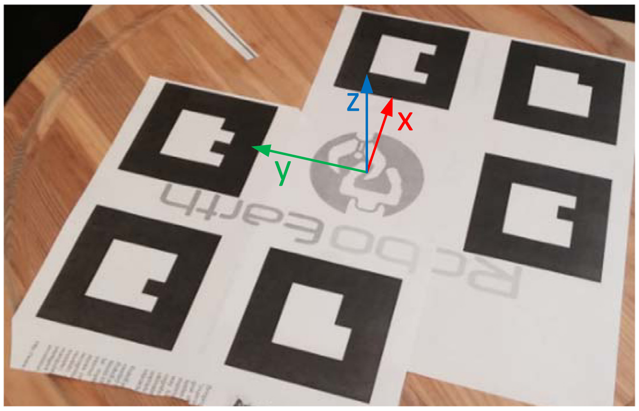


Fig. 8. The coordinate system defined by the 6 markers [44].

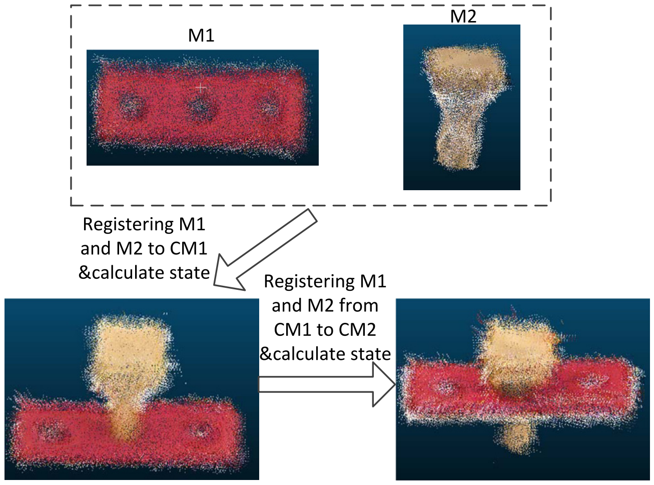

M1
M2
Registering M1 and M2 to CM1 &calculate state
Registering M1 and M2 from CM1 to CM2 &calculate state
CM1
CM2

Fig. 9. State estimation for a two-step assembly task. M_1 and M_2 are the 3D models of the base board and the bolt respectively. CM_1 is the 3D model of the assembled part after the first action. CM_2 is the 3D model of the assembled part after the second action.

If two parts are involved in the task, the Eq. (12) can be derived as follows:

$$P(P_{m1}, P_{m2}|\psi_{p1}, \psi_{p2}) = P(P_{m1}|\psi_{p1}) \cdot P(P_{m2}|\psi_{p2}) \tag{13}$$

$P(P_{m1}|\psi_{p1})$ and $P(P_{m2}|\psi_{p2})$ are the part P1 and P2 classification accuracy respectively.

3.4. *Assembly state estimation*

To estimate the assembly state, we first define a common coordinate system based on 6 markers, as shown in Fig. 8. One of the two parts is treated as the reference part. Therefore, the assembly state is the pose of the non-reference part with respect to the reference part.

Fig. 9 shows a two-step assembly task that inserts and hammers a square bolt into a hole through the baseboard. With the 3D scanner, first we create the 3D models of these two parts. Since they are in the same coordinate frame, the initial state S_0 can be represented by a 4×4 identity matrix. The 3D model of the assembled part is created after an action is applied to these two parts. The 3D model of each part is registered to the 3D model of the assembled part.

We adopt a two-step point cloud registration approach. In the first step, Fast Point Feature Histograms (FPFH) [45] are extracted from both object and scene models. It is an informative pose-invariant local feature which represents the underlying surface model properties at a point. Based on the correspondences between features, Sample Consensus Initial Alignment (SAC-IA) [46] is used to find the initial alignment. In the second step, the ICP (Iterative Closest Point) [47] algorithm is adopted to calculate the transformation based on the initial alignment. With this approach, the position and orientation of the individual models can be arbitrary with respect to the model of the assembled part. The transformation matrix obtained by our approach is shown in Eq. (14).

$$T = \begin{pmatrix} R_{11} & R_{12} & R_{13} & D_x \\ R_{21} & R_{22} & R_{23} & D_y \\ R_{31} & R_{32} & R_{33} & D_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{14}$$

The translation along each axis is given by D_x, D_y, D_z. To calculate the Euler angles about each axis, let ψ, θ, ϕ be the Euler angles about the X, Y, Z axis, respectively, then we have the following equations [48].

$$\begin{cases} \psi = atan2(R_{32}/R_{33}) \\ \theta = -sin^{-1}R_{31} \\ \phi = atan2(R_{21}/R_{11}) \end{cases} \tag{15}$$

Here we define two transformation matrices. T_{ri} represents the pose change of the reference part after action i; while T_{pi} represents the pose change of the non-reference part after action i. The assembly state S_i after action i can be calculated as

$$S_i = (T_{ri})^{-1} \cdot S_{i-1} \cdot T_{pi} \tag{16}$$

Here, S_{i-1} is the precondition, S_i is the postcondition.

3.5. Skill scripts implementation

The skill script is a 5-tuple symbolic description, $\Sigma = \langle A, T, P_1, P_2, S\rangle$, where A is the primitive action to be implemented by the robot, T is the tool used, P_1, P_2 are the two parts involved. At the beginning of the demonstration, the parts and tools are placed onto the turntable for recognition. The recognized action tells the robot which primitive action should be applied. Before the action is applied, it checks if the precondition is met based on the assembly state. Once met, the robot reproduces the action according to the observation. The action is repeated until the postcondition is satisfied. Meanwhile, in order to use the primitive actions for general purpose task planning, the actions have to be represented in a symbolic, scenario-independent way. The preconditions and effects for each action in the knowledge base are derived from the assembly state. Fig. 10 shows a snippet of the planning formalization. : *objects* is used to specify all the objects in the PDDL problem [32]. They are listed according to the type. : *init* and : *goal* specify the initial and final state described using positive ground literals. The goal condition is formalized as a logical conjunction.

4. Experiment and results

Experiments are conducted to verify the proposed method. An assembly scenario is used to validate our theoretical framework. Fig. 11 shows the parts and tools used in our experiment.

```
(define (problem p1) (: domain assembly)
  (:objects
   l-gripper  -- gripper
   screwdriver  wrench  hammer  -- tool
   baseboard  nut  screw  bolt  -- part
   workstation  -- station
  )
  (:init
    (gripper-can-reach  l-gripper  baseboard
     gripper-able  insert
     gripper-able  screw
     gripper-able  hammer
     gripper-able  wrench
     part-at  screw  workstation
     ...
     ...)
   )
   (:goal
      (and  (part-state  screw  screwed-in)
            (part-state  bolt  hammered-in)
            (part-state  bolt  wrenched-to-certain-angle)
      )
   )
   (:constraints
   (and (before(part-state  bolt  wrenched-to-certain-angle)
         (part-state  bolt  hammered-in) )
   (and (before(part-state  screw  screwed-in)
         (part-state  screw  inserted) )
   )
)
```

Fig. 10. The snippet of the planning formalization.

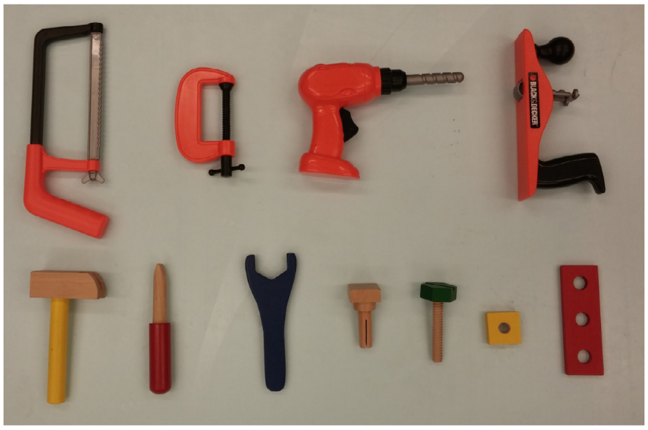


Fig. 11. The parts and tools. From left to right, the top row: saw, G clamp, drill, and wood planner. The bottom row: hammer, screw driver, wrench, bolt, screw, nut and red board.

4.1. Object recognition

4.1.1. Template extraction

The point cloud of the parts on top of the turntable is extracted. They are used for part recognition. The point cloud of the base board is shown in Fig. 12. It shows that our approach can effectively extract the points that belong to the part from the scene point cloud. It is worth mentioning that the plane function of the turntable corresponds to the camera pose. If the camera pose is

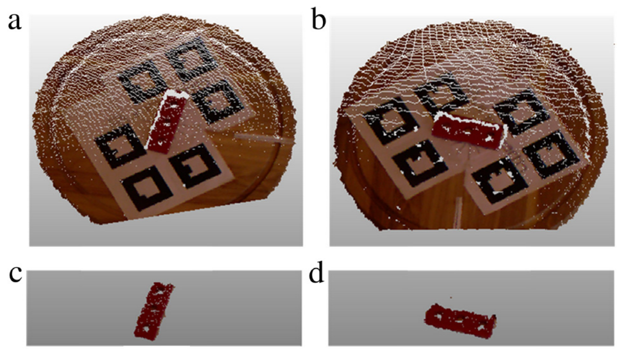


Fig. 12. Extraction of the based board's points from the scene point cloud. (a) and (b) show the scene point cloud with two different rotation angles of the turntable. (c) and (d) show the extracted points of the base board.

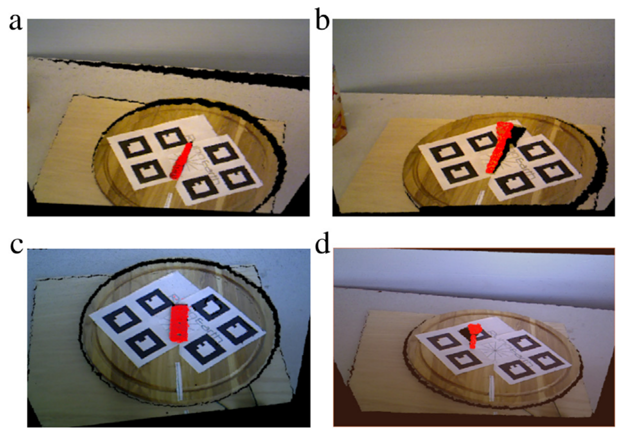


Fig. 13. Results of tool and part recognition, where the recognized objects are highlighted in red. (a) the screw driver. (b) the hammer. (c) the base board. (d) the screw. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

changed, the plane function should be recalculated. To reliably recognize objects, two templates are needed for symmetrical objects and four for asymmetrical objects.

4.1.2. Object recognition

The top row in Fig. 13 shows the recognition results of the screw driver and the hammer respectively. The bottom row shows the recognition of the base board and the screw. The recognized objects are highlighted in red. In order to recognize small objects like the screw, the downsampling radius for the scene has to be small so that enough keypoints can be extracted. In addition, we evaluate the robustness of our approach under occlusion and in cluttered scenes. Figs. 14(a) and (b) show that with minor occlusions, the board can still be recognized. Figs. 14(c) and (d) show that our algorithm works correctly in cluttered scenes. The two red blocks in the cluttered scene which are similar to the base board are not misclassified as the base board. To evaluate the recognition accuracy, for each object type, 100 trials are conducted with different object poses and camera viewpoints. The recognition accuracy is shown in Table 1.

4.2. Action recognition

For action recognition, segmentation is implemented implicitly by the HMMs. If the outputs of all the HMMs are very small, it indicates that there is no action or the action has already finished. Two

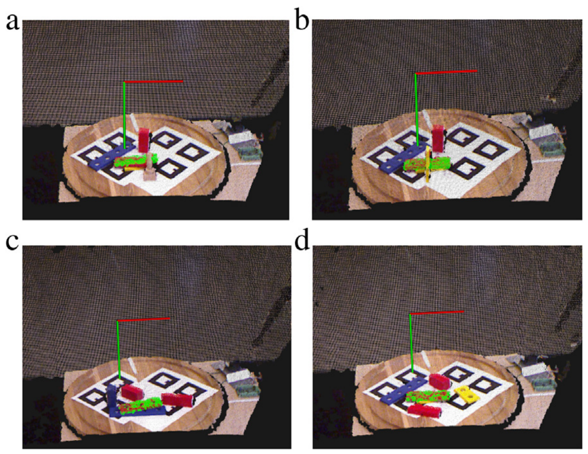


Fig. 14. Results of board recognition, where the recognized objects are highlighted in green. (a) the board with occlusion from a bolt. (b) the board with occlusion from another board. (c) the board in cluttered scene 1. (d) the board in cluttered scene 2. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
The object recognition accuracy.

Object	saw	G clamp	drill	wood planner
Accuracy	0.92	0.93	0.95	0.92
Object	hammer	screw driver	wrench	bolt
Accuracy	0.97	0.87	0.88	0.85
Object	screw	nut	red board	
Accuracy	0.86	0.87	0.89	

kinds of manipulative actions are tested. For the actions that involve tools, both tool/action and part/action dependencies are considered. The priority of the tool/action dependencies is higher than that of the part/action dependencies, since the tool used usually offers more clue about the action than the part does. While for the actions that do not involve tools, only part/action dependencies are considered. As the human hand (with or without tool in the hand) approaches the part, the Bayesian network takes effect. To evaluate the action recognition system, the training dataset is collected from one subject while the HMMs and Bayesian model are tested on four other subjects. The trainer conducts a sequence of assembly tasks. Two kinds of training data are collected: the training data for each HMM and the object/action sequence which is used to calculate the object/action dependencies used in the Bayesian network. After that, four other subjects conduct a sequence of assembly tasks to verify the trained models. Each assembly action is conducted 50 times by each subject for each cross validation. The accuracy of the HMMs and the Bayesian model are shown in Tables 2 and 3 respectively. The accuracy of the Bayesian model is shown in Table 3; Compared with HMMs, the Bayesian model can differentiate actions more effectively.

4.3. Assembly state estimation

In this section, the point cloud registration method used is evaluated first. Then the assembly state estimation results are given.

4.3.1. Evaluation of point cloud registration

The computation cost of the point cloud registration is proportional to the size of the point cloud. However, reducing the size of point cloud may impact the accuracy of registration. Therefore, we design an experiment to evaluate how the size of the point cloud affects the accuracy of a pure rotation assembly. The accuracy of

Table 2
Recognition accuracy of the HMMs.

test type	decision type										accuracy
	hammering	screwing	wrenching	sawing	drilling	rasping	fixing	inserting	aligning	missed	
hammering	**0.90**	–	–	0.02	–	0.04	–	–	–	0.04	**0.90**
screwing	–	**0.75**	–	–	–	–	0.22	–	–	0.03	**0.75**
wrenching	–	–	**0.93**	0.02	0.01	–	0.02	–	–	0.02	**0.93**
sawing	–	–	–	**0.66**	–	0.32	–	–	–	0.02	**0.66**
drilling	0.02	–	0.02	–	**0.89**	–	–	–	–	0.07	**0.89**
rasping	–	–	–	0.34	–	**0.62**	–	–	–	0.04	**0.62**
fixing	–	0.25	–	–	–	–	**0.72**	–	–	0.03	**0.72**
inserting	–	–	–	–	–	0.02	–	**0.86**	0.10	0.02	**0.86**
aligning	–	–	–	–	–	0.01	–	0.06	**0.87**	0.06	**0.87**

Table 3
Recognition accuracy of the Bayesian model.

test type	decision type										accuracy
	hammering	screwing	wrenching	sawing	drilling	rasping	fixing	inserting	aligning	missed	
hammering	**0.94**	–	–	0.02	–	0.02	–	–	–	0.02	**0.94**
screwing	–	**0.95**	–	–	–	–	0.02	–	–	0.03	**0.95**
wrenching	–	–	**0.98**	–	–	–	–	–	–	0.02	**0.98**
sawing	–	–	–	**0.96**	–	0.02	–	–	–	0.02	**0.96**
drilling	–	–	–	–	**0.95**	–	–	–	–	0.05	**0.95**
rasping	–	–	–	0.04	–	**0.92**	–	–	–	0.04	**0.94**
fixing	–	0.02	–	–	–	–	**0.92**	–	–	0.06	**0.92**
inserting	–	–	–	–	–	–	–	**0.91**	0.07	0.02	**0.91**
aligning	–	–	–	–	–	–	–	0.05	**0.90**	0.05	**0.90**

Table 4
The procedure of part assembly.

Step	p1 (Reference)	p2	Action⊕	p1⊕p2
1	red board	bolt	inserting	CP1
2	red board	bolt	hammering	CP2
3	red board	bolt	wrenching	CP3
4	CP3	nut	aligning	CP4
5	CP4	screw	inserting	CP5
6	CP4	screw	screwing	CP6

the calculated angle ϕ is evaluated against the portion of the points kept. Fig. 15 indicates that as long as more than half of the points are kept, the accuracy will be consistently high.

Another experiment is implemented to test the robustness of our registration approach. The model of a bolt is registered to its flipped model. Figs. 16(a) and (b) are the input point cloud and the target point cloud respectively. Fig. 16(c) is the registration result of the ICP algorithm without initial alignment. The registration failed since without initial alignment, the ICP algorithm falls into a local minimum. Fig. 16(d) is the result of the Sample Consensus Initial Alignment algorithm. It roughly aligns the input cloud to the target cloud. However some misalignment still exists, since the correspondences found are not always correct. The last figure is the result of the proposed approach. The ICP is applied after initial alignment which can overcome the limitation of each individual algorithm and the registration results are better.

4.3.2. *Assembly state estimation*

An assembly scenario is designed to evaluate the assembly state estimation. The experimental procedure is shown in Table 4, while Fig. 17 shows the six steps of the parts assembly and Fig. 18 shows the parts and the corresponding models after each step. CP means the combined part after an primitive action. First, the bolt is inserted into one hole of the base board. Second, the bolt is hammered into the hole. Third, the bolt is wrenched by 30° with respect to the base board. Fourth, a nut is aligned to the other hole. Fifth, a screw is inserted into this hole. Finally, a screwdriver is used to screw the base board and the nut.

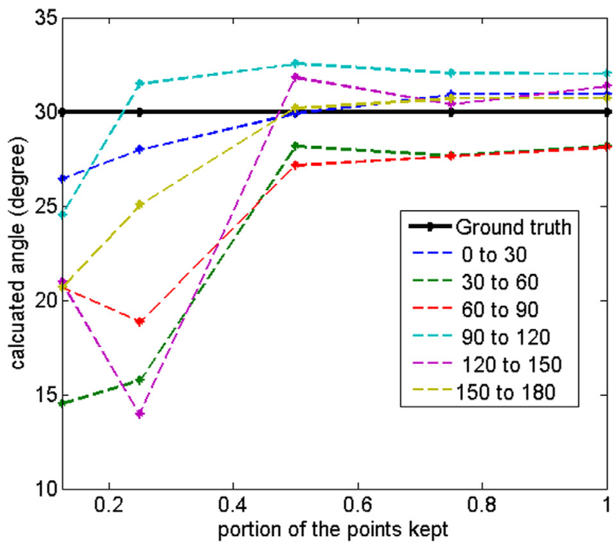


Fig. 15. Accuracy of the rotation estimation with respect to the portion of point cloud kept. Different colors denote registration from the models with different initial angles. The angular difference between the input model and the target model of each registration is 30 degree. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The initial state S_0 is

$$\begin{aligned} S_0 &= \{D_x, D_y, D_z, \psi, \theta, \phi\} \\ &= \{0.0\text{ mm}, 0.0\text{ mm}, 0.0\text{ mm}, 0.00°, 0.00°, 0.00°\} \end{aligned} \quad (17)$$

Fig. 19-(a) gives the results of registering ${M_1}^1$ and ${M_2}^1$ to CM_1 after the "inserting" action. The ground truth of state S_1 which represents the pose of M_2 with respect to M_1 is $\{43.0\text{ mm}, 0.0\text{ mm}, 0.0\text{ mm}, 0.00°, 0.00°, 0.00°\}$, the estimation is:

$$\hat{S}_1 = \{43.2\text{ mm}, 1.7\text{ mm}, -2.8\text{ mm}, 0.50°, 0.12°, 1.01°\} \quad (18)$$

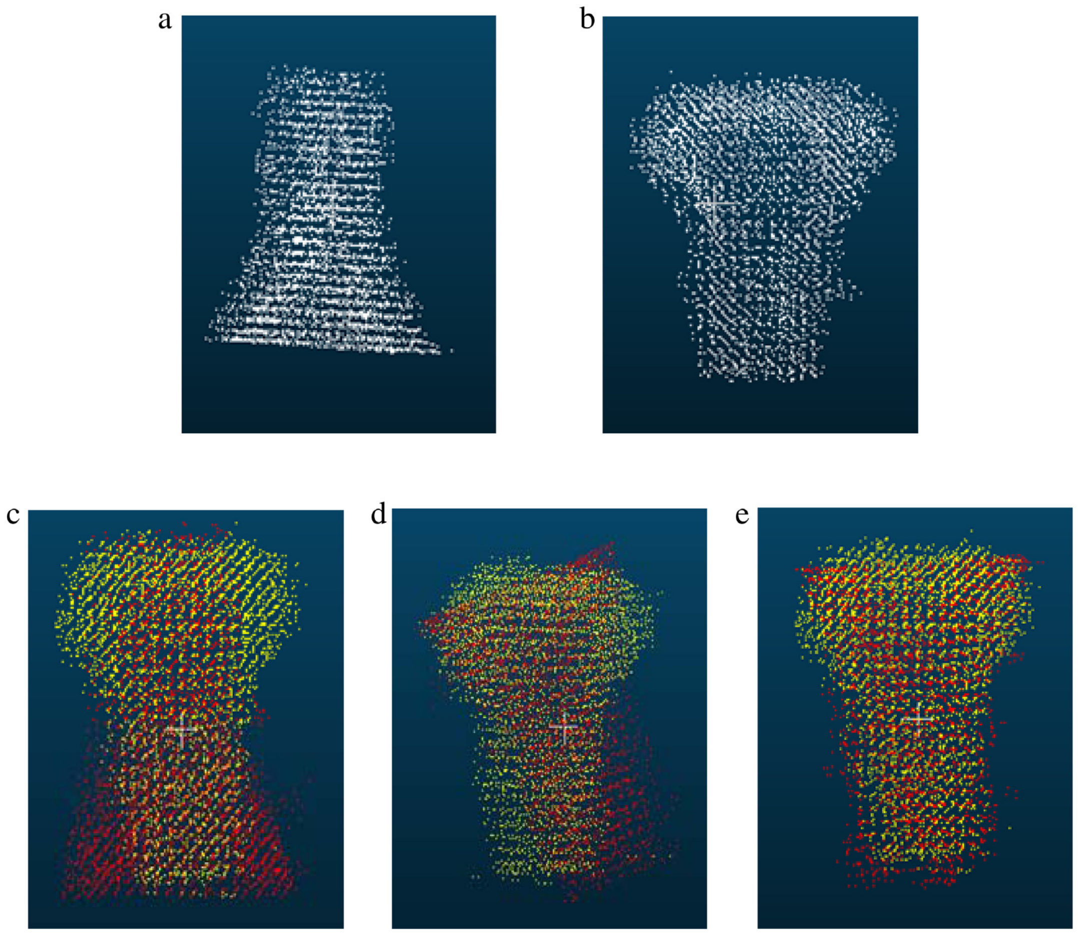


Fig. 16. Registration results. (a) the input 3D model. (b) the target model. (c) registration result of ICP. (d) registration result of SAC-IA. (e) registration result of our approach. ((c) - (e): Red ones indicate the registration results. Yellow ones indicate the target model). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

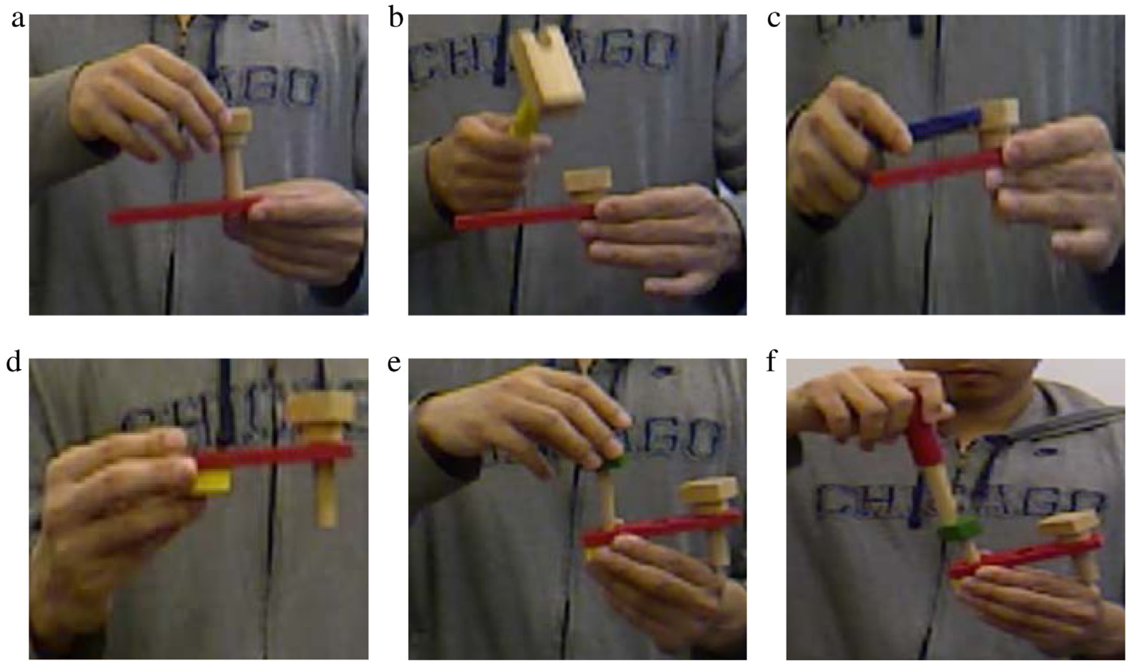


Fig. 17. The actions used in the experiment. (a) inserting the bolt. (b) hammering the bolt. (c) wrenching the bolt. (d) aligning the nut to the hole on the base board. (e) inserting the screw. (f) screwing the screw.

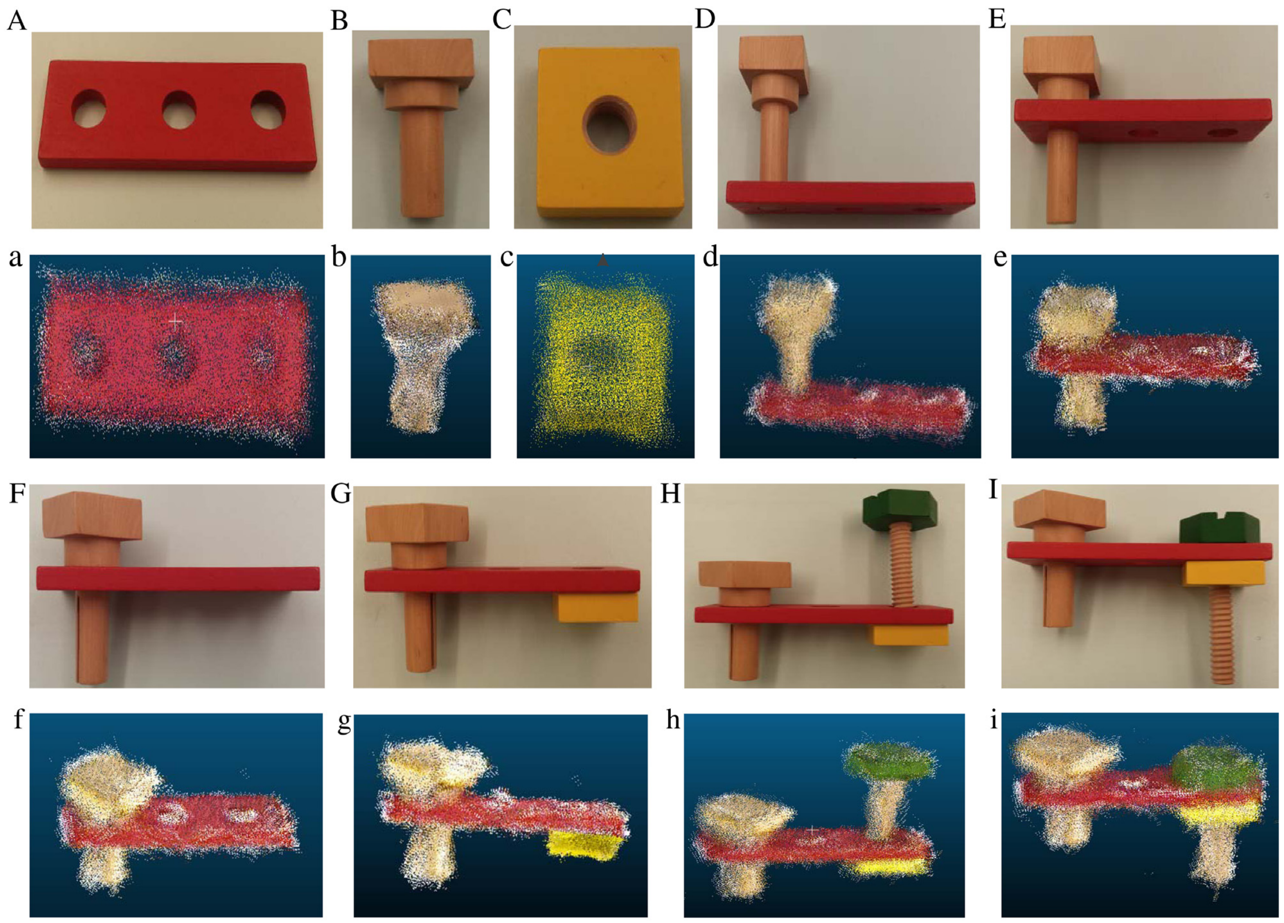


Fig. 18. The parts and the associated model. (A) red board. (B) bolt (C) nut (D) parts pose (CP1) after action 1. (E) parts pose (CP2) after action 2. (F) parts pose (CP3) after action 3. (G) parts pose (CP4) after action 4. (H) parts pose (CP5) after action 5. (I) parts pose (CP6) after action 6. (a) - (i) are the models of (A)-(I).

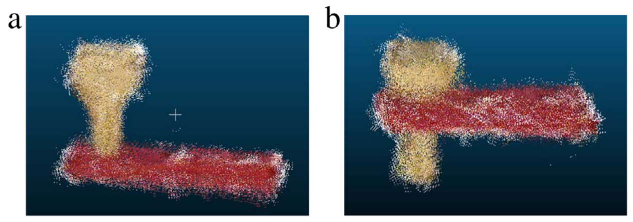


Fig. 19. Registration results. (a) register ${M_1}^1$ and ${M_2}^1$ to CM_1 after action 1 "inserting". (b) register ${M_1}^2$ and ${M_2}^2$ to CM_2 after action 2 "hammering".

Table 5
The registration performance.

Step	1	2	3
RMS error (mm)	1.280	1.662	1.862
Step	4	5	6
RMS error (mm)	1.7014	1.9014	2.4624

Fig. 19-(b) shows the registration of ${M_1}^2$ and ${M_2}^2$ to CM_2. The major movement of ${M_1}^2$ is translation along the vertical axis (Z axis). The ground truth of S_2 is {43.0 mm, 0.0 mm, −34.0 mm, 0.00°, 0.00°, 0.00°}. The estimated S_2 is

$$\hat{S}_2 = \{44.1\text{ mm}, 3.5\text{ mm}, -35.9\text{ mm}, 0.95°, -0.74°, 1.41°\} \quad (19)$$

After wrenching, the bolt is rotated anticlockwise by 30° around the base board. The ground truth of S_3 is {43.0 mm, 0.0 mm, −34.0 mm, 0.00°, 0.00°, 30.00°}. The estimated S_3 is

$$\hat{S}_3 = \{43.5\text{ mm}, -4.1\text{ mm}, -33.2\text{ mm}, 2.21°, 1.36°, 31.47°\} \quad (20)$$

As mentioned before, the estimated angle is not unique and only the one that reflects the actual rotation is selected. Similarly, this process goes on until the final part is assembled. The estimated assembly state $\hat{S}_4, \hat{S}_5, \hat{S}_6$ are as follows,

$$\hat{S}_4 = \{-42.7\text{ mm}, 2.2\text{ mm}, -2.8\text{ mm}, 3.65°, 2.74°, 4.21°\} \quad (21)$$

$$\hat{S}_5 = \{-46.2\text{ mm}, -1.0\text{ mm}, 0.9\text{ mm}, 4.39°, 4.26°, 5.12°\} \quad (22)$$

$$\hat{S}_6 = \{-45.2\text{ mm}, 5.2\text{ mm}, -51.2\text{ mm}, 3.88°, 3.54°, 7.26°\} \quad (23)$$

The registration result of the last step is shown in Fig. 20. As the assembled part becomes more complicated, the registration result gets less accurate. This is because the chance of occlusion increases. The registration performance of each step is given in Table 5 in terms of root mean square (RMS in mm). It is worth noting that for some symmetrical parts, the rotation angle cannot be uniquely determined, and other sources of information may be required.

The assembly process is repeated 10 times. The average error for translation and rotation is within 10 mm and 10° respectively. The assembly state is used for precondition and action effects definition. Finally, after the human demonstration, the following

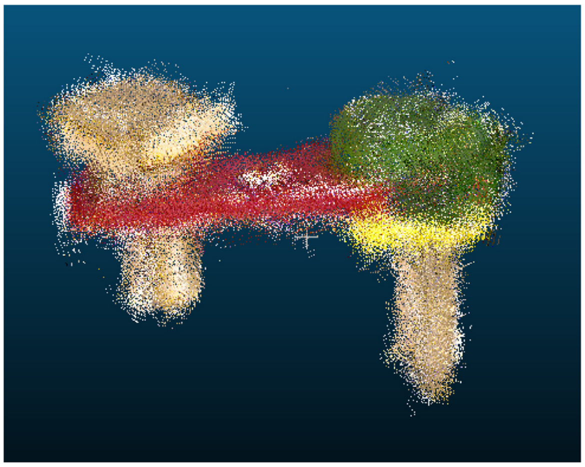


Fig. 20. Registration results. Register $M_1{}^5$ and $M_2{}^5$ to CM_6 after action 6 "screwing".

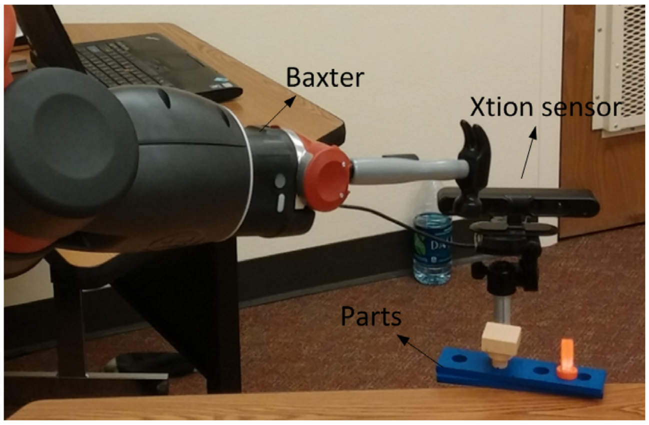


Fig. 21. The robot implementation setup.

scripts are generated automatically:

$$\begin{cases} \Sigma_1 = \langle inserting, N/A, bolt, baseboard, \hat{S}_1 \rangle \\ \Sigma_2 = \langle hammering, hammer, bolt, baseboard, \hat{S}_2 \rangle \\ \Sigma_3 = \langle wrenching, wrench, bolt, baseboard, \hat{S}_3 \rangle \\ \Sigma_4 = \langle aligning, N/A, nut, CP3, \hat{S}_4 \rangle \\ \Sigma_5 = \langle inserting, N/A, screw, CP4, \hat{S}_5 \rangle \\ \Sigma_6 = \langle screwing, screwdriver, screw, CP4, \hat{S}_6 \rangle \end{cases} \tag{24}$$

where N/A stands for "Not Applicable". In summary, with the action recognition and assembly state estimation methods developed in the paper, we can automatically obtain the assembly script from human demonstration. Overall, the average error for translation and rotation is within 10 mm and 10° respectively.

5. Robot implementation

We implement the acquired skill scripts on the Baxter robot as shown in Fig. 21. The Baxter is a dual-arm robot which is capable of performing a variety of production tasks, while safely and intelligently working next to people. Its 360° sonar and front camera can be used for human presence detection, vision-guided movement, and object detection. Each arm has 7 degrees of freedom for maximum flexibility and range [49]. We first teach the Baxter the primitive actions through Kinesthetic teaching. During the demonstration, the assembly parts are tracked by the AR markers using an Kinect sensor. Due to the limited capabilities of the Baxter robot, the following assumptions are made:

- The robot always grasps the part from an overhead position.
- Once a part is grasped, there is no relative motion between the part and the gripper.
- The tool is fixed on the gripper manually.

There are four actions in total: inserting, hammering, wrenching and screwdriving. For each action, the robot arm is guided to the desired pose multiple times from different initial poses. Let $\{\varepsilon_j\}_{j=1}^{M}$ denote the M demonstrations. Each demonstration is normalized to 40 time steps using interpolation. Each data point in ε_j, $\{t_i, \gamma_i\}$, consists of a time step t_i and a pose γ_i which is the pose of the left gripper with respect to the part, ${}^{lg}P_{pt}$. Figs. 22 and 23 show the results of applying GMM/GMR to multiple "hammering" demonstrations which characterize the left gripper's pose with respect to the bolt. The results indicate that the positional constraints of the hammering action are tight at the end, since the hammer always needs to strike the small top surface of the bolt. On the other hand, the rotational constraints are loose, which also makes sense since the hammer can approach the bolt from multiple directions. This method is applied to all the other basic actions.

When a new pose of the part is obtained by the Kinect sensor, it can be transformed into the pose with respect to the Baxter base frame ${}^{pt}P_{base}$. Then the reproduced pose can be derived as follows:

$$ {}^{lg}P_{base} = {}^{lg}P_{pt} \cdot {}^{pt}P_{base} \tag{25}$$

Ultimately, ${}^{lg}P_{base}$ must be converted from the task space to the joint space using the given Inverse Kinematic function.

The robot then implements the demonstrated tasks after the human demonstration. The task has five steps.

- Step 1: Inserting a bolt into a baseboard.
- Step 2: Hammering the bolt down.
- Step 3: Inserting a screw into the baseboard.
- Step 4: Screwdriving the screw down.
- Step 5: wrenching the bolt.

The generated scripts for each step are interpreted as mentioned in Section 3-E. Tracking the parts in real-time is very challenging because of the limited size and occlusion between the assembly parts. Therefore, AR markers are used. By making the marker reasonably small (2 cm by 2 cm), we can reduce the chance of occlusion. The assembly state S describes the relative pose of the two assembly parts. The coordinate of the 3D model of each part is defined by the marker frame on the turntable as shown in Fig. 8. On the other hand, the robot uses an AR marker to track each part. Therefore, for each part, the transformation between the 3D model frame and the AR marker frame has to be calculated. After that, the assembly state is transformed into the AR marker frame. S' and S'_{i-1} are calculated as follows.

$$ S'_i = (T_{ri} * T_{r2a})^{-1} \cdot S'_{i-1} \cdot T_{pi} * T_{p2a} \tag{26}$$

S'_i is the transformed assembly state after action A_i. T_{r2a} is the transformation from the reference part 3D model frame to its AR marker frame. T_{p2a} is the transformation from the non-reference part 3D model frame to its AR marker frame. The robot repeats the assembly action A_i until S'_i is reached.

Fig. 24 shows the robot inserting the bolt into the base board. Fig. 25 shows the robot hammering the bolt down to the base board. It hammers the bolt multiple times (three times in this example) until it reaches the desired pose. Fig. 26 shows the screw being inserted into the slot. Fig. 27 shows the robot screwdriving the screw down to a hole in the base board. This action is repeated 8 times until the assembly state is reached. To track the screw, two

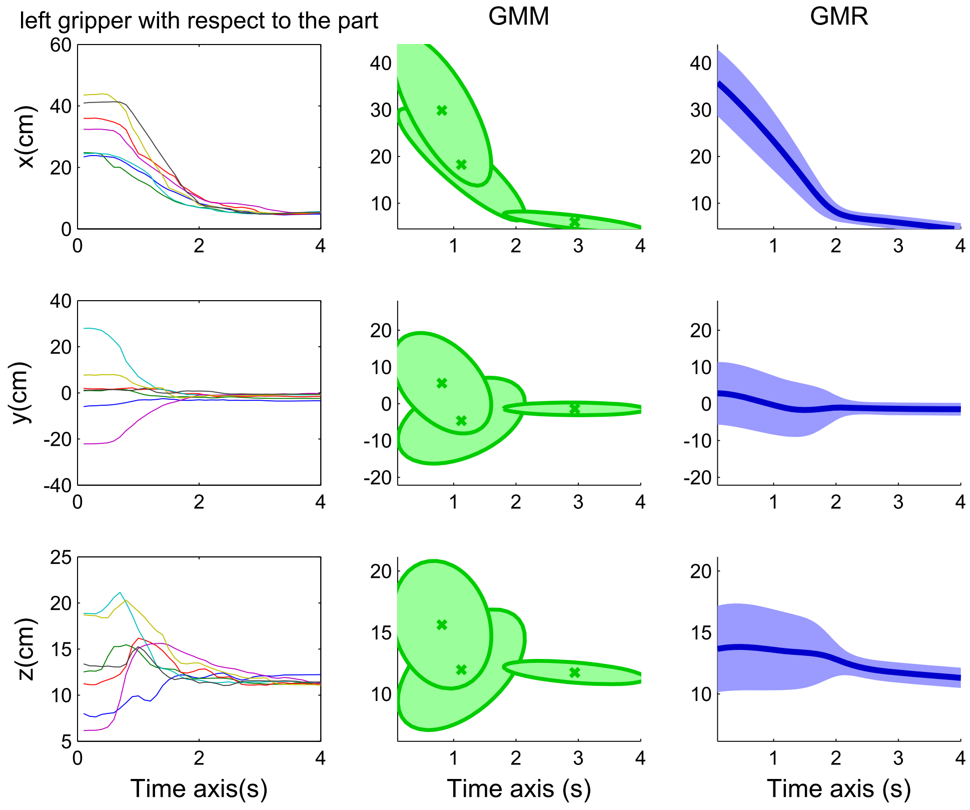


Fig. 22. Trajectory encoding and generalization 1. The figures on the left column represent the trajectories of x,y, and z positions of each demonstration (hammering). The figures in the middle column show the GMMs to model the trajectories. The figures on the right column show the generalized trajectories of each dimension and the corresponding variances.

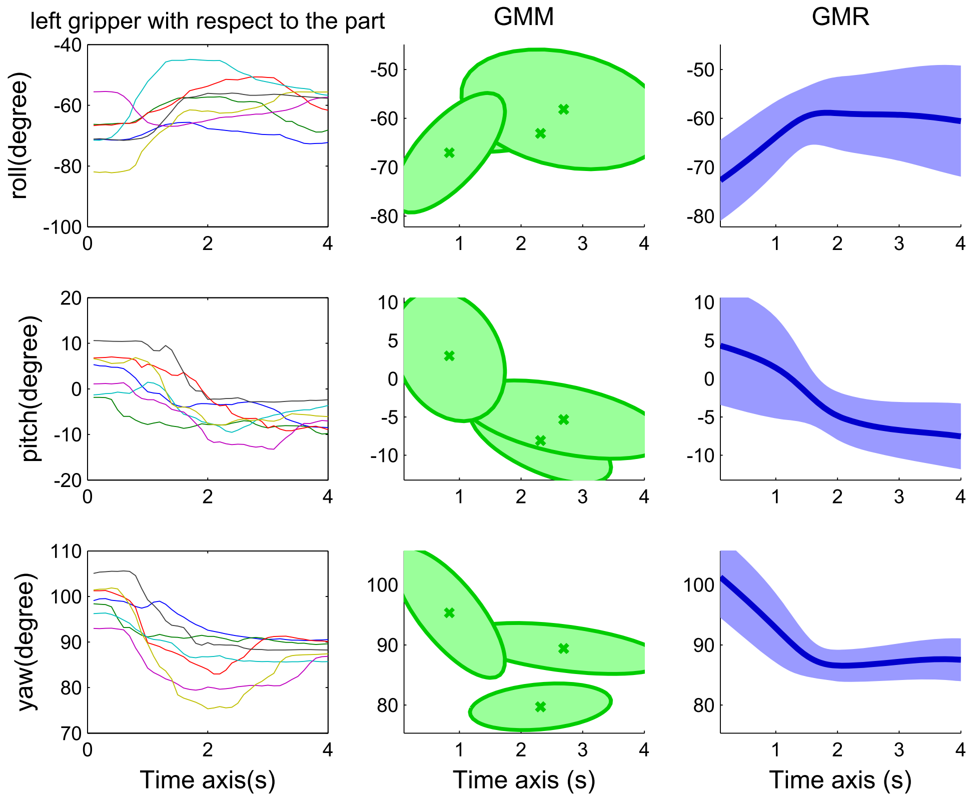


Fig. 23. Trajectory encoding and generalization 2. The figures on the left column represent the roll, pitch, and yaw rotations of each demonstration (hammering). The figures in the middle column show the GMMs to model the trajectories. The figures on the right column show the generalized trajectories of each dimension and the corresponding variances.

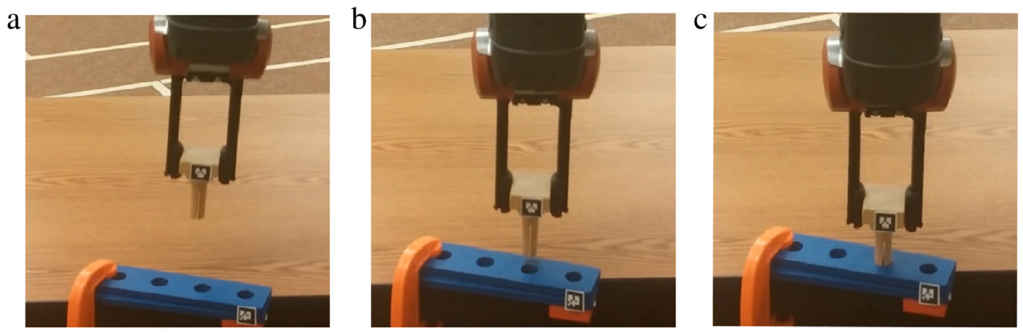


Fig. 24. The robot implements script 1 (inserting bolt). $\Sigma_1 = \langle \mathit{inserting}, N/A, \mathit{bolt}, \mathit{baseboard}, S_1' \rangle$.

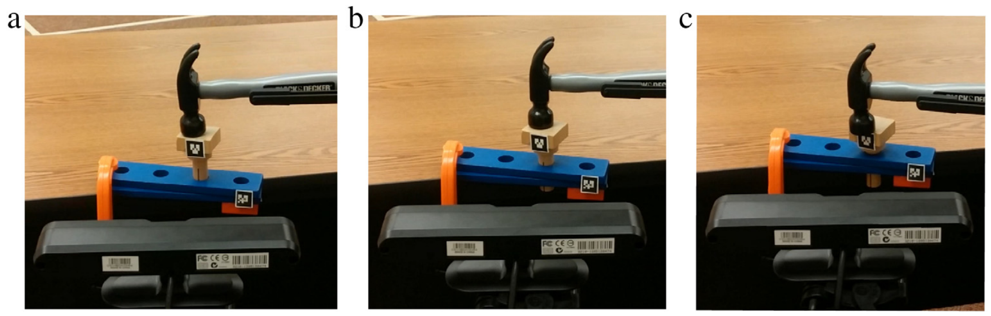


Fig. 25. The robot implements script 2 (hammering bolt). $\Sigma_2 = \langle \mathit{hammering}, \mathit{hammer}, \mathit{bolt}, \mathit{baseboard}, S_2' \rangle$.

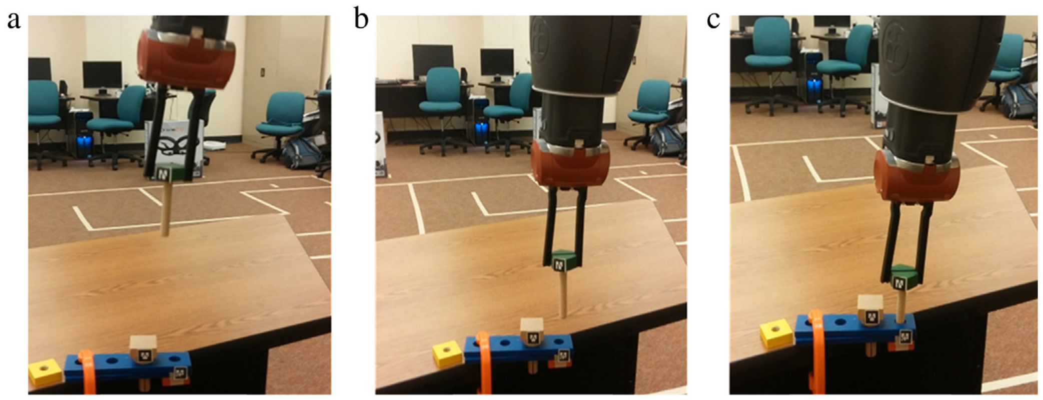


Fig. 26. The robot implements script 3 (inserting screw). $\Sigma_3 = \langle \mathit{inserting}, N/A, \mathit{screw}, \mathit{baseboard}, S_3' \rangle$.

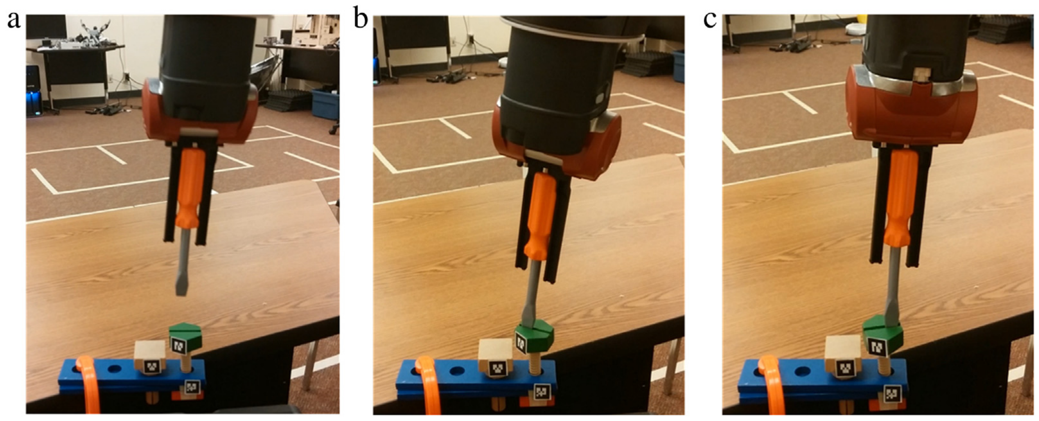


Fig. 27. The robot implements script 4 (screwdriving). $\Sigma_4 = \langle \mathit{screwdriving}, \mathit{screwdriver}, \mathit{screw}, \mathit{baseboard}, S_4' \rangle$.

Table 6
The success rate of script implementation.

Script	1	2	3	4
Success rate	0.70	0.90	0.80	0.60
Script	5			
Success rate	0.75			

identical markers are attached on the two opposite sides of the screw. The reason is twofold. First, the screw rotates 180 degrees after one screwdriving action, so one marker is not enough to track it. Secondly, due to the symmetric geometry of the screw, the use of two identical markers enables the robot to apply the same learned constraint in either configuration.

The success rate of implementing each script is also evaluated and is shown in Table 6. Each script is implemented 20 times. The script describing the action "hammering" is the simplest for the robot, since the hammer only needs to hit the top of the bolt. On the other hand, the most challenging task for the robot is script 4 which involves "screwdriving", since the slot on the screw is very small. Due to the errors of the robot and the AR marker tracking algorithm, the robot sometimes fails to insert the screwdriver into the slot. To judge if a robot implementation is a success or a failure, a criterion is defined. As long as the relative orientation between the two parts is close enough to the desired one, it is considered as a successful implementation. On the other hand, the following will be considered as failures: 1. Massive contact force, which is judged by the torque measurement of each joint. If too much torque is put on any one of the joint, the robot stops immediately by itself and the implementation is considered as failed. 2. The robot fails to reach the goal within a given amount of time.

6. Conclusion

In this paper, we propose a Portable Assembly Demonstration (PAD) system for robots to learn complex assembly skills from humans. Based on a RGB-D camera, tools and parts used in the assembly are recognized, which allows to effectively recognize complex assembly actions. To estimate the assembly state, 3D models of the individual parts and the assembled parts are created using the PAD system. A two-step registration approach is adopted to estimate the assembly state. This registration method is robust to minor occlusions and works well for small parts. Experiments have verified and evaluated the proposed PAD system and the associated theoretical framework. Our PAD system can be used for robots to acquire assembly skills, which can help automate the future factories. Currently the limited accuracy of the created 3D models restricts the current PAD system to only simple parts. Compared to [12,23], the proposed system is more tightly integrated. Since the action recognition considers both human motion and object information, it performs more robustly. Considering the limitations of the current PAD system, our future work will focus on improving the system on the following aspects: (1) The current human demonstration and robot implementation do not involve force. We will investigate using force sensors to better capture human assembly and realizing force control during implementation. (2) The current part tracking is based on AR markers. We will develop more robust, highly accurate part tracking method without using AR markers. (3) The current system cannot handle significant occlusions during part recognition and tracking. We will improve the recognition and tracking method to address this issue.

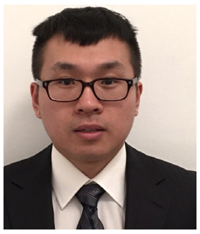
Acknowledgments

This project is supported by the National Science Foundation (NSF) Grant CISE/IIS 1231671/IIS 1427345, National Natural Science Foundation of China (NSFC) Grants 61328302, 61222310, and U1613210, the Open Research Project of the State Key Laboratory of Industrial Control Technology, Zhejiang University, China (No. ICT170314) and the Shenzhen Overseas High Level Talent (Peacock Plan) Program (No. KQTD20140630154026047).

References

[1] H. Wang, X. Liu, Adaptive shared control for a novel mobile assistive robot, IEEE/ASME Trans. Mechatronics 19 (6) (2014) 1725–1736.
[2] J. Pile, N. Simaan, Modeling, design, and evaluation of a parallel robot for cochlear implant surgery, IEEE/ASME Trans. Mechatronics 19 (6) (2014) 1746–1755.
[3] Robohub. [Online] Available: http://robohub.org/foxbots-being-deployed-in-china/.
[4] Baxter robot. http://www.rethinkrobotics.com/, 2014.
[5] Nextage robot. http://nextage.kawada.jp/en/, 2014.
[6] K. Lai, L. Bo, X. Ren, D. Fox, Sparse distance learning for object recognition combining rgb and depth information, in: IEEE International Conference on Robotics and Automation, ICRA, May 2011, pp. 4007–4013.
[7] J. Lei, X. Ren, D. Fox, Fine-grained kitchen activity recognition using rgb-d, in: Proceedings of the 2012 ACM Conference on Ubiquitous Computing, 2012, pp. 208–211.
[8] N. Payet, S. Todorovic, From contours to 3d object detection and pose estimation, in: IEEE International Conference on Computer Vision, Nov 2011, pp. 983–990.
[9] J. Takamatsu, K. Ogawara, H. Kimura, K. Ikeuchi, Recognizing assembly tasks through human demonstration, Int. J. Robot. Res. 26 (7) (2007) 641–659.
[10] J. Aleotti, S. Caselli, M. Reggiani, Toward programming of assembly tasks by demonstration in virtual environments, in: IEEE International Workshop on Robot and Human Interactive Communication, 2003, pp. 309–314.
[11] P. Azad, T. Asfour, R. Dillmann, Toward an unified representation for imitation of human motion on humanoids, in: IEEE International Conference on Robotics and Automation, 2007, pp. 2558–2563.
[12] R. Dillmann, Teaching and learning of robot tasks via observation of human performance, Robot. Auton. Syst. 47 (2–3) (2004) 109–116.
[13] H. Friedrich, S. Münch, R. Dillmann, S. Bocionek, M. Sassin, Robot programming by demonstration (rpd): Supporting the induction by human interaction, Mach. Learn. 23 (2–3) (1996) 163–189.
[14] S. Calinon, A. Billard, A probabilistic programming by demonstration 1017 framework handling constraints in joint space and task space, in: IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008, pp. 367–372.
[15] M. Denisa, A. Gams, A. Ude, T. Petric, Learning compliant movement primitives through demonstration and statistical generalization, IEEE/ASME Trans. Mechatronics PP (99) (2016) pp. 1–1.
[16] D. Leidner, A. Dietrich, M. Beetz, A. Albu-Schäffer, Knowledge-enabled parameterization of whole-body control strategies for compliant service robots, Auton. Robots 40 (3) (2016) 519–536.
[17] C. Zhu, W. Sheng, Realtime human daily activity recognition through fusion of motion and location data, in: 2010 IEEE International Conference on Information and Automation, June 2010, pp. 846 –851.
[18] R. Zollner, O. Rogalla, R. Dillmann, Integration of tactile sensors in a programming by demonstration system, in: IEEE International Conference on Robotics and Automation, vol. 3, 2001, pp. 2578–2583.
[19] S. Lee, H. Suh, Skill learning and inference framework for skilligent robot, in: IEEE International Conference on Intelligent Robots and Systems, 2013, pp. 108–115.
[20] L. Kunze, A. Haidu, M. Beetz, Skill learning and inference framework for skilligent robot, in: IEEE International Conference on Intelligent Robots and Systems, 2013, pp. 108–115.
[21] Z. Fang, G. Bartels, M. Beetz, Learning models for constraint-based motion parameterization from interactive physics-based simulation, in: Intelligent Robots and Systems, IROS, 2016 IEEE/RSJ International Conference on IEEE, 2016, pp. 4005–4012.
[22] S.R. Ahmadzadeh, P. Kormushev, D. Caldwell, Visuospatial skill learning for object reconfiguration tasks, in: IEEE International Conference on Intelligent Robots and Systems, 2013, pp. 685–691.
[23] S.R. Ahmadzadeh, A. Paikan, F. Mastrogiovanni, L. Natale, P. Kormushev, D.G. Caldwell, Learning symbolic representations of actions from human demonstrations, in: IEEE International Conference onRobotics and Automation, ICRA, 2015, pp. 3801–3808.
[24] W. Takano, Y. Yamada, Y. Nakamura, Generation of action description from classification of motion and object, in: Robotics and Autonomous Systems, vol. 91, 2017, pp. 247–257.
[25] C. Matuszek, L. Bo, L. Zettlemoyer, D. Fox, Learning from unscripted deictic gesture and language for human–robot interactions, in: Twenty-Eighth AAAI Conference on Artificial Intelligence, 2014, pp. 2556–2563.

[26] K. Ramirezamaro, M. Beetz, G. Cheng, Understanding the intention of human activities through semantic perception: Observation, understanding and execution on a humanoid robot, Adv. Robot. 29 (5) (2015) 345–362.
[27] Y.C. Song, H.A. Kautz, A testbed for learning by demonstration from natural language and rgb-depth video, in: AAAI, 2012.
[28] G. Hovland, P. Sikka, B. McCarragher, Skill acquisition from human demonstration using a hidden markov model, in: IEEE International Conference on Robotics and Automation, vol. 3, 1996, pp. 2706–2711.
[29] L.R. Rabiner, A tutorial on hidden markov models and selected applications in speech recognition, in: Proceedings of the IEEE, 1989, pp. 257–286.
[30] N. Dantam, I. Essa, M. Stilman, Linguistic transfer of human assembly tasks to robots, in: International Conference on Intelligent Robots and Systems, IROS, 2012, pp. 237–242.
[31] Y. Wang, R. Xiong, L. Shen, K. Sun, J. Zhang, L. Qi, Towards learning from demonstration system for parts assembly: A graph based representation for knowledge, in: IEEE 4th Annual International Conference on Cyber Technology in Automation, Control, and Intelligent Systems, CYBER, June 2014, pp. 174–179.
[32] J. Huckaby, S. Vassos, H.I. Christensen, Planning with a task modeling framework in manufacturing robotics, 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5787–5794, 2013.
[33] D. Di Marco, A. Koch, O. Zweigle, K. Haussermann, B. Schiessle, P. Levi, D. Galvez-Lopez, L. Riazuelo, J. Civera, J. Montiel, M. Tenorth, A. Perzylo, M. Waibel, R. van de Molengraft, Creating and using roboearth object models, in: IEEE International Conference on Robotics and Automation, ICRA, May 2012, pp. 3549 –3550.
[34] M. Quigley, K. Conley, B.P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A.Y. Ng, Ros: An open-source robot operating system, in: ICRA Workshop on Open Source Software, 2009.
[35] H. Gonzalez-Jorge, B. Riveiro, E. Vazquez-Fernandez, J. Martí nez Sánchez, P. Arias, Metrological evaluation of microsoft kinect and asus xtion sensors, Measurement 46 (6) (2013) 1800–1806.
[36] R.B. Rusu, S. Cousins, 3d is here: Point cloud library (pcl), in: IEEE International Conference on Robotics and Automation, ICRA, Shanghai, China, May 2011.
[37] L.A. Alexandre, 3D descriptors for object and category recognition: A comparative evaluation, in: Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, October 2012.
[38] F. Tombari, S. Salti, L. Di Stefano, A combined texture-shape descriptor for enhanced 3d feature matching, in: IEEE International Conference on Image Processing, ICIP, 2011, pp. 809–812.
[39] F. Tombari, S. Salti, L. Di Stefano, Unique signature of histograms for local surface description, in: Proceedings of the 11th European Conference on Computer Vision, 2010, pp. 356–369.
[40] F. Tombari, L. Di Stefano, Object recognition in 3d scenes with occlusions and clutter by hough voting, in: Fourth Pacific-Rim Symposium on Image and Video Technology, PSIVT, 2010, pp. 349–355.
[41] Openni. [Online] Available: http://www.openni.org/Documentation/.
[42] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the em algorithm, J. Roy. Statist. Soc. B Methodol. 39 (1) (1977) 1–38.
[43] Y. Gu, H. Do, J. Evert, W. Sheng, Human gesture recognition through a kinect sensor, in: IEEE International Conference on Robotics and Biomimetics, Dec. 2012.
[44] M. Waibel, M. Beetz, J. Civera, R. D'Andrea, J. Elfring, D. Galvez-Lopez, K. Haussermann, R. Janssen, J. Montiel, A. Perzylo, B. Schiessle, M. Tenorth, O. Zweigle, R. van de Molengraft, Roboearth, IEEE Robot. Autom. Mag. 18 (2) (2011) 69–82.
[45] R.B. Rusu, Semantic 3d Object Maps for Everyday Manipulation in Human Living Environments, Ph.D Dissertation, Computer Science department, Technische Universitaet Muenchen, Germany, 2009.
[46] R. Rusu, N. Blodow, M. Beetz, Fast point feature histograms (fpfh) for 3d registration, in: IEEE International Conference on Robotics and Automation, 2009, pp. 3212–3217.
[47] P. Besl, N.D. McKay, A method for registration of 3-d shapes, IEEE Trans. Pattern Anal. Mach. Intell. 14 (2) (1992) 239–256.
[48] K. Shoemake, Animating rotation with quaternion curves, in: Proceedings of the 12th annual conference on Computer graphics and interactive techniques, 1985, pp. 245–254.
[49] E. Guizzo, E. Ackerman, The rise of the robot worker, IEEE Spectr. 49 (10) (2012) 34–41.



Ye Gu was born in China in 1985. He received the B.E. from Harbin Institute of Technology, Harbin, China, in 2007, M.E. from University of Minnesota Duluth, Duluth, USA, in 2010 and Ph.D. degrees from Oklahoma State University, Stillwater, USA, in 2015. He joined Zebra Technologies, in 2015. His main areas of research interest are robot skill learning from demonstration, computer vision, image and point cloud processing.

Weihua Sheng is currently an associate professor at the School of Electrical and Computer Engineering, Oklahoma State University (OSU), USA. He is the Director of the Laboratory for Advanced Sensing, Computation and Control (ASCC Lab, http://ascc.okstate.edu) at OSU. Dr. Sheng received his Ph.D degree in Electrical and Computer Engineering from Michigan State University in May 2002. He obtained his M.S and B.S. degrees in Electrical Engineering from Zhejiang University, China in 1997 and 1994, respectively. He is the author of more than 150 peer-reviewed papers in major journals and international conferences. Seven of them have won best paper or best student paper awards in major international conferences. His current research interests include mobile robotics, wearable computing, human robot interaction and intelligent transportation systems. His research has been supported by US National Science Foundation (NSF), Department of Defense (DoD), Oklahoma Transportation Center (OTC) /Department of Transportation (DoT), etc. Dr. Sheng is a senior member of IEEE and serves as an Associate Editor for IEEE Transactions on Automation Science and Engineering.

Christopher Crick received his Ph.D. from Yale University in 2009. He is currently Assistant Professor of Computer Science at Oklahoma State University and director of the Cognitive Robotics Laboratory. His work addresses the many challenges of humans and robots working together in teams.

Yongsheng Ou is a Professor at Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences. He holds a B.Sc. degree in Mechanical & Electrical Engineering (Beijing University of Aeronautics and Astronautics, 1995) and a M.Sc. degree in Electrical Engineering (Institute of Automation, Chinese Academy of Sciences, 1998). Ou received a Ph.D. degree in Automation & Computer-Aided Engineering from the Chinese University of Hong Kong in 2004. He is a coauthor of the monograph on Control of Single Wheel Robots (Springer, 2005). His research interests include control of complex systems, learning control by demonstrations, and service robotics.