Neurocomputing 259 (2017) 85–93

Contents lists available at ScienceDirect

Neurocomputing

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

Teaching robots to do object assembly using multi-modal 3D vision

Weiwei Wan[a,*], Feng Lu[b], Zepei Wu[a], Kensuke Harada[a,c]

[a] Intelligent System Research Institute, National Institute of Advanced Industrial Science and Engineering, Japan
[b] State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, and the International Institute for Multidisciplinary Science, Beihang University, China
[c] Graduate School of Engineering Science, Osaka University, Japan

ARTICLE INFO

Article history:
Received 25 January 2016
Revised 2 October 2016
Accepted 16 January 2017
Available online 7 February 2017

Keywords:
3D visual detection
Robot manipulation
Motion planning

ABSTRACT

The motivation of this paper is to develop an intelligent robot assembly system using multi-modal vision for next-generation industrial assembly. The system includes two phases where in the first phase human beings demonstrate assembly to robots and in the second phase robots detect objects, plan grasps, and assemble objects following human demonstration using AI searching. A notorious difficulty to implement such a system is the bad precision of 3D visual detection. This paper presents multi-modal approaches to overcome the difficulty: It uses AR markers in the teaching phase to detect human operation, and uses point clouds and geometric constraints in the robot execution phase to avoid unexpected occlusion and noises. The paper presents several experiments to examine the precision and correctness of the approaches. It demonstrates the applicability of the approaches by integrating them with graph model-based motion planning, and by executing the results on industrial robots in real-world scenarios.

1. Introduction

The motivation of this paper is to develop an intelligent rob ot assembly system using multi-modal vision for next-generation industrial assembly: (1) A human worker assembles objects in front of a vision system. (2) The system detects the position and orientation of the objects and learns how to do assembly following the human worker's demonstration. (3) An industrial robot performs assembly tasks using the data learned from human demonstration. It detects objects in its workspace, picks up them, and does assembly using assembly planning and motion planning.

The difficulty in developing such a system is precise visual detection. Two problems exist where the first one is in the human teaching phase, namely how to precisely detect the position and orientation of the objects in human hands during manual operation; The second one is in the robot execution phase, namely how to precisely detect the position and orientation of objects in the workspace and perform assembly.

This paper develops a novel multi-modal approach to solve the two problems. First, we attach AR markers to the objects for assembly and track them by detecting and transforming the marker positions during human demonstration. We don't need to worry about occlusions in this process since the teaching phase is manual and is performed by human beings who are smart enough to actively avoid occlusions and ensure good exposure to vision systems. The modal employed in this phase is the markers (RGB image) and the geometric relation between the markers and the object models. The tag "AR(RGB)" is used to denote this modal. Second, during robot execution, we roughly detect the pose of the objects by matching the object model to the point clouds obtained from depth camera and use the geometric constraints of horizontal table surface to refine the detected results. The robot execution phase is automatic and is not as flexible as human teaching. Therefore we use markerless approaches to avoid occlusions. We fuse the point clouds and geometric constraints to make up the partial loss and noises, and improve the detected results. The geometric constraints are based on an assumption that when an object is placed on the surface of a table, it stabilizes at a limited number of poses. These poses freeze some Degree of Freedom and improve precision. The modal employed in this phase is the cloud point data and the geometric constraints of a horizontal surface. The tag "Depth+Geom" is used to denote it. Moreover, we propose an improved graph model based on our previous work to perform integrated assembly planning and motion planning.

Our contribution is we use different modals according to the requirements and limitations of different phases. In the human teaching phase, AR markers are used since human beings could control the operation and actively avoid occlusion. In the robot execution phase, point clouds are used to find a rough pose since occlusion happens frequently. Geometric constraints are used to improve the rough results. Experiments are performed to examine

* Corresponding author.
E-mail address: wanweiwei07@gmail.com (W. Wan).

http://dx.doi.org/10.1016/j.neucom.2017.01.077
0925-2312/

the precision and correctness of our approaches. We quantitatively show the advantages of "AR(RGB)" and "Depth+Geom" in next-generation industrial assembly and concretely demonstrate the process of searching and planning using the improve graph model. The developed approaches are integrated with Kawada Nextage Robots to show the applicability in real-world scenarios.

2. Related work

This paper is highly related to studies in 3D object detection for robotic manipulation and assembly and the literature review part of this paper concentrates on it. For general studies on robotic grasping, manipulation, and assembly, refer to [1,2], and [3]. Also, the literature review emphasizes on model-based approaches since the paper is motivated by next-generation industrial assembly and is about the industrial applications where precision is crucial and object models are available. For model-less studies, refer to [4] and [5]. For appearance-based studies, refer to [6,7], and [8]. Moreover, learning approaches are not reviewed since they are not precise. Refer to [9] and [10] if interested.

The related work is archived and reviewer according to the modals used for 3D detection, including RGB images, markers, point clouds, haptic sensing, extrinsic constraints, and multi-modal fusion.

2.1. RGB images

RGB images are the most commonly used modal of robotic visual perception. Using RGB images to solve the model-based 3D position and orientation detection problem is widely known as the "model-to-image registration problem" [11] and is under the framework of POSIT (Pose from Orthography and Scaling with Iterations) [12]. When looking for objects in images, the POSIT-based approaches match the feature descriptors by comparing the most representative features of an image to the features of the object for detection. The features could either be values computed at pixel points or histograms computed on a region of pixels. Using more than three matches, the 3D position and orientation of an object can be found by solving polynomial equations [13–15]. A good material that explains the matching process can be found in [16] which studied multi-view image matching. It is not directly related to 3D detection, but explains well how to match the feature descriptors.

Some of the most common choices of features include corner features [17] which are applied in [18] and [12], line features which are applied in [19,20] and [21], cylinder features which are applied in [21] and [22], and SIFT features [23] which are applied in [24] and [25]. Especially, [24] stated clearly the two stages of model-based detection using RGB images: (1) The modeling stage where the textured 3D model is recovered from a sequence of images; (2) The detection stage where features are extracted and matched against those of the 3D models. The modeling stage is based on the algorithms in [16]. The detection stage is open to different features, different polynomial solving algorithms, and some optimizations like Levenberg–Marquardt [26] and Mean-shift [27]. [28] compared the different algorithms in the second stage.

2.2. Markers

In cases where the objects don't have enough features, markers are used to assist image-based detection. Possible marker types include: AR markers, colored markers, and shape markers, etc. The well known AR libraries [29,30] provide robust libraries and the Optitrack device provides easy-to-use systems for the different marker types. However, the applications with markers require some manual settings and there are limitations on marker sizes, view directions, etc.

[31] used circular concentric ring fiducial markers which are placed at known locations on a computer case to overlay the hidden innards of the case on the camera's video feed. [32] used AR markers to locate the display lots during virtual conference. It explained the underneath computation well. [33] didn't directly use markers, but used the offline matched keyframes, which were essentially the same thing, to correct online tracking. [34] used AR markers to recognize and cage objects. [35] used both balls (circles) and AR markers to estimate and track the position of objects. More recently, [36] used AR markers to perform localization and tracking in wide area. [37] used AR markers to track the position of human hands and the operating tools and used the tracked motion to teach robots. The later work shared the same assumption with us that human beings can actively avoid occlusions. However, it didn't need high precision since the goal of tracking was in task level, instead of the low level trajectories.

2.3. Point clouds

Point clouds can be acquired using a structured light camera [38], stereovision camera [39], and LIDAR device [40], etc. The recent availability of low cost depth sensors and the handy Point Cloud Library (PCL) [41] has widely disseminated 3D sensing technology in research and commercial applications. The basic flow of using point clouds for detection is the same as image-based detection: The first step is to extract features from models and objects; The second step is to match the features and detect the 3D poses.

[42] is one of the early studies that used point clouds to detect the pose of an object. It was based on the Iterative Closest Point (ICP) algorithm [43] which iteratively minimizes the mean square distance of nearest neighbor points between the model and the point clouds. Following work basically uses the same technique, with improvements in feature extraction and matching algorithms. The features used in point clouds detection are more general than those in image-based detection, including local descriptors like signature of histograms of orientation (SHOT) [44] and Radius-based Surface Descriptor (RSD) [45], and global descriptors like Clustered Viewpoint Feature Histogram (CVFH) [46] and Ensemble of Shape Functions (ESF) [47]. The matching algorithms didn't change much, sticking to Random Sample Consensus (RANSAC) [13] and ICP. A complete review and usage of the features and matching algorithms can be found in [48].

2.4. Extrinsic constraints

Like the markers, extrinsic constraints are used to improve point clouds based detection. When the point clouds fail provide enough features or when the point clouds are noisy and occluded, it is helpful to take into account the extrinsic constraints. For example, the detection pipeline in [48] used hypothesis to verify the result of feature matching. The hypothesis is one example of extrinsic constraints. [49] analyzed the functions of connected object parts and used them to refine grasps. The paper is not directly related to detection but is an example of improving performance using extrinsic constraints caused by adjacent functional units.

Some other work uses geometric constraints to reduce ambiguity of ICP. [50] and [51] segmented 3D clutter using the geometric constraints. They are not directly related to detection but are used widely as the initial steps of many detection algorithms. [52] clustered point clouds into polygon patches and uses RANSAC with the multiple polygon constraints to improve the precision of detection. [53] used table constraints for segmentation and uses Hough voting to detect object poses. [54] used the sliced 2D contours of 3D stable placements to reduce the noises of estimation. It is similar to our approach but is contour-based and suffers from ambiguity.

2.5. Multi-modal fusion

Multi-modal approaches are mostly the combination or repetition of the previous five modals. For example, some work fuses repeated modals to improve object detection. One representative publication is [55] which fuses colour, edge and texture cues predicted from a textured CAD model of the tracked object to recover the 3D pose, and is open to additional cues.

Some other work uses visual tracking to correct the noises caused by fast motions and improve the precision of initial matches. The fused modals include the RGB image modal and the motion modal where the later one could be either estimated using image sequences or third-part sensors like Global Positioning System (GPS) or gyros. [56] is one representative work which fuses model motion (model-based tracking) and model detection in RGB images to refine object poses. [20] fuses gyro data and line features of RGB images to reinforce the pose estimation for head-mounted displays. [57] uses gyro data, point descriptors, and line descriptors together to improve the performance of pose estimation for outdoor applications. [58] uses gyro data, image sequences, and geometric constraints to implement visual location and recognition.

[59] uses point clouds to cluster the scene and find the Region-of-Interests (ROIs), and uses image modal to estimate the object pose at respective ROIs. The fused modals are RGB images and point clouds. [60] also combines image and depth modals. Image gradients found on the contour of images and surface normals found on the body of point clouds are used to estimate object poses. [61] performs high-precision 3D reconstruction by combining RGB and depth information.

To our best knowledge, contemporary object detection studies do not meet our requirements on precision. The most demanding cases in the contemporary publications are robotic grasping and simple manipulation, which are far less strict than regrasp and assembly. In order to improve the precision, we develop a novel approach by fusing different modals to deal with the problems in the teaching phase and the robot execution phase respectively: (1) We use AR markers to detect the 3D object positions and orientations during human teaching; (2) We use the cloud point data and the geometric constraints from planar table surface during robot execution. The details are presented in following sections.

3. System overview

We present an overview of the next-generation industrial assembly system and make clear the positions of the 3D detection in this section. The details of the "AR(RGB)" and "Depth+Geom" approaches will be described in the sections following it.

Fig. 1 shows the flow of the next-generation industrial assembly system. It is composed of a human teaching phase and a robot execution phase. In the human teaching phase, a human worker demonstrates how to assemble objects in front of a vision system. The system remembers the relationship of the two objects and saves it as an intermediate value.

In the robot execution phase, the robot uses another vision system to find the objects in the workspace, picks up them, and performs assembly. The relative position and orientation of the objects in an assembled structure are the values perceived in the human teaching phase. The grasping configurations of robot hands and motions are computed online using motion planning algorithms.

An advantage of this system is it doesn't need direct robot programming and is highly flexible. It only requires a human worker to attach the markers and pre-save the pose of the marker in the object's local coordinate system so that the system can compute the pose of the object from the detected markers.

The 3D detection of the two phases are denoted by the two "object detection" sub-boxes in Fig. 1. The one in the human teaching phase uses AR markers since human beings can intentionally avoid unexpected partial occlusions by human hands or the other objects, and as well as ensure high precision. The one in the motion planning phase uses point clouds to roughly detect the object pose, and uses the geometric constraints from planar table surface to correct the noises and improve precision. The details will be explained in following sections. The symbols are as follows.

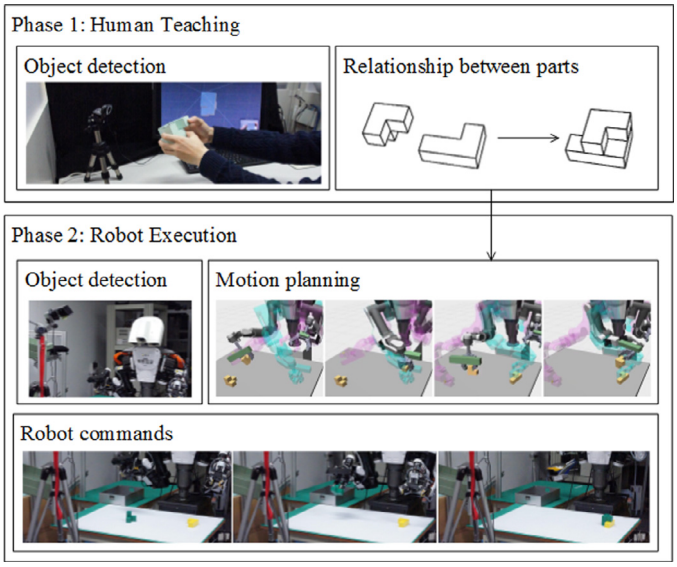


Fig. 1. The flow of the next-generation industrial assembly. It is composed of a human teaching phase and a robot execution phase. In the human teaching phase, a human worker demonstrates assembly with marked objects in front of an RGB camera. The computer detects the relationship of the assembly parts. In the robot execution phase, the robot detects the parts in the workspace using depth camera and geometric constraints, picks up them, and performs assembly.

- $\boldsymbol{p}_X{}^s$ The position of object X on a horizontal table surface. We use A and B to denote the two objects and consequently use $\boldsymbol{p}_A{}^s$ and $\boldsymbol{p}_B{}^s$ to denote their positions.
- $\mathbf{R}_X{}^s$ The orientation of object X on a horizontal table surface. Like $\boldsymbol{p}_X{}^s$, X is to be replaced by A or B.
- $\boldsymbol{p}_X{}^a$ The position of object X in the assembled structure.
- $\mathbf{R}_X{}^a$ The orientation of object X in the assembled structure.
- $\boldsymbol{p}_X{}^p$ The pre-assembly positions of the objects. The robot will plan a motion to move the objects from initial positions to the pre-assemble positions.
- $\mathbf{g}_X{}^f$ The force-closure grasps of object X. The letter f indicates the object is free. It is neither in an assembled structure nor laying on something.
- $\mathbf{g}_X{}^{s'}$ The force-closure grasps of object X on a horizontal table surface. It is in the local coordinate system decided by ($\boldsymbol{p}_X{}^s$, $\mathbf{R}_X{}^s$).
- $\mathbf{g}_X{}^s$ The collision-free, IK (Inverse Kinematics) feasible, and force-closure grasps of object X on a horizontal table surface. It is in the local coordinate system decided by ($\boldsymbol{p}_X{}^s$, $\mathbf{R}_X{}^s$) and is a subset of $\mathbf{g}_X{}^{s'}$.
- $\mathbf{g}_X{}^{a'}$ The force-closure grasps of object X in an assembled structure. It is in the local coordinate system decided by ($\boldsymbol{p}_X{}^a$, $\mathbf{R}_X{}^a$).
- $\mathbf{g}_X{}^a$ The collision-free, IK (Inverse Kinematics) feasible, and force-closure grasps of object X in the assembled structure. It is in the local coordinate system decided by ($\boldsymbol{p}_X{}^a$, $\mathbf{R}_X{}^a$) and is a subset of $\mathbf{g}_X{}^{a'}$.
- $\mathbf{g}_X{}^{p'}$ The force-closure grasps of object X at a pre-assembly position.
- $\mathbf{g}_X{}^p$ The collision-free, IK feasible, and force-closure grasps of object X at a pre-assembly position. It is a subset of $\mathbf{g}_X{}^{p'}$.

Fig. 2. Object detection using AR markers. Beforehand, human worker attaches the markers and presaves the pose of the marker in the object's local coordinate system. During detection, vision system computes the pose of the object from the detected marker poses (the three subfigures). The output is the $(\boldsymbol{p}_A{}^a, \mathbf{R}_A{}^a)$ and $(\boldsymbol{p}_B{}^a, \mathbf{R}_B{}^a)$.

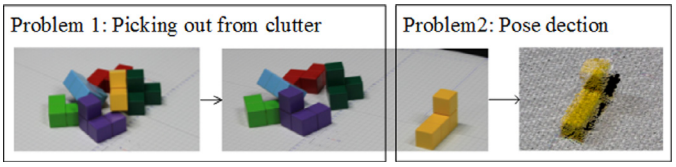


Fig. 3. Overcome the clutter problem by dividing the pose detection into two subproblems. The first one is picking out from clutter where the system doesn't care what the object is or the pose of the object and its only goal is to pick something out. The problem is well studied. The second one is to detect the pose of a single object on an open plenary surface. It is not trivial since the precision of Kinect is bad.

4. 3D detection during human teaching

The object detection in the human teaching phase uses AR markers and RGB cameras. Fig. 2 shows the flow of the detection and the poses of the markers in the object model's local coordinate system. The markers are attached manually by human beings.

During teaching, a human worker assembles two objects in front of a camera. We assume the worker has enough intelligence to expose the markers to the camera without occlusion. The detection process is a common approach which could be found in many AR literature: Given some features in the markers' local coordinate system, find a transform matrix which converts the features onto the coordinate system of a captured image.

In the example shown in Fig. 2, two objects are detected where the results are represented by $(\boldsymbol{p}_A{}^a, \mathbf{R}_A{}^a)$ and $(\boldsymbol{p}_B{}^a, \mathbf{R}_B{}^a)$. During robot execution, the $(\boldsymbol{p}_B{}^a, \mathbf{R}_B{}^a)$ is set to:

$$\boldsymbol{p}_B{}^a = \boldsymbol{p}_B{}^a - \boldsymbol{p}_A{}^a,\ \ \mathbf{R}_B{}^p = \mathbf{R}_B{}^a \cdot (\mathbf{R}_A{}^a)' \tag{1}$$

and $\boldsymbol{p}_A{}^a$ and $\mathbf{R}_A{}^a$ are set to zero and identity matrix respectively. Only the relative poses between the two objects are used.

5. 3D detection during robotic execution

The object pose detection during robot execution is done using Kinect, PCL, and geometric constraints. Point clouds are used instead of markers since: (1) There are many target objects during execution and it is impossible to attach markers to all of them. (2) Markers might be occluded from time to time during robotic pick-and-place. Image-based approaches are not considered because: Target objects are mono colored and textureless, image features are not only helpless but even harmful.

Using Kinect is not trivial due to its low resolution and precision. For example, the objects in industrial applications are usually in clutter. It is difficult to segment one from another using Kinect's resolution. Our solution is to divide the difficulty caused by clutter into two sub-problems: First, the robot selects one object and picks out it from the clutter. The robot places down the object randomly on an open plenary surface. Second, the robot estimates the pose of the single object on the open plenary surface.

The goal of the first sub-problem is to pick something out, without considering object identifiers nor object poses. It is known as a pick-and-place problem in contemporary literature and is illustrated in the left part of Fig. 3. It is well studied in [62].

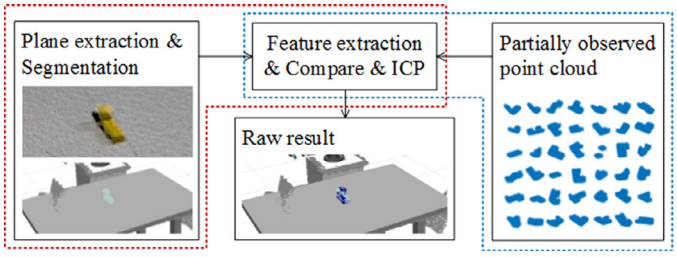


Fig. 4. Rawly detecting the pose of an object using model matching and CVFH and CRH features. In a preprocessing process before the detection, we precompute the CVFH and CRH features of 42 different views and save them as the template. The preprocessing process is shown in the dashed blue frame. During the detection, we segment the remaining point clouds, compute the CVFH and CRH features of each segmentation, and match them to the precomputed views using ICP. The best match is used as the output. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The goal of the second sub-problem is to detect the pose of the single object on an open plenary surface and is shown in the right part of Fig. 3. It is relatively easier but still requires much process to meet the requirements of precision for assembly. This section presents approaches to solve the second sub-problem.

5.1. Rough detection using point clouds

First, we roughly detect the pose of the object using CVFH and CRH features. In a preprocessing process before starting the detection, we pose the camera to 42 viewpoints on a unit sphere, save the view of the camera, and pre-compute the CVFH and CRH features of each view. This step is virtually performed using PCL and is shown in the dashed blue frame of Fig. 4. During the detection, we extract the horizontal table surface from the point clouds, segment the remaining point clouds, and compute the CVFH and CRH features of each segmentation. Then, we match the precomputed features with the features of each segment and estimate the orientation of the segmentations. This step is shown in the dashed red frame of Fig. 4. The matched segments are further refined using ICP to ensure good matching. The segmentation that has highest ICP matches and smallest outlier points wi ll be the result of the rough detection. An example is shown in the "Raw result" framebox of Fig. 4.

5.2. Noise correction using geometeric constraints

The result of rough detection is further refined using geometric constraints. Since the object is on horizontal table surface, its stable poses are limited [63] and can be used to correct the noises of the roughly estimated result. The limited stable poses are computed in a preprocessing process using placement planning. It includes two steps: First, we compute the convex hull of the object's mesh model and perform surface clustering on the convex hull to find some facets. Each facet is a candidate surface where the object may stand on. Second, we check the stability of the objects standing on the candidate surfaces. The unstable placements (the pose where the projection of center of mass is outside the candidate surface or too near to its boundary) are removed. An example of the stable placements for an L-shape block is shown in the stable placements framebox of Fig. 5.

Given the raw detection result using CVFH and CRH features, the robot computes its distance to the stable placements and performs noise correction following Algorithm. 1 .

Here, $\mathbf{I}$ indicates a 3×3 identity matrix. Functions `rotFromRpy` and `rpyFromRot` convert *roll, pitch, yaw* angles to rotation matrix and vice verse. The distance between two rotation matrices is computed in line 4. The corrected result is updated at lines 11 and 12.

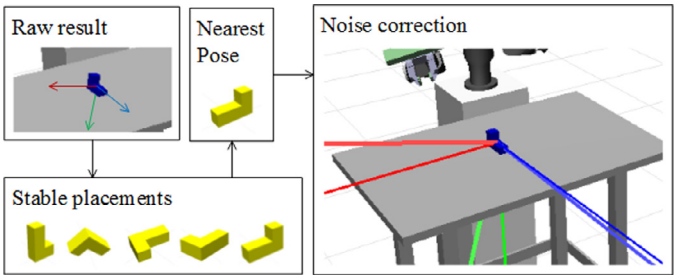


Fig. 5. Correcting the raw result using the stable placements on a plenary surface (geometric constraints). In a preprocessing process before the correction, we compute the stable placements of the object on a plenary surface. An example is shown in the stable placements framebox. During the correction, we compute the distance between the raw results and each of the stable placements, and correct the raw results using the nearest pose.

Algorithm 1: Noise correction.

Data: Raw result: $\boldsymbol{p}_r$, $\mathbf{R}_r$;
Stable placements: $\{\mathbf{R}_p(i),\ i = 1, 2, \ldots\}$
Table height: h_t
Result: Corrected result: $\boldsymbol{p}_c$, $\mathbf{R}_c$

$d_{min} \leftarrow +\infty$;
$\mathbf{R}_{near} \leftarrow \mathbf{I}$;
for $i \leftarrow 1$ **to** $\mathbf{R}_p$.size() **do**
  $d_i \leftarrow ||\log(\mathbf{R}_p(i)\mathbf{R}_p{}')||$;
  if $d_i < d_{min}$ **then**
    $d_{min} \leftarrow d_i$;
    $\mathbf{R}_{near} \leftarrow \mathbf{R}_p(i)$;
  end
end
$\mathbf{R}_{yaw} \leftarrow$ rotFromRpy(*0, 0,*rpyFromRot($\mathbf{R}_r$).y);
$\mathbf{R}_c \leftarrow \mathbf{R}_{yaw} \cdot \mathbf{R}_{near}$;
$\boldsymbol{p}_c \leftarrow [\boldsymbol{p}_r.x, \boldsymbol{p}_r.y, h_t]'$

6. Grasp and motion planning

After finding the poses of the objects on the horizontal table surface, the robot grasps the parts and assembles them. The process includes two steps: A grasp planning step and a motion planning step.

6.1. Grasp planning

In the grasp planning step, we set the object at free space and compute the force-closure and collision-free grasps. Each grasp is represented using $\mathbf{g}_X{}^f = \{\boldsymbol{p}_0, \boldsymbol{p}_1, \mathbf{R}\}$ where $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$ are the contact positions of the finger tips, $\mathbf{R}$ is the orientation of the palm. The whole set is represented by $\mathbf{g}_X{}^f$, which includes many $\mathbf{g}_X{}^f$. Namely, $\mathbf{g}_X{}^f = \{\mathbf{g}_X{}^f\}$.

Given the pose of an object on the plenary surface, say $\boldsymbol{p}_X{}^s$ and $\mathbf{R}_X{}^s$, the IK-feasible and collision-free grasps that the robot can use to pick up the object is computed following

$$\mathbf{g}_X{}^s = \mathrm{IK}\ (\mathbf{g}_X{}^{s'}) \cap \mathrm{CD}\ (\mathbf{g}_X{}^{s'},\ \textit{horizontal table surface}) \tag{2}$$

where

$$\mathbf{g}_X{}^{s'} = \mathbf{R}_X{}^s \cdot \mathbf{g}_X{}^f + \boldsymbol{p}_X{}^s \tag{3}$$

$\mathbf{R}_X{}^s \cdot \mathbf{g}_X{}^f + \boldsymbol{p}_X{}^s$ transforms the grasps at free space to the local coordinate system of the object. $\mathbf{g}_X{}^{s'}$ denotes the transformed grasp set. IK() finds the IK-feasible grasps from the input set. CD() checks the collision between the two input elements and finds the collision-free grasps. $\mathbf{g}_X{}^s$ denotes the IK-feasible and collision-free grasps that the robot can use to pick up the object.

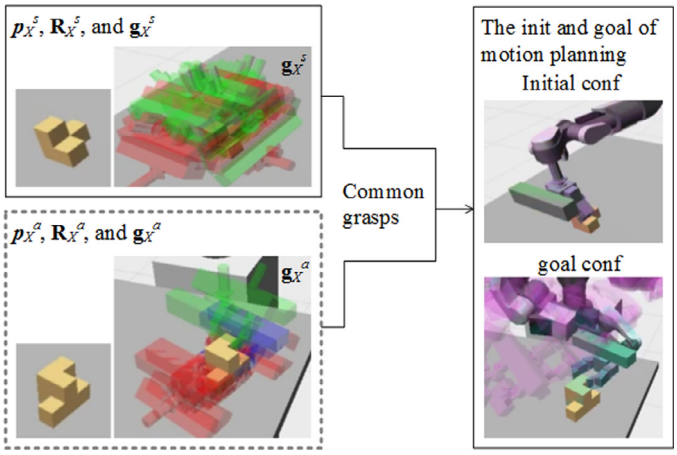


Fig. 6. The flow of motion planning. Given initial and goal poses of an object (left images in the upperleft and lowerleft frameboxes), we search its feasible initial and goal grasps and use the common grasps and IK to get the initial and goal configurations of the robot arm. Then, we do motion planning repeatedly between the initial and goal configurations to find a solution to the desired task.

Likewise, given the pose of object A in the assembled structure, say $\boldsymbol{p}_A{}^a$ and $\mathbf{R}_A{}^a$, the IK-feasible and collision-free grasps that the robot can use to assemble it is computed following

$$\mathbf{g}_A{}^a = \mathrm{IK}\ (\mathbf{g}_A{}^{a'}) \cap \mathrm{CD}\ (\mathbf{g}_A{}^{a'}, \mathrm{objB}(\boldsymbol{p}_B{}^a, \mathbf{R}_B{}^a)) \tag{4}$$

where

$$\mathbf{g}_A{}^{a'} = \mathbf{R}_A{}^a \cdot \mathbf{g}_A{}^f + \boldsymbol{p}_A{}^a \tag{5}$$

$\mathbf{R}_A{}^a \cdot \mathbf{g}_A{}^f + \boldsymbol{p}_A{}^a$ transforms the grasps at free space to the local coordinate system of the object in the assembled structure. $\mathbf{g}_X{}^{a'}$ denotes the transformed grasp set. IK() and CD() are the same as those in Eq. (2). objB($\boldsymbol{p}_B{}^a$, $\mathbf{R}_B{}^a$) indicates the mesh model of object B at pose $\boldsymbol{p}_B{}^a$, $\mathbf{R}_B{}^a$. $\mathbf{g}_A{}^a$ denotes the IK-feasible and collision-free grasps that the robot can use to assemble the object.

6.2. Motion planning

In the motion planning step, we build a graph using the elements in $\mathbf{g}_X{}^s$ and $\mathbf{g}_X{}^a$, search the grasp to find high-level keyposes, and perform Transition-based Rapidly-Exploring Random Tree (Transition-RRT) [64] motion planning between the keyposes to find assemble motions.

Fig. 6shows the flow. The object X in this graph is a wooden block shown in the left image of the upper-left frame box. The image also shows the pose of this object on a horizontal table surface, $\boldsymbol{p}_X{}^s$ and $\mathbf{R}_X{}^s$. When the object is assembled in the structure, its pose $\boldsymbol{p}_X{}^a$ and $\mathbf{R}_S{}^a$ is shown in the left image of the bottom-left frame box. The grasps associated with the poses are shown in the right images of the two frame boxes. They are rendered using different colors: Green denotes collision-free and IK-feasible grasps. Blue denotes grasps with infeasible IK. Red denotes collided grasps. We build a graph to find the common green grasps and employ Transition-RRT to find a motion between the initial configuration and goal configuration.

In practice, the flow in Fig. 6 doesn't work since the goal configuration is in the assembled structure and is in the narrow passages or on the boundaries of configuration space. The motion planning problem is a narrow-passage [65] or peg-in-hole problem [66] which is difficult to solve. We overcome the difficulty by adding a pre-assemble configuration: For the two objects A and B, we retract them from the structure following the approaching direction $\boldsymbol{v}^a$ of the two objects and get the pre-assemble poses $\boldsymbol{p}_A{}^p$, $\mathbf{R}_A{}^p$, and $\boldsymbol{p}_B{}^p$, $\mathbf{R}_B{}^p$ where

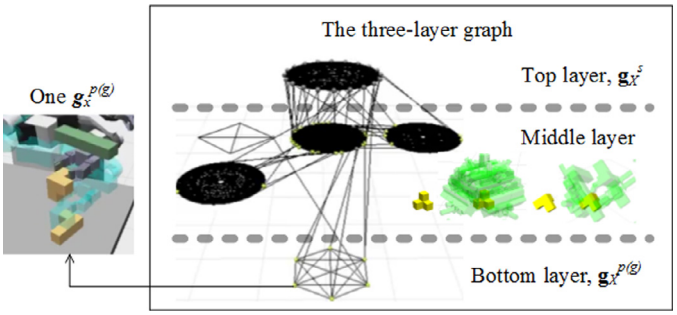


Fig. 7. The grasp graph built using $\mathbf{g}_X{}^s$ and $\mathbf{g}_X{}^{p(g)}$. It has three layers where the top layer encodes the grasps associated with the initial configuration, the middle layer encodes the grasps associated with placements on horizontal table surfaces, and the bottom layer encodes the grasps associated with the assemble pose. The left image shows one $\mathbf{g}_X{}^p(g)$ (the virtual grasp illustrated in cyan). It corresponds to a node in the bottom layer. The subimages in the frame box illustrate the placements (yellow) and their associated grasps (green). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$$\boldsymbol{p}_A{}^p = \boldsymbol{p}_A{}^a + 0.5\boldsymbol{v}^a, \ \ \mathbf{R}_A{}^p = \mathbf{R}_A{}^a \tag{6}$$

$$\boldsymbol{p}_B{}^p = \boldsymbol{p}_B{}^a - 0.5\boldsymbol{v}^a, \ \ \mathbf{R}_B{}^p = \mathbf{R}_B{}^a \tag{7}$$

The grasps associated with the retracted poses are

$$\mathbf{g}_A{}^p = \mathrm{IK}\ (\mathbf{g}_A{}^{p'}), \ \text{where}\ \mathbf{g}_A{}^{p'} = \mathbf{g}_A{}^a + 0.5\boldsymbol{v}^a \tag{8}$$

$$\mathbf{g}_B{}^p = \mathrm{IK}\ (\mathbf{g}_B{}^{p'}), \ \text{where}\ \mathbf{g}_B{}^{p'} = \mathbf{g}_B{}^a - 0.5\boldsymbol{v}^a \tag{9}$$

Note that the poses in Eq. (6)–(9) are in the local coordinate of object A where $\boldsymbol{p}_A{}^a$ is a zero vector and $\boldsymbol{R}_A{}^a$ is an identity matrix. Given the pose of object A in world coordinate, $\boldsymbol{p}_A{}^{a(g)}$ and $\boldsymbol{R}_A{}^{a(g)}$, the grasps in the world coordinate are computed using

$$\mathbf{g}_A{}^{a(g)} = \mathbf{g}_A{}^{p(g)} \tag{10}$$

$$\mathbf{g}_B{}^{a(g)} = \boldsymbol{p}_A{}^{a(g)} + \boldsymbol{R}_A{}^{a(g)} \cdot \mathbf{g}_B{}^a \tag{11}$$

$$\mathbf{g}_A{}^{p(g)} = \boldsymbol{p}_A{}^{a(g)} + \boldsymbol{R}_A{}^{a(g)} \cdot \mathbf{g}_A{}^p \tag{12}$$

$$\mathbf{g}_B{}^{p(g)} = \boldsymbol{p}_A{}^{a(g)} + \boldsymbol{R}_A{}^{a(g)} \cdot \mathbf{g}_B{}^p \tag{13}$$

The motion planning is then to find a motion between one initial configuration in $\mathbf{g}_X{}^s$ to a goal configuration in $\mathbf{g}_X{}^p(g)$ where X is either A or B. There is no motion between $\mathbf{g}_A{}^p(g)$ and $\mathbf{g}_A{}^a(g)$ since they are equal to each other. The motion between $\mathbf{g}_B{}^p(g)$ and $\mathbf{g}_B{}^a(g)$ is hard coded along $\boldsymbol{v}^a$.

Which initial and goal configuration to use is decided by building and searching a grasp graph which is built using $\mathbf{g}_X{}^s$ and $\mathbf{g}_X{}^{p(g)}$. It is shown in the frame box of Fig. 7. The graph is basically the same as [63], but has three layers. The top layer has only one circle and is mapped to the initial configuration. The bottom layer also has only one circle and is mapped to the goal configuration. The middle layers are composed of several circles where each of them maps a stable placement on a plenary surface. Each node of the circles represents a grasp: The ones in the upper layers are from $\mathbf{g}_X{}^s$, and the ones in the bottom layers are from $\mathbf{g}_X{}^{p(g)}$). The ones from the middle layers are the grasps associated with the stable placements. The orientations of the placements are evenly sampled online. Their positions are fixed to the initial position $\mathbf{p}_A{}^s$. If the circles share some grasps (grasps with the same $\boldsymbol{p}_0$, $\boldsymbol{p}_1$, $\mathbf{R}$ values in the object's local coordinate system), we connect them at the correspondent nodes. We search the graph to find the initial and goal configurations and a sequence of high-level keyposes, and perform motion planning between the keyposes to find robot motions. An exemplary result will be shown in the experiment section.

7. Experiments and analysis

Experiments are performed to examine the precision of the developed approaches, analyze the process of grasp and motion planning, and demonstrate the applicability of the study using a Kawada Nextage Robot. The camera used in the human teaching phase is a logicool HD Webcam C615. The computer system is a Lenovo Thinkpad E550 laptop (Processor: Intel Core i5-5200 2.20GHz Clock, Memory: 4G 1600 MHz DDR3). The depth sensor used in the robotic execution phase is Kinect. The computer system used to compute the grasps and motions is a Dell T7910 workstation (Processor: Intel Xeon E5-2630 v3 with 8CHT, 20 MB Cache, and 2.4 GHz Clock, Memory: 32 G 2133 MHz DDR4).

7.1. Precision of the object detection in human teaching

First, we examine the precision of object detection in human teaching. We use two sets of objects and examine the precision of five assembly structures for each set. Moreover, for each assembly structure, we examine the values of at five different orientations.

The two sets and ten structures (five for each) are shown in the first row of Fig. 8. Each structure is posed at five different orientations to examine the precision. The five data rows under the subimage row in Fig. 8 are the result of different orientations. Each grid of the data rows is shown in the form $\Delta d(\Delta r, \Delta p, \Delta y)$ where Δd is the difference in the measured $|\boldsymbol{p}_A{}^a - \boldsymbol{p}_B{}^a|$ and the actual value. $(\Delta r, \Delta p, \Delta y)$ are the difference in the measured *roll, pitch*, and *yaw* angles. The last row of the figure shows the average detection error of each structure in the form $|\Delta d|(|\Delta r|, |\Delta p|, |\Delta y|)$ where $|\cdot|$ indicates the absolute value. The metrics are *millimeter* (*mm*) for distance and *degree* (°) for orientation. The precision in position is less than 1*mm* and the precision in orientation is less than 2° on average.

7.2. Precision of the object detection in robotic execution

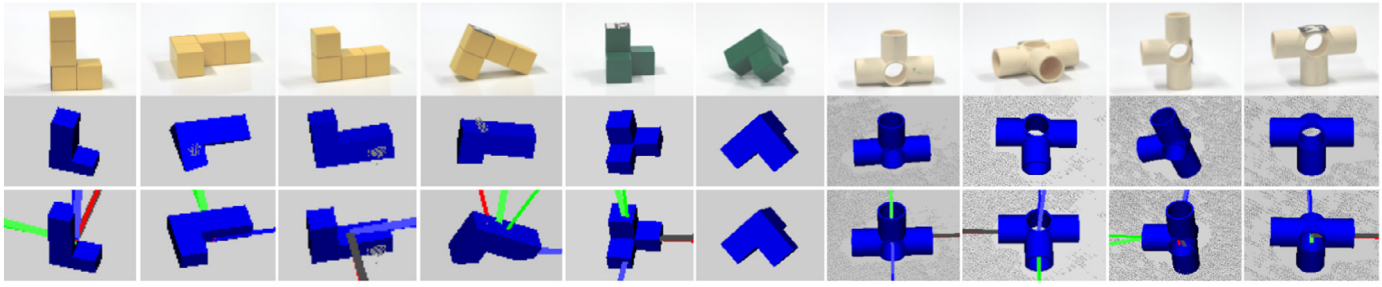
Then, we examine the precision of object detection in the robot execution phase. Three objects with eight placements are used during the process. Their real poses are shown in the top row of Fig. 9. The horizontal table surface is in front of the robot and is divided into four quarters. We place objects into each quarter to get the average values. There are five data rows divided by dashed or solid lines in Fig. 9 where the first four of them show the individual detection precision at each quarter and the last one shows the average detection precision (The difference between the detected value and the groundtruth. Since we know the exact model of the object and the height of the table, the groundtruth is known beforehand. The average detection precision is the mean of the absolute difference).

Inside each data grid there are four triples where the upper two are the roughly detected position and orientation. The lower two are the corrected values. The roughly detected results are marked in red shadows and the corrected results are marked in green. The three values of the position triples are the *x, y, z* coordinates. Their metrics are *millimeters* (*mm*). The three values of the orientation triples are the *roll, pitch, yaw* angles. Their metrics are *degree* (°).

The results show that the maximum errors of rough position detection are −1.1 *mm*, 3.3 *mm*, and 2.3 *mm* along *x, y*, and *z* axis. They are marked with red boxes in Fig. 9. After correction, the maximum position errors change to −1.1 *mm*, 3.6 *mm*, 0.0 *mm* respectively. They are marked with green boxes in Fig. 9. The maximum errors of rough orientation detection are −26.3°, 26.1°, and −19.5° in *roll, pitch*, and *yaw* angles. They are marked with red boxes. After correction, the errors change to 0.0°, 0.0°, and −21.4°. The correction using geometric constraints completely removes the errors along *z* axis and in *roll* and *pitch* angles. It might slightly

-1.4(0.5, 0.0, 2.2)	-0.2(1.0, 1.5, 1.2)	-1.2(2.0, 1.0, 0.6)	0.0(0.4, -2.0, 1.7)	0.0(0.0, -0.2, 2.0)	0.0(3.0, 2.0, 3.0)	-0.2(2.0, 3.0, -1.0)	-1.0(2.0, 1.0, 1.0)	0.5(1.0, 0.0, 1.0)	1.0(1.0, -1.0, -1.0)
0.1(1.3, 1.0, 1.0)	-0.2(1.0, 0.7, 1.0)	-3.5(1.0, 1.0, 0.5)	-1.0(0.2, -2.0, 2.0)	0.0(1.5, -2.0, 3.0)	0.0(2.0, 2.0, 3.0)	0.3(3.0, 1.0, 1.0)	-1.9(3.0, 1.0, 2.0)	-0.4(3.0, 0.0, 0.0)	1.0(3.0, 0.0, -1.0)
-0.5(1.0, 0.6, 0.3)	3.3(0.0, 1.0, 2.0)	-0.5(0.5, -1.0, 1.8)	-0.7(1.3, 0.0, 1.0)	-0.1(1.5, -2.0, 0.8)	0.0(1.0, 3.0, 3.0)	-0.3(1.0, 0.0, 0.0)	-0.4(2.0, 2.0, 1.0)	0.3(2.0, -1.0, 0.0)	-2.0(2.0, 0.0, -1.0)
-0.5(0.0, 4.0, 0.8)	-0.4(0.0, 0.0, 2.3)	-0.2(0.7, -1.0, 0.7)	-1.5(0.4, -1.5, 1.5)	2.7(1.2, -1.0, 2.2)	-0.6(1.0, 4.0, 2.0)	3.4(1.0, 4.0, -6.0)	-5.0(5.0, 2.0, 0.0)	-3.0(2.0, 0.0, 0.0)	1.0(2.0, 0.0, 0.0)
-1.4(2.0, -0.3, 0.7)	-0.5(0.0, 1.0, 0.3)	0.3(0.9, 1.0, 1.7)	-0.5(0.7, 0.0, 1.4)	-1.8(1.0, -1.0, 1.1)	-0.5(2.0, 1.0, 0.0)	3.4(2.0, 0.0, -2.0)	-2.0(1.0, 0.0, 0.0)	-2.0(3.0, 2.0, 0.0)	1.0(2.0, -1.0, 0.0)
0.7(1.0, 1.0, 1.0)	0.9(0.4, 0.8, 1.4)	1.1(1.0, 1.0, 1.0)	0.7(0.6, 1.1, 1.5)	1.0(1.0, 1.2, 2.0)	0.6(1.8, 2.4, 2.2)	1.5(1.8, 1.6, 2.0)	2.0(2.6, 1.2, 0.8)	1.2(2.2, 0.6, 0.2)	1.2(2.0, 0.4, 0.6)

Fig. 8. Results of the object detected during human teaching. The image row shows the structure to be assembled. Each structure is posed at five different orientations to examine the precision and the detected error in distance and orientation are shown in the five data rows below. Each grid of the data rows is shown in the form $\Delta d(\Delta r, \Delta p, \Delta y)$ where Δd is the difference in the measured $|\boldsymbol{p}_A{}^a - \boldsymbol{p}_B{}^a|$ and the actual value. $(\Delta r, \Delta p, \Delta y)$ are the difference in the measured roll, pitch, and yaw angles. The last row is the average detection error. The metrics are *millimeter* for distance and *degree* for orientation.



(-0.7, -2.1, 2.3)	(-0.8, -2.6, 2.1)	(-0.3, 1.9, 1.0)	(-0.5, 3.3, 0.1)	(-0.7, 1.2, -0.4)	(-1.0, 1.4, -0.3)	(-0.2, 1.1, -0.2)	(-0.4, 1.2, -0.1)	(-0.5, 1.0, 0.1)	(-0.6, 1.5, -0.1)
(-3.3, -0.1, 3.3)	(1.5, -15.7, 9.4)	(8.1, 13.4 -6.9)	(-0.9, 9.3, 8.3)	(1.1, -1.5, 2.4)	(0.6, 4.2, 10.5)	(0.5, -2.1, -6.0)	(-0.3, -0.5, 1.7)	(-2.2, -2.7, 9.0)	(-3.7, 0.7, -0.6)
(0.5, -2.0, 0.0)	(-0.8, -2.6, 0.0)	(-0.3, 1.9, 0.0)	(-0.5, 3.6, 0.0)	(-0.7, 1.2, 0.0)	(-1.0, 1.4, 0.0)	(-0.3, 1.1, 0.0)	(-0.4, 1.2, 0.0)	(-0.5, 1.0, 0.0)	(-0.6, 1.5, 0.0)
(0.0, 0.0, 2.3)	(0.0, 0.0, 9.8)	(0.0, 0.0, -6.1)	(0.0, 0.0, 8.5)	(0.0, 0.0, 2.5)	(0.0, 0.0, 10.2)	(0.0, 0.0, -6.0)	(0.0, 0.0, 1.7)	(0.0, 0.0, 9.0)	(0.0, 0.0, -0.6)
(0.1, -1.6, 1.3)	(-0.9, -2.4, 1.3)	(-1.1, 1.8, 1.5)	(0.2, 3.1, 1.1)	(-0.2, 2.1, -0.3)	(-1.0, 1.3, -0.1)	(-0.1, 1.3, 0.1)	(-0.3, 1.4, 0.0)	(0.1, 1.8, 2.1)	(-0.2, 1.0, -0.1)
(1.9, -0.3, 11.1)	(0.5, -17.3, 14.8)	(4.8, 8.7, -2.8)	(4.8, 26.1, 17.1)	(11.4, -9.7, -5.0)	(7.2, 6.7, -19.5)	(10.8, -0.4, 4.1)	(4.1, 0.5, -7.4)	(-16.2, -9.2, 9.4)	(-12.5, 2.8, -3.4)
(0.1, -1.6, 0.0)	(-0.9, -2.4, 0.0)	(-1.1, 1.8, 0.0)	(0.2, 3.1, 0.0)	(-0.2, 2.1, 0.0)	(-1.0, 1.3, 0.0)	(-0.1, 1.3, 0.0)	(-0.3, 1.4, 0.0)	(0.1, 1.8, 0.0)	(-0.2, 1.0, 0.0)
(0.0, 0.0, 11.1)	(0.0, 0.0, 14.7)	(0.0, 0.0, 2.4)	(0.0, 0.0, 14.9)	(0.0, 0.0, -3.8)	(0.0, 0.0, -15.3)	(0.0, 0.0, 4.1)	(0.0, 0.0, -7.4)	(0.0, 0.0, 17.6)	(0.0, 0.0, -2.7)
(0.4, 2.0, 1.9)	(0.2, 2.3, -0.1)	(0.0, 2.2, -0.9)	(0.7, 2.7, 1.1)	(0.5, 0.8, -0.2)	(-0.2, 1.0, -0.1)	(1.0, 1.2, -0.1)	(0.7, 1.2, -0.1)	(0.1, 1.3, 0.1)	(0.8, 0.9, -0.1)
(-0.1, 6.0, -10.2)	(1.3, -7.1, 11.4)	(-2.1, 3.7, 1.9)	(-7.4, 19.0, 12.4)	(1.0, 0.1, -2.7)	(5.1, -10.9, 3.9)	(-26.3, -0.7, -1.5)	(3.6, -1.2, 1.5)	(-3.0, 4.9, 2.3)	(-21.9, 0.2, -1.7)
(0.4, 2.0, 0.0)	(0.2, 2.3, 0.0)	(0.0, 2.2, 0.0)	(0.7, 2.7, 0.0)	(0.5, 0.8, 0.0)	(-0.2, 1.0, 0.0)	(1.0, 1.2, 0.0)	(0.7, 1.2, 0.0)	(0.1, 1.3, 0.0)	(0.8, 0.9, 0.0)
(0.0, 0.0, -10.2)	(0.0, 0.0, 11.5)	(0.0, 0.0, 1.9)	(0.0, 0.0, 14.8)	(0.0, 0.0, -2.7)	(0.0, 0.0, 0.9)	(0.0, 0.0, -1.3)	(0.0, 0.0, 1.4)	(0.0, 0.0, 1.1)	(0.0, 0.0, -1.6)
(-0.1, 1.7, -0.6)	(0.4, 2.0, -0.5)	(-0.7, 1.2, -0.5)	(0.7, 2.6, -0.5)	(0.1, 0.7, -0.4)	(-0.3, 0.8, -0.6)	(0.4, 1.0, -0.6)	(0.8, 1.4, -0.4)	(0.7, 1.2, -0.4)	(0.4, 1.1, -0.5)
(-8.5, 10.4, -19.9)	(-3.3, -3.3, 2.7)	(3.6, -2.2, 0.1)	(1.6, 21.6, 11.9)	(-4.6, 1.5, 8.0)	(-19.3, -4.9, -13.7)	(0.6, -1.3, 2.8)	(-5.5, 6.3, 11.9)	(1.5, 4.0, 3.0)	(-20.1, -0.8, 4.2)
(-0.1, 1.7, 0.0)	(0.4, 2.0, 0.0)	(-0.7, 1.2, 0.0)	(0.7, 2.6, 0.0)	(0.1, 0.7, 0.0)	(-0.3, 0.8, 0.0)	(0.4, 1.0, 0.0)	(0.8, 1.4, 0.0)	(0.7, 1.2, 0.0)	(0.4, 1.1, 0.0)
(0.0, 0.0, -21.4)	(0.0, 0.0, 2.5)	(0.0, 0.0, 0.0)	(0.0, 0.0, 11.3)	(0.0, 0.0, 8.2)	(0.0, 0.0, -2.6)	(0.0, 0.0, 2.8)	(0.0, 0.0, 11.9)	(0.0, 0.0, 4.5)	(0.0, 0.0, 3.9)
(0.3, 1.8, 1.5)	(0.6, 2.3, 1.0)	(0.5, 1.8, 1.0)	(0.5, 2.9, 0.7)	(0.4, 1.2, 0.3)	(0.6, 1.1, 0.3)	(0.4, 1.1, 0.3)	(0.6, 1.3, 0.1)	(0.4, 1.3, 0.7)	(0.5, 1.1, 0.2)
(3.5, 4.2, 11.1)	(1.7, 10.9, 9.6)	(4.6, 7.0, 2.9)	(3.7, 19.0, 12.4)	(4.5, 3.2, 4.5)	(8.0, 6.7, 12.0)	(9.6, 1.1, 3.6)	(3.4, 2.1, 5.6)	(5.7, 5.2, 5.9)	(14.6, 1.1, 2.5)
(0.3, 1.8, 0.0)	(0.6, 2.3, 0.0)	(0.5, 1.8, 0.0)	(0.5, 3.0, 0.0)	(0.4, 1.2, 0.0)	(0.6, 1.1, 0.0)	(0.4, 1.1, 0.0)	(0.6, 1.3, 0.0)	(0.4, 1.3, 0.0)	(0.5, 1.1, 0.0)
(0.0, 0.0, 11.3)	(0.0, 0.0, 9.6)	(0.0, 0.0, 2.6)	(0.0, 0.0, 12.4)	(0.0, 0.0, 4.3)	(0.0, 0.0, 7.3)	(0.0, 0.0, 3.6)	(0.0, 0.0, 5.6)	(0.0, 0.0, 8.0)	(0.0, 0.0, 2.2)

Fig. 9. Results of the object detected during robotic execution. The figure includes three image rows and five data rows where the first data row show the target object pose, the second and third image rows plot some examples of the roughly detected poses and the corrected poses. The five data rows are divided by dashed or solid lines where the first four of them show the individual detection precision at four different positions and the last one shows the average detection precision. Each data grid of the data rows include four triples where the upper two (under red shadow) are the roughly detected position and orientation and the lower two (under green shadow) are the corrected values. The three values of the position triples are the *x, y, z* coordinates. Their metrics are *millimeters* (*mm*). The three values of the orientation triples are the *roll, pitch, yaw* angles. Their metrics are *degree* (°). The maximum values of each data element are marked with colored frameboxes. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

increase the errors along *x* and *y* and in *yaw*. The average performance is shown in the rows under the double solid line of Fig. 9.

In addition, the second and third image rows of Fig. 9 show some examples of the roughly detected poses and the corrected poses. Readers may compare them to better understand the data rows.

7.3. Simulation and real-world results

Fig. 10 shows the correspondence between the paths found by the searching algorithms and the configurations of the robot. The structure to be assembled in this task is the one shown in the first figure of Fig. 8. Motion planning is done repeatedly along the paths found by the searching algorithms.

The assembly process is divided into two step where in each step the robot manipulates one object. In the first step, the robot finds object A on the table and moves it to a goal pose by searching the three-layer graph. The subfigures (1)–(4) of Fig. 10 shows this step. In Fig. 10(1), the robot computes the grasps associated with the initial pose and goal pose of object A. The associated grasps are shown in green, blue, and red colors like Fig. 1. They correspond to the nodes in the top and bottom layers of the graph shown in Fig. 10(1′). In Fig. 10(2), the robot chooses one feasible (IK-feasible and collision-free) grasp from the associated grasps and does motion planning to pick up the object. The selected grasp corresponds to one node in the top layer of the graph, which is marked with red color in Fig. 10(2′). In Fig. 10(3), the robot picks up object A and transfers it to the goal pose using a

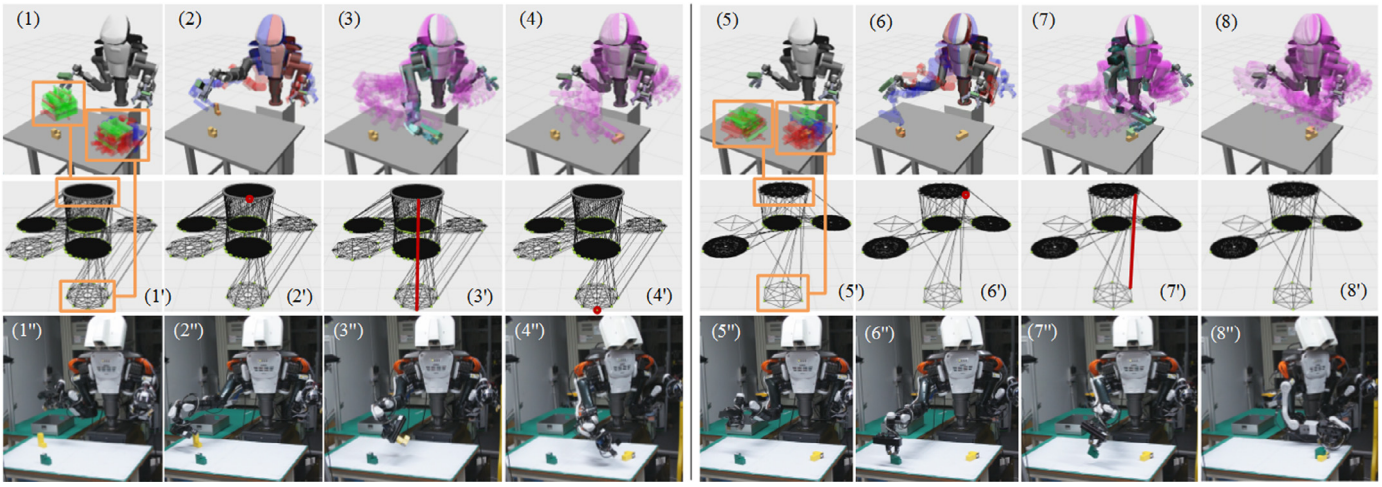


Fig. 10. The snapshots of assembling the structure shown in Fig. 8. It is divided into two step with the first step shown in (1)–(4) and the second step shown in (5)–(8). In the first step, the robot picks up object A and transfers it to the goal pose. In the second step, the robot finds object B on the table and assembles it to object A. The subfigures (1′)–(8′) shows correspondent path edges and nodes on the three-layer graph. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article.)

second motion planning. This corresponds to an edge in Fig. 10(3′) which connects the node in one circle to the node in another. The edge directly connects to the goal in this example and there is no intermediate placements. After that, the robot moves its arm back at Fig. 10(4), which corresponds to a node in the bottom layer of the graph shown in Fig. 10(4′).

In the second step, the robot finds object B on the table and assembles it to object A. The subfigures (5)–(8) Fig. 10(b) show it. In Fig. 10(5), the robot computes the grasps associated with the initial pose and goal pose of object B. They are rendered in green, blue, and red colors like Fig. 10(1) and are correspondent to the top and bottom layer of the grasp shown in Fig. 10(5′). In Fig. 10(6), the robot chooses one feasible grasp and does motion planning to pick up the object. The selected grasp corresponds to the marked node in Fig. 10(6′). In Fig. 10(7), the robot picks up object B and assembles it to the goal pose using a second motion planning which corresponds to an edge in Fig. 10(7′). Finally, the robot moves its arm back at Fig. 10(8) and (8′).

The subfigures (1′′)–(8′′) in the third row show how the robot executes the planned motion. They correspond to (1)–(8) and (1′)–(8′) in the first two rows.

8. Conclusions and future work

We presented precise 3D visual detection approaches in this paper which fulfills the requirements of a smart assembly system for next generation industrial manufacturing. The approaches include two phases. In a human teaching phase where human beings control the operation and actively avoid occlusion, AR markers are used to find the poses of the objects. In a robot execution phase where occlusions happen unexpectedly, point clouds are used to find a rough pose and geometric constraints are used to correct the noises. The precision of the approaches are examined using a graph model and an industrial robot in the experiment part which demonstrates that the precision fulfills assembly tasks.

The future work will be on the manipulation and assembly aspect. The current result is kinematics-based assembly. It will be extended to force-based assembly tasks like inserting, snapping, etc., in the future.

Acknowledgment

This paper is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO). It is also partially supported by the NSFC (No. 61602020, U1533129).

References

[1] T. Lozano-Perez, J.L. Jones, E. Mazer, P.A. O'Donnell, HANDEY: A Robot Task Planner, The MIT Press, Boston, USA, 1992.
[2] M.T. Mason, Mechanics of Robotic Manipulation, The MIT Press, Boston, USA, 2001.
[3] M. Dogar, A. Spielberg, S. Baker, D. Rus, Multi-robot grasp planning for sequential assembly operations, Proceedings of IEEE International Conferene on Robotics and Automation, 2015, pp. 193–200.
[4] C. Goldfeder, Data-Driven Grasping, Columbia University, New York, USA, 2002 Ph.D. thesis.
[5] I. Lenz, H. Lee, A. Saxena, Deep learning for detecting robotic grasps, Int. J. Robotics Res. 34 (2014) 705–724.
[6] H. Murase, S.K. Nayar, Visual learning and recognition of 3d objects from appearance, Int. J. Comput. Vis. 14 (1995) 5–24.
[7] P. Mittrapiyanuruk, G.N. Desouza, A.C. Kak, Calculating the 3D-pose of rigid-objects using active appearance models, in: Proceedings of the IEEE International Conference on Robotics and Automation, 2004.
[8] S. Zickler, M. Veloso, Detection and localization of multiple objects, in: Proceedings of IEEE-RAS International Conference on Humand Robots, 2006.
[9] M. Stark, M. Goesele, B. Schiele, Back to the future: learning shape models from 3D CAD data, in: Proceedings of British Machine Vision Conference, 2011.
[10] J. Liebelt, C. Schmid, Multi-view object class detection with a 3D geometric model, in: Proceedings of Computer Vision and Pattern Recognition, 2010.
[11] P. Wunsch, G. Hirzinger, Registration of CAD models to images by iterative inverse perspective matching, in: Proceedings of International Conference on Pattern Recognition, 1996.
[12] P. David, D. DeMenthon, R. Duraiswami, H. Samet, SoftPOSIT: Simultaneous pose and correspondence determination, in: Proceedings of European Conference on Computer Vision, 2002.
[13] M.A. Fischler, R.C. Bolles, Random sample consensus: a paradigm for model fitting and applications to image analysis and automated cartography, Assoc. Comput. Mach. 24 (1981) 381–395.
[14] D. DeMenthon, L.S. Davis, Model-based object pose in 25 lines of code, Int. J. Comput. Vis. 15 (1995) 123–141.
[15] C.-P. Lu, G.D. Hager, E. Mjolsness, Fast and globally convergent pose estimation from video images, IEEE Trans. Pattern Anal. Mach. Intell. 22 (2000) 610–622.
[16] F. Schaffalitzky, A. Zisserman, Multi-view matching for unordered image sets, or "How do i organize my holiday snaps?", in: Proceedings of European Conference on Computer Vision, 2002.
[17] C.J. Harris, A combined corner and edge detector, in: Proceedings of Alvey Vision Conference, 1988.
[18] K.W. Chia, A.D. Cheok, S.J. Prince, Online 6-DOF augmented reality registration from natural features, in: Proceedings of European Conference on Computer Vision, 2002.

[19] P. David, D. DeMenthon, R. Duraiswami, H. Samet, Simultaneous pose and correspondence determination using line features, in: Proceedings of Computer Vision and Pattern Recognition, 2003.
[20] G. Klein, T. Drummond, Robust visual tracking for non-instrumented augmented reality, in: Proceedings of International Symposium on Mixed and Augmented Reality, 2003.
[21] E. Marchand, F. Chaumette, Virtual visual servoing: A framework for real-time augmented reality, in: Proceedings of Eurographic, 2002.
[22] K. Harada, K. Nagata, T. Tsuji, N. Yamanobe, A. Nakamura, Y. Kawai, Probabilistic approach for object bin picking approximated by cylinders, in: Proceedings of IEEE International Conference on Robotics and Automation, 2013.
[23] D.G. Lowe, Distinctive image features from scale-invariant key-points, Int. J. Comput. Vis. 60 (2004) 91–110.
[24] I. Gordon, D.G. Lowe, What and where: 3D object recognition with accurate pose, Lect. Notes Comput. Sci. 4170 (2006) 67–82.
[25] A. Collet, D. Berenson, S.S. Srinivasa, D. Ferguson, Object recognition and full pose registration from a single image for robotic manipulation, in: Proceedings of IEEE International Conference on Robotics and Automation, 2009.
[26] W. Press, S. Teukolsky, W. Vetterling, Numerical Recipes in C: The Art of Scientific Computing, The Cambridge Press, New York, USA, 1992.
[27] Y. Cheng, Mean shift, mode seeking, and clustering, IEEE Trans. Pattern Anal. Mach. Intell. 17 (1995) 790–799.
[28] A. Ramisa, D. Aldavert, S. Vasudevan, R. Toledo, R.L. de Mantaras, Evaluation of three vision based object perception methods for a mobile robot, J. Intell. Robotic Syst. 68 (2012) 1–24.
[29] M. Fiala, ARTag: a fiducial marker system using digital techniques, in: Proceedings of Computer Vision and Pattern Recognition, 2005.
[30] D. Wagner, D. Schmalstieg, ARToolKitPlus for pose tracking on mobile devices, in: Proceedings of Computer Vision Winter Workshop, 2007.
[31] V. Sundareswaran, R. Behringer, Visual servoing-based augmented reality, in: Proceedings of International Workshop on Augmented Reality, 1998.
[32] H. Kato, M. Billinghurst, Marker tracking and HMD calibration for a video-based augmented reality conferencing system, in: Proceedings of International Workshop on Augmented Reality, 1999.
[33] L. Vacchetti, V. Lepetit, P. Fua, Stable real-time 3D tracking using online and offline information, IEEE Trans. Pattern Anal. Mach. Intell. 26 (2004) 1385–1391.
[34] S. Makita, K. Okita, Y. Maeda, Motion planning for 3D multifingered caging with object recognition using AR picture markers, in: Proceedings of IEEE International Conference on Mechatronics and Automation, 2015.
[35] F. Suligoj, B. Sekoranja, M. Svaco, bojan Jerbic, Object tracking with a multiagent robot system and a stereo vision camera, Procedia Eng. 69 (2014) 968–973.
[36] B. Wei, T. Guan, L. Duan, J. Yu, T. Mao, Wide area localization and tracking on camera phones for mobile augmented reality systems, Multimedia Syst. 21 (2015) 381–399.
[37] K. Ramirez-Amaro, M. Beetz, G. Cheng, Transferring skills to human robots by extracting semantic representations from observations of human activities, Artif. Intell. In Press (2015). Available Online http://www.sciencedirect.com/science/article/pii/S0004370215001320
[38] B. Freedman, A. Shpunt, M. Machline, Y. Arieli, Google Patents: Depth Mapping using Projected Patterns, 2012. Publication number: US20080240502 A1.
[39] S.M. Choi, E.G. Lim, J.I. Cho, D.H. Hwang, Google Patents: Stereo Vision System and Stereo Vision Processing Method, 2012. Publication number: US20090060280 A1.
[40] P.E. Bauhahn, B.S. Fritz, B.C. Krafthefer, Google Patents: Systems and Methods for Safe Laser Imaging, 2009. Publication number: US20090273770 A1.
[41] R.B. Rusu, S. Cousins, 3D is here: point cloud library (PCL), in: Proceedings of IEEE International Conference on Robotics and Automation, 2011.
[42] C. Schutz, H. Hugli, Augmented reality using range images, in: SPIE Photonics West, The Engineering Reality of Virtual Reality, 1997.
[43] P.J. Besl, N.D. McKay, A method for registration of 3-D shapes, IEEE Trans. Pattern Anal. Mach. Intell. 14 (1995) 239–256.
[44] F. Tombari, S. Salti, L.D. Stefano, Unique signatures of histograms for local surface description, in: Proceedings of European Conference on Computer Vision, 2010.
[45] Z.C. Marton, D. Pangercic, N. Blodow, M. Beetz, Combined 2D-3D categorization and classification for multimodal perception systems, Int. J. Robotic Res. 30 (2011) 1378–1402.
[46] A. Aldoma, et al., CAD-model recognition and 6DOF pose estimation using 3D cues, in: IEEE Int. Conf. Comput. Vis. Workshops, 2011.
[47] W. Wohlkinger, M. Vincze, Ensemble of shape functions for 3D object classification, in: Proceedings of IEEE International Conference on Robotics and Biomimetics, 2011.
[48] A. Aldoma, Z.C. Maron, F. Tombari, M. Vincze, Tutorial: Point cloud library: three-dimensional object recognition and 6 DOF pose estimation, IEEE Robotics Autom. Mag. 19 (2012) 80–91.
[49] Y. Shiraki, K. Nagata, N. Yamanobe, A. Nakamura, K. Harada, D. Sato, D.N. Nenchev, Modeling of everyday objects for semantic grasp, in: Proceedings of International Symposium on Robot and Human Interactive Communication, 2014.
[50] M.J. Schuster, J. Okerman, H. Nguyen, J.M. Rehg, C.C. Kemp, Perceiving clutter and surface for object placement in indoor environment, in: Proceedings of IEEE-RAS International Conference on Humanoid Robots, 2010.
[51] G. Smonath, M. Rohith, D. Metaxas, C. Kambhamettu, D-Clutter: building object model library from unsupervised segmentation of cluttered scenes, in: Proceedings of Computer Vision and Pattern Recognition, 2010.
[52] M. Saval-Calvo, J. Azorin-Lopez, A. Fuster-Guillo, J. Garcia-Rodriguez, Three-dimensional planar model estimation using multi-constraint knowledge based on K-means and RANSAC, Appl. Soft Comput. 34 (2015) 572–586.
[53] L. Goron, Z.-C. Marton, G. Lazea, M. Beetz, Robustly segmenting cylindrical and box-like objects in cluttered scenes using depth camera, in: Proceedings of German Conference on Robotik, 2012.
[54] E.C. Cheung, C. Cao, J. Pan, Multi-contour initial pose estimation for 3D registration, in: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, 2015.
[55] G. Taylor, L. Kleeman, Fusion of multimodal visual cues for model-based object tracking, in: Proceedings of Australasian Conference on Robotics and Automation, 2003.
[56] T. Kempter, A. Wendel, H. Bischof, Online model-based multi-scale pose estimation, in: Proceedings of Computer Vision Winter Workshop, 2012.
[57] G. Reitmayr, T. Drummond, Going Out: Robust model-based tracking for outdoor augmented reality, in: Proceedings of International Symposium on Mixed and Augmented Reality, 2006.
[58] Y. Zhang, T. Guan, L. Duan, B. Wei, J. Gao, T. Mao, Inertial sensors supported visual descriptors encoding and geometric verification for mobile visual location recognition applications, Signal Process. 112 (2015) 17–26.
[59] D. Pangercic, V. Haltakov, M. Beetz, Fast and robust object detection in household environments using vocabulary trees with sift descriptors, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Workshop on Active Semantic Perception and Object Search in the Real World, 2011.
[60] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, V. Lepetit, Multimodal Templates for real-time detection of texture-less objects in heavily cluttered scenes, in: Proceedings of IEEE International Conference on Computer Vision, 2011.
[61] H. Pan, T. Guan, Y. Luo, L. Duan, Y. Tian, L. Yi, Y. Zhao, J. Yua, Dense 3D reconstruction combining depth and RGB information, Neurocompputing 175 (2016) 644–651.
[62] Y. Domae, D. Okuda, Y. Taguchi, K. Sumi, T. Hirai, Fast graspability evaluation on single depth maps for bin picking with general grippers, in: Proceedings of IEEE International Conference on Robotics and Automation, 2014.
[63] W. Wan, K. Harada, Developing and comparing single-arm and dual-arm regrasp, IEEE Robotics Autom. Lett. 1 (2016) 243–250.
[64] L. Jaillet, J. Cortes, T. Simeon, Transition-based RRT for path planning in continuous cost spaces, in: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008.
[65] H. Liu, D. Ding, W. Wan, Predictive model for path planning using K-near dynamic brdige builder and inner Parzen window, in: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008.
[66] S. kook Yun, Compliant manipulation for peg-in-hole: is passive compliance a key to learn contact motion, in: Proceedings of IEEE International Conference on Robotics and Automation, 2008.

Weiwei Wan received his Ph.D. degree from the Department of Mechano-Informatics, Graduate School of Information Science and Engineering, the University of Tokyo, Japan, in 2013. He worked as a postdoctoral research fellow of the Japanese Society for the Promotion of Science (JSPS), Japan and was a visiting scholar at Carnegie Mellon University, USA. Since 2015, he has been a research scientist at the National Institute of Advanced Industrial Science and Technology (AIST). His research insterest includes robotic grasping and manipulation planning for next-generation manufacutring. He is a member of IEEE and is the winner of the IEEE RAS Japan Chapter Young Award.

Feng Lu received the B.Sc. and M.Sc. degrees in automation from Tsinghua University, in 2007 and 2010, respectively, and the Ph.D. degree in information science and technology from The University of Tokyo, in 2013. After working with the Institute of Industrial Science, the University of Tokyo, he joined the State Key Laboratory of Virtual Reality Technology and Systems, School of Computer Science and Engineering, Beihang University, China, in 2015. His research interests include shape recovery, reflectance analysis, and human gaze estimation.

Zepei Wu received his the B.Sc.degree in mechanical engineering from Hohai University, China, in 2014. He is currently working toward the M.Sc. degree with the the Tukuba University. Since 2015, he has been a visiting student at the National Institute of Advanced Industrial Science and Technology (AIST). His research interest is robotic manipulation planning and control.

Kensuke Harada received his B.Sc., M.Sc., and Doctoral degrees in Mechanical Engineering from Kyoto University in 1992, 1994, and 1997, respectively. He worked as a Research Associate at Hiroshima University from 1997 to 2002. From 2002, he has been working at the National Institute of Advanced Industrial Science and Technology (AIST). For one year from 2005 to 2006, he was a visiting scholar at the computer science department of Stanford University, Currently, he is a leader of the vision and manipulation research group. His research interest includes mechanics and control of robot manipulators and robot hands, biped locomotion, and motion planning of robotic systems. He is a member of IEEE, JSME, RSJ, and SICE.