Multimedia Tools and Applications
https://doi.org/10.1007/s11042-019-08302-9

Robotic grasp detection based on image processing and random forest

Jiahao Zhang[1] · Miao Li[2] · Ying Feng[1] · Chenguang Yang[3]

Received: 6 March 2019 / Revised: 6 September 2019 / Accepted: 30 September 2019 /
Published online: 21 November 2019

Abstract
Real-time grasp detection plays a key role in manipulation, and it is also a complex task, especially for detecting how to grasp novel objects. This paper proposes a very quick and accurate approach to detect robotic grasps. The main idea is to perform grasping of novel objects in a typical RGB-D scene view. Our goal is not to find the best grasp for every object but to obtain the local optimal grasps in candidate grasp rectangles. There are three main contributions to our detection work. Firstly, an improved graph segmentation approach is used to do objects detection and it can separate objects from the background directly and fast. Secondly, we develop a morphological image processing method to generate candidate grasp rectangles set which avoids us to search grasp rectangles globally. Finally, we train a random forest model to predict grasps and achieve an accuracy of 94.26%. The model is mainly used to score every element in our candidate grasps set and the one gets the highest score will be converted to the final grasp configuration for robots. For real-world experiments, we set up our system on a tabletop scene with multiple objects and when implementing robotic grasps, we control Baxter robot with a different inverse kinematics strategy rather than the built-in one.

Keywords Real-time grasp detection · Novel objects · Improved graph segmentation method · Morphological image processing · Random forest · Baxter

1 Introduction

Robotic grasping in an unknown scene involves environmental perception, motion planning, and some issues about robot control. In this paper, we mainly concentrate on the perception problem and partially about robot control with robot inverse kinematics. Perception is a necessary skill for robot grippers to interact with environments. The motivation of visual

✉ Chenguang Yang
cyang@ieee.org

[1] The Key Laboratory of Autonomous Systems and Networked Control, College of Automation Science and Engineering, South China University of Technology, Guangzhou, China

[2] The Institute of Technological Sciences, Wuhan University, Wuhan, China

[3] Bristol Robotics Laboratory, University of the West of England, Bristol, BS16 1QY, UK

perception in the research of robots is to recognize the correct poses to grasp objects. It is an easy task for human beings to identify the perfect grasp poses for novel objects. However, a visual task like grasp detection for unknown objects on a given image is a complex problem for robots in recent years.

For general use of robots, detecting grasps of unknown objects fast and accurately on a tabletop scene is indispensable. The main challenges for robotic grasp are:

- **How to do object detection:** If there are a pile of objects, it is important to figure out where they are and separate them to find individual grasp configuration for each object.
- **How to decide the best grasp configurations:** For each object, the best way to grab it is to find the proper pose, then approach it and close the gripper.
- **How to control robot move to our desired pose:** When we decide to grasp objects, the control strategy can ensure the success rate when grasping the same object at the same configuration.

With the incredible development of machine learning and deep learning methods, they are widely used in computer vision tasks such as objects classification, objects detection, and localization [12, 35–37, 39]. Their results show that deep learning can achieve high accuracy of classification and precision of localization. Inspired by that, recent works try to solve the grasp detection problem by a convolutional neural network (CNN) [34], deep residual network (ResNet) [3] and other similar deep neural networks [18, 25]. All their works utilized a famous human-made dataset - the Cornell Dataset which consists of 885 images (both RGB data and depth map included) with different types of objects and the corresponding labeled rectangles, the labeled rectangles have two parts: positive and negative (Samples in Fig. 1).

For deciding the best grasp configuration, approaches in [13, 26, 28, 29] shows that machine learning methods like SVM and deep learning methods like CNN have great power to determine whether it can be graspable.

In this paper, we propose a machine-learning algorithm to decide the best grasp configuration and improve a graph segmentation method to localize objects based on our previous work [41]. The key advantages of the system are:

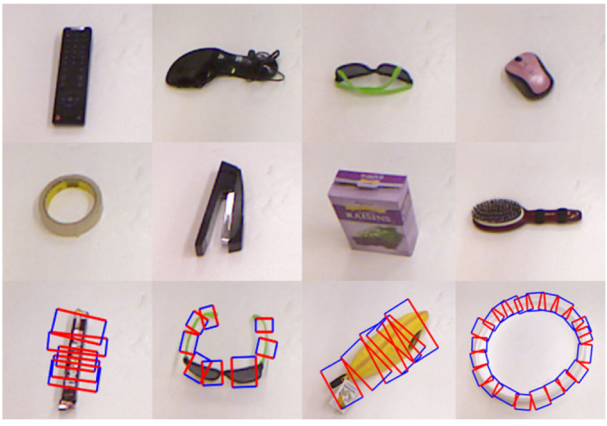

Fig. 1 Examples from the Cornell Grasp Dataset. There are many different kinds of objects. For each object in the bottom, some positive grasp rectangles in the dataset are shown in the bottom

- We use a graph segmentation algorithm to localize objects rather than deep learning methods. Compared to deep learning methods, graph segmentation method does not need to collect large amounts of data, train model and test, which makes our image pre-processing task simpler. And We also use a background mask to reduce the time on calculating the boundary between two regions.
- Unlike the sliding window approach which overlooks the area around tabletop objects, morphological image processing tries to locate the graspable area around the object itself and uses an iterative method to generate a candidate grasp set. Our results show that by randomly choose a grasp from the candidate grasp set, we can also achieve a 79.12% grasp success rate. And an important part is that there is no machine learning methods is employed.
- We train a random forest (RF) model on the Cornell Grasp Dataset and get a high accuracy around 94.26%. Intending to predict the optimal grasp in our candidate grasp set rather than searching globally, it runs efficiently with a speed of 28 fps when we implement grasp detection.
- For Baxter robot, the built-in inverse kinematic (IK) algorithm can crash, when the object is hard to be reached. We change the strategy with a numerical inverse kinematic method using the Jacobian matrix of the robot.

Our works focus on implementing robotic grasp detection in real-time and find a good configuration to grasp objects. And the results of our experiments show that our algorithm is really fast to detect a good grasp for novel objects.

In the next section, we will introduce recent works about graph segmentation and robotic grasp. Section 3 introduces our entire system and explains how each part of our system works in detail. Section 4 is the detailed introduction for our experiments. This section shows all the results in our grasp pipeline. Comparison results with other state-of-the-art also listed. Finally, we make our conclusions in Section 5.

2 Related work

The problem of graspable feature detection still needs to be explored. A lot of researchers try to find a solution to this. Old approaches in [2, 7, 20] wanted to solve the grasping problem by form-closure or force-closure grasp. While these types of approach can grasp some objects stably, it overlooked the facts that it was unrealistic to obtain the full model of objects in the real situation. Until recent years, with the development of RGB-D sensor, researchers can get a high-resolution image, as the work in [33], an active triangulation sensor was used and with the accurate information, Deepak et al. successfully used graph segmentation and a supervised localization algorithm to separate the graspable and ungraspable regions, it didn't rely on the shape of objects, and that made their algorithm robust to grasp objects. To get the full model of objects, there also have lots of great works: 3D grasp simulation using GraspIt or OpenRAVE had been made by [9–11, 23, 24]. OpenRAVE [6] is a platform to simulate and analyze geometric information of objects and robotic kinematics. Through simulation, researchers can save a lot of time from collecting data and be more convenient to do experiments. Also, it is very helpful for validating the effectivities of different algorithms. In [9–11], they trained the grasp classifier through simulation. They also further developed a weighting system to make graspable judgments located at every point. That made their works reliable and easy to generalize to more complex system equipped with diverse hands. Another example is similar to 3D reconstruction. Huaman [15] and

Makhal et al. [22], tried to find the geometry representation of every object. They mirrored the incomplete point cloud obtained from the sensor to approximate the real point cloud of objects. Apparently, it was not accurate and processed all the point cloud for every object was time-consuming. Most of the previously introduced methods failed in real time and not so effective to detect the feasible grasps of novel objects.

With the great power of machine learning in classification, researchers find that novel objects grasp detection can be classified into two parts, which is graspable or ungraspable. SVM has been widely used in grasp feature classification [11, 28]. Pas and Platt [28] used two sensors to generate a two-view registered cloud, and then they used it to generate a grasp dataset. Their methods are that first, they sampled lots of hand configuration as the input features, and then they labeled every hand configuration through judging whether this configuration satisfied antipodal grasp. By their labeled dataset, they can train their classifier to detect graspable features. And deep learning methods are also been applied in grasp detection as a classifier since [19] implemented a stacked autoencoder (SAE) and ran at 13.5 seconds per frame with an accuracy of 93.7%. SAE is time-consuming when it was implemented on grasp detection using sliding window approached. Redmon and Angelova [34] used AlexNet [17] to solve it which ran at 13 frames per second on a GPU. It was a significant work because it shows the fact that grasp detection is not only a classification problem but also it is a regression problem. YOLO (You only look once) proposed by Redmon et al. [37] can classify objects with high accuracy and localize the recognized objects with coordinates of the bounding box simultaneously in real time. Inspired by the simultaneous implementation, [38] used a residual convolutional neural network to predict the confidence map and the rectangle for grasps for single objects in an image. But for the all-purpose utility, a robot should find grasps in object cluttered scenes. A GG-CNN is proposed by [25]. Their work is to use the depth image to predict the grasp quality and grasp pose of every pixel. Other methods in [40] developed a roi-based detection system which can grasp objects in a pile of objects scene. All their works [3, 18, 34] made the full use of convolutional neural network which need no pre-processing and could automatically extract features. Nevertheless, our works show that image pre-processing is helpful to reduce the detection time for our trained model. Our use of decision trees is novel and perform better than most of the current research.

The inverse kinematics is a mapping problem, which converts the end-effector pose to robot self-configuration [30]. For solving the inverse kinematics problem, there are three main solutions that are numerical, analytical and approximating. Aristidou et al. [1] introduced that numerical methods tried to attain qualified solutions through iterations while analytical methods wanted to find all satisfied solution through a mapping function. Maciejewski and Klein [21] solved the inverse kinematics of kinematically redundant manipulator through approximate solutions. They tried to solve it by calculating the mapping function between the joint velocities and the end-effector's velocities. And the mapping function can be found through the generalized inverse [32].

3 The system of robotic grasp

In this section, we propose a robotic grasp system based on image processing and random forest. We will introduce this system from four parts. Section 3.1 is the statement of our grasp detection problem. The architecture of our system is also introduced. Section 3.2 gives a detail explanation on how we change the metric of graph segmentation relative to previous approaches. The results of our method are also presented with the comparison of others'

approaches. And the morphological image processing is also shown. Section 3.3 explains how we preprocess our data and train our random forest classifier on Cornell Dataset. For the robot control parts, we also introduce the robot inverse kinematics in Section 3.4.

3.1 Problem statement

By capturing the colored image and corresponding aligned depth data, our final goal is to find out a stable way to detect grasps of novel objects. According to the methods proposed by Jiang et al. [16], our representation of robotic grasps is based on five-dimensional rectangles. This representation is a simplified version of the seven-dimensional methods [16, 34] and by getting the normal vector of a rectangle's centroid, we can project the rectangle into six-dimensional space, so it can be converted to the final pose (including position and orientation) of a parallel plate gripper (Fig. 2).

As shown in Fig. 3, the rectangle (r) which can be represents by:

$$r = \{\mathrm{x}, \mathrm{y}, \theta, \mathrm{h}, \mathrm{w}\} \tag{1}$$

where r is the grasp of ground truth (final grasp), (x, y) and θ is the centroid's coordinate and orientation of r respectively, and the height and width are shown by (h, w).

By using the representation of a five-dimensional rectangle, we convert the problem of robotic grasps to detect objects in an image and find the reliable rectangles on the detected objects. Our grasp detection pipeline is presented in Fig. 2. From it, we can see our improved graph segmentation is intended to set the object and background apart. This will be helpful for later implementation. And by processing the image with morphology, we can get a candidate grasps set which can be represented by:

$$G = \{\mathrm{g}_1, \mathrm{g}_2, \ldots\ldots, \mathrm{g}_\mathrm{n}\} \tag{2}$$

where $g_i \in G$ represents a robotic grasp rectangle.

Because morphological image processing cannnot distinguish the positives from candidate grasps, $g_i \in G$ can result in either positive or negative grasp. So we train a random forest classifier on Cornell Dataset, then use this classifier to score every candidate grasp. We select the rectangle which has the highest score as our final rectangle, using the idea of conversion, we can determine the final grasp configuration for our robot.

3.2 Image processing

3.2.1 Image segmentation

Image segmentation remains lots of issues in the task of computer vision. Our goal is trying to remove the background of an image taken from a tabletop scene. Some of existing

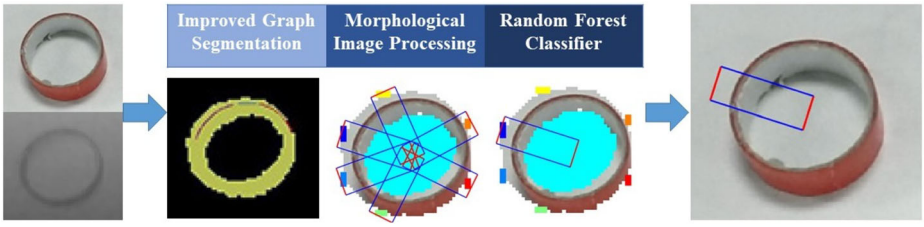

Fig. 2 The architecture of our proposed system

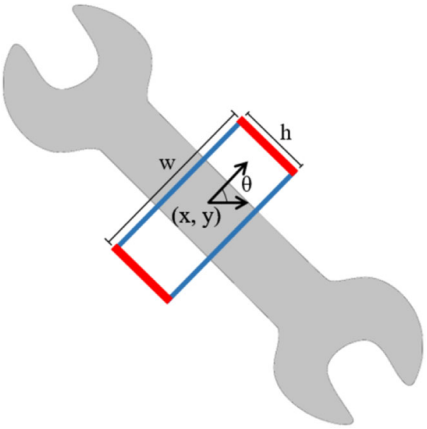

Fig. 3 The grasp of ground truth is represented by a rectangle with five dimensions. The red line remarks the final pose of a gripper plate while the blue line shows the width that a gripper should open before executing an object's grasp

approaches [8, 33] are powerful to distinguish background from scene. By a graph-based representation of an image, Daniel et al. are devoted to looking for a boundary between two different regions. Their idea of segmentation relies on the difference between the intensity of pixels. Equation (4) represents their calculations:

$$\omega = \left| I\left(p_i\right) - I\left(p_j\right) \right| > \tau(c) \tag{3}$$

$$\omega = \sqrt{(r_{p_i} - r_{p_j})^2 + (g_{p_i} - g_{p_j})^2 + (b_{p_i} - b_{p_j})^2} \tag{4}$$

where $I(p_i)$ and $I(p_j)$ represent the pixel intensity of p_i and p_j respectively, $\tau(c)$ is the function for threshold.

Apparently, the two pixels are regarded as belonging to different regions when ω is greater than $\tau(c)$. However, as mentioned in [33], it is not a reasonable way to segment only with RGB information. As a extension of this approach, Deepak et al. [33] added depth map giving the reason that objects exist in 3D space. Their metric is shown in (5).

$$\omega = \left| W^T * F\left(p_i\right) - F\left(p_j\right) \right| > \tau(c) \tag{5}$$

where $F\left(p_i\right) \in R^4$ is equal to $I(p_i)$ plus an extra corresponding depth information, $W \in R^4$ represents the weight function for each dimension of F.

However, Deepak's method didn't separate background total from the tabletop scene so they need to train an additional supervised classifier to help recognize segments.

In our work, we propose a method that combines depth information with a graph-based algorithm depicted in [8], which is different from [33] and can separate the whole background from the tabletop scene. And we also use the segmentation method to help us to figure out how many objects are contained in our scene, which is important for our morphological image processing. Our improvement is shown in (6). That also means we don't have to calculate the difference of intensity for two pixels when the two all belong to the background. We simplify this metric by using the logical mask(M) times the results from (3).

$$\omega = (\mathrm{M}_i \parallel \mathrm{M}_j) * \left| I\left(p_i\right) - I\left(p_j\right) \right| > \tau(c) \tag{6}$$

where M_i is equal to the logical value of ith point from our logical mask(M) in Algorithm 1, $M_i \in \{0, 1\}$ and 0 stands for background while 1 stands for objects.

Algorithm 1 Graph segmentation.

Input:
a RGB image, I and the aligned depth map, D
the corresponding depth data of background, D_b
Output:
a RGB image with separate objects and without background, I_o
1: Calculate the difference between D and D_b
2: Set a threshold to obtain a mask(M) by the difference
3: Use convolution filters and area opening to remove noise
4: Get I_o by using the scheme of graph segmentation algorithm from [36] and updating the metric with our metric (specified in Equation (6))

Our approach has the following steps (see Algorithm 1). In Step 1 and Step 2, we subtract the background depth map from the front depth map and use a threshold to get the logical mask(M) from the tabletop scene. In Step 3, because of the existing of noise, which is the pixels that segmented from background image but does not belong to the objects in the foreground image (See Fig. 4b), we implement a two-dimensional convolutional filter that is a 5×5 all one matrix to smooth the mask map and then use area opening to remove small components which has less than 100 pixels in the image. From the above procedures, we can obtain the background mask. Finally, in Step 4, to make the graph-based algorithm more adapted to the tabletop scene, we update the metric using the following equation (6). By the metric, we can easily separate objects and figure out the number of them, at the same time, it avoids time-consuming.

Our results of segmentation is shown in Fig. 4. From the results in Fig. 4b and c, we can figure out that [8, 33] have no ability to segment the bottom part of the red tape (shown in the left-down corner of Fig. 4a) from the scene.

3.2.2 Morphological image processing

Through graph segmentation with a depth mask, we have obtained an image which contains only objects. There have two main strategies to process it: one is to sample image patches randomly (or sample with sequential importance) like [27]. The other is to implement a sliding window to search the grasp rectangles globally. Obviously, the two methods are time-consuming and the first method is also not so accurate enough. To avoid those problems, our methods is to do morphological image processing (MIP) on the result of our graph segmentation, I_o, which can get the candidate grasps effectively.

Algorithm 2 Morphological image processing.

Input:
a RGB image which is the results of segmentation, I_o
Output:
candidate rectangles, G
1: **for** i = 1 **to** NumOfObjects **do**
2: Mark the regions of interest for each object in I_o
3: Estimate candidate parts where the gripper could fall
4: Calculate the centers of the candidate parts, C
5: Iteratively choose two different centers from C and transform them into candidate rectangles.
6: **end for**

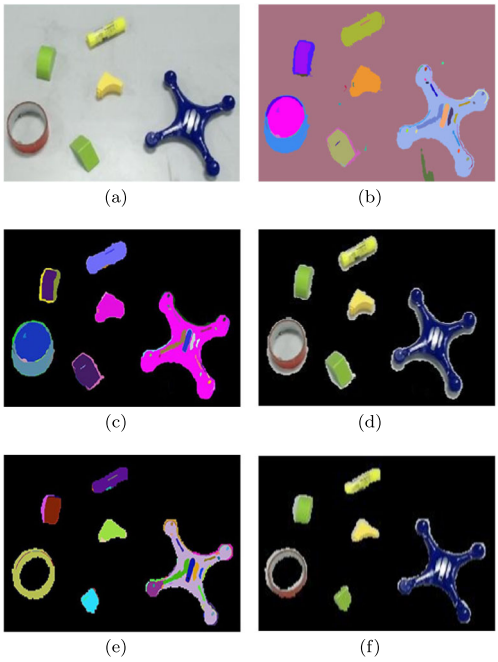

Fig. 4 The results of different segmentation approaches: **a** our test scene; **b** results by implementing the open-source code in [8]; **c** represents the segmentation results we implemented by following the procedure in [33]; **d** is the corresponding to the results in **c**; **e** and **f** are the two different representations for our improved segmentation algorithm

A strategy based on MIP is illustrated in Algorithm 2. Step 2 and Step 3 are the most important part of our algorithm. Firstly, from the results of graph segmentation, or by doing blob detection, we can figure out how many objects exist in I_o and separate them into different parts. In the multiple objects detection scene, the two steps also convert it into the problem of single object grasp detection. Secondly, for each of them, a divide-and-conquer algorithm [5] is implemented to calculate the smallest convex polygon that contains the object. Then, we use a convolutional filter to expand the convex polygon. This step can avoid the problem of boundary coinciding between the convex polygon and the object. Finally, we subtract the expand convex polygon from I_m that is a binary version of I_o. And the results are our regions of interest (ROI). Intuitively, ROI is also the place that our gripper should fall to grasp an object. Our ROI results are shown on the top part of Fig. 5, which are represented by the colorful area. Step 4 and Step 5 connect centers of two different with a straight line, which represents the width that our robotic gripper should open and it also shows the width of our candidate rectangle. But for real experiments, the width of our gripper is a constant. So we don't need to focus on the height of our grasp rectangle. For preserving the five-dimensional representation of grasp detection, we set the height of our rectangle to 0.4 times the width of the candidate rectangle.

With the setting of height and width, we can convert the combination of two different centers of ROI to candidate rectangles G. For illustration, we randomly sample the candidate rectangles for different objects, which is shown on the bottom part of Fig. 5. And from Fig. 5, we find that morphological processing can help us generate a candidate grasp set. Intuitively, we can find not every grasp rectangle in this set is positive. So for the stability of grasping objects, we need to train a classifier to evaluate each grasp rectangle in our candidate set.

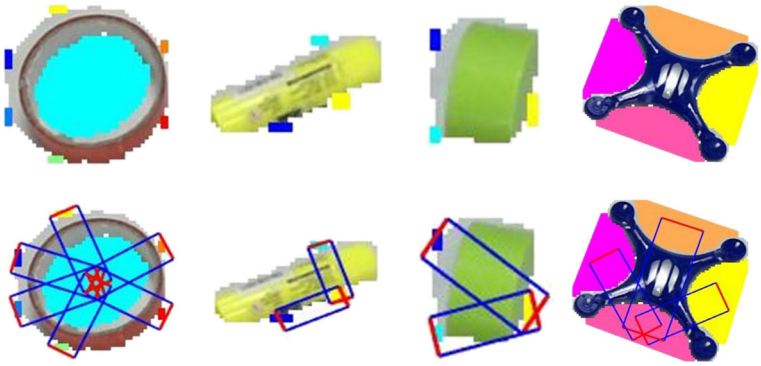

Fig. 5 Results for morphological image processing. The colorful area is marked as the regions of interest. And the bottom part samples some candidate rectangles for four types of objects

Interestingly, our real-world experiments show that the MIP algorithm by itself can detect the object grasp pose and be applied to grasp objects, though our proposed system also need an additional classifier. In one round of our experiments, we cancel the random forest classifier, randomly choose one rectangle from our MIP algorithm output G, and then we assume it is a positive grasp, convert it to grasp pose and do grasping. We can get a surprising grasp success rate about 79.12% (The experimental details and results are shown in Table 1). The reason for this success rate is that it is easy to form an antipodal-like grasp by randomly choose two different areas near objects. And our MIP output G has a high relation to our final grasp accuracy because the input of our classifier is the candidate grasp set.

Table 1 Results of Baxter grasping in different scene

Experiments		Grasp attempts	Grasp success rate (%)		Average detection
			No RF	With RF	time (s)
Single object scene	pen	15	86.7	100	0.02
	cable	15	73.3	100	0.033
	knife	15	80.0	100	0.032
	tape	15	93.3	100	0.045
	bottle	15	73.3	93.3	0.031
	grip exerciser	15	73.3	93.3	0.047
	scissor	15	66.7	86.7	0.05
	stapler	15	80.0	93.3	0.05
	wire nipper	15	73.3	86.7	0.063
	plier	15	66.7	86.7	0.04
Multiple objects scene	5 objects	15	85.3	100	0.129
	8 objects	15	82.5	93.8	0.289
	8 objects	15	70.8	87.5	0.32

3.3 Random forest

3.3.1 Data pre-processing

Our method is evaluated on the Cornell Grasping Dataset [16]. Referring to the image pre-processing approaches from earlier works [19], our process methods are: firstly, we try to extract all the data inside the labeled rectangles of datasets. The data which consists of RGB information and the corresponding depth data is a $w * h * 4$ matrix (W and H describe the width and height of the rectangle respectively).

After loading the image, we remove all the nan values in the image and replace it with zero. And then, we do interpolation to inpaint the image. Secondly, by comparing different color space such as HSV and LAB (See Fig. 6), YUV space which sets the image brightness (also called intensity) and color components apart can make objects more distinguishable in the image. For that reason, we convert the RGB space of the image into YUV space. Thirdly, we calculate the surface normals of every pixel on the depth map and the surface normal of each pixel is represented by three-dimensional vectors. And we can obtain a $w * h * 7$ matrix which contains seven-dimensional feature included YUV, depth data and vector of the surface normal.

However, for each rectangle, w and h is different. For the consistency of input features' size, it is crucial to scale all grasp rectangles to the same size. According to [19], direct scaling can cause negative grasp to appear positive. Our approach is shown in (7), we resize our rectangle by a ratio of ρ. Using the max scale makes the resized rectangle less than $w_f * h_f$, and then we pad the missing part by zero value.

$$\rho = \max \left\{ \frac{w}{w_f}, \frac{h}{h_f} \right\} \tag{7}$$

where ρ is our scaling factor, w_f and h_f represent the final size of our grasp rectangle.

Finally, our input features have a size of $w_f * h_f * 7$. And in this paper, we set $w_f = h_f = 24$. For general purpose, we do normalization processing by $(D - \mu)/\sigma$, where D is the depth data, μ is the mean value and σ indicates the standard deviation.

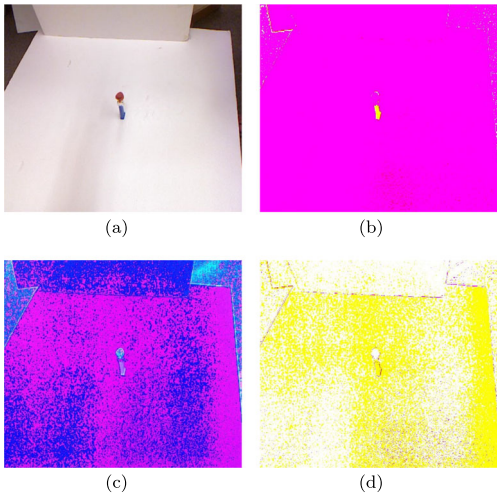

Fig. 6 The image in different color space: **a** RGB space(original scene); **b** YUV space; **c** HSV space which describes colors by hue, saturation, and value; **d** LAB space which includes three components: luminosity and two other color-related components

3.3.2 Grasp detection model

Random forest classifier (RF) has shown excellent accuracy in both classification and regression among current machine learning approaches. And it is very easy to train and set user-defined parameters.

In our work, we use the random forest to classify our grasp rectangle and Fig. 7 shows the architecture of our random forest model.

Our input dataset is represented by (8).

$$T = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)\} \tag{8}$$

where x_i represents the input features and y_i is the bool value (-1 and 1 represent negative and positive grasp respectively). N is the number of grasp instance. And the dimensional of our input feature x_i called M is equal to $w_f * h_f * 7 = 4032$.

The procedures of training random forest classifier are shown as follow:

1. Sample n training examples from these N instances randomly but with replacement.
2. Select m features ($m < M$) at random out of M. Then a random subset of the features is specified and m is a constant when we grow up the forest.
3. Create a decision tree using the n training examples with the m selected features. We set no limit on the depth of each tree and there is no pruning.
4. Repeat Step 1, 2 and 3 for k times, then we get our RF model.

After finishing our training phase, we predict the test set by aggregating the results of all trees in our RF model. For our grasp classification problem, we can take the majority votes as our final decision.

During the process of testing, we find k value (the number of trees) is important for the accuracy of classification. And for the other parameters of our random forest, we set them as default value in scikit-learn package [31]. We train our RF model for every k value in the range of (2, 110). From Fig. 8, we can assure that when k is up to some value, the accuracy always keeps the same level. Finally, we set $k = 52$ and we can get an accuracy about 94.26%.

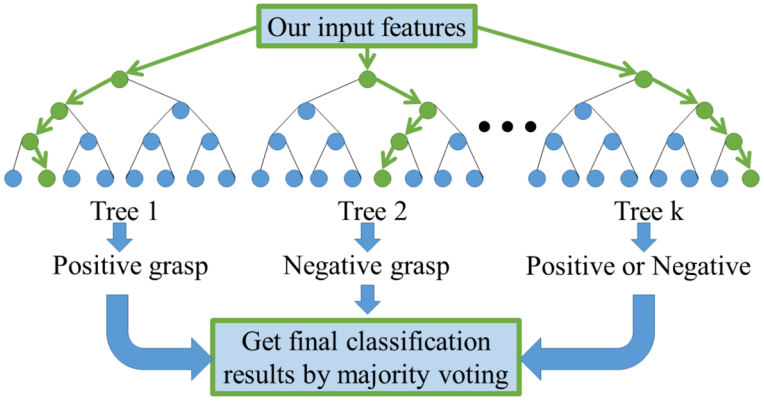

Fig. 7 The simplified architecture of our random forest classifier

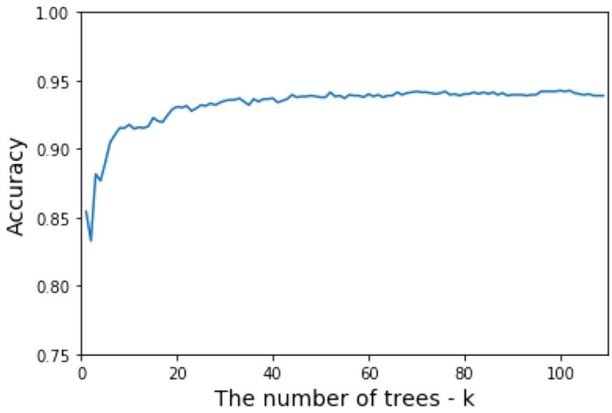

Fig. 8 Our training accuracy with different k value

We also take the random forest to compare with other classifiers. Support vector machine (SVM) which can map the high dimensional data into low dimensional data has an outstanding performance on classification. For comparison, we use a linear SVM and a radial based radial basis function kernel SVM to train our data. And both of them we get an accuracy of 94.7% on our test set. From the accuracy, we can see that in off-line classification, SVM perform a little better than random forest. However, for real-world detecting, SVM is very time-consuming than our random forest classifier, which makes real-time detection impractical.

3.4 Robot inverse kinematics

A simple way for Baxter robot to grasp an object is to send a goal pose, then the end-effector can move to the desired configuration, finally, the robot closes its gripper and use the friction force to grasp the object. To avoid the IK failed problems and for the purpose of controlling the trajectory of the robot's end-effector link, we implement a numerical cartesian IK solution which is described in [21] to calculate the joint velocities.

Denote the velocities of the joint and the end-effector as $\dot{\theta}$ and $\dot{x}$ respectively. And J is represented for the robot arm's Jacobian matrix. The relationship between the joint velocities and the end-effector velocities is shown in (9).

$$\dot{x} = J\dot{\theta} \tag{9}$$

where $\dot{x}$ is the differential results of the pose, which is in 3-D space and it is a six-dimensional vector. The dimension of $\dot{\theta}$ depends on the degree of freedom (DOF) of the manipulator's arm.

The arm of our manipulator is kinematically redundant when its DOF is larger than the end-effector velocities' dimension. In this situation, J is not defined because the number of rows and columns of J matrix is not equal. According to the methods in [21], when it comes to the IK solution of the kinematically redundant arm, there are infinite sets of solution. And the best approximate solution for (9) is shown in (10).

$$\dot{\theta} = J^{\dagger}\dot{x} + (I - J^{\dagger}J)z \tag{10}$$

where $J^{\dagger}$ is the pseudo-inverse. And (11) gives the relationship between $J^{\dagger}$ and J. I is an identity matrix, and the rank of I is equal to the number of DOF (n). z is an arbitrary n-dimensional vector.

$$J^{\dagger} = J^{T}(JJ^{T})^{-1} \tag{11}$$

For our real control, we give the moving velocities towards our desired goal by (12). $v_{\max}$ is the six-dimensional limitation velocities for robot motion. And through several time periods, we can achieve reaching the desired goal.

$$\dot{x} = \min\left\{\frac{(x_{\mathrm{d}} - x_c)}{\mathrm{rate}}, v_{\max}\right\} \tag{12}$$

where x_{d} is the desired goal pose, x_c is the current pose, $v_{\max}$ represents the maximum moving velocities in 3-D space. The value of rate shows the time period that we recalculate our current pose x_c.

4 Experimental results

We performed two steps of robotic grasp experiments. The first one is to implement grasp detection to test the predicted accuracy of our model and acquire the average detection time for objects. Another step is to using the robot to grasp novel objects in the real world.

Equipment preperation For the purpose of evaluating the performance of our methods, we performed robotic grasping experiments using the Baxter Research robot in the real scene. Baxter is a two-armed robot, and each arm has seven degrees of freedom which is kinematically redundant. It is equipped with an antipodal gripper, which has only one DOF. In our experiments, we only used the right arm of Baxter. Microsoft Kinect Sensor V2 was used as our RGB-D data-acquired device, which gave a 1920×1080 RGB image and a 512×424 depth image.

Due to the inconsistency of coordinate space of RGB and depth, we need to calibrate the camera and get the intrinsic parameters, then we used it to unify the cooridinate frame of RGB and depth by mapping the depth data to the RGB cooridinate frame. Finally, the size of our depth image was the same as RGB data, which was 1920×1080. We put our Kinect V2 sensor in front of our robot about 1.2 meters and placed the table between the Baxter robot and Kinect. And we used Matlab 2014B with a computer which had a memory of 16GB and an Intel (R) Core (TM) i5-6400 CPU to implement our algorithm.

Experimental preparation For objects in our scene, we collected some mechanic equipment, official tools, and daily supplies. Their shapes varied from cuboid, cylinder, fork, triangle and other complex shapes. And because of the size limitation of our Baxter's gripper, our objects are not too large. The number of objects is about 40 and all these objects did not exist in the Cornell Grasp Dataset and they were novel for our algorithm. Some typical objects were shown in Fig. 9.

Before placing objects in our tabletop scene, we took a RGB and an aligned depth map as our background data. This trick was used to get a depth mask that is an input for improved graph segmentation. Due to there are lots of missing data in the left and right border of the depth map, we placed objects in the center-view field of Kinect sensor. And also because of the limitation of Baxter's reachability, the objects should be placed into the workspace of it.

Fig. 9 Typical objects in our experiments

4.1 Grasp detection

A typical single grasp detection pipeline was performed as follow. For each object in our collection, we placed it on the table randomly. Then, we took a RGB image and an aligned depth map as our input, and our algorithm generated a five-dimensional grasp rectangle. And as mentioned before, our rectangle was not always the best grasp for objects, it was an optimal grasp in our candidate grasp set resulted from MIP algorithm.

Overall, we tested our methods by an extensive series of experiments. We performed single object grasp for 40 different objects and also three groups of multiple objects (Some of them was illustrated in Fig. 10). For each trial, we implemented 15 attempts to try to grasp. For placing the object, the object's position and orientation were different. When

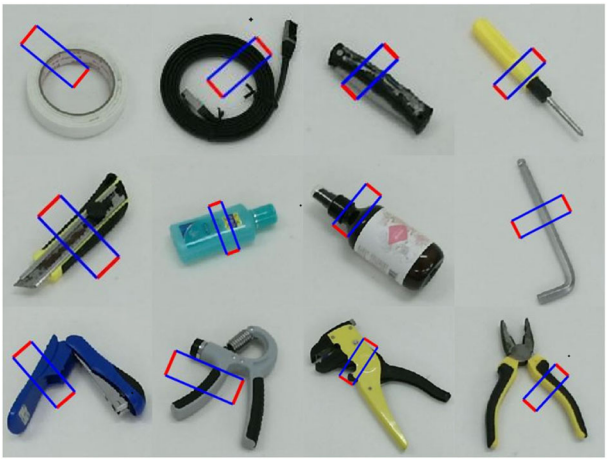

Fig. 10 Examples from our experiment collections. There are different type of objects

Table 2 The comparsion results of different algorithms

Algorithm	Accuracy (%)	Speed (fps)
Two-step learning [16]	89.6	0.02
SAE [19]	93.7	0.07
CNN [34]	88.0	3.31
2 × ResNet-50[18]	89.2	16.03
ZF-net[14]	93.2	–
Ours	94.26	28

detecting objects on the table, we computed our detection time for each trial. The result of average detection time is shown in Table 1. And the detection results are shown in Fig. 10.

Comparisons in accuracy and average detection time had been made between different algorithms such as CNN, ResNet. The results of comparison are shown in Table 2. All their model were trained on the Cornell Grasping DataSet. From that we can see, our MIP methods reduced the time for RF classifier and distinguished it from other time-consuming machine learning approaches like [16] and [19], and the combination of our grasp segmentation and MIP eliminate the useless data in RGB and depth map which made it perform better than other end-to-end deep learning detection approach like [14, 18, 34].

4.2 Robotic grasp in real world

Because of the difference between the camera space and the robot space, we need to do eye to hand calibration before grasping. And we use the dual quaternions methods described in [4] to calculate T_C^R, which is the transformation matrix of the camera to the robot. As it was described in Section 3.4, because of the IK crash problem, we replace the built-in IK algorithm in Baxter robot as our IK methods. This can make Baxter do trajectory planning without failing only if the objects were placed in the workspace of the Baxter robot.

For real-world experiments, two different approaches (No RF and With RF) were performed. We set 15 rounds of robotic grasp to clear the table for each group with each approach. Our grasp success rate is shown in Table 1. By calculating the average, the grasp success rate of No RF and With RF is 79.12% and 93.1%. And a typical clear table scene is illustrated in Fig. 11.

Fig. 11 A table clearing sence. Eight different objects in our experiments

The results showed that the success rate of grasping (See in Table 1) is not always equal to our algorithm detection accuracy (94.26%). The first reason for failing grasps is that we always tried to grasp objects with an antipodal parallel-style gripper. Limitation of this gripper is that it tries to grasp objects relying on friction force and when grasping handle-featured objects such as bottle and pen, it was easy to slip away. Another reason is that our algorithm was not trying to find the best grasps for objects. That was easy to fail when our candidate grasps set didn't contain a good grasp relative to global search methods. Our algorithm can be very useful when detecting odd-feature objects such as grip exerciser, wire nipper, and other mechanical tools. And for real-world grasp, when converting the grasp rectangle to final grasp pose, the accuracy of the hand-eye calibration results T_C^R is also related to the grasp success rate.

An interesting result in our experiments was that when we tried to grasp objects without RF, we could achieve a mean accuracy of 79.12%. As was described in Section 3.2.2, our random selection of grasp can be either a positive grasp or negative grasp. In our MIP algorithm, every rectangle contained two different candidate parts, when it was converted to grasp, it is kind of like antipodal grasp which apply force at two points and the forced direction of the two points are opposite and co-linear. The difference between our grasp and antipodal grasp is that we neglected whether the normals at the two different points are co-liner. In our case, it was impossible to check which element in our candidate grasp set satisfied antipodal grasp for the reason that we only used one RGBD sensor, we can just get the normals when points were in the front-side of the RGBD sensor.

5 Conclusions

We presented a novel robotic grasp pipeline to clear the table in a RGB-D view, which relied on graph segmentation, morphological image processing and machine learning (random forest). Compared to previous approaches, our graph segmentation can completely distinguish the objects from the background, our image pre-processing methods (including graph segmentation and MIP) which are used to generate a candidate grasp set can reduce the detection time for our classifier. And our pipeline without RF also can be used to detect grasp though there is no principle to judge whether it is a positive grasp. Our RF model is evaluated on the famous Cornell Dataset and we made a comparison with the state-of-the-art deep learning methods, our results show that our whole approach can have a better performance than many of them for the reason that image pre-processing can help us avoid detect grasps on background while deep learning methods take the whole RGB and depth map as input. When robot grasping objects, it is important to get the best IK solution so we can grasp objects more robust. We implement a different IK method rather than the Baxter built-in algorithm. This method also reduces the time for the trajectory plan and robot motion. We also perform two-stage grasp detection experiments. One is to detect the grasp pose for our self-collected objects and the other does real-world grasping using Baxter robot. For real-world grasp, we achieve a grasp success rate about 93.1% while grasp success rate without RF is about 79.12%. Our comparison experiments show the effectiveness of our random forest classifier.

However, there is also a lot need to be improved. One of our future works is trying to focus on grasp objects using the multi-finger hand rather than the two-finger hand. For our five-dimensional rectangle representation, it can be extremely difficult to be adaptable to the hand with five fingers. In our experiments, for the reason that we didn't consider objects' center of gravity when the Baxter robot tried to grasp, some objects were easy to

slip away. Next, when using morphological image algorithm to generate a candidate grasp set, it is not always containing the best global grasp rectangle. In our system, the final grasp configuration is only the optimal grasp in the candidate grasp set. Another extension work may try to adjust the rectangle in the candidate grasp rectangle set so it can always hold the best grasp in global grasp.

Acknowledgments This work was partially supported by Engineering and Physical Sciences Research Council (EPSRC) under Grant EP/S001913.

References

1. Aristidou A, Lasenby J, Chrysanthou Y, Shamir A (2018) Inverse kinematics techniques in computer graphics: a survey. In: Computer graphics forum, vol 37, pp 35–58. Wiley online library
2. Bicchi A, Kumar V (2000) Robotic grasping and contact: a review. In: Proceedings 2000 ICRA. Millennium Conference. IEEE international conference on robotics and automation. Symposia Proceedings (Cat. No.00CH37065), vol 1, pp 348–353
3. Chu F-J, Vela PA (2018) Deep grasp: Detection and localization of grasps with deep neural networks
4. Daniilidis K (1999) Hand-eye calibration using dual quaternions. Int J Robot Res 18(3):286–298
5. De Berg M, Van Kreveld M, Overmars M, Schwarzkopf O (1997) Computational geometry. In: Computational geometry. Springer, pp 1–17
6. Diankov R, Kuffner J (2008) Openrave: a planning architecture for autonomous robotics. Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34,79
7. Ding D, Liu Y-H, Zhang J, Knoll A (2001) Computation of fingertip positions for a form-closure grasp. In: Proceedings 2001 ICRA. IEEE international conference on robotics and automation (Cat. No.01CH37164), vol 3, pp 2217–2222
8. Felzenszwalb PF, Huttenlocher DP (2004) Efficient graph-based image segmentation. Int J Comput Vis 59(2):167–181
9. Fischinger D, Vincze M (2012) Empty the basket-a shape based learning approach for grasping piles of unknown objects. In: Iros, pp 2051–2057
10. Fischinger D, Vincze M (2012) Shape based learning for grasping novel objects in cluttered scenes. In: Syroco, pp 787–792
11. Fischinger D, Weiss A, Vincze M (2015) Learning grasps with topographic features. Int J Robot Res 34(9):1167–1194
12. Girshick R (2015) Fast r-cnn. In: Proceedings of the IEEE international conference on computer vision, pp 1440–1448
13. Gualtieri M, Pas AT, Saenko K, Platt R (2016) High precision grasp pose detection in dense clutter. In: 2016 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 598–605
14. Guo D, Sun F, Kong T, Liu H (2016) Deep vision networks for real-time robotic grasp detection. Int J Adv Robot Sys 14(1):1729881416682706
15. Huaman AC (2016) Grasp selection strategies for robot manipulation using a superquadric-based object representation. PhD thesis, Georgia Institute of Technology
16. Jiang Y, Moseson S, Saxena A (2011) Efficient grasping from rgbd images: Learning using a new rectangle representation. In: 2011 IEEE International conference on robotics and automation, pp 3304–3311
17. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105
18. Kumra S, Kanan C (2017) Robotic grasp detection using deep convolutional neural networks. In: 2017 IEEE/RSJ International conference on intelligent robots and systems (IROS). IEEE, pp 769–776
19. Lenz I, Lee H, Saxena A (2015) Deep learning for detecting robotic grasps. Int J Robot Res 34(4–5):705–724

20. Li J-W, Jin M-H, Liu H (2003) A new algorithm for three-finger force-closure grasp of polygonal objects. In: 2003 IEEE International conference on robotics and automation (cat. no.03CH37422), vol 2, pp 1800–1804
21. Maciejewski AA, Klein CA (1985) Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments. Int J Robot Res 4(3):109–117
22. Makhal A, Thomas F, Gracia AP (2018) Grasping unknown objects in clutter by superquadric representation. In: 2018 Second IEEE international conference on robotic computing (IRC). IEEE, pp 292–299
23. Miller AT, Allen PK (2004) Graspit! a versatile simulator for robotic grasping. IEEE Robot Autom Mag 11(4):110–122
24. Miller AT, Knoop S, Christensen HI, Allen PK (2003) Automatic grasp planning using shape primitives. In: IEEE international conference on robotics and automation, 2003. Proceedings. ICRA'03, vol 2. IEEE, pp 1824–1829
25. Morrison D, Corke P, Leitner J (2018) Closing the loop for robotic grasping: a real-time, generative grasp synthesis approach. arXiv:1804.05172
26. Pas AT, Platt R (2015) Localizing antipodal grasps in point clouds. arXiv:1501.0
27. Pas AT, Platt R (2016) Localizing handle-like grasp affordances in 3D point clouds. In: Experimental robotics. Springer, pp 623–638
28. Pas AT, Platt R (2018) Using geometry to detect grasp poses in 3d point clouds. In: Robotics research. Springer, pp 307–324
29. Pas AT, Gualtieri M, Saenko K, Platt R (2017) Grasp pose detection in point clouds. Int J Robot Res 36(13-14):1455–1473
30. Paul RP (1981) Robot manipulators: mathematics, programming, and control: the computer control of robot manipulators. Richard Paul
31. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: Machine learning in Python. J Mach Learn Res 12:2825–2830
32. Rao CR, Mitra SK (1972) Generalized inverse of a matrix and its applications. In: Icams conference
33. Rao D, Le QV, Phoka T, Quigley M, Sudsang A, Ng AY (2010) Grasping novel objects with depth segmentation. In: 2010 IEEE/RSJ international conference on intelligent robots and systems (IROS). IEEE, pp 2578–2585
34. Redmon J, Angelova A (2015) Real-time grasp detection using convolutional neural networks. In: 2015 IEEE international conference on robotics and automation (ICRA). IEEE, pp 1316–1322
35. Redmon J, Farhadi A (2017) Yolo9000: better, faster, stronger. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 7263–7271
36. Redmon J, Farhadi A (2018) Yolov3: an incremental improvement. arXiv:1804.02767
37. Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 779–788
38. Trottier L, Giguere P, Chaib-Draa B (2017) Convolutional residual network for grasp localization. In: 2017 14Th conference on computer and robot vision (CRV). IEEE, pp 168–175
39. Zeng X, Ouyang W, Yan J, Li H, Xiao T, Wang K, Liu Y, Zhou Y, Yang B, Wang Z, et al. (2018) Crafting gbd-net for object detection. IEEE Trans Pattern Anal Mach Intell 40(9):2109–2123
40. Zhang H, Lan X, Zhou X, Zheng N (2018) Roi-based robotic grasp detection in object overlapping scenes using convolutional neural network. arXiv:1808.10313
41. Zhang J, Yang C, Li M, Feng Y (2018) Grasping novel objects with real-time obstacle avoidance. In: International conference on social robotics. Springer, pp 160–169

Jiahao Zhang received the B.Eng. degree in automation from the South China University of Technology, Guangzhou, China, in 2018, and is currently pursuing the M.S. degree in the South China University of Technology, Guangzhou, China. His research interests include machine learning, intelligent control and image processing.

Miao Li received the Bachelor and Master's degrees from the School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan, China, in 2008 and 2011, respectively, and Ph.D. degree in Department of Mechanical Engineering, École Polytechnique Fédérale de Lausanne in Switzerland in 2015. His research interests are in robotics, machine learning and applied nonlinear control. They encompass robot learning and control, object grasping and manipulation, human-robot interaction, robotic hand and tactile sensing, and neuroscience.

Ying Feng received the B.E. degree and M.E. degree in electrical engineering from Zhejiang University, China, in 2000 and 2003, respectively, and Ph.D. degree in control engineering from South China University of Technology, China, in 2006. She currently serves as Associate Professor with the School of Automation Science and Engineering, South China University of Technology, China.

Her research interests include robot control and smart materials actuating control.

Chenguang Yang is a Professor of Robotics with Bristol Robotics Laboratory, University of the West of England. He received the Ph.D. degree in control engineering from the National University of Singapore, Singapore, in 2010 and performed postdoctoral research in human robotics at Imperial College London, London, UK from 2009 to 2010. He has been awarded EU Marie Curie International Incoming Fellowship, UK EPSRC UKRI Innovation Fellowship, and the Best Paper Award of the IEEE Transactions on Robotics as well as over ten conference Best Paper Awards. His research interest lies in human robot interaction and intelligent system design.