Robotic object recognition and grasping with a natural background

A Hui Wei¹,² and B Yang Chen¹,²

Abstract
In this article, a novel, efficient grasp synthesis method is introduced that can be used for closed-loop robotic grasping. Using only a single monocular camera, the proposed approach can detect contour information from an image in real time and then determine the precise position of an object to be grasped by matching its contour with a given template. This approach is much lighter than the currently prevailing methods, especially vision-based deep-learning techniques, in that it requires no prior training. With the use of the state-of-the-art techniques of edge detection, superpixel segmentation, and shape matching, our visual servoing method does not rely on accurate camera calibration or position control and is able to adapt to dynamic environments. Experiments show that the approach provides high levels of compliance, performance, and robustness under diverse experiment environments.

Keywords
Object recognition, robotic grasping

Introduction
An aging population and rising labor costs are acute challenges facing society, resulting in a high demand for indoor service robots. Service robots working in indoor environments, such as homes or offices, often need to handle a variety of grasping tasks that require the ability to recognize the target object in a complex or dynamic background environment.¹,² Due to uncertain factors, such as illumination, occlusion, and object posture, as well as the challenges of executing a real-time response by indoor service robots and choosing the proper gripping positions, it is difficult to design a lightweight recognition algorithm that can define different target objects whenever necessary.

Research on robotic grasping has resulted in many different grasping methods.³–⁶ Recently, deep-learning techniques have emerged as the most preferred methods among the approaches in the field of grasp synthesis.⁷,⁸ These methods use various versions of convolutional neural networks (CNNs) to identify the objects to be grasped,⁹,¹⁰ which means they demand a large amount of data as well as time for training and testing; the approaches also require an expensive hardware environment. However, the results of these methods often include problems with overfitting and lack reasonable generalization ability and the ability to be well-interpreted. Therefore, the methods based on deep-learning technology are difficult to apply to indoor robotic grasp tasks with variable target objects, viewing angles, and a dynamic environment.

In this article, a novel, fast, and lightweight method is proposed for robotic object recognition and grasping tasks.
The method can extract the contour information of objects contained in an image using edge detection and superpixel segmentation techniques and calculate the similarity between the two contours with a shape descriptor technique to complete the object recognition. Then, using the relative distance between the object centroid and the gripper, the algorithm guides the robot to move the gripper to the object and form a proper grabbing posture to complete the grasping task.

When compared with the prevailing deep-learning methods, our approach has the following advantages. First, it can flexibly adapt to the variable positions and postures of the target objects and the changes in the environment because the recognition method is based on the shape information of the objects, which is a stable, long-lasting, and essential feature. Second, since the object is identified by shape features, this method does not require a large number of training samples, which means that it saves cumbersome manual labeling work and greatly lowers the requirements for a computing hardware. Third, this method combines the object recognition module with the robot control module to form a hand-eye coordination mechanism with feedback, that is, a closed-loop control process. This process means it is not necessary to calculate the exact absolute coordinate values, only the relative positional offset between the object and the gripper, which greatly simplifies the calculation of the conversion between multiple coordinate systems. This process also improves the robot’s adaptability to the environment and the response speed to the tasks. Fourth, this method is highly interpretable. The human visual system recognizes objects mainly based on the contour information,11,12 which means our method utilizes the results of cognitive research.

Related work

Robotic grasping is a widely studied topic. Generally, these techniques can be grouped into two categories: empirical methods and analytic methods. Analytic methods3,13 use mathematical and physical models of geometry, kinematics, and dynamics to calculate stable grasping strategies. However, such methods are not easily applied to real-world scenarios, since it is difficult to model the physical interaction between the gripper and the object. Empirical methods14–16 tend to avoid the computation of physical or mathematical models that mimic human grasping strategies. These techniques associate the appropriate grasp points with a database storing object model or shape information based on object type definitions.

Recently, techniques based on deep learning have become popular.8,10,17–19 The strategies are similar: A certain number of grasp candidates are extracted from the image or point cloud, and the algorithm ranks them with a CNN and considers the object of the highest score the one to be grasped. Once the object is identified, the robot performs an open-loop grasp, which requires the precise calibration between the camera and the gripper, as well as a completely static grasping environment. These methods require a large number of labeled samples for training and testing, which not only imposes high requirements on the hardware environment but also requires a lot of manpower and time. The models usually lack reasonable generalization ability. However, CNNs often contain millions of parameters and rank grasp candidates with a sliding window at discrete intervals of offset and rotation, which results in a long processing time that can be up to tens of seconds. Deep-learning methods often only achieve coarse positioning with bounding boxes, which is not enough for precise grasping tasks.

The approach proposed in this article identifies an object based on the shape information, can quickly complete the pixel-level recognition tasks, and does not require a pre-training process or expensive hardware environment. In addition, this method is highly adaptable to changes in the environment as well as the position and posture of the object because of the shape-based recognition algorithm. Instead of bounding boxes, the recognition result is the pixel-level outline of the object, which is more conducive to the following grasping tasks.

Shape-based object detection with background interference

Shape-based object representation method

In the early work of one of the authors, multiscale triangular centroid distance (MTCD) descriptors were proposed to represent shapes.20,21,22–25 MTCD descriptors can be adapted to translation, scale, rotation, and deformation. In addition, it is convenient and quick to calculate the difference between shapes represented by MTCD descriptors, so here we used this method to calculate the similarity between two contours.

Given a shape \( S \), let sequence \( P(x_i, y_i) \) (\( i = 1, 2, \ldots, N \)) denote the sample points of its outer contour, where \((x_i, y_i)\) are the coordinates of the sample points and \( N \) is the number of the sample points (Figure 1). We calculate the centroid point \( G(x_G, y_G) \) of \( S \):

\[
\begin{align*}
 x_G &= \frac{1}{2N} \sum_{i=1}^{N} x_i \\
 y_G &= \frac{1}{2N} \sum_{i=1}^{N} y_i
\end{align*}
\]

(1)

Given a certain point \( P(x_i, y_i) \), we can find other points \( P_j(x_j, y_j), (i \neq j) \) of \( S \). A triangle \( \Delta P_i P_j G \) can be formed with \( P_i, P_j \) and the centroid point \( G \). As shown in Figure 2, the coordinates of the centroid point \( G_{ij}(x_{G_{ij}}, y_{G_{ij}}) \) of \( \Delta P_i P_j G \) can be calculated as follows:

\[
\begin{align*}
 x_{G_{ij}} &= \frac{x_i + x_j + x_G}{3} \\
 y_{G_{ij}} &= \frac{y_i + y_j + y_G}{3}
\end{align*}
\]

(2)

\( T \) triangles can be obtained for each sample point \( P_i \), where \( T \) represents the scale number, which is set to
We then calculate the distances \( d_{ij} \) between \( P_i \) and the centroid points of all triangles

\[
\begin{align*}
  d_{ij} &= \sqrt{(x_i - x_{G_{ij}})^2 + (y_i - y_{G_{ij}})^2} \\
  v_i &= \begin{bmatrix} d_{i1} \\ d_{i2} \\ \cdots \\ d_{iT} \end{bmatrix}
\end{align*}
\]

Then, we can obtain a column vector \( v_i \)

\[
M = \begin{bmatrix} v_1, v_2, \ldots, v_N \end{bmatrix} = \begin{bmatrix} d_{11}, d_{21}, \ldots, d_{N1} \\ d_{12}, d_{22}, \ldots, d_{N2} \\ \cdots \\ d_{1T}, d_{2T}, \ldots, d_{NT} \end{bmatrix}
\]

Thus, given a certain shape \( S \), we can obtain an \( T \times N \) matrix \( M \)

It is easy to prove that \( M \) has intrinsic invariance to translation of the contour of \( S \) from its definition. In addition, for each row of \( M \), we normalize \( M \) by dividing the elements by the maximal absolute value of each of row as follows

\[
M = \begin{bmatrix} \max_{i \in [1,N]} d_{i1} \\ \max_{i \in [1,N]} d_{i2} \\ \cdots \\ \max_{i \in [1,N]} d_{iT} \end{bmatrix}
\]

Next, to obtain invariance to the starting point of our shape descriptor, Fourier transform is applied to each row of \( M \) and the phase information is discarded. For easy explanation, let \( r_i \) denote each row of \( M \). Then, the discrete Fourier transform for \( r_i \) can be calculated as

\[
F_i(u) = \frac{1}{N} \sum_{i=1}^{N} r_i(u) e^{-j2\pi(u-1)(i-1)/N}, \quad i = 1, 2, \ldots, N, \quad j^2 = -1
\]

It is not difficult to prove that \( \text{abs}(F_i(u)) \) is invariant to the starting point of the contour of \( S \). Furthermore, to improve the efficiency and effectiveness of the following shape matching, the dimensionality of \( M \) is reduced from \( N \) to \( Q \), where \( Q \ll N \). Thus, the final definition of our shape descriptor is

\[
M = \{ \text{abs}(F_i(v)) | t = 1, 2, \ldots, T; v = 1, 2, \ldots, Q \}
\]

Here, we set \( Q = 16 \) in all our experiments, see Figure 3.

Given two shapes \( S_1 \) and \( S_2 \), whose shape descriptors are \( M_{S_1} = \{ \text{abs}(F_i^S(v)) | t = 1, 2, \ldots, T; v = 1, 2, \ldots, Q \} \)

\[
M_{S_2} = \{ \text{abs}(F_i^S(v)) | t = 1, 2, \ldots, T; v = 1, 2, \ldots, Q \}
\]

Figure 1. (a) The original image of an elephant model; (b) the contour shape of the model; and (c) the set of points obtained by equidistant sampling of the contour.

Figure 2. Acquisition of centroid point \( G_{ij} \) of \( \Delta P_iP_jG \).
and \( M_{S_1} = \{ \text{abs}(F_{1}^{S_1}(v)) | t = 1, 2, \ldots, T; v = 1, 2, \ldots, Q \} \), respectively. Then, the measure of dissimilarity between \( S_1 \) and \( S_2 \) can be obtained by the \( L_1 \) distance

\[
D(S_1, S_2) = \frac{1}{T \times Q} \sum_{t=1}^{T} \sum_{v=1}^{Q} \text{abs}(F_{t}^{S_1}(v)) - \text{abs}(F_{t}^{S_2}(v))
\]

(9)

The smaller the dissimilarity \( D(S_1, S_2) \) is, the more similar the two shapes are.

Elimination of redundant information in images

To obtain the line information in the image, we referred to Dollár and Zitnick’s work\(^26\) on edge detection and the work of Radhakrishna et al.\(^27\) on superpixel segmentation. In one study,\(^26\) the use of the structured forest technique for edge detection achieved good results.

However, since real images often contain a lot of noise, the results of edge detection involve much redundant information for object recognition tasks. On the other hand, superpixel segmentation, which groups pixels into perceptually meaningful atomic regions, can effectively eliminate the effects of noise in the image.

The superpixel segmentation algorithm we apply is easy to understand, and it requires only one parameter provided by users, that is, \( k \), which is the desired number of the superpixels.

Given a color image in Commission Internationale de l’Eclairage Lab (CIELAB) color space, the algorithm firstly divides the \( N \) pixels into a regular grid, whose interval is \( S = \sqrt[2]{N/k} \). \( k \) initial cluster centers \( C_i = [l_i, a_i, b_i, x_i, y_i]^T \) are then sampled from the grid. To avoid centering a superpixel on an edge and seeding a superpixel with a noisy pixel, the centers are moved to positions corresponding to the lowest gradient in a \( 3 \times 3 \) neighborhood.

Next, we traverse all the centers and associate each pixel with the nearest cluster center whose search region overlaps its position. The algorithm searches a limited region, the size of which is set to \( 2S \times 2S \) in our experiments. Given a pixel \( P_i = [l_i, a_i, b_i, x_i, y_i]^T \), we can obtain its color distance \( d_c \) and space distance \( d_s \) from a certain cluster center \( C_j = [l_j, a_j, b_j, x_j, y_j]^T \) by:
Let $D$ denote this measure, whose definition is as follows:

\[
D = \sqrt{\left(\frac{d_c}{N_c}\right)^2 + \left(\frac{d_s}{N_s}\right)^2} = \sqrt{\left(\frac{d_c}{m}\right)^2 + \left(\frac{d_s}{S}\right)^2} \quad (12)
\]

where $N_c = S = \sqrt{\frac{S}{r}}$, $N_s = m$, $m$, which we set to 10 in our experiments, weighs the relative importance between spatial proximity and color similarity.

The results of edge detection, shown in Figure 4, and superpixel segmentation to extract the true contour information of the objects contained in the image.

Figure 6 shows this algorithm in action. Assume that the input image is $I$, of which the size is $c \times r$. $I_{ed}$ is obtained by performing edge detection on $I$, and $I_{sp}$ is obtained by performing superpixel segmentation on $I$, where both $I_{ed}$ and $I_{sp}$ are grayscale images whose gray values are $[0, 1]$. The larger the gray value of a pixel, the greater the probability that it belongs to an edge. For a pixel $p$ in $I_{ed}$, assume its coordinate in the image is $(x, y)$, $(1 \leq x \leq c, 1 \leq y \leq r)$, and its gray value is $g$, $(0 \leq g \leq 1)$, corresponding to the pixel $p'$ in $I_{sp}$, whose coordinate is $(x, y)$, and the gray value is $g'$. If $\frac{|g-g'|}{2} > t$, it is considered that the pixel whose coordinate is $(x, y)$ is more likely to be the actual contour of some object, where $t$ is the set threshold. Thus, the value for $I_{C}$ is obtained that contains the possible contour information of the objects involved in the original image $I$, where $I_{C}$ is a binary map, wherein a pixel value of 1 indicates that the pixel is a contour pixel, and a pixel value of 0 indicates that the pixel is a background pixel.

Next, we refine the contour lines in $I_{C}$, that is, reduce the width of lines to one pixel and then take the branch points, whose number of adjacent points is more than two pixels, as the line end points to extract the contour line information $C$, which is a set of lines, where a line is a set of coordinate values of pixels.

**Extraction of contour template**

To obtain the shape contour of a given object, we photograph the object with a relatively monotonous background and remove redundant information of the image captured using the method above. As is shown in Figure 8, the image of animal models with a white background captured by a camera is processed by edge detection and superpixel segmentation, respectively. Then, the remaining pixels are filtered, and the lines are refined so that the shape contour of the models is obtained. The shape contour of a given object obtained above can be used as contour template in the following process, as shown in Figure 7.
Line segments combination based on heuristic search

Heuristic search, also known as informed search, reduces the search scope and complexity of the problem to be solved by referring to the heuristic information. The objective of heuristic search is to produce a solution in a reasonable time frame that is good enough for solving the problem. Heuristic search can avoid combinatorial explosions by guiding search to the most promising direction using heuristic information. The stronger the heuristic information, the less the search branches. The function used to evaluate the importance of search nodes is called valuation function, which is generally in the form of:

\[ f(x) = g(x) + h(x) \]  

where \( g(x) \) denotes the actual cost from the initial node to node \( x \), and \( h(x) \) denotes the estimated cost of the optimal path from node \( x \) to the target node. Heuristic information is mainly included in \( h(x) \) and is determined according to the characteristics of the problem.

Given an image, after the preprocessing described above of eliminating the redundant information, we can turn the image to a set of line segments which includes the contour information. If we traverse all the combinations of the line segments in the set, then the combination of line segments that is most similar to the shape contour of a given object can always be found, which is obviously very time-consuming. By using the shape descriptor and shape dissimilarity we introduced above, we can guide the search path to avoid unnecessary search nodes and thus greatly improving search efficiency.

Given the contour template \( M \) of the object to be grasped, \( M \) is a binary map where pixels have a value of 1, indicating they belong to contours. As shown in Figure 9, the algorithm looks for a seed line \( C_s \) in \( C \) as the starting
state for the following search, using the heuristic search strategy. \( C_s \) should have a certain length because the short lines correspond to very few sampling points, which result in finding too many similar parts on the template. \( C_s \) should also have a certain degree of curvature, since a real scene in images always involves a lot of line segments that tend to be mismatched with the line segment of the template. With these restrictions, the search domain for \( C_s \) is greatly reduced, and \( C_s \), which indicates the line most similar to some part of the object to be grasped, is searched exhaustively in the remaining line set \( C_0 \). Specifically, for a line \( c \) length of \( l \) in \( C' \), the algorithm slides a window with size of \( l \) onto the contour template \( M \). In each iteration, the line \( c \) in the window is taken out to calculate the shape similarity \( s \) with \( c \). The maximum value of \( s \) is considered to be the similarity between \( c \) and \( M \), and the corresponding part of \( M \) is recorded as \( c_m \).

After all the lines in \( C' \) have been calculated in regards to their similarity with \( M \), the algorithm takes the \( c \) with the largest \( s \) as the seed line \( C_s \), corresponding to the contour line \( C_m \) on \( M \).

After \( C_s \) is found, the following searches only consider the line set \( C_{nei} \) connected to the ends of \( C_s \), where \( C_{nei} = \{c_1', c_2', \ldots, c_n' | c_i' \in C' \cap (\text{startpoint}(c_i') = \text{endpoint}(C_s) \cup \text{endpoint}(c_i') = \text{startpoint}(C_s))\} \). For each iteration, a line \( c_{nei} \in C_{nei} \) is selected to combine with \( C_s \) to form a new line \( C_s' \), and \( C_m' \) is obtained by extending the corresponding length in the same direction of \( C_m \) on \( M \), and the similarity between \( C_s' \) and \( C_m' \) is calculated. At the end of each iteration, the \( C_s' \) most similar to \( M \) is taken as the result of this round and set as the seed line \( C_s \). The search proceeds according to this rule until \( C_s \) becomes a closed line and the template match is completed, as shown in Figure 10.

**Figure 8.** Animal models in a relatively monotonous background are photographed, and the images captured are processed with edge detection and superpixel segmentation, and then the remaining pixels are filtered and lines are refined. The left side of each row shows the original images and the right side of each row shows the obtained contour templates.

**Figure 9.** The process of matching an object to be grasped is shown in order from left to right and from top to bottom. (a) Search for the seed line. (b) to (e) Process of each iteration of the search. The candidates are marked in white, and the most similar line is marked in yellow. (f) Matching result.
Finally, the centroid $P$ of $C_s$ is calculated to guide the robot to execute the grasping task. Suppose $C_s=\{(x_1,y_1), (x_2,y_2), \ldots, (x_n,y_n)\}$. The coordinate of $P$ is $(x, y)$, then $x = \sum_{i=1}^{n} x_i \frac{1}{n}$ and $y = \sum_{i=1}^{n} y_i \frac{1}{n}$.

**Determination of the grasp position with contour detection**

After the recognition task is completed, the shape contour $C_s=\{(x_1,y_1), (x_2,y_2), \ldots, (x_n,y_n)\}$ of the object has been obtained, let $P(x,y)$ denotes the centroid point of the shape contour, where

$$x = \frac{\sum_{i=1}^{n} x_i}{n} \quad \text{and} \quad y = \frac{\sum_{i=1}^{n} y_i}{n} \quad (14)$$

Since the gripper is open to a certain extent, an appropriate gripping position is needed to guide the robot to rotate the sixth joint to form a proper grip posture to execute the grasp task. We took into consideration the general size of the gripper mounted on the robotic arms and the

**Figure 10.** The first row of each column shows the contour template; the second row of each column shows the original image captured by the camera; the third row of each column shows the line segments extracted from the image; the fourth row of each column shows the seed line found by the algorithm; and the fifth row of each column shows the final recognition result.
irregularity of the shape of the object to be grasped. The proposed approach uses the relatively narrow concave portion of the object outline as the grasping position so that it can handle different situations, such as when the gripper is too small to grasp a relatively large object and make the grasping state as strong as possible.

The result of the recognition method discussed in this article, a precise outline of the contour of the object, indicates that the gripping position that meets the requirements above can be obtained in a simple way. First, make a straight line \( l \) passing through the centroid \( C \) of the object outline and measure the width of the contour using the Euclidean distance \( d \) between the intersections \( p_1 \) and \( p_2 \) of \( l \) and contour \( C \). Rotate \( l \) at a certain angle interval \( \theta \), and obtain sets of intersection points and corresponding distance values. After a rotation of \( 180^\circ \), the set of intersection points with the smallest distance value is taken as the final grasping position.

As shown in Figure 11, the normal line between the two clips of the gripper is initially collinear with the horizontal axis. After the appropriate grasping position is calculated, it is only necessary to control the gripper to rotate \( \theta \) degrees counterclockwise to form the optimal grasping posture.

Guiding the robot to approach the object to be grasped using the recognition results

Interaction between the recognition module and the robot control module

As is shown in Figure 12, the computer is connected with the control cabinet of the robotic arm through a twisted pair. The
world coordinates (the origin is in the center of the base of the robotic arm) the gripper to be moved to are transferred to operating system of the robotic arm using transmission control protocol protocol. The coordinate data are then transformed to the rotation angles of each of the six joints.

As shown in Figure 13, the images captured by the camera are transmitted to the computer; the latter takes the images and the appropriate template image as parameters and invokes the recognition module to execute the recognition tasks. Next, the relative position of the object centroid to the center of the camera is calculated. If the centroid is in the central area of the camera field of view, the control module moves the gripper down to grasp the object. Otherwise, it moves the gripper to the centroid position of the object according to the relative position and makes the camera capture an image.

Object recognition flowchart

The flowchart in Figure 14 corresponds to Algorithms 1 and 2.

Robot control module flow chart

As shown in Figure 15, the robot control module accesses the memory location storing the object recognition result at

---

**Algorithm 1.** Contour segments generation.

**Input:** image l, size c*r  
**Output:** A set of contour segments C  
set $I_{ed} = EdgeDetector(l)$  
set $I_{sp} = Superpixel(l)$  
for each pixel $p(x, y, g)$ in $I_{ed}$, $p'(x, y, g')$ in $I_{sp}$ do  
  set $I_C(x, y) = \frac{(g+g')}{2} > t ? 1 : 0$  
end for  
set $I_C = Refining(I_C)$  
for each pixel $p$ in $I_C$ do  
  if NeighborsNum($p$) > 2 then  
    set EndPoints = EndPoints $\cup$ $(x, y)$  
  end if  
end for  
for each point $p$ in Endpoints do  
  for each point $p'$ in Endpoints / $p$ do  
    if HasCurveBetween($p, p'$) then  
      $C = C \cup Curve(p, p')$  
    end if  
  end for  
end for
a certain frequency and reads out if it is updated, where the recognition result refers to the relative position of the object centroid to the center of the camera field of view. If the centroid is in the center, control the gripper to move down for grasping, otherwise move it closer to the centroid of the object to be grasped.

**Experiments**

**Our environment**

Figure 16 shows our experiment environment, which consists of a computer connected to the robot control cabinet, an SD700E industrial robot arm (yellow) with six degrees of freedom, ±0.03 mm repeatability, and 700 mm radius. An EFG20 electric gripper (silver white) was mounted on the end flange of the arm, above which a simple camera (Logitech C310) was mounted. The total cost of the hardware set was less than US$15,000.

The surface of the workbench was set up to be complicated and messy on purpose. The state of the table was changed during the experiments, such as when we disturbed the relative order of the objects and changed their postures and positions, to highlight the robustness of our approach.

**Experiment result**

Figures 17 to 19 show the application of our method in the real environment. Each row in Figure 18 shows the relative positions between the gripper and the object to be grasped, that is, rhino model, before and after the recognition process was done. To present the robustness of our method, each time when the gripper was moved above the object referring to the recognition...
Camera mounted on the end of the robotic arm captures image $I$.

Edge detection and super-pixel segmentation $I$ on to obtain binary map $I_c$.

Refine the lines in $I_c$. Extract the line information in $I_c$ by taking the branch points as the endpoints to get a set of lines $C$.

Searching for seed line $C_s$ in $C$. $C_s$ should be similar to some section of contour $C_t$ in the template and has a certain length and curvature.

Consider line set $C_{net}$ connected to the ends of $C_s$. Each time, take $c_t \in C_{net}$ to form a new line $C'_t = C_s \cup c_t$ with $C_s$, and extend with the same length in the same direction on the template contour to obtain the contour segment $C'_t$ and calculate the similarity. Finally, take the most similar $C'_s$ with the template as the new seed line $C_s$.

Is $C_s$ closed?

Yes

Calculate the centroid $P$ of $C_s$.

Is the centroid in the center of the camera’s field of view?

No

Yes

Move the gripper down to grasp.

End

Figure 14. Sketch map of the object recognition module.
The camera mounted on the end of the robotic arm captures image $I$, which will be transmitted to the recognition module for processing.

Is there feedback from the recognition module?

Is the centroid of the object in the center of the camera's field of view?

Move the gripper down to grasp the object

End

Figure 15. Sketch map of the robot control module.

Figure 16. Our experiment hardware set.
result, the object was moved to a different position with a different posture as well as the surrounding objects, which resulted in another round of recognition process. Once the state of the object was not changed after the gripper was moved above it, the gripper would be moved down to execute the grasp, as is shown in the last row.

Figure 18 shows the details of recognition process corresponding to different positions of the gripper displayed in Figure 17. From left to right on each row, the image captured
by camera mounted on the gripper, the line segments representing the contour information extracted from the image and the recognition result are displayed respectively. The red line in the right side of each row connecting the centers of the camera view and the object to be grasped shows the relative position between the gripper and the object, which can be used to guide the robotic arm to move the gripper above the object.

After the precise contour of the object to be grasped is obtained, the appropriate grasp point can be calculated in a simple way to guide the robot to perform grasp tasks, as is shown in Figure 19.

We are unable to include enough pictures to show the entire recognition and grasp process owing to space constraints. However, a video clip is provided to show the whole experiment process.

Discussion

Although the object recognition methods based on deep learning are outstanding for classification tasks, they can only generate bounding boxes containing the object to be grasped when guiding a robot to perform grasp tasks. In addition, deep-learning methods require a lot of training and test data and computing time as well as an extremely extensive hardware environment. Therefore, these techniques are not suitable for robot grasp tasks, especially those that should be defined temporarily.

The object recognition module required for robot grasp tasks should be sufficiently lightweight and fast, while still being able to handle the noisy environment because the environment affects the motion planning of the robot. The recognition proposed in this article starts with the shape of the object and extracts the contour information from the original image which is stored in the form of lines. In addition, the heuristic search strategy greatly reduces the search domain so that the object can be effectively identified from the chaotic environment, which makes our method fast and robust enough to be more suitable for various robot applications.
Declaration of conflicting interests
The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding
The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was supported by NSFC under Project 61771146 and Project 61375122.

Supplemental material
Supplemental material for this article is available online.

References
1. Göngör F and Tutsoy Ö. Design and Implementation of a Facial Character Analysis Algorithm for Humanoid Robots. *Robotica* 2019; 37(11): 1850–1866.
2. Gongor F and Tutsoy O. Two steps for chair recognition: feature extraction and shape descriptors. In: *International Mediterranean Science and Engineering Congress*, Adana, Turkey, October 2016, pp. 26–28.
3. Bicchi A and Kumar V. Robotic grasping and contact: a review. In: *Proceedings 2000 ICRA millennium conference. IEEE international conference on robotics and automation. Symposia proceedings* (Cat. No. 00CH37065), 24–28 April 2000, pp. 348–353. San Francisco, California: IEEE.
4. Bohg J, Morales A, Asfour T, et al. Data-driven grasp synthesis—a survey. *IEEE Trans Rob* 2014; 30(2): 289–309.
5. Sahbani A, El-Khoury S and Bidaud P. An overview of 3D object grasp synthesis algorithms. *Rob Auton Syst* 2012; 60(3): 326–336.
6. Shimoga KB. Robot grasp synthesis algorithms: a survey. *Int J Rob Res* 1996; 15(3): 230–266.
7. Johns E, Leutenegger S and Davison AJ. Deep learning a grasp function for grasping under gripper pose uncertainty. In: *Intelligent robots and systems (IROS), 2016 IEEE/rsj international conference on Intelligent Robots and Systems (IROS)*, 9 October 2016, pp. 4461–4468. IEEE.
8. Lenz I, Lee H and Saxena A. Deep learning for detecting robotic grasps. *Int J Rob Res* 2015; 34(4–5): 705–724.
9. Kumra S and Kanan C. Robotic grasp detection using deep convolutional neural networks. In: *2017 IEEE/RSJ international conference on intelligent robots and systems* (IROS), Vancouver, BC, Canada, 24–28 September 2017, pp. 769–776. IEEE.
10. Redmon J and Angelova A. Real-time grasp detection using convolutional neural networks. In: *2015 IEEE international conference on robotics and automation (ICRA)*, 26 May 2015, pp. 1316–1322. IEEE.
11. Hoffman DD and Richards WA. Parts of recognition. *Cognition* 1984; 18(1–3): 65–96.
12. Hoffman DD and Singh M. Salience of visual parts. *Cognition* 1997; 63(1): 29–78.
13. Prattichizzo D and Trinkle JC. Grasping. In: *Springer handbook of robotics* 2016 (pp. 955–988). Springer, Cham.
14. Detrey R, Kraft D, Kroemer O, et al. Learning grasp affordance densities. *Paladyn, Journal of Behavioral Robotics* 2011; 2(1): 1–7.
15. Goldfeder C, Allen PK, Lackner C, et al. Grasp planning via decomposition trees. In: *2007 IEEE international conference on robotics and automation*, Roma, Italy, 10 April 2007, pp. 4679–4684. IEEE.

Figure 19. Once the precise shape contour of the object to be grasped is obtained, the appropriate grasping points are calculated to guide the robotic arm to execute the grasp task. (a) The initial state of the robotic arm; (b) the gripper was moved above the object according to the recognition result; (c) the gripper was moved down to the object; and (d) the gripper was rotated to form an appropriate grasp posture according to the grasp points. It is difficult to form such a posture if there is no such a pixel-level shape contour.

ORCID iD
A Hui Wei https://orcid.org/0000-0003-2696-0707
16. Miller AT, Knoop S, Christensen HI, et al. Automatic grasp planning using shape primitives. In: Proceedings. ICRA’03. IEEE international conference on robotics and automation, 2003, Vol. 2, Taipei, Taiwan, 14 September 2003, pp. 1824–1829. IEEE.

17. Mahler J, Liang J, Niyaz S, et al. Dex-net 2.0: deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. arXiv preprint arXiv:1703.09312, 2017.

18. Pinto L and Gupta A. Supersizing self-supervision: learning to grasp from 50 k tries and 700 robot hours. In: 2016 IEEE international conference on robotics and automation (ICRA), Stockholm, Sweden, 16 May 2016, pp. 3406–3413. IEEE.

19. Wang Z, Li Z, Wang B, et al. Robot grasp detection using multimodal deep convolutional neural networks. Adv Mech Eng 2016; 8(9): 1687814016668077.

20. Yang C, Wei H and Yu Q. A novel method for 2D nonrigid partial shape matching. Neurocomputing 2018; 275: 1160–1176.

21. Yang C, Wei H and Yu Q. Multiscale triangular centroid distance for shape-based plant leaf recognition. In: Proceedings of the twenty-second European conference on artificial intelligence, 29 August 2016, pp. 269–276. IOS Press.

22. Xiao J and Wei H. Scale-invariant contour segment context in object detection. Image Vision Comput 2014; 32(12): 1055–1066.

23. Yu Q, Wei H and Yang C. Local part chamfer matching for shape-based object detection. Pattern Recognit 2017; 65: 82–96.

24. Wei H, Yang C and Yu Q. Efficient graph-based search for object detection. Inform Sci 2017; 385: 395–414.

25. Wei H, Yang C and Yu Q. Contour segment grouping for object detection. J Vis Commun Image R 2017; 48: 292–309.

26. Dollár P and Zitnick CL. Structured forests for fast edge detection. In: Proceedings of the IEEE international conference on computer vision. December 2013, pp. 1841–1848. IEEE.

27. Achanta R, Shaji A, Smith K, et al. SLIC superpixels compared to state-of-the-art superpixel methods. IEEE Trans Pattern Anal Mach Intell 2012; 34(11): 2274–2282.