Chinese Journal of Aeronautics, (2015), 28(5): 1451–1463

Chinese Society of Aeronautics and Astronautics
& Beihang University

Chinese Journal of Aeronautics

cja@buaa.edu.cn
www.sciencedirect.com

On-orbit real-time robust cooperative target identification in complex background

Wen Zhuoman [a,b], Wang Yanjie [a,*], Arjan Kuijper [c,d], Di Nan [a], Luo Jun [a,b], Zhang Lei [a,b], Jin Minghe [e]

[a] *Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences, Changchun 130033, China*
[b] *University of the Chinese Academy of Sciences, Beijing 100049, China*
[c] *Fraunhofer Institute for Computer Graphics Research, Darmstadt 64283, Germany*
[d] *Technische Universität Darmstadt, Darmstadt 64283, Germany*
[e] *State Key Laboratory of Robotics and System, Harbin Institute of Technology, Harbin 150080, China*

Received 9 January 2015; revised 30 March 2015; accepted 4 May 2015
Available online 28 August 2015

KEYWORDS

Circle detection;
Edge extraction;
Edge tracking;
Line detection;
Robot vision;
Space robot arm

Abstract Cooperative target identification is the prerequisite for the relative position and orientation measurement between the space robot arm and the to-be-arrested object. We propose an on-orbit real-time robust algorithm for cooperative target identification in complex background using the features of circle and lines. It first extracts only the interested edges in the target image using an adaptive threshold and refines them to about single-pixel-width with improved non-maximum suppression. Adapting a novel tracking approach, edge segments changing smoothly in tangential directions are obtained. With a small amount of calculation, large numbers of invalid edges are removed. From the few remained edges, valid circular arcs are extracted and reassembled to obtain circles according to a reliable criterion. Finally, the target is identified if there are certain numbers of straight lines whose relative positions with the circle match the known target pattern. Experiments demonstrate that the proposed algorithm accurately identifies the cooperative target within the range of 0.3–1.5 m under complex background at the speed of 8 frames per second, regardless of lighting condition and target attitude. The proposed algorithm is very suitable for real-time visual measurement of space robot arm because of its robustness and small memory requirement.

1. Introduction

In space station, robot arms[1–4] are used to reach out from space shuttles to deploy, maneuver and capture payloads. For instance, they deploy and capture satellites, support spacewalking astronauts and help assemble the space station. All of these performances consist of three stages: object capturing, moving and releasing. During the capturing and releasing process, the

* Corresponding author. Tel.: +86 431 86176560.
E-mail address: wangyj@ciomp.ac.cn (Y. Wang).

Peer review under responsibility of Editorial Committee of CJA.

Production and hosting by Elsevier

http://dx.doi.org/10.1016/j.cja.2015.08.006

space robot arm keeps calculating its relative position and orientation to the to-be-captured object and adjusting its moving path. As shown in Fig. 1, a cooperative target[5,6] and a to-be-arrested device are fixed on the object; a visual sensor (hand-eye camera) and an arresting device are placed on the space robot arm. The hand-eye camera takes images of the cooperative target, calculates their relative position and orientation using visual measurement methods and then transfers it to the POS (position and orientation) between the arresting and to-be-arrested device. According to the POS parameters, the moving path of the space robot arm is planned.

In order to precisely control the moving trajectory, the camera must calculate the POS between the arm and the object on orbit and in real-time. The initial step of POS calculation is identifying the cooperative target on the object. However, in order to guarantee the flexibility of the robot arm, the size and weight of the hand-eye camera are limited. Therefore, the chips inside the camera, including DSPs (Digital Signal Processors), FPGA (Field Programmable Gate Array) and MCU (Micro Controller), should also have limited size and weight. Performing in outer space, the chips in the camera must have low power supply, resist radiation and high-speed particles and endure large temperature range (lower than minus 100 to more than 120 °C). Chips meeting these requirements have lower speed and less memory storage than civil products.[7] Hence, the minimization of computational cost in target identification is a constant focus. Another obstacle is that the target background is complex. Various objects may appear in the background, including planet, star, spaceship and tool box. In day-light, the target is probably backlighted and the light intensity may be too weak, too strong or uneven. The metal on the space robot arm, satellite or spaceship is likely to generate glittering spots in the image.

Fig. 2 represents the cooperative target we use to calculate the relative POS parameters. It is painted by two flat paints, one black as the background and one white as the foreground. The pattern in the foreground consists of a ring, three lines and three dots. The ring and lines are designed for identification. Because circles rarely appear in outer space, the ring reduces the misidentification rate of the target. The dots are for POS measurement. Make point O the target center and column OA with dot A on its top is perpendicular to the target plane. Dots A, B and C consist of an isosceles triangle, with A as the vertex. Using the image coordinates of the centroids of the three dots, the POS between the space robot arm and the object is calculated by perspective-three-point (P3P) algorithm.

Using the characteristics of circle and lines, we propose an on-orbit real-time robust algorithm that identifies this cooperative target in complex background. The proposed algorithm

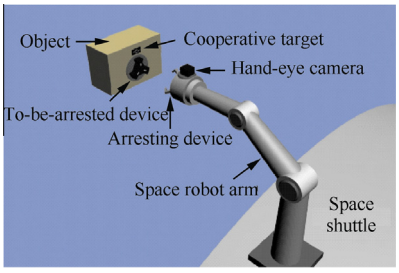


Fig. 1 Space robot arm capturing an object.

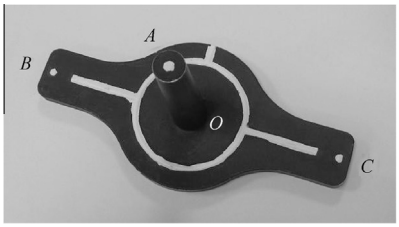


Fig. 2 Cooperative target.

first extracts the interested single-pixel-width edges using an adaptive threshold and improved non-maximum suppression. The memory amount is substantially decreased because a large number of irrelevant edges are ignored in this step. Then it follows a novel tracking algorithm to cluster pixels which change smoothly in tangential directions. According to a reliable criterion, the circles are retrieved with a small amount of computation and time. Then we set up two different sized square boundaries around each circle and detect straight lines in their complement areas. Finally, the cooperative target is identified if there are certain numbers of straight lines whose relative positions with the circle match the known target pattern. The proposed algorithm works in real-time and has a high accuracy rate. Regardless of lighting condition, distance and target attitude, it accurately identifies the target in complex background. Most importantly, it uses a very small amount of memory and can be easily deployed on DSPs. Therefore, it is very suitable for real-time robotic applications, such as POS measurement of space robot arm and robotic arms' manipulating objects on a production line.

2. Related work

During the middle 20th century to the early 21st century, various cooperative targets are used in space by different countries. Advance video guidance sensor (AVGS)[8,9] is the most widely used cooperative target on orbital vehicles by the United States of America. To identify it, two lasers of different wavelengths are used to illuminate the target. One passes through the filter in front of the retro-reflectors on the target and generates the foreground image; the other is absorbed by the filters and produces the background image. Then the foreground image subtracts the background image thresholds the result, leaving an image with only the target's retro-reflectors visible. AVGS determines the retro-reflectors' centroids and calculates the target's pose by solving the perspective-N-point problem. However, it is not suitable for visual measurement of space robot arm because of the huge size of AVGS. And its requirement of two lasers as the illumination sources increases the burden of hardware design of the hand-eye camera. When the Chinese Tiangong-1 performed docking with the Spaceships Shenzhou, 9 and 10, a cross target was used. In manual mode, it was identified by human eye; in automatic mode, it was identified using a laser and radar system. In only visual light, a cross target can be easily misidentified because lines appear constantly in outer space. And because the target is two-dimensional, it cannot be used to measure the six degrees of freedom (DOF) POS between an object and the space robot arm. Japanese Engineering Test Satellite-7(ETS-VII)[10] used a three-point non-coplanar marker

and a two-point marker. The six parameters of relative position and orientation are measured from the three-point marker, while five parameters except yaw information are measured from two-point marker. However, their recognition process depends on commands from the ground.

Two of the main steps in identifying a target are feature extraction and supervised machine learning. Most popular feature detection methods include scale invariant feature transform (SIFT),[11] Histogram of oriented gradient (HOG)[12] and wavelet transform (WT).[13] Widely used machine learning methods include neutral network, Markov random field and support vector machine. It is practical to perform machine learning off-line on the ground to obtain the feature vector of the cooperative target in Fig. 2, but it is unpractical to extract features using the above methods on-orbit in real time in outer space. The computational amount is the barrier, because the storages of hardware[14] that tolerate radiation, high-speed particles, large temperature range and low-power supply are limited. Detecting symmetry[15] can be another approach, but it is far from real-time. The computational cost is huge because the gradient product transform calculates the symmetry score of every point from a minimum radius to a maximum radius. And the method fails to detect the target if the chosen parameter which indicates the radius of the symmetry region is much larger than the actual value.

To identify the target in Fig. 2, we use a combination of circle and lines, and circle detection is the key. Traditionally, the most popular circle detection techniques are based on the famous circle Hough transform (CHT); however, these techniques are very slow and memory-demanding and produce many false detections. To overcome the limitations of the classical CHT-based methods, many variants have been proposed including randomized HT[16], GPU rasterizer[17] and local voting.[18] All these methods try to correct different shortcomings of CHT but are still memory-demanding and slow to be of any use in real-time applications. Apart from the CHT-based methods, there are several circle detection algorithms based on other methods including clonal selection algorithm (CSA)[19], center clustering[20] and electro-magnetism optimization[21], but they are still far from being real-time. The authors' research group proposes a real-time robust algorithm for circle detection based on the circle-fitting technique presented by Wu Jianping.[22] The new algorithm has a lower false detection rate, a higher detection speed and a smaller memory amount.

3. Overview of the proposed algorithm

The general idea of the proposed algorithm is to identify circles in the image and then identify the target by selecting the one that has straight lines beside the circle.

As presented in Fig. 3, it follows several steps to recognize the cooperative target in a given image. First, a 5×5 Gaussian filter with $\delta = 1.0$ is hired to reduce noises. Then, it extracts edges by adaptive thresholding and improved non-maximum suppression. The adaptive thresholding keeps only the interested edges and ignores a large amount of irrelevant edges in the background. The improved non-maximum suppression refines the edges to mostly single-pixel-width. Taking advantage of the circular edge's smooth change in tangential direction, a unique edge tracking method is performed. Then we detect circles by invalid edge removal, valid arc extraction and circle reassembly. A large number of invalid edges are removed by applying the principle that the mid-perpendiculars of two chords on the same circle intersect at the center of the circle. According to a reliable validation criterion, valid circular arcs are extracted and reassembled. Acquiring the circles, we identify the target by choosing the correct circle. Two different sized square boundaries are set around each circle according to its radius and center. We detect lines in the complement area of the two squares by adopting least square line fitting. The cooperative target is identified if there are certain numbers of lines whose relative positions with the circle match the known target pattern.

The following sections will discuss the main steps of the proposed algorithm in detail.

3.1. Edge extraction

The background of the cooperative target is complex. When the lighting condition, background or distance varies, the amount and strength of the edges in the image are different. Take lighting condition for instance. As shown in Fig. 4, both of the amount and strength of the edges grow as the lighting

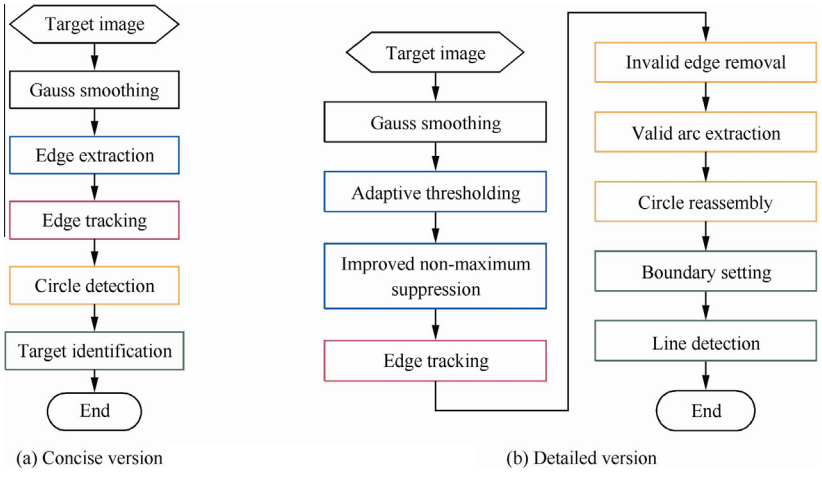


Fig. 3 Architecture of the proposed algorithm.

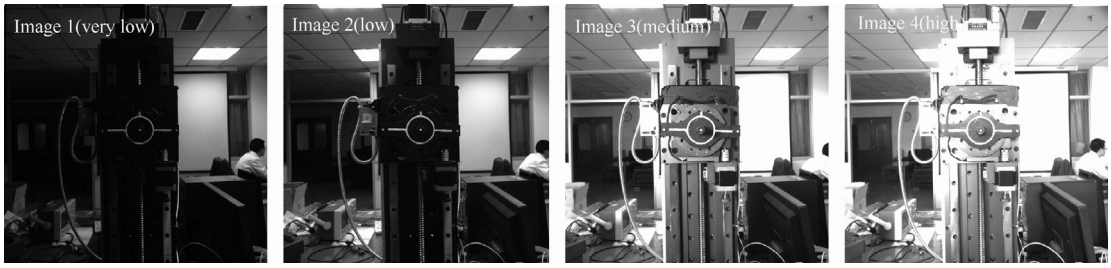


Fig. 4 Images of cooperative target under different lighting conditions.

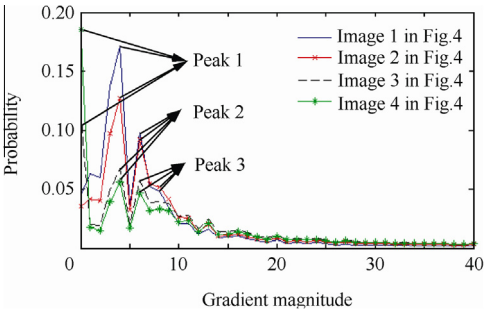


Fig. 5 Gradient magnitude histogram.

intensity increases. Fig. 5 shows the gradient magnitudes histogram of the Gaussian-smoothed images 1, 2, 3, 4 in Fig. 4. Peak 1, the largest peak appears near the origin and corresponds to smooth areas. After Peak 1, the curves continue to fluctuate, and generate the second largest peak Peak 2, the third largest peak Peak 3 and other smaller peaks. After Peak 3, the curves gradually decrease and approach approximately 0 when the gradient magnitude becomes 35. With the increase of the illumination intensity, the fluctuations after the largest peak decrease and the overall distributions of the histogram become more average.

For the cooperative target identification, we aim to keep the target edge while removing other irrelevant edges in the background. As shown in image 1 in Fig. 4, the strengths of the target edges are weaker than those of the edges in the background due to uneven illumination intensity. From Fig. 4, it is clear that excessive irrelevant edges appear when the lighting is strong, thus leading to the increase of time and storage capacity in the subsequent steps. Therefore, the key of edge extraction is to remain the edges of interest.

The well-known Canny[23] algorithm thinks that the proportion of edge pixels in a given image belongs to a certain interval, hence detecting excessive irrelevant edges. We calculate the gradient magnitude using Sobel operator, use an adaptive threshold to extract rough edges and then use improved non-maximum suppression to refine the edges. The adaptive threshold saves the target edges and removes a large amount of useless background edges; the improved non-maximum suppression has better refining effects. Fig. 6 gives a comparison of the edges obtained from Fig. 4 by Canny and our edge extraction algorithm.

As shown in Fig. 6(a), most of the edges extracted cvCanny are not single-pixel-width; the number of edges increases rapidly as the lighting becomes stronger. Fig. 6(b) shows that the edges generated using the proposed edge extraction algorithm are mostly single-pixel-width. In each of the 4 lighting conditions, the target edges remain clear, whereas the number of background edges is relatively smaller than that of cvCanny.

The adaptive thresholding and the improved non-maximum suppression are explained in detail as follows.

3.1.1. Adaptive thresholding

Having performed Gauss smoothing and edge strength calculation, the gradient magnitude of each pixel $g(x, y)$ is obtained:

$$g(x,y) = \sqrt{g_x^2 + g_y^2} \tag{1}$$

where g_x and g_y are the gradient of pixel (x, y) in X direction and Y direction, respectively.

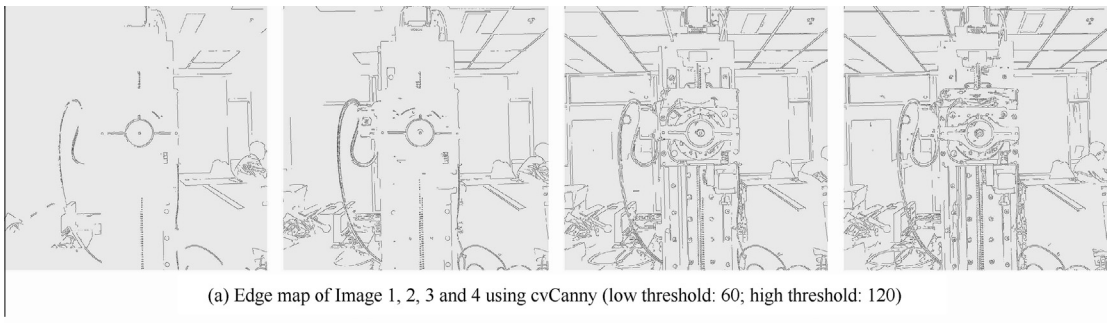
As shown in Fig. 5, the majority of the pixels in a given image have low gradient magnitudes and the probability shows a decreasing trend as the magnitude increases. Therefore, as shown in Fig. 7, we non-uniformly discretize the gradient magnitude percentage histogram. Make $G_{\max}$ and $G_{\min}$ the largest and smallest gradient magnitude in a given picture. The magnitude of an image can be divided into several non-uniform intervals: $[G_{i+1}, G_i]$ $(i = 0, 1, \ldots, n)$, where $G_0 = G_{\max}$ and $G_{i+1} = (G_i + G_{\min})/2$ $(i = 1, 2, 3, \ldots, n)$; therefore, the probability of each interval H_i $(i = 0, 1, \ldots, n)$ can be calculated.

Let H_{edge} be the percentage of pixels that would be classified as edges and H_{smooth} the smooth areas. H_{edge} and H_{smooth} are determined by H_i $(i = 0, 1, \ldots, n)$:

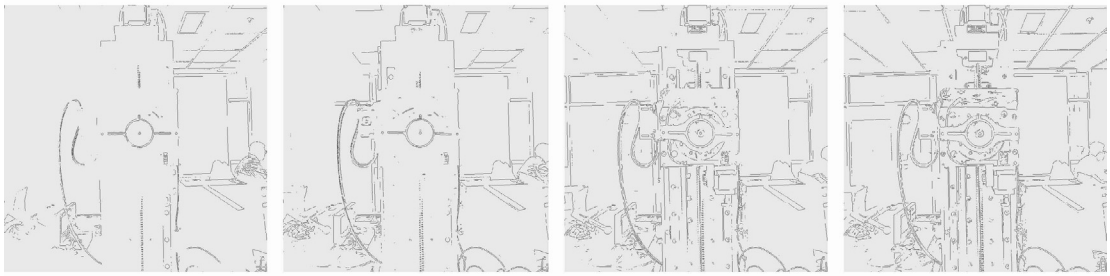
$$\begin{cases} H_{\text{edge}} = \sum_{i=0}^{3} H_i \\ H_{\text{smooth}} = 1 - \sum_{i=0}^{5} H_i \end{cases} \tag{2}$$

As mentioned in Section 1, the target is painted by two flat paints. The reflectivity of the white paint is above 60%, and that of the black one is under 10%. Because of such diffuse reflection, the gradient magnitudes of the target edges are relatively strong. It is clear from Fig. 7 that G_4 is a medium strong gradient-magnitude. According to the values of H_{edge} and H_{smooth}, an adaptive gradient-magnitude threshold is chosen by multiplying different coefficients with G_4.

As shown in Table 1, the highest threshold is chosen when the proportions of the edge pixels H_{edge} are greater than 20% and those of the smooth pixels are lower than 70%. In these situations, the gradient magnitude distributions are similar to the green curve in Fig. 5. It indicates that the target edges are quite strong, hence we multiply G_4 with a high coefficient as the threshold. The case 7 in Table 1 indicates that more than



(a) Edge map of Image 1, 2, 3 and 4 using cvCanny (low threshold: 60; high threshold: 120)



(b) Edge map of Image 1, 2, 3 and 4 using the proposed algorithm (adaptive threshold: 55, 87, 99, 111)

Fig. 6 Edge maps using cvCanny and the proposed algorithm.

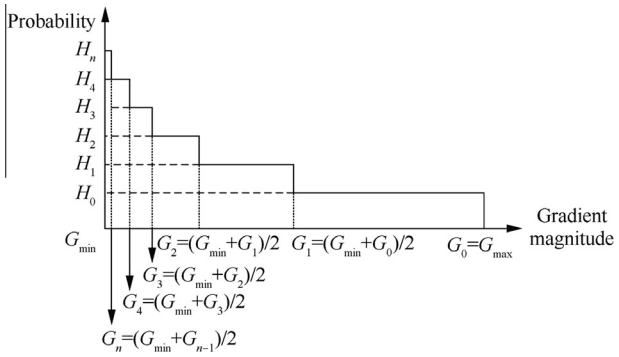


Fig. 7 Discretization of gradient magnitude.

90% of the pixels belong to smooth areas while only less than 10% pixels have relatively strong magnitude gradients. That means the target edges have medium gradient magnitudes as Image 1 in Fig. 4. Therefore, the lowest threshold is chosen.

Given an image, an adaptive threshold T_G is calculated using Table 1 and pixels whose gradient-magnitudes are greater than T_G are extracted as edges.

3.1.2. Improved non-maximum suppression

In order to generate single-pixel-width edges, an improved non-maximum suppression method has been proposed. Its basic idea is to change the weighting coefficients of the comparing points to obtain local maximum. As shown in Fig. 8, we divide all the edge pixels into 4 cases according to their gradient in X direction g_x and Y direction g_y.

Let the current edge pixel be (x, y) and its gradient magnitude be $g(x, y)$. In each of the 4 cases, we choose different 4 pixels in the 8 neighborhood of (x, y) as comparing points. Make g_i ($i \in [1, 4]$) the gradient magnitude of the comparing points and w a weighted coefficient which will be mentioned later.

Table 1 Gradient-magnitude threshold choosing criterion.

Case	H_{smooth}	H_{edge}	T_G
1	$\in[0, 0.7)$	$\in[0.2, 1]$	$2G_4$
2		$\in[0.1, 0.2)$	$1.8G_4$
3		$\in[0, 0.1)$	$1.6G_4$
4	$\in[0.7, 0.9)$	$\in[0.2, 1]$	$1.4G_4$
5		$\in[0.1, 0.2)$	$1.2G_4$
6		$\in[0, 0.1)$	G_4
7	$\in[0.9, 1)$	$\in[0, 0.1]$	$0.8G_4$

In Table 2, both coefficient k_1 and k_2 are at the interval of (0, 1]. Edges become thinner as their values decrease and will be broken if they decrease to a certain extent.

Let w be the weighted coefficient, g_{12} be the weighted average values of g_1 and g_2, and g_{34} be the weighted average value of g_3 and g_4. Using Eq. (3), we obtain the values of g_{12} and g_{34}.

$$\begin{cases} g_{12} = wg_1 + (1-w)g_2 \\ g_{34} = wg_3 + (1-w)g_4 \end{cases} \tag{3}$$

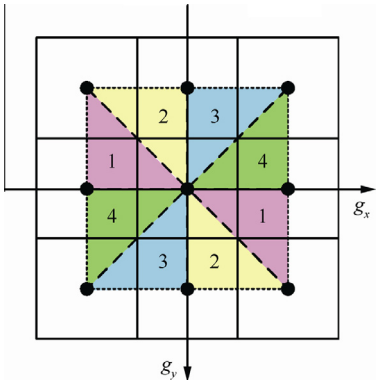


Fig. 8 Dividing diagram along gradient direction.

Table 2 Values for magnitude gradient of comparing points and weighted coefficient in 4 cases.

Case	Gradient value		Parameter value				
	$g_x\ g_y$	$\lvert g_y\rvert/\lvert g_x\rvert$	g_1	g_2	g_3	g_4	w
1	>0	<0	$g(x+1, y+1)$	$g(x+1, y)$	$g(x-1, y-1)$	$g(x-1, y)$	$k_2\cdot\lvert g_y\rvert\lvert g_x\rvert$
2	>0	>0	$g(x-1, y-1)$	$g(x, y-1)$	$g(x+1, y+1)$	$g(x, y+1)$	$k_1\cdot\lvert g_x\rvert\lvert g_y\rvert$
3	<0	>0	$g(x+1, y-1)$	$g(x, y-1)$	$g(x-1, y+1)$	$g(x, y+1)$	$k_1\cdot\lvert g_x\rvert\lvert g_y\rvert$
4	<0	<0	$g(x+1, y-1)$	$g(x+1, y)$	$g(x-1, y+1)$	$g(x-1, y)$	$k_2\cdot\lvert g_y\rvert\lvert g_x\rvert$

If the gradient magnitude of the current pixel $g(x, y)$ is greater than both g_{12} and g_{34}, this point is defined as a true edge pixel and is remained; otherwise, the point is not an edge pixel and removed.

As shown in Fig. 9(a), using the convolution of 2-dimensional Gauss template and a perfect circle as the original image, Fig. 9(b) and Fig. 9(c) are the results obtained by algorithm in Ref. 24 and the proposed algorithm, respectively. Fig. 9(d) and Fig. 9(e) are the local images of Fig. 9(b) and Fig. 9(c). The edges obtained by the traditional method are wider. Most edges are not single-pixel-width, especially on left-top, left-bottom, right-top and right-bottom. However, most of the edges obtained by our algorithm are single-pixel-width.

3.2. Edge tracking

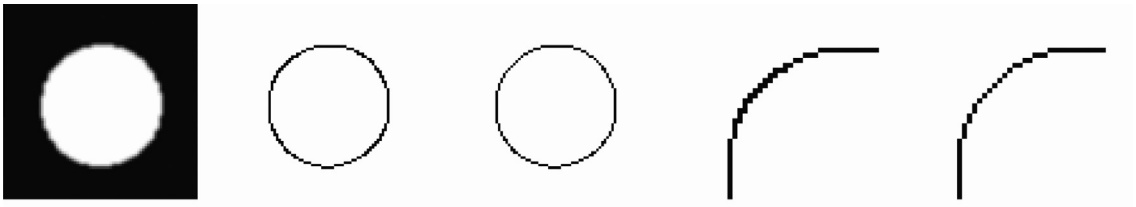
After edge extraction, we cluster the edge pixels. As shown in Fig. 10(a), the arc on the target is no longer complete when the column in the target center blocks part of the arc. Fig. 10(b) is the edges extracted from Fig. 10(a) and Fig. 10(c) is the local image of Fig. 10(b). From the two red circles in Fig. 10(c), it is clear that the inner and outer edges of the ring on the target are connected into one edge because of the occlusion. Circular edges change smoothly in tangential directions; hence, a novel edge tracking technique is proposed to break the edges at the points which change abruptly in tangential directions. Fig. 10(d) shows the result of edge tracking using the proposed method and Fig. 10(e) is the local image. It is clear from Fig. 10(e) that our tracking strategy disconnects the edge where the arc is blocked, therefore reducing the difficulty of subsequent circle detection. (In order to show the tracking results more clearly, the first and last points of each edge is not drawn in Fig. 10(d) and Fig. 10(e).) The following describes the steps of our edge tracking strategy in detail.

For each edge point $P(x, y)$, we define that its adjacent points are in the 8 directions as shown in Fig. 11. Based on this definition, our edge tracking follows the steps as below to cluster edge pixels:

Step 1. Find the start point. Start at the left-top of the given image and search the first untracked edge pixel from left to right, from up to down. This point is the top of an edge and is denoted as StartP.

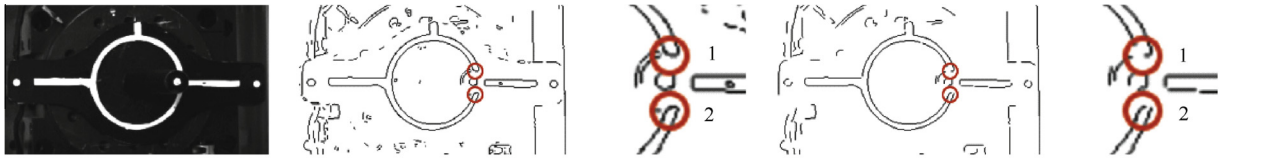
Step 2. Find the second. Search the next untracked edge point in the 0–3 directions of StartP, because direction 4–7 have already been searched in Step 1. If a new point is found, we define it as SecondP; if not, delete StartP because it is an isolated pixel and turn to Step 1.

Step 3. Track point by point. Make CurrentP the current point, SeachDir the searching direction from the last point to CurrentP and Avedir the average value of the search directions of last n points. As shown in Fig. 12, search for the next edge point still on SearchDir that is direction D_1. If there is no untracked edge point, look for one clockwise from direction D_2 to D_7 in turn; however, do not search in those directions, if the difference between Avedir and search direction is greater than 2 and more than a minimal pixel length, say, 10 pixels have been tracked from StartP to CurrentP. If an untracked edge point is found on the required directions as above, pass the coordinate value of this point to CurrentP and turn to Step 3. If not, turn to Step 4 because the edge has ended.



(a) Original image (b) Algorithm in Ref.[18] (c) Proposed algorithm (d) Local image of Fig.9(b) (e) Local image of Fig.9(c)

Fig. 9 Results of different non-maximum suppression methods.



(a) Original image (b) Edge map of Fig.10(a) (c) Local image of Fig.10(b) (d) Traced edges using the proposed algorithm (e) Local image of Fig.10(d)

Fig. 10 Blocking of arc by target column.

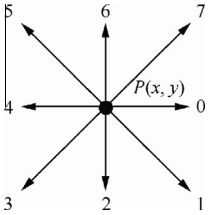


Fig. 11 Eight directions from $P(x, y)$ to its adjacent points.

Step 4. Output an edge. If the length of the current edge is longer than ten pixels, output the coordinates of all the points on the edge from StartP to the end.

3.3. Circle detection

After clustering edge pixels, circles in the image are identified with a very small amount of computational cost by 3 steps. Initially, a large amount of invalid edges are easily removed using the principle that the mid-perpendiculars of two chords on the same circle intersect at the center of the circle. Then each of the remained edges is split into two halves and fitted into a circle respectively. If the fitting results of the two halves are similar, the edge is identified as a circular one. Lastly, we reassemble the edges that belong to the same circle. Each of the three steps will be discussed in detail in the following section.

3.3.1. Invalid edge removal

As shown in Fig. 13, edge E is split into E_1 and E_2 from the middle. P_1, P_2 and P_3 are the first point, midpoint and last point of E_1 respectively; P_4, P_5 and P_6 are the first point, midpoint and last point of E_2 respectively. l_1 is the mid-perpendicular of chord P_1P_2, and l_2 is that of chord P_2P_3. l_1 and l_2 intersect at point P_{c1}, and the Euclidean distance between P_{c1} and P_2 is R_{c1}. l_3 is the mid-perpendicular of chord P_4P_5 and l_4 is that of chord P_5P_6. l_3 and l_4 intersect at point P_{c2} and the Euclidean distance between P_{c2} and P_5 is R_{c2}.

If edge E is an ideal circular edge, P_{c1} and P_{c2} will overlap at the center of the circle, and both R_{c1} and R_{c2} will be equal to the circle's radius. In actual situation, P_{c1} and P_{c2} do not coincide in the center of the circle, but are very close to each other. Likewise, R_{c1} and R_{c2} are approximately the same as the radius.

Assuming that n is the length of edge E and $\{(x_i, y_i), i \in [1, n]\}$ denotes the coordinates of the pixels on edge E, the coordinates of P_1 to P_6 are shown in Table 3. Symbol $\lfloor\ \rfloor$ in Table 3 means rounding down, *i.e.* $\lfloor y \rfloor$ is the largest integer that does not exceed y.

Using the above coordinates, the linear equations of l_1, l_2, l_3 and l_4 are

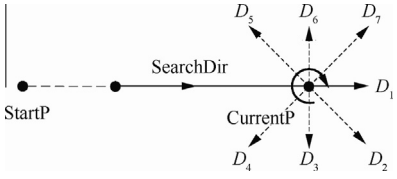


Fig. 12 Point by point edge tracking directions.

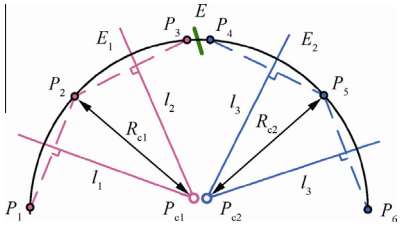


Fig. 13 Schematic of invalid edges removal.

$$\begin{cases} l_1: y = a_1x + b_1 \quad \left(a_1 = -\frac{x_{P_1} - x_{P_2}}{y_{P_1} - y_{P_2}},\right. \\ \quad \left. b_1 = \frac{1}{2}\left[y_{P_1} + y_{P_2} - a_1(x_{P_1} + x_{P_2})\right]\right) \\ l_2: y = a_2x + b_2 \quad \left(a_2 = -\frac{x_{P_2} - x_{P_3}}{y_{P_2} - y_{P_3}},\right. \\ \quad \left. b_2 = \frac{1}{2}\left[y_{P_2} + y_{P_3} - a_2(x_{P_2} + x_{P_3})\right]\right) \\ l_3: y = a_3x + b_3 \quad \left(a_3 = -\frac{x_{P_4} - x_{P_5}}{y_{P_4} - y_{P_5}},\right. \\ \quad \left. b_3 = \frac{1}{2}\left[y_{P_4} + y_{P_5} - a_3(x_{P_4} + x_{P_5})\right]\right) \\ l_4: y = a_4x + b_4 \quad \left(a_4 = -\frac{x_{P_5} - x_{P_6}}{y_{P_5} - y_{P_6}},\right. \\ \quad \left. b_4 = \frac{1}{2}\left[y_{P_5} + y_{P_6} - a_4(x_{P_5} + x_{P_6})\right]\right) \end{cases} \tag{4}$$

where a_i ($i \in [1, 4]$) and b_i ($i \in [1, 4]$) are the slope and intercept of l_i, respectively.

If l_1 and l_2 are collinear, i.e. the angle between them is smaller than 10° then l_1 is approximately parallel to l_2. Hence E is not a circular edge because the intersection of $l_1$and l_2 is too far away. Likewise, E is not a circular edge if l_3 and l_4 are collinear. If l_1 and l_2 are not collinear and neither are l_3 and l_4, P_{c1}, P_{c2}, R_{c1} and R_{c2} are calculated:

$$\begin{cases} (x_{P_{c1}}, y_{P_{c1}}) = \left(-\frac{b_1 - b_2}{a_1 - a_2}, \quad a_1x_{P_{c1}} + b_1\right), \\ \quad R_{c1} = \sqrt{(x_{P_2} - x_{P_{c1}})^2 + (y_{P_2} - y_{P_{c1}})^2} \\ (x_{P_{c2}}, y_{P_{c2}}) = \left(-\frac{b_3 - b_4}{a_3 - a_4}, \quad a_3x_{P_{c2}} + b_3\right), \\ \quad R_{c2} = \sqrt{(x_{P_5} - x_{P_{c2}})^2 + (y_{P_5} - y_{P_{c2}})^2} \end{cases} \tag{5}$$

Define parameters S_1, S_2, S to describe the similarities between P_{c1}, R_{c1} and R_{c2}, P_{c2}:

$$S = S_1 \cdot S_2 = \left(1 - \frac{2|R_{c1} - R_{c2}|}{R_{c1} + R_{c2}}\right)\left(1 - \frac{2\sqrt{(x_{P_{c1}} - x_{P_{c2}})^2 + (y_{P_{c1}} - y_{P_{c2}})^2}}{R_{c1} + R_{c2}}\right) \tag{6}$$

For a perfect circle, parameters S_1, S_2 and S are all equal to 1. Therefore, if S_1, $S_2 \in (0, 1]$ and $S \in (k_3, 1]$, where $\{k_3 \in \mathrm{R} | k_3 > 0, k_3 < 1\}$, E is likely to be a circular edge. We save the edges that meet these requirements and remove those that do not. In this way, a large amount of invalid edges are removed with a small amount of time and calculation.

3.3.2. Valid arc extraction

Each of the remained edge E is divided into E_1 and E_2 from the middle, as shown in Fig. 14. Then we use least square fitting method to fit E_1 into a circle, and E_2 likewise. P_{c1} and R_{c1} are the center and radius of E_1 respectively; P_{c2} and R_{c2} are the center and radius of E_2 respectively. If the fitting results of E_1 and E_2 are similar, the entire edge E is fitted to a circle.

Table 3 Coordinates of P_1–P_6.

Point	P_1	P_2	P_3	P_4	P_5	P_6
Coordinate	(x_{P_1}, y_{P_1})	(x_{P_2}, y_{P_2})	(x_{P_3}, y_{P_3})	(x_{P_4}, y_{P_4})	(x_{P_5}, y_{P_5})	(x_{P_6}, y_{P_6})
Value	(x_1, y_1)	$(x_{\lfloor n/4 \rfloor}, y_{\lfloor n/4 \rfloor})$	$(x_{\lfloor n/2 \rfloor}, y_{\lfloor n/2 \rfloor})$	$(x_{\lfloor n/2 \rfloor+1}, y_{\lfloor n/2 \rfloor+1})$	$(x_{\lfloor 3n/4 \rfloor}, y_{\lfloor 3n/4 \rfloor})$	(x_n, y_n)

P_c and R_c are its center and radius, respectively. Take an edge E as an example and the circle fitting procedure is shown in Fig. 14.

Make n the length of edge E. For each point (x_i, y_i), $i \in [1, n]$ on E, the following equation stands:

$$(x - x_{P_c})^2 + (y - y_{P_c})^2 = R_c^2 \tag{7}$$

Make $a = -2x_{Pc}$, $b = -2y_{Pc}$, $c = x_{Pc}^2 + y_{Pc}^2 - R_c^2$, hence another form of the above equation is

$$x^2 + y^2 + ax + by + c = 0 \tag{8}$$

The fitting results are obtained by calculating the values of a, b and c. Therefore, we set up an objective function $F(a, b, c)$:

$$F(a,\ b,\ c) = \sum_{i=1}^{n} (x_i^2 + y_i^2 + ax_i + by_i + c)^2 \tag{9}$$

When $F(a, b, c)$ reaches its minimum, the corresponding values of a, b and c are the best choice. Hence, we calculate the partial differentials of $F(a, b, c)$ to a, b and c respectively and let all the results be 0:

$$\begin{cases} \dfrac{\partial F}{\partial a} = \displaystyle\sum_{i=1}^{n} 2(x_i^2 + y_i^2 + ax_i + by_i + c)x_i = 0 \\ \dfrac{\partial F}{\partial b} = \displaystyle\sum_{i=1}^{n} 2(x_i^2 + y_i^2 + ax_i + by_i + c)y_i = 0 \\ \dfrac{\partial F}{\partial c} = \displaystyle\sum_{i=1}^{n} 2(x_i^2 + y_i^2 + ax_i + by_i + c) = 0 \end{cases} \tag{10}$$

Using elimination method, we derive the following results:

$$\begin{cases} a = \dfrac{T_2T_5 - T_3T_4}{T_1T_4 - T_2^2} \\ b = \dfrac{T_1T_5 - T_2T_3}{T_2^2 - T_1T_4} \\ c = -\dfrac{\displaystyle\sum_{i=1}^{n}(x_i^2+y_i^2) + a\sum_{i=1}^{n}x_i + b\sum_{i=1}^{n}y_i}{n} \end{cases} \tag{11}$$

where coefficients T_i ($i = 1, 2, \ldots, 5$) are

$$\begin{cases} T_1 = n\displaystyle\sum_{i=1}^{n}x_i^2 - \sum_{i=1}^{n}x_i\sum_{i=1}^{n}x_i \\ T_2 = n\displaystyle\sum_{i=1}^{n}x_iy_i - \sum_{i=1}^{n}x_i\sum_{i=1}^{n}y_i \\ T_3 = n\displaystyle\sum_{i=1}^{n}x_i^3 + n\sum_{i=1}^{n}x_iy_i^2 - \sum_{i=1}^{n}(x_i^2 + y_i^2)\sum_{i=1}^{n}x_i \\ T_4 = n\displaystyle\sum_{i=1}^{n}y_i^2 - \sum_{i=1}^{n}y_i\sum_{i=1}^{n}y_i \\ T_5 = n\displaystyle\sum_{i=1}^{n}x_i^2y_i + n\sum_{i=1}^{n}y_i^3 - \sum_{i=1}^{n}(x_i^2 + y_i^2)\sum_{i=1}^{n}y_i \end{cases} \tag{12}$$

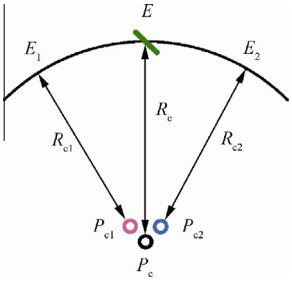


Fig. 14 Schematic of valid edges extraction.

The center point and radius of the fitted circle are calculated from a, b and c:

$$\begin{cases} (x_{P_c},\ y_{P_c}) = \left(-\dfrac{a}{2},\ -\dfrac{b}{2}\right) \\ R_c = \dfrac{\sqrt{a^2 + b^2 - 4c^2}}{2} \end{cases} \tag{13}$$

A relative standard deviation is defined as below to indicate the computational accuracy:

$$\sigma_c = \frac{1}{R_c}\sqrt{\frac{\displaystyle\sum_{i=1}^{n}\left|(x_i - x_{P_c})^2 + (y_i - y_{P_c})^2\right|}{n}} \tag{14}$$

If $\sigma_c < k_4R_c$ where k_4 is a constant and $k_4 \in (0, 1)$, it means that the edge points fall around the circle defined by center P_c and radius R_c closely, and E is likely to be a circular edge. If not, E is not likely to be a circular edge.

If both the standard deviations of E_1 and E_2 meet the above requirement, use Eq. (6) to calculate the similarity coefficients S_1, S_2 and S. If S_1, $S_2 \in (0, 1]$ and $S \in [k_5, 1]$ where constant $k_5 \in (0.7, 1)$, E_1 and E_2 belong to the same circle. Then the final results, center P_c and radius R_c, are obtained by fitting the entire edge E using least square fitting method.

3.3.3. Circle reassembly

Some of the above extracted valid arcs belong to the same circles, because an entire arc is probably divided into several sections at the edge extraction or clustering stage. Use Eq. (6) to obtain these edges and reassemble them to a bigger pixel cluster, then calculate the new fitting results using Eq. (13). This will eliminate duplicate circle information and results in more accurate parameters for the circle.

Next, some of the unwanted circles are removed if they meet either of the following 2 conditions:

1. The ratio of the radius of the fitted circle to the width of the target image is bigger than 1/5 or less than 1/100. This is because that the distance between the target and the hand-eye camera is in the range of [0.3, 1.5] m.

2. The length of the edge is less than 1/4 of its fitted circle's perimeter. The arc on the target is a complete circle under most circumstances; hence we do not identify minor arcs as circles.

3.4. Target identification

Even through circles rarely appear in space station, it is not practical to identify the cooperative target only by circle detection. We improve the accuracy of target identification by detecting the straight lines on the target. The line detection is first assisted by boundary setting.

3.4.1. Boundary setting

For each detected circle, two differently sized square boundaries are set around it as shown in Fig. 15. Once the cooperative target is manufactured, the physical sizes of the circle, lines and dots are known. Here some symbols are defined to describe them. Make R the inner radius of the ring with W as its width, D the distance between the target center and the end of the lines and Δ a constant determined by the size of the target. By applying the circle detection algorithm described in Section 3.4, we can easily obtain the circle parameters in a given target image. They are radius R_c and circle center (x_{Pc}, y_{Pc}).

We set up two boundaries using the above parameters, and (x_1, y_1) to (x_4, y_4) are denoted to describe their positions in the image. (x_1, y_1) and (x_2, y_2) are the left top of the larger and smaller boundaries respectively; (x_3, y_3) and (x_4, y_4) are the right bottom of the smaller and larger boundaries respectively. According to the pinhole camera model, each point in the world coordinate is projected on the image plane. Therefore, the values of (x_1, y_1) to (x_4, y_4) are obtained according to geographical relationships:

$$\begin{cases} (x_1,y_1) = \left(x_c - \frac{R_c}{R}(D-\Delta)\cos 45^\circ,\ y_c - \frac{R_c}{R}(D-\Delta)\sin 45^\circ\right) \\ (x_2,y_2) = \left(x_c - \frac{R_c}{R}(R+W),\ y_c - \frac{R_c}{R}(R+W)\right) \\ (x_3,y_3) = \left(x_c + \frac{R_c}{R}(R+W),\ y_c + \frac{R_c}{R}(R+W)\right) \\ (x_4,y_4) = \left(x_c + \frac{R_c}{R}(D-\Delta)\cos 45^\circ,\ y_c + \frac{R_c}{R}(D-\Delta)\sin 45^\circ\right) \end{cases} \tag{15}$$

3.4.2. Line detection

In the complement area of the two squares, we perform edge tracking and save the edges whose lengths are larger than k_6R_c, where $\{k_6 \in \mathrm{R}|\ k_6 > 0,\ k_6 < 1\}$. Fig. 16 displays the ideal tracking results. The solid lines are the edges tracked in the complement area, while the dashed lines are the edges outside that area. Then edges in the complement area are fitted into straight lines using least square fitting method. For each edge, we calculate σ_L and d using Eq. (16), which are the relative standard deviation of the fitting results and the Euclidean distance from the lines to the circle center respectively.

$$\begin{cases} \sigma_E = \dfrac{1}{R_c}\sqrt{\dfrac{\sum_{i=1}^{n}(y_i - ax_i - b)^2}{n}} \\ d = \dfrac{|y_c - ax_c - b|}{\sqrt{1+a^2}} \end{cases} \tag{16}$$

If the distance d is smaller than k_7R_c, where k_7 is a constant and $\{k_7 \in \mathrm{R}|\ k_7 > 0,\ k_7 < 1\}$, it means that the edge is close to circle center. If the deviation σ_E is less than 1, it means the edge is a straight line. An edge that meets both requirements is identified as a straight line on the target.

As shown in Fig. 16 and Fig. 4 straight lines, No. 1–4, are in the complement areas of the 2 boundaries in ideal situation; however, in actual situation, it is not the case. Illumination may be uneven, too strong or too week. Glittering spots may appear in the target image. The straight lines may be jagged when looked closely because of the manufacturing. Therefore, straight lines on the target are probably broken or curved due to various reasons. A figure is identified as the cooperative target if 2–6 such lines are found.

4. Experiments

In order to validate the target recognition algorithm, we conducted experiments from 3 aspects: circle detection, line detection and target recognition.

4.1. Circle detection

We fixed the cooperative target on the to-be-arrested device. The hand-eye camera on the space robot arm took the images of the target as the distance between them varies from 1.5 m to 0.3 m. Fig. 17(a) shows the original target images when the distances are at 1.5 m, 0.9 m, 0.6 m and 0.3 m respectively. Fig. 17 (b)–Fig. 17(d) demonstrate the performance of 3 different circle detection algorithms on these target images. The running times were measured in a PC with a 3.4 GHz Intel (R) Core (TM) i3-2130 CPU and 1.96G SDRAM. The size of each image is 1024 × 1024 pixels.

As shown in Fig. 17(b), cvHough detected a small amount of circles in each image, whereas a false circle was detected in the second image. From Fig. 17(c), it is clear that the algorithm proposed by Wu Jianping[16] detected more than 10 circles in each image and a large proportion of circles are false. Fig. 17(d) shows

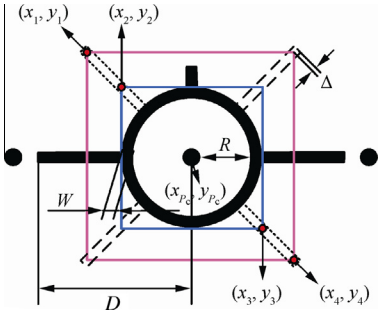


Fig. 15 Schematic diagram of coordinate calculation.

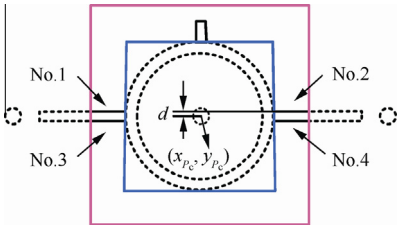


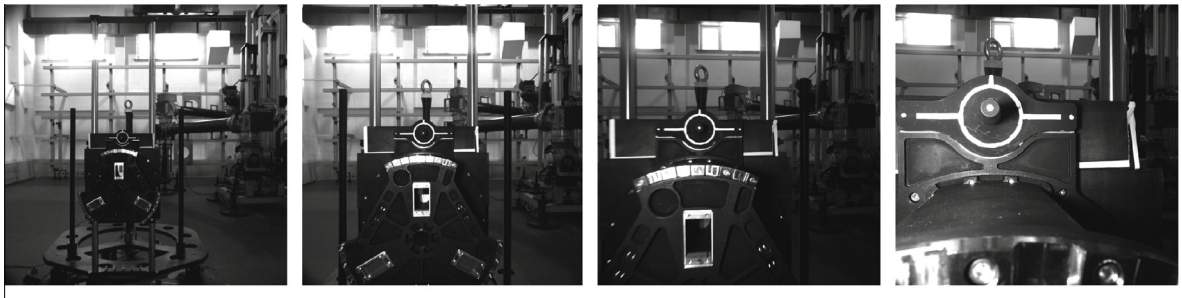
Fig. 16 Edges in complement area of two boundaries.

the detection results obtained by our algorithm. It detected the ring on the target accurately and ignored a large number of the circles in the background. No false circle was detected.

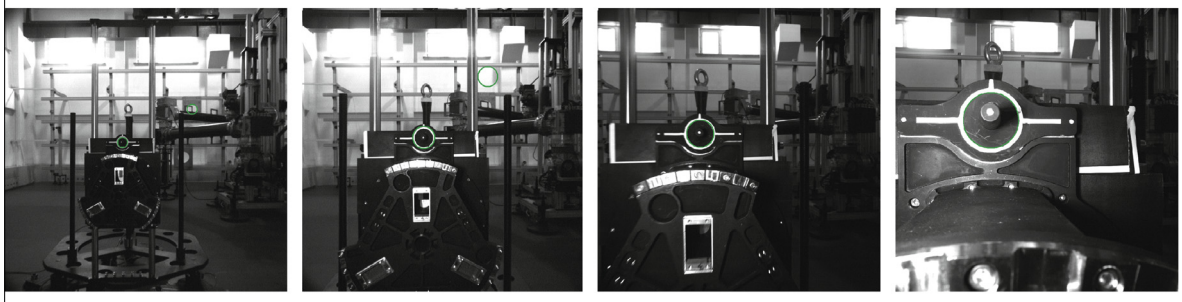
Table 4 dissects the running time of the proposed circle detection algorithm for each image in Fig. 17 and we compares it with cvHoughCircles and the algorithm proposed in Ref.[22]. Clearly, our algorithm runs real-time for hand-eye camera input sizes of 1024 × 1024 with much of the time spent on edge detection and clustering of edge pixels rather than circle detection. The proposed algorithm is approximately 1.33 times faster than cvHoughCircles and 1.96 times faster than the algorithm proposed by Wu Jianping.

4.2. Line detection

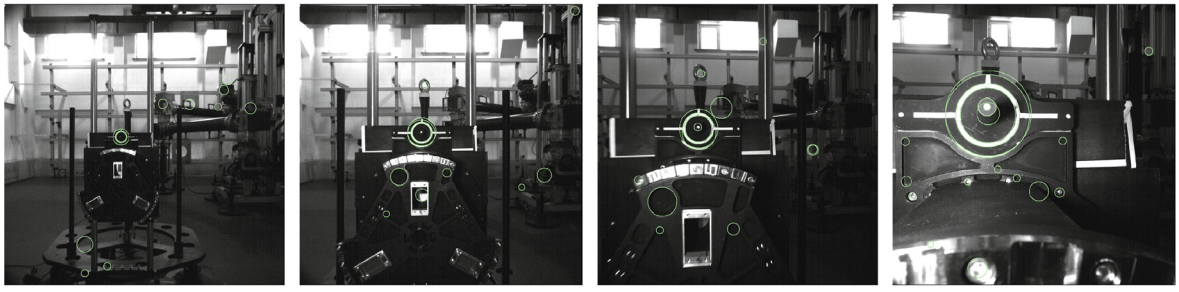
Having obtained the center and radius of each detected circle in a given image, we set up two square boundaries around each circle as mentioned in Section 4.1. Take the third image in Fig. 17(a) as an example; two circles were detected as shown in Fig. 18(a) and Fig. 18(b) is its local image. Table 5 gives the parameters of the 2 detected circles as well as the left-top and right-bottom corners of the 2 squares beside each circle. Fig. 18(c) shows the enlarged image of edges detected in the complement of the 2 different sized squares. Clearly, there are 10 edges near circle No. 1 and 2 edges near circle No. 2.



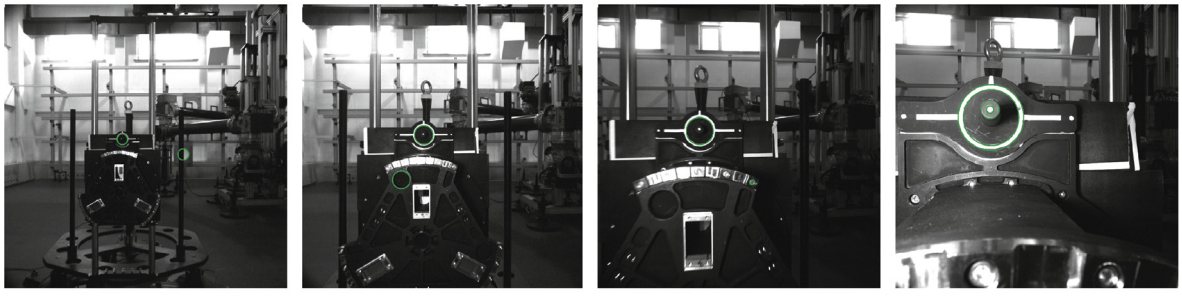
(a) Original image of the cooperative target (distance: 1.5 m, 0.9 m, 0.6 m, 0.3 m)



(b) Circles detected using cvHoughCircles (results: 2 circles in 42 ms, 2 circles in 45 ms, 1 circles in 36 ms, 1 circles in 37 ms)



(c) Circles detected using the Wu Jianping algorithm (results: 12 circles in 15 ms, 12 circles in 31 ms, 12 circles in 16 ms, 18 circles in 31 ms)

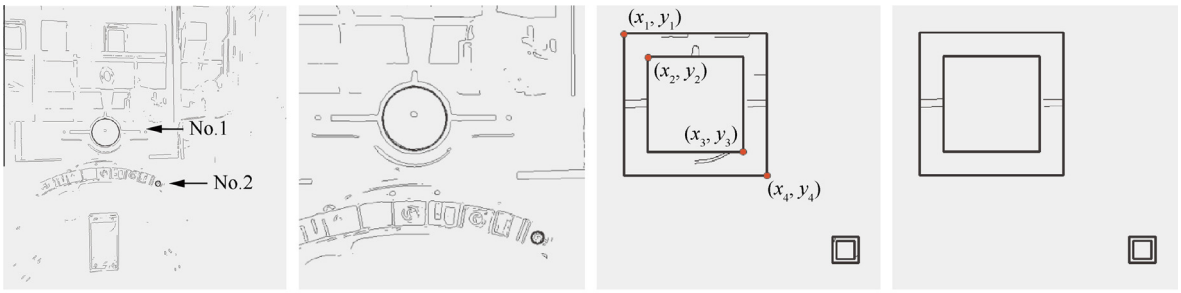


(d) Circles detected using the proposed algorithm (results:2 circles in 15 ms, 2 circles in 16 ms,2 circles in 14 ms, 4 circles in 16 ms)

Fig. 17 Circle detection results with different algorithms.

Table 4 Dissection of running time with different circle detection algorithms.

Distance (m)	The proposed circle detection algorithm (ms)				cvHoughCircles (ms)	Wu Jianping' algorithm (ms)
	Edge detection	Edge tracking	Circle detection	Total		
1.2	21.7	9.5	4.8	36.1	42.8	78.2
0.5	22.1	9.7	4.7	36.4	47.4	62.5
0.4	21.8	8.8	4.1	34.7	46.3	63.3
0.3	21.6	10.2	5.2	36.9	54.6	78.6



(a) Circles detected (b) Local image of Fig.18(a) (c) Edges tracked near circles (d) Straight lines beside circles

Fig. 18 Straight line detection results.

Table 5 Parameters of detected circles and two squares near them.

			Pixel			
Circle	Center	Radius	(x_1, y_1)	(x_2, y_2)	(x_3, y_3)	(x_4, y_4)
No. 1	(370.3, 456.0)	49.2	(279.2, 366.2)	(309.5, 396.5)	(430.5, 517.5)	(460.8, 547.8)
No. 2	(560.5, 641.4)	9.1	(543.2, 625.2)	(548.8, 630.8)	(571.2, 653.2)	(576.8, 658.8)

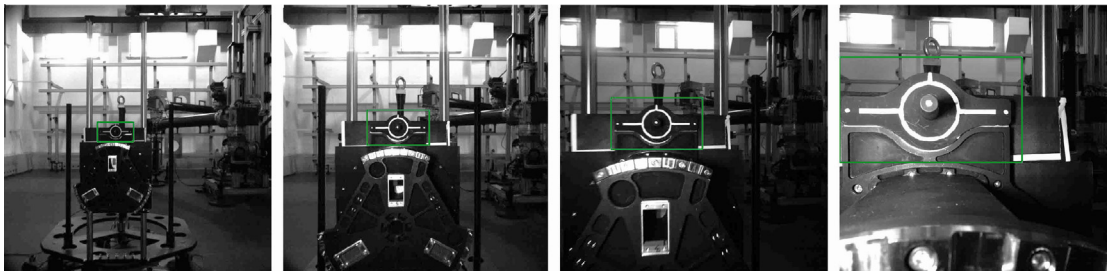


Fig. 19 Recognition results of different distances (distance: 1.5 m, 0.9 m, 0.6 m, 0.3 m).

Using the criterion mentioned in Section 3.4.2, 4 straight lines were discovered near circle No. 1 and none near circle No. 2, as shown in Fig. 18(d). Therefore, circle No. 1 was identified as the cooperative target. For this image, it took 13.4 ms to perform Gauss smoothing; 34.7 ms to detect circles and 4.8 ms to detect the straight lines near circles. In total, it took 53.0 ms for our algorithm to recognize the cooperative target.

4.3. Cooperative target recognition

To test the robustness of the proposed algorithm, we designed the experiments from 3 different aspects: distance, lighting condition and target attitude. Each experiment lasted 3.5 h with the frame rate set at 8 FPS. No frame was lost during all the experiments; hence, 100800 frames of images have been processed in each experiment.

In the first experiment, the cooperative target was fixed on the to-be-arrested device and the hand-eye camera on the space robot arm took the target images as the distance between them varies from 1.5 m to 0.3 m. Fig. 19 shows the recognition results when the distance is at 1.5 m, 0.9 m, 0.6 m and 0.3 m, respectively. The overall recognition accuracy is at 98.7%.

For the second experiment, the cooperative target and the hand-eye camera remained relatively static. During the 3.5 h, we occasionally changed the lighting condition by switching on or off the fluorescent lamps on the ceiling and opening or closing the curtains. Fig. 20 shows the recognition results of the cooperative target when the lighting intensity is very low, low, medium and high. The total accuracy rate of this experiment is a little lower than the first one, at 97.5%.

The last experiment was aimed at measuring the performance of the proposed algorithm as the attitude of the

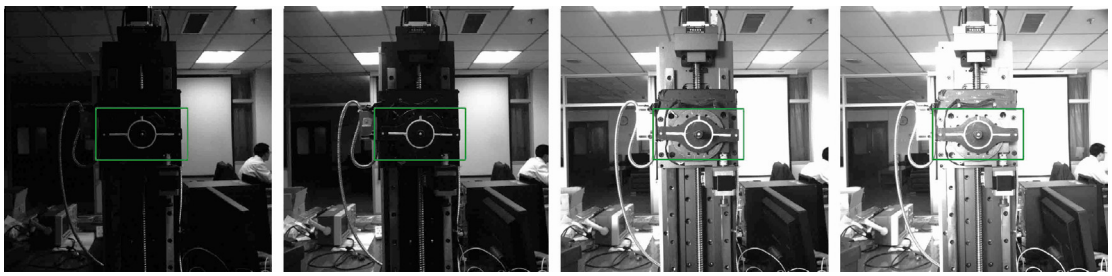


Fig. 20 Recognition results with different lighting conditions (light intensity: very low, low, medium, high).

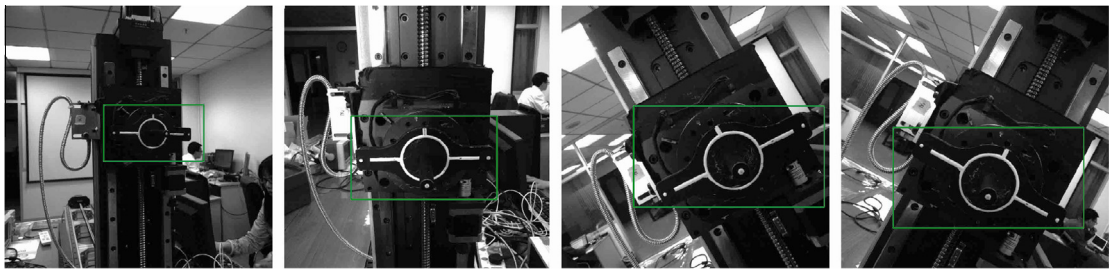


Fig. 21 Recognition results when position and orientation of target vary.

cooperative target changes. The relative distance between the target and the hand-eye camera varied from 0.3 m to 0.6 m because of the change of attitude. Fig. 21 demonstrates the recognition results when the target is of different attitudes. From the first and the second image in Fig. 21, it is clear that the proposed algorithm recognized the target when the ring on the target was partially blocked. Even though the ring on the second image became an ellipse due to the target attitude, our algorithm still recognized the target accurately. In the last two images of Fig. 21, the target turned 15° counterclockwise and 30° clockwise respectively. Due to our straight line detection strategy, the rotation of the target did not affect the recognition results. In this experiment, 98.5% of the identification results were correct.

5. Conclusion

(1) An edge extraction algorithm has been proposed. Using an adaptive threshold, it extracts only the interested edges and largely reduces the memory amount, and the edges are mostly single-pixel-width due to improved non-maximum suppression.
(2) A novel tracking approach is proposed to cluster the pixels that change smoothly in tangential directions.
(3) With a small amount of calculation and storage, circles are detected accurately and in real-time.
(4) Using two square boundaries set around each circle to help detecting lines increases the line detection speed.
(5) A cooperative target identification algorithm is proposed. Regardless of lighting condition and target attitude, it accurately identifies the target within the distance of 0.3–1.5 m under complex background at the processing speed of 8 frames per second. It is practical and effective for visual measurement of space robot arm and can be easily applied to real-time industrial robotic applications.

Acknowledgement

This project was supported by the National Basic Research Program of China (No. 2013CB733103).

References

1. Flores-A Angel, Ou M, Khanh P, Steve U. A review of space robotics technologies for on-orbit servicing. *Progress in Aerospace Sciences* 2014;**68**:1–26.
2. Gibbs G, Sachdev S, Marcotte B. Canada and the International Space Station Program overview and status. *Acta Astronautica* 2002;**51**(1–9):591–600.
3. Xu WF, Wang XQ, Xue Q, Liang B. Study on trajectory planning of dual-arm space robot keeping the base stabilized. *Acta Automat Sin* 2013;**39**(1):69–80.
4. Zhai G, Zhang JR, Zhou ZC. Coordinated target localization base on pseudo measurement for clustered space robot. *Chin J Aeronaut* 2013;**26**(6):1524–33.
5. Wen ZM, Wang YJ, Di N, Jin MH. On-orbit hand-eye calibration using cooperative target. *Chin J Sci Instrum* 2014;**35**(5):1005–12 Chinese.
6. Wen ZM, Wang YJ, Di N, Chu GL, Jin MH. Fast recognition of cooperative target used for position and orientation measurement of space station's robot arm. *Acta Aeronaut Astronaut Sin* 2015;**25**(4):1330–8 Chinese.
7. Donohoe Gregory W. Low-power reconfigurable processor. *Aerospace Conference Proceedings,* 2002 Mar 9-16; Big Sky, USA. Piscataway, NJ: IEEE Press; 2002. p. 1969–73.
8. Stephen RG, Jerry LeCroyb. Analysis and design of solid corner cube reflectors for a space navigation application. *Proceedings of SPIE,* 2005 March 28; Orlando, USA. Bellingham: SPIE; 2005. p. 120–9.
9. Richard TH, Thomas CB, Linda LB. Proximity operations and docking sensor development. *Aerospace conference,* 2009 March 7-14; Big Sky, USA. Piscataway, NJ: IEEE Press; 2009. p. 1–10.
10. Noriyasu I, Mitsushige O. Autonomous satellite capture by a space robot: world first on-orbit experiment on a Japanese robot satellite ETS-VII. *Proceedings of the 2000 IEEE International*

Conference on Robotics & Automation, 2000 Apr 24-28; San Francisco, USA. Piscataway, NJ: IEEE Press. p. 1169–74.
11. David GL. Distinctive image features from scale-invariant keypoints. *Int J Comput Vision* 2004;**60**(2):91–110.
12. Navneet D, Bill T. Histograms of oriented gradients for human detection. *Proceedings of the 2005 IEEE computer society conference on computer vision and pattern recognition,* 2005 Jun 25–25; San Diego, CA, USA. Piscataway, NJ: IEEE Press; 2005. p. 886–93.
13. Stephane GM. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Trans Pattern Anal Mach Intell* 1989;**11**(7):674–93.
14. Giovanni I, Fabio C, Ferrante N, Ernesto M. Robot base disturbance optimization with compact differential evolution light. *Evo Applications* 2012;**7248**:285–94.
15. Christoph D, Regina P-F, Fabian S, Manuel J. The gradient product transform for symmetry detection and blood vessel extraction. *International conference on computer vision theory and applications,* 2015 March 11-14; Berlin, Germany. Berlin: Springer; 2015. p. 177–84.
16. Jiang LY. Efficient randomized Hough transform for circle detection using novel probability sampling and feature points. *Optik – Int J Light Electron Opt* 2012;**123**(20):1834–40.
17. Manuel U, Antonio R, Nicolas G. On the computation of the circle Hough transform by a GPU rasterizer. *Pattern Recogn Lett* 2008;**29**(3):309–18.
18. Yang HP, Luo JC, Shen ZF, Wu W. A local voting and refinement method for circle detection. *Optik – Int J Light Electron Opt* 2014;**125**(3):1234–9.
19. Erick C, Valentin OE, Fernando W, Daniel Z, Marco PC. Automatic multiple circle detection based on artificial immune systems. *Expert Systems with Applications* 2012;**39**(1):713–22.
20. Rudolf S, Tomislav M. Multiple circle detection based on center-based clustering. *Pattern Recogn Lett* 2014;**52**:9–16.
21. Erick C, Diego O, Daniel Z, Marco PC, Humberto S. Circle detection using electro-magnetism optimization. *Information Sciences* 2012;**182**(1):40–55.
22. Wu JP, Li JX, Xiao CS, Tan FY, Gu CD. Real-time robust algorithm for circle object detection. *The 9th International Conference for Young Computer Scientists,* 2008 Nov 18-21; Zhang Jia Jie, China. Piscataway (NJ): IEEE Press; 2008. p. 1722–7.
23. John C. A computational approach to edge detection. *IEEE Trans Pattern Anal Mach Intell* 1986;**8**(6):679–97.
24. Pratt WK. *Digital image processing*. Deng LH, Zhang YH, Trans. Beijing: China Machine Press; 2005. p. 330–1 [in Chinese].

Wen Zhuoman received the B.S. degree in electronic engineering from Northeast Normal University in 2012. She is currently a Ph.D. student in University of the Chinese Academy of Sciences. Her research interests include computer vision artificial intelligence and digital image processing.

Wang Yanjie received B.S. degree of computer science from Jilin University in 1988 and M.S. degree from Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences in 1999. He is a researcher and a Ph.D. supervisor in CIOMP. His current research interests are real-time image processing TV tracking and automatic target recognition.