IEICE TRANSACTIONS

on Fundamentals of Electronics, Communications and Computer Sciences

VOL. E101-A NO. 2
FEBRUARY 2018

A PUBLICATION OF THE ENGINEERING SCIENCES SOCIETY
The Institute of Electronics, Information and Communication Engineers
Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3 chome, Minato-ku, TOKYO, 105-0011 JAPAN

LETTER

Lug Position and Orientation Detection for Robotics Using Maximum Trace Bee Colony

Phuc Hong NGUYEN[†], **Jaehoon (Paul) JEONG**[††], ***Nonmembers***, ***and*** **Chang Wook AHN**[†a)], ***Member***

SUMMARY We propose a framework to detect lug position and orientation in robotics that is insensitive to the lug orientation, incorporating a proposed optimization based on the artificial bee colony genetic algorithm. Experimental results show that the proposed optimization method outperformed traditional artificial bee colony and other meta-heuristics in the considered cases and was up to 3 times faster than the traditional approach. The proposed detection framework provided excellent performance to detect lug objects for all test cases.

key words: *artificial bee colony, object detection*

1. Introduction

Object detection has been applied to many applications, such as vehicle assistant systems, security, and robotics [1]. In robotics, object detection is often performed in advance to provide information for subsequent steps, which requires fast computation for mobile systems. This study proposes a detection method to detect lug position and orientation in an image. When the location and orientation is known, a robot can be instructed to pick up the lug. While picking up the detected lug, detection continues to identify subsequent lug(s) in the image. The contributions of this latter include:

- a maximum trace bee colony (MTBC) method to efficiently find the optimal solution;
- a robust and accurate detection framework (MTBC-D);
- an evaluation of the proposed MTBC-D in real environments.

Qualitative and quantitative experimental results show that the proposed MTBC-D achieved promising accuracy for all considered test cases.

2. Maximum Trace Bee Colony

Artificial bee colony (ABC) mimics the foraging behavior of real honeybees and includes three groups. From the view showing the process of MTBC, where the stop criterian is usually the maximum number of function evaluations or iterations, we observe that the best solution of a population often can be improved with a local search. However, ABC depends significantly on exploration, so it often takes time (due to, function evaluation) to reach a better neighbor solution. For example, assume a lug with an exact position (120, 250) and with exact degree 60, and the current best solution a position (121, 249) and degree 59, ABC does not exploit the best solution effectively and just lets it explore randomly. This prevents ABC from operating efficiently and takes many more computation cost to achieve an optimal solution. In this situation, the best solution should be exploited rather than explored. This can help to find a best solution much faster.

We propose MTBC that is based on ABC and exploits efficiently a best solution of the current population **P**. MTBC includes the best solution exploitation step, which allows the current best solution to look around for a better solution. As shown later, this idea can help MTBC to find a best solution much faster, compared to ABC, in our experiments. Figure 1 shows the process of MTBC, in which the stop criterian can be the number of function evaluations or the number of cycles.

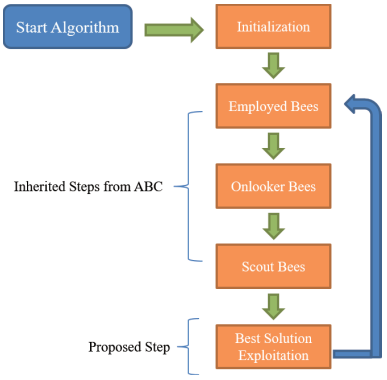


Fig. 1 Process of the proposed MTBC optimization method.

Alg. 1 Exploitation of Current Best Solution in MTBC.

1. Memorize a best solution **g** achieved so far
2. **for each** $\mathbf{g}_i \in \mathbf{g}$
3. $t \leftarrow$ a random number 1 or -1
4. $\mathbf{s}_j = \mathbf{g}_j + t$
5. $\mathbf{P}_w \leftarrow$ greedy selection between **s** and **g**

2.1 Best Solution Exploitation Step

Firstly, we select the solution **g** that has the best fitness in

Manuscript received March 15, 2017.
Manuscript revised October 7, 2017.

[†]The author is with School of Electrical Engineering and Computer Science, Gwangju Institute of Science and Technology (GIST), Republic of Korea.
[††]The authors are with Department of Interaction Science, Sungkyunkwan University, Republic of Korea.
a) E-mail: cwan@gist.ac.kr (Corresponding author)

DOI: 10.1587/transfun.E101.A.549

the current population such that each variable j in **g** is minus with 1 or plus with 1 randomly as follows:

$$s_j = g_j + t, \tag{1}$$

where t receives a value of -1 or 1 randomly. Then **g** and **s** are compared to determine whether **s** is better than **g** or not. If so, **s** will replace the current worst population solution. Alg. 1 shows the proposed MTBC procedure, where w is the index of a worst solution.

3. Proposed Detection Framework

3.1 Process of Proposed Method

Figure 2 shows the proposed MTBC-D method, comprising 7 steps.

- A background model is constructed from images that do not contain any lugs, and in normal condition. This step can be performed offline.
- For each search image, background subtraction highlights changes, referred to as foreground areas.
- Noise filtering removes small foreground areas.
- The proposed detection algorithm identifies the location and orientation of a lug in the filtered image.
- The detected lug area is added to the background image to avoid being included in future detection.
- Details of the detected lug are transmitted to robot.
- The search image is re-checked to determine if there are further lugs or not. If a lug is identified, return to step 4. Otherwise, detection stops.

3.2 Back-Ground Modeling and Subtraction and Noise Removal

We capture a set of N images in a normal condition to construct a background model. Let I_i be the i^{th} image in the image set. The background model B is computed using a mean filter technique as follows:

$$B(x,y) = \frac{1}{N}\sum_{i=1}^{N} I_i(x,y), \tag{2}$$

where $B(x,y)$ is the value at pixel (x,y) in B, and $I_i(x,y)$ is the pixel value for the pixel (x,y) in I_i.

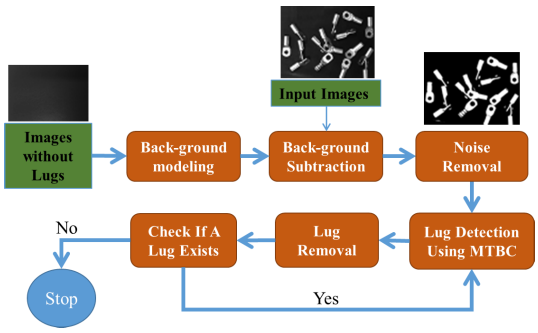


Fig. 2 Process of the proposed detection framework using MTBC.

Let F be a foreground image and I is a new coming image in a detection process. The foreground image F is computed as follows:

$$F(x,y) = \begin{cases} 255 & if\ |I(x,y) - B(x,y)| > th_b, \\ 0 & otherwise. \end{cases} \tag{3}$$

The foreground image F in this stage may contain small foreground areas because of errors from the background subtraction step. The problems can be solved by using an erode technique.

3.3 Lug Detection Using MTBC

Locations and orientations in an image are random. Therefore, the object detection method must be able to detect objects with different orientations. We propose MTBC to detect lug objects. The parameters for optimization include the left-top position (x,y) and rotation α.

The general form of rotation matrix T_R is defined as follows:

$$T_R = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4}$$

The following sections provide the detailed method to find optimized parameters following the proposed MTBC approach.

3.3.1 Structure of Problem

The rotation matrix requires the degree value of to determine the amount of rotation for the template T; hence, $\alpha \in \mathrm{N}$ and $0 \le \alpha \le 359$. In addition, the left-top position has two parameters (x,y); hence, $x, y \in \mathrm{N}$ and $0 \le x \le W_I - W_T$ and $0 \le y \le H_I - H_T$ where W_I and H_I are the width and height of an input image, and W_T and H_T are the width and height of the template image, respectively. Therefore, a solution,**s**, can be defined as an array of length 3. That is $\mathbf{s} = [\alpha \quad x \quad y]$. The $lb = [0 \quad 0 \quad 0]$ and $ub = [359 \quad W_I - W_T \quad H_I - H_T]$ are the lower bound and upper bound for **s** respectively.

3.3.2 Cost Function

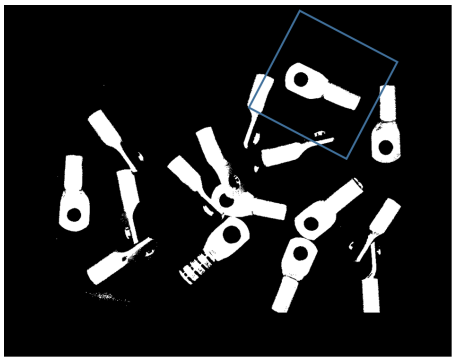
Given a rotation value α and a left-top pixel q, a rotated template R can be computed by rotating the template T around the center position of T with an α degree. After the rotated template R is computed, the cost value between R and I is computed as follows:

$$C(R,I,q) = \sum_{p \in R} \left| I_{(q+p)} - R_p \right| \times \delta\left(R_p\right) \tag{5}$$

where

$$\delta\left(R_p\right) = \begin{cases} 1 & if\ R_p = 255, \\ 0 & otherwise. \end{cases} \tag{6}$$

The function $\delta(.)$ aims to compute the similarity cost only



(a) An Input image I

(b) Template T

Fig. 3 Illustration of a detected lug. After that, its position and orientation are used to remove it from the current processing image.

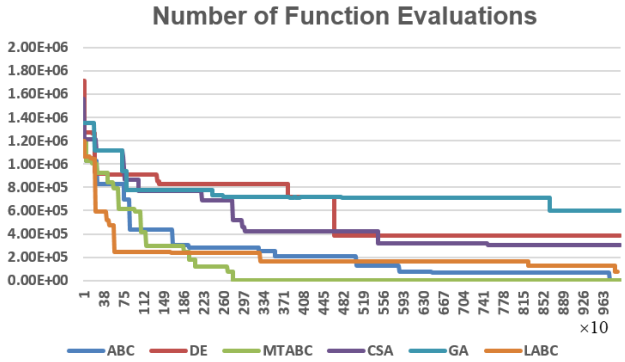


Fig. 4 Efficiency comparison of the testing methods. MTBC performed much more efficiently than ABC and outperformed the other testing methods.

for a lug area, which only includes the identified lug area. Thus, MTBC can obtain the rotation and top-left position parameters, which minimizes an error.

3.4 Lug Removal and Lug Existing Checking

After a lug detected and its parameters identified, the process continues to look for other lugs within the image. Therefore, we remove the detected lug region to avoid duplication using R as an image mask. For each pixel p in R, we set the value for the corresponding pixel $(m + p)$ in I to zero if $R_p = 255$. Figure 3 illustrates the case for a detected lug with the known position and rotation. We propose a simple technique to estimate if a lug appears in an image: If the white pixels (intensity = 255) is larger than a threshold th_r, then it is likely at least one lug existing in the image.

4. Experimental Results

The sign test was adopted [2] to evaluate overall method performances by counting the number of cases where the proposed algorithm was superior (+) as a winner and where it was inferior (-) as a loser. The p-value provides the degree of comparison.

We compared the proposed MTBC approach with current best practice population based meta-heuristics, including GA [3], DE [4], CSA [5], ABC [6], and LABC [7] on our captured images. The corresponding detection methods, using the optimization method discussed above, are denoted as MTBC-D, GA-D, DE-D, CSA-D, ABC-D, LABC-D, respectively. These testing methods are in turn used as an optimization method in the proposed detection framework. Each experiment was run 50 times with the same random seed. Parameters for CSA and DE/rand/1/bin (DE) were carefully selected. Parameter values for ABC and LABC were selected as described in the original papers [6] [7]. To be similar to ABC and LABC, we chose $N_{MTBC} = 100$.

We used a camera to capture a set of 1280×672 pixel images from a real system. To evaluate each method, we visually inspected each image and identified the position and orientation for each lug. A thresholds for x-and y-coordinates is $th_p = 5$ pixels and a threshod for orientation is $th_o = 3$ degrees.

Table 1 Quantitative results of a lug detection for the testing methods.

MTBC	ABC	CSA	DE	GA	LABC
0	285175	519359	831500	715373	130305

4.1 Efficiency Evaluation

We compared the efficiency of a lug detection using the input image I, as shown in Fig. 3; however, instead of using a general template T, we extracted a template T' from the image I. Thus, the cost function of the optimal detected lug is 0. We fixed the number of function evaluations for all the testing methods to be 10,000.

Figure 4 compares the different method performances. With the limited number of function evaluations, GA, DE, and CSA failed to reach an optimal solution, whereas the proposed MTBC method required less than 2,970 function evaluations, and ABC required more than three times this number to obtain the optimal solution. For meta-heuristics, faster methods are also more accurate, because for real problems we cannot specify a correct solution, hence we must estimate an appropriate number of evaluations that balance computation cost and accuracy. Table 1 presents the quantitative results of the testing methods after 2,970 times of function evaluations. With the limited number of function evaluations, MTBC performed the most efficient to with the smallest minimized value.

We also evaluated computation time for ABC and MTBC to detect a lug, using a PC with core i7-4790k CPU, 4.00 GHz, and 8GB RAM. ABC took approximately 2.5 s to detect a lug, whereas the proposed MTBC approach required approximately 0.8 second. This latter solution is fast enough for a robot, because it requires at least 2 s to pick up a detected lug.

4.2 Method Comparison

As discussed above, we used the sign test technique [2] to

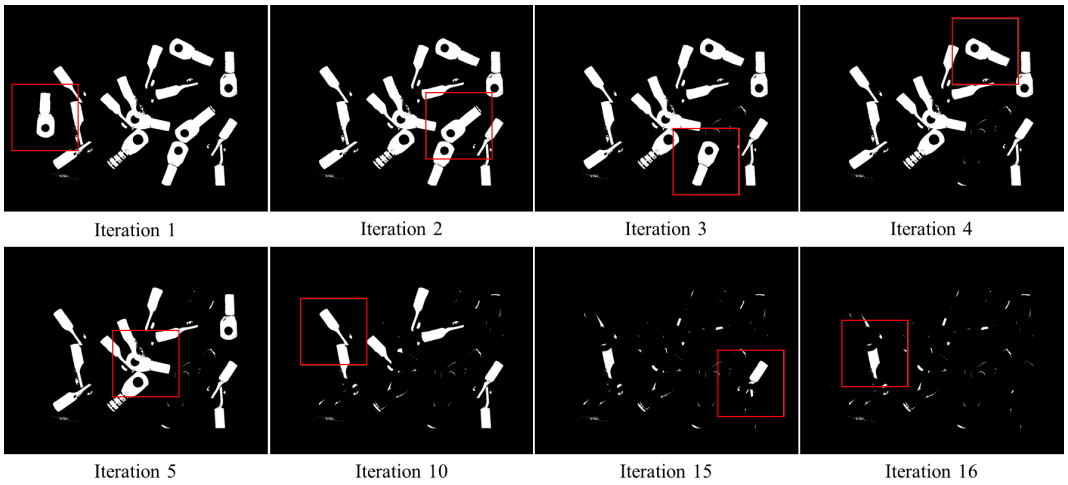


Fig. 5 Qualitative results of the proposed detection method for lug location and orientation detection for an image. Each iteration try to find a lug in an processing image.

Table 2 Results of lug detection for the testing methods.

Delta	MTBC-D	ABC-D	CSA-D	DE-D	GA-D	LABC-D
image 1	Win (+)	47	48	49	49	45
	Lose (-)	3	2	1	1	5
	p-value	0.05	0.05	0.05	0.05	0.05
image 2	Win (+)	47	49	48	47	44
	Lose (-)	3	1	2	3	6
	p-value	0.05	0.05	0.05	0.05	0.05
image 3	Win (+)	48	48	47	48	47
	Lose (-)	2	2	3	2	3
	p-value	0.05	0.05	0.05	0.05	0.05
image 4	Win (+)	47	48	49	47	44
	Lose (-)	3	2	1	3	6
	p-value	0.05	0.05	0.05	0.05	0.05

compare the proposed MTBC-D against the other test detection methods. We set the maximum number of function evaluations to 4,000, the background subtraction threshold $th_b = 20$ and the lug removal threshold $th_r = 150$ for all methods.

Table 2 compares the method performance of the methods with different images. With the limited number of function evaluations, ABC-D, CSA-D, DE-D, GA-D, and LABC-D failed to achieve optimal solutions for all lugs in an image, hence, the nominally optimized solutions were mostly incorrect with thresholds th_p and th_o. On the other hand, MTBC-D efficiently provided the actual optimal solution, and performed significantly better than ABC and all other methods for all the test cases.

Figure 5 shows qualitative MTBC-D results for lug location and orientation detection. The process continued until the detection method determined that no lug existed in the processed image.

5. Conclusion

We proposed an optimization detection method for lug objects within an image, MTBC-D, that outperformed the state-of-the-art current methods. In addition, we proposed a framework for detecting lug objects for robots. Thus, the proposed method can efficiently detect lug objects with different rotations sufficiently quickly for real cases in real time. These excellent experimental outcomes motivate to extend the method to detect other objects in manufacturing environments.

Acknowledgments

This work was supported by the ICT R&D program of MSIT/IITP [2014-0-00077, Development of global multi-target tracking and event prediction techniques based on real-time large-scale video analysis] and also by X-Project funded by the MSIT (NRF-2016R1E1A2A02946533).

References

[1] S. Haykin, Neural Networks: A Comprehensive Foundation, 3rd ed., Prentice-Hall, Upper Saddle River, NJ, USA, 2007.
[2] J. Derrac, S. Garcia, D. Molina, and F. Herrera, "A practical tutorial on the use of non-parametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," Swarm Evol. Comput., vol.1, no.1, pp.3–18, 2011.
[3] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, 1975.
[4] R. Storn and K. Price, "Differential evolution — A simple and efficient heuristic for global optimization over continuous spaces," J. Global Optim., vol.11, no.4, pp.341–359, 1997.
[5] N.C. Leandro and F.J. Von Zuben, "Learning and optimization using the clonal selection principle," IEEE Trans. Evol. Comput., vol.6, no.3, pp.239–251, 2002.
[6] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (abc) algorithm," J. Global Optim., vol.39, no.3, pp.459–471, 2007.
[7] N.H. Phuc and W.A. Chang, "Linkage artificial bee colony for solving linkage problems," Expert. Syst. Appl., vol.61, pp.378–385, 2016.