Model-based Pose Estimation for Texture-less Objects with Differential Evolution Algorithm

Linh Tao , Tinh Nguyen and Hiroshi Hasegawa

Graduate School of Engineering and Science,Shibaura Institute of Technology,Japan

Abstract. This paper proposes a novel object-tracking method to estimate three dimensions position of texture-less objects using one camera system and 3D model. The system uses efficient chamfer matching method to calculated distances between 2D edge templates of pose hypotheses with edges from the Canny edge query image. Differential Evolution algorithm uses those distances as inputs to ensure the close optimum results and find the most suitable position of objects. For initialization the exhaustive searching is employed. With the good initialization, a smaller searching space is set to guaranty the online tracking ability. The first results showed the potential of the method in solving object tracking and detection problem.

1 Introduction

In the last decade, object detection and recognition have gained significant improvement by using keypoint features [1]. Since geometric transformations and illumination changes has no effect on finding keypoints, they have been widely used for matching images from slightly different viewpoints [2]. Keypoint-based approaches work well in textured objects but texture-less objects. Textured objects have various keypoints, those have high potential appearing on both images. After finding keypoints, sample consensus such as RANSAC [3] calculates the most suitable transformation of the object from reference position to current position. The more matched keypoints, the more accurate the transformation is.

On texture-less objects lack of keypoint repeatability and stability on texture-less regions neither reduces the accuracy of sample consensus method nor leads to wrong results. Like keypoints, edges are also invariant to general geometric transformations and illumination changes [4]. Using edges are more suitable as a general approach even with texture-less objects.

In early computer vision research, to find the best alignment between two edge maps, a given priori set of edge templates compare their suitability to the current edge maps to draw the mot suitable transformation. The current proposed method of chamfer distance matching [5] enhances the cost functions enable for applying global searching algorithm into object tracking problem.

Harris [6] and various proposed edge-based tracking system such as [7] used edges and contours for visual tracking task. One drawback of using edges is that they are not distinctive enough to provide effective discrimination in complex background or occlusions, there have been efforts to enhance the previous one by unifying interest points or considering multiple but limited hypotheses on edge correspondences. For consideration of multiple hypotheses in a more general sense global searching algorithm should come into consideration.

We propose an approach of using Differential Evolution (DE) [8] as the global searching method to continuously search for the 3D position of object in camera coordinate.

2 Methodology

Initialization is the most important step, a subject of the paper's interest, in tracking algorithm. Following steps presents the implementation pipeline of the initialization:

- Canny edge image is employed to archive edge images from query images.
- Distance maps or chamfer matching maps is calculated from edge images.
- Differential Evolution is employed to calculate the best fit pose of the object which create an 2D edge images fitted into the chamfer matching maps for initialization.

After initialization, narrower searching boundary is used to get the accurate results at on-line speed. If the cost function goes large, initialization is required.

2.1 Chamfer matching maps from query images.

© The Authors, published by EDP Sciences. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (http://creativecommons.org/licenses/by/4.0/).

2.1.1 Canny edge detection

The white lines in Fig1 are output of Canny [9] edge detection method. Canny method includes five different steps:

1. Apply Gaussian filter to smooth the image in order to remove the noise.
2. Find the intensity gradients of the image
3. Apply non-maximum suppression to get rid of spurious response to edge detection
4. Apply double threshold to determine potential edges.
5. Track edge by hysteresis: Finalize the detection of edges by suppressing all the other edges that are weak and not connected to strong edges.

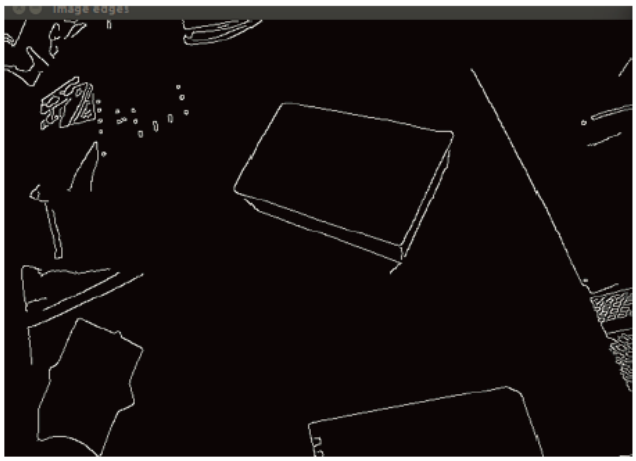

Figure 1. An edge image from Canny method

OpenCV [10] library documentation gave us above implementation [11] in detail.

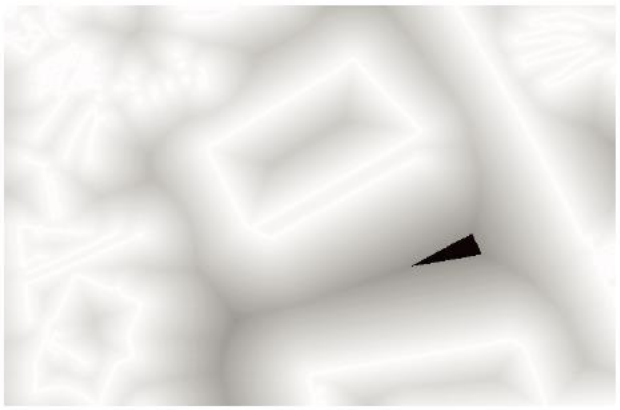

Figure 2. Chamfer maps of edge image example

2.1.2 Chamfer matching maps

In the comparison step, points get higher score when they come closer to the edge point. By indexing the value of pixel with its distance to the closest edge point, we get the distance map as shown in Fig 2.

2.2 Camera model and edges from CAD

From a camera with prior-known configuration and object CAD model, we are able to archive ideal visible edges of objects by using camera model matrix. This matrix convert a point with coordinate of (x, y, z) in real coordinate to a image point (u, v) as Equation 1.

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x/z \\ y/z \\ 1 \end{bmatrix} \quad (1)$$

To determine visibility of object edges we use edge features based method from [12]. Figure 3 shows an edge between adjacent faces $A =< v_0, v_1, v_2 >$ and $B =< v_0, v_1, v_3 >$ with unit face normal n_A and n_B calculated as in Equation 2,3.

$$n_A = \text{normalize}([v_1 - v_0] \times [v_2 - v_0]) \quad (2)$$

$$n_B = \text{normalize}([v_3 - v_0] \times [v_1 - v_0]) \quad (3)$$

To determine the visibility of edge $E =< v_0, v_1 >$, we uses additional vector v_e with direction from v_0 to the camera center. E is visible edge if cross manipulation value of v_e with n_A or v_e with n_B positive.

2.3 Initial pose searching

2.3.1 Cost function calculation

The cost function for global searching algorithm is a comparison result between edge maps from image and edge maps from CAD model. To gain a equivalent between ideal edge maps, a re-sampling step is applied, so the number of edge points in different ideal maps is set equally at N=200 points.

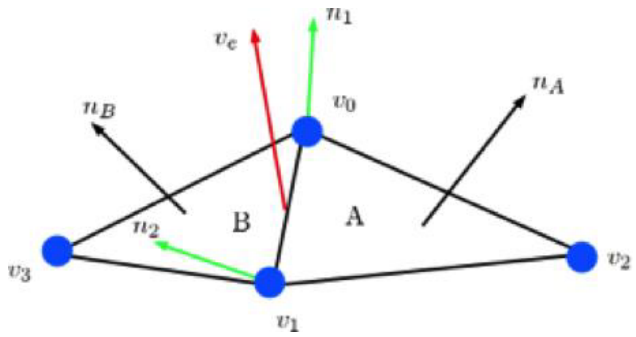

Figure 3. Edge of two surfaces

The error cost function is calculated as in Equation 2,

$$F(R,t) = f(n)\frac{1}{n^2}\sum_1^n (E_i - M_i)^2 \quad (4)$$

where f(n) is function depended on number of inlier (n) as in Equation 3. E_i is value of real edge images at inlier i, M_i is value of CAD model edge images at inlier number i.

$$f(n) = 1 - \frac{n}{N} \quad (5)$$

2.3.2 Differential evolution

Differential evolution (DE), proposed by Storn and Price, is a very popular EA. Like other EAs, DE is a population-based stochastic search technique. It uses mutation, crossover and selection operators at each generation to move its population toward the global optimum minimum.

a) Initialization in DE

The initial population was generated uniformly at random in the range lower boundary (LB) and upper boundary (UB).

$$X_{i,j}^{G=0} = lb_j + rand_j(0,1)(ub_j - lb_j) \quad (6)$$

where $rand_j(0,1)$ a random number in [0,1].

b) Mutation operation

In this process, DE creates a mutant vector, $X_i^G = (X_{i,1}^G, X_{i,2}^G, \dots, X_{i,n}^G)$. For each individual at each generation X_i^G is a target vector in the current population.

There are several variants of DE based on mutation schemes, which are: DE/rand/1, DE/best/1, DE/current to best/1, DE/rand/2, DE/best/2, DE/rand to best/1.

c) Crossover operation

After mutation process, DE performs a binomial crossover operation on X_i^G and V_i^G to generate a trial vector $U_i^G = (U_{i,1}^G, U_{i,2}^G, \dots, U_{i,n}^G)$ for each particle i as shown as Equation 5.

$$U_i^G = \begin{cases} V_{i,j}^G & \text{if } rand_j \le CR \text{ or } j = j_{rand} \\ X_{i,j}^G & \text{otherwise} \end{cases} \quad (7)$$

where i=1, …, NP. j=1,…, D is randomly chosen integer in $[1, D]$, $CR \in [0,1]$ is the crossover control parameter.

d) Selection

The selection operator is to select the better vector between the target vector $X_{i,j}^G$ and the trial vector $U_{i,j}^G$ to enter the next generation.

$$X_i^{G+1} = \begin{cases} U_i^G & \text{if } f(U_i^G) \le f(X_i^G) \\ X_i^G & \text{otherwise} \end{cases} \quad (8)$$

where i =1, …, NP, X_i^{G+1} is target vector in the next generation.

3 Experiment and results

3.1 Experiment setup

To implement the algorithm, the system hardware included a iBuffalo BSW20KKM11BK camera. We used small box as a tracking object. All code is implemented on C++ code on a standard Desktop Computer powered with Intel Core i7-4790 CPU 3.6x8.

In the experiment, we used the box object with size of 145x95x40(mm) in white colour as the tracking object. The object "*.ply" extension is use the input model.

The searching boundary for the objects translation are in [-100, 100] and $[-\pi/2, \pi/2]$.for rotation angles of roll-pitch-raw.

Parameters for Differential Evolution searching algorithm are presented in Table 1.

Table 1. DE parameters

F0	Cr	Cross over	Max gen	Pop
0.8	Cr	DE/rand/1/bin	150	300

3.2 Results

Fig. 4 and Fig. 5 show results from Canny edge detection method.

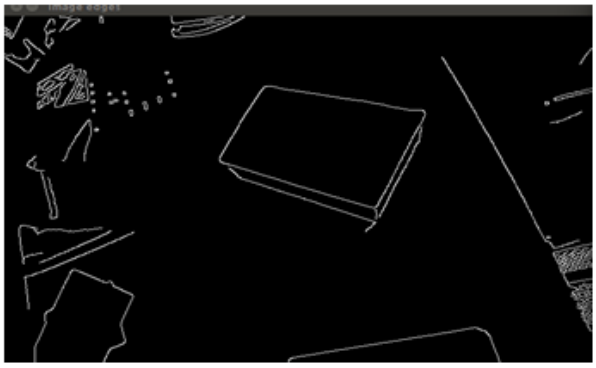

Figure 4. A canny edge detection result

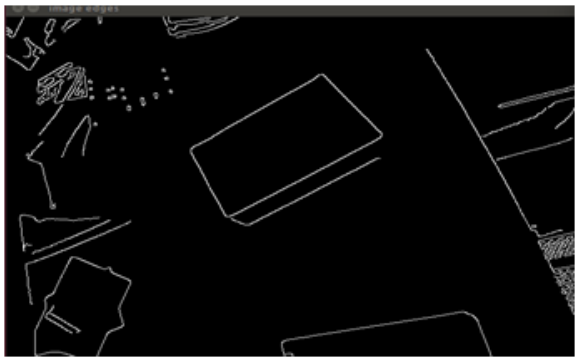

Figure 5. A canny edge detection result

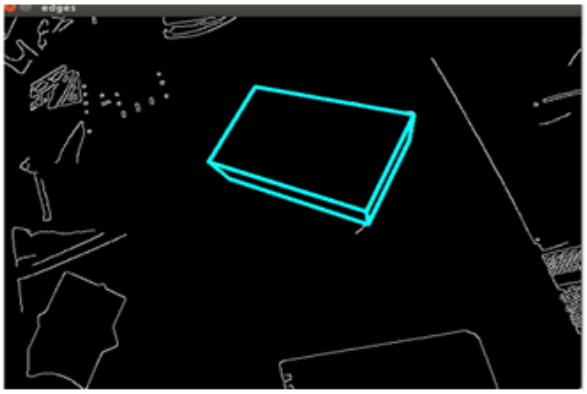

Figure 5. A canny edge detection result

Fig. 6 and Fig. 7 show results of boundary search of the box with different positions from Fig 4 and Fig 5. The box boundary is in blue colour in the left image.

The result images showed that, the method were able to find the position of the object so the boundary could fit into the edge maps.

Depend on the objects, symmetric or non-symmetric, we need to set different searching boundaries for the searching algorithm (DE). Table 2 shows consuming time depend on number of population of DE.

Table 2. Time consuming on population size

Pop size	400	300	200	100	20	10
ms	2817	2190	1404	741	161	76

4 Discussion and conclusions

Object tracking has been always a challenging task in computer vision. Recently, evolution based global searching methods have proved its potential of tackling the tracking problem with ability of finding robust and accurate global optima solutions.

We proposed a novel approach of using differential evolution as a global searching method to find the best 3D position of objects. The experimental results showed promising results.

In the future work, we would like to improve cost function with the method to narrow searching area for more accuracy but smaller generation of searching. By doing so, we expect to reduce the runtime but remains the accuracy and robustness.

References

1. D. G. Lowe, “Distinctive image features from scale-invariant key- points,” IJCV, vol. 60, no. 2, pp. 91–110, 2004.
2. K. Grauman and T. Darrell, “The pyramid match kernel: Discrimina- tive classification with sets of image features,” in ICCV, vol. 2, 2005, pp. 1458–1465 Vol. 2.
3. M. A. Fischler, R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Comm. of the ACM, Vol 24, pp 381-395, 1981.
4. Luc Vosters, Caifeng Shan, Tommaso Gritti, Real-time robust background subtraction under rapidly changing illumination conditions, Image and Vision Computing, Volume 30, Issue 12, December 2012, Pages 1004-1015, ISSN 0262-8856, http://dx.doi.org/10.1016/j.imavis.2012.08.017.
5. H. Barrow, J. Tenenbaum, R. Bolles, and H. Wolf, “Parametric correspondence and chamfer matching: Two new techniques for image matching,” in IJCAI, 1977, pp. 659–663.
6. C. Harris, “Tracking with Rigid Objects”. MIT Press, 1992.
7. A. I. Comport, E. Marchand, and F. Chaumette, “Robust model-based tracking for robot vision,” in IROS, vol. 1, 2004.
8. Storn, Rainer “Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces”, Journal of Global Optimization, doi: 10.1023/A:1008202821328, 2007.
9. Canny method, http://docs.opencv.org/master/da/d22/tutorial_py_canny.html#gsc.tab=0
10. Bradski, G. “opencv_library”, Dr. Dobb's Journal of Software Tools, 2000.
11. OpenCV Edge Detection Implementation. http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html
12. Morgan McGuire and John F. Hughes, “Hardware-Determined Feature Edges”, Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering, 2004, doi: http://doi.acm.org/10.1145/987657.987663