Real-Time Edge Template Tracking via Homography Estimation

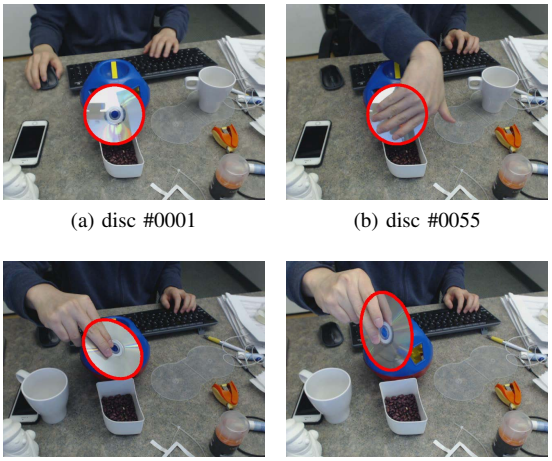
Xuebin Qin*, Shida He, Zichen Zhang, Masood Dehghan, Jun Jin and Martin Jagersand

***Abstract*— In this paper, we propose a novel real-time method for tracking planar edge templates. This method tracks an edge template by estimating its homography transformations with respect to the sampled edge pixels detected from the incoming frames. Particularly, we define a cost function based on a new feature map of the to-be-tracked edge template and optimize it by a Lucas-Kanade-like algorithm. The feature map is defined as the fourth root of the distance transform. Our method operates on just edges so that it is good at tracking those low textured targets, such as hollow targets (mug rim), thin targets (cable, ring) and non-Lambertian objects (disc). We validate and compare our method with four other methods on five newly collected real-world video sequences. The results achieves the lowest overall average error (1.58 pixels) and also outperforms others in terms of success rate. The per frame processing time of about 30 ms proves that our method is acceptable in real-time applications. The code and dataset are publicly available at: `http://webdocs.cs.ualberta.ca/~xuebin/`.**

I. INTRODUCTION

Visual tracking is a hot yet challenging issue in computer vision and robotics communities. Many trackers have been proposed in the past decades [1], [2]. The bounding boxes and quadrilaterals are the most commonly used targets representations of low Degree-of-Freedom (DoF) [1], [3] and high DoF trackers [2] respectively. Those trackers operates on the relatively rich texture information enclosed by the bounding boxes and quadrilaterals. The assumption is that these textures are rich and relatively stable. However, this assumption fails when the targets are hollow objects, thin objects and non-Lambertian objects (see Fig. 1). To represent these targets, edge template is a good alternative. In this paper, we are aiming at tracking these special targets via edge templates homography estimation.

Chamfer Matching (CM) based methods [4], [5] are the most commonly used in edge templates matching and detection [6], [7]. However, most of them are designed for locating the templates in two dimensional (2D) images other than estimating their homography transformations. Although it is possible to adapt them for detecting shapes with high DoF transformations [8], it is hard to achieve real-time performance. Another possible idea is to solve the 2D planar edge pixels alignment problem by the iterative closest point (ICP) methods [9], [10]. However, the direct ICP methods are not able to recover non-rigid transformations, such as affine and homography. Additionally, the speed of ICP is often relatively slow and thus the real-time performance is not guaranteed.



(a) disc #0001 (b) disc #0055

(c) disc #0178 (d) disc #0287

Fig. 1. Non-Lambertian object: a disc. These frames are sampled from the same video sequence. Their intensities in the disc regions are extremely different from each other.

Hofhauser *et al.* developed an edge template detection method using the metric defined on the differences between the template edge pixels' geometric gradient directions and the image gradient directions [11]. Their method estimates the perspective transformation based on the 2D translation of the detected feature points. These feature points are locally stable patterns comprised of clustered template edge pixels. Hence, this approach is likely to fail in tracking simple templates because of the aperture problem. Holzer *et al.* estimated the homography between two closed boundaries by adapting the Lucas-Kanade (LK) algorithm [12] in which they substitute the image intensities for distance transform maps [13]. This method works well with low texture targets in a large scale range. One drawback is that the two input closed boundaries have to be correctly recognized and no outliers are allowed. Besides, it is not real-time.

Qin *et al.* proposed to track salient closed boundaries by graph based line segments perceptual grouping in [14]. To obtain more accurate results, they replaced the straight line segments with smooth edge fragments in [15]. The prerequisite of this method is that the to-be-tracked targets have to be closed. Additionally, this method is very sensitive to occlusions, especially large area occlusions. Because its grouping cost is usually defined proportional to the ratio of the total gap length along the to-be-tracked boundary and

The authors are with the Dept. of Computing Science, University of Alberta, Canada. {xuebin, shida3, zichen2, masood1, jjin5, mj7}@ualberta.ca.

Xuebin was supported by China Scholarship Council (CSC) and Alberta Innovates Graduate Student Scholarship (AITF).

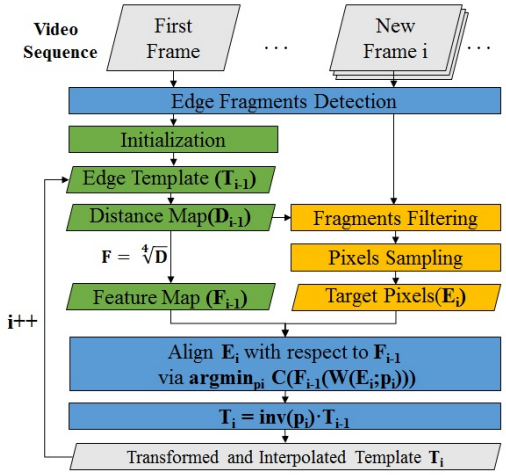


Fig. 2. Workflow of our method

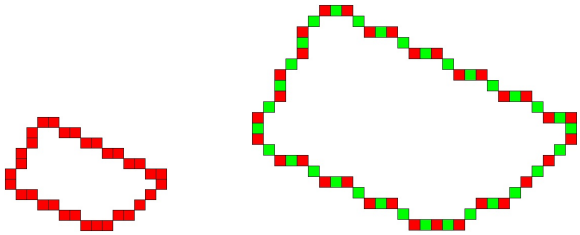
the boundary's perimeter. Large area occlusions generate too many big gaps which make the grouping cost meaningless.

To address the problems mentioned above, we propose a novel edge template tracking method. We provide the code for a real-time CPU implementation. Our contributions are threefold: 1) We generate a new feature map of the to-be-tracked edge template to suppress impacts of outliers. 2) We introduce a smoothness term to define a new tracking cost function for edge template homography estimation. The cost function is optimized by a LK-like algorithm. 3) We implement this edge template based object tracking method and validate it on our newly collected real-world video dataset. It outperforms the pertinent methods [16], [14], [15] and achieves state-of-the-art performances.

The reminder of this paper is organized as follows. Section II illustrates the method details. Section III presents the experimental results. The conclusion is given in Section IV.

II. METHOD

In this paper, our goal is to track edge templates through video sequences. Specifically we estimate the homography transformation that relates the edge template between incoming frames. As illustrated in Fig. 2, our method operates on detected edge fragments instead of image intensities, as is common in conventional template tracking [1], [2], [12]. Given a video sequence, we manually initialize an edge template at its first frame. Then, a feature map of this template is generated. As the next frame comes, we sample the most relevant pixels (we call them target pixels) of the to-be-tracked target from its detected and filtered edge fragments. We estimate the homography of the current frame with respect to the template feature map by aligning those target pixels and the feature map. The alignment problem is solved by minimizing our newly defined cost function using a LK-like algorithm [2]. The outputs of our tracking method are transformed edge templates.



(a) initial template (b) scaled and completed template

Fig. 3. Template completing after scaling. (a) the red pixels are sequentially stored, (b) the red pixels are computed from the coordinates of those red pixels in (a) and the green pixels are filled by line segments between two adjacently stored red pixels.

A. Edge Fragments Detection

Edge fragments are well organized short one-pixel width edge segments. Each edge fragment is comprised of several sequentially stored edge pixels which belong to the boundary of the same object. To obtain high quality edge fragments, we use the fragments detection method proposed in [15]. Given a new frame, this method has two steps: detection and splitting. First, it detects edge segments from the frame by Edge Drawing (ED) [17]. The detected edge segments are one-pixel width pixel chains. But most of the segments are long and sometimes the edge pixels of the same edge segment are erroneously grouped from different objects (such as foreground and background). To avoid this kind of errors, we split every detected edge segment into multiple shorter and more straight edge fragments by a fast splitting method proposed in [15]. For a 640×480 (width$\times$height) image, the edge fragments detection process needs less than five milliseconds (ms).

B. Edge Template and Feature Map

Edge templates, as the main outputs of our method, are represented by sets of sequentially stored edge pixels. In the initial frame (the first frame or the frame where tracking starts), we manually initialize the edge templates by a method adapted from a boundary based annotation tool [18]. This method allows users to select detected edge fragments for creating edge templates. Compared with common boundary initialization methods based on straight line segments drawing, this method can produce more accurate and smoother edge templates with light human workload. In an non-initial frame, its edge template is obtained by transforming that of its previous frame using estimated homography. The template pixel chains manually selected in the initial frame are guaranteed to be continuous (see Fig. 3a). However, adjacent pixels are likely to be detached because of transformation, such as scaling (see Fig. 3b). To get edge templates with continuous pixel chains, we fill the gaps between those detached adjacent pixels by drawing straight line segments automatically, as shown in Fig. 3. Although this way of template completing is less accurate than spline fitting, it is more efficient and

easy to implement.

In our method, the feature map is a transform of the edge template which facilitates the alignment and optimization. In [13], the distance transform map D [7], [4] (see Fig. 4c) of the template is directly taken as the feature map. But it is sensitive to outliers. Hence, we propose to use $F = \sqrt[4]{D}$ as the feature map (see Fig. 4d). The red and green lines in Fig. 4e illustrate the comparison between D and F. The blue and black lines show the main difference between D^2 and F^2. As we can see, F^2 grows more gently so that it is able to suppress the impact of outliers.

C. Target Pixels

As the new (the i-th) frame comes, our method detects its edge fragments first, as illustrated in Fig. 2. Our purpose is to align the detected edge fragments with the edge template tracked from the last (the $(i-1)$-th) frame. However, most of the edge fragments in the new frame are outliers and redundant. Hence, we remove those irrelevant edge fragments based on the following two rules:

1) mean distance $md > 10$

$$md = \frac{\sum_{k=1}^{s} D_{i-1}(\mathrm{x}_i^k)}{s}, \tag{1}$$

2) mean absolute distance difference $madd > 0.8$

$$madd = \frac{\sum_{k=1}^{s-1} |D_{i-1}(\mathrm{x}_i^{k+1}) - D_{i-1}(\mathrm{x}_i^k))|}{s} \tag{2}$$

where D_{i-1} denotes the distance map of the edge template tracked from the $(i-1)$-th frame. x_i^k represents the k-th pixel of the current edge fragment, which was detected from the current (i-th) frame. s is the pixel number of the current edge fragment. Additionally, to achieve real-time performance in the optimization process, if the total pixels' number of these retained edge fragments is more than 100, we uniformly sample 100 edge pixels from them.

D. Tracking Cost

Given the feature map F of the last ($(i-1)$-th) frame and the target pixels of the current (i-th) frame, we define the tracking cost as:

$$C(\mathbf{p}) = E_{align}(\mathbf{p}) + \lambda E_{smooth}(\mathbf{p}) \tag{3}$$

where E_{align} is the alignment error and E_{smooth} is the smoothness term:

$$E_{align}(\mathbf{p}) = \sum_{j=1}^{n}\sum_{k=1}^{m_j}[F(\mathbf{W}(\mathbf{x_k};\mathbf{p}))]^2 \tag{4}$$

$$\begin{aligned} E_{smooth}(\mathbf{p}) = \sum_{j=1}^{n}\sum_{k=2}^{m_j}[F(\mathbf{W}(\mathbf{x_k};\mathbf{p})) \\ -F(\mathbf{W}(\mathbf{x_{k-1}};\mathbf{p}))]^2 \end{aligned} \tag{5}$$

where $\mathbf{p}$ indicates the homography parameter of warping the current frame back to the last frame. n and m_j denote the number of the retained edge fragments and the sampled pixels at the j-th edge fragment. $\mathbf{x}$ represents the coordinate of a target edge pixel. k and $k-1$ are the indices of two adjacently sampled pixels which belong to the same edge fragment. λ is the a weight of the smoothness term. We use a homography warp $\mathbf{W}$ [12]:

$$\mathbf{W}(\mathbf{x};\mathbf{p}) = \frac{1}{1+p_7x+p_8y}\begin{pmatrix}(1+p_1)x+p_3y+p_5\\ p_2x+(1+p_4)y+p6\end{pmatrix}. \tag{6}$$

In the cost function of the traditional LK [12], both template (I_T) and target (I) are images (or feature maps), and LK tries to minimize their differences $\sum[I_T - I]^2$. In $E_{align}(\mathbf{p})$, (4), however, although the template is still a gray scale image like feature map (F), the target $\mathbf{x}$ can be taken as a zero/one mask (M). Hence, $\sum_{j=1}^{n}\sum_{k=1}^{m_j}[F(\mathbf{W}(\mathbf{x_k};\mathbf{p}))]^2$ equals to $\sum[F \cdot M]^2$ where $\cdot$ denotes element-wise multiplication.

E. Optimization

Our goal is to estimate $\mathbf{p}$ by minimizing the tracking cost as $\arg\min_{\mathbf{p}} C(\mathbf{p})$. In this paper, we solve this optimization problem using LK-like algorithm [12]. Our solver initializes $\mathbf{p}$ to an identity homography matrix. Then it iteratively solves the following minimization problem:

$$\arg\min_{\mathbf{\Delta p}} C(\mathbf{\Delta p}), \tag{7}$$

$$\begin{aligned} C(\mathbf{\Delta p}) = &\sum_{j=1}^{n}\sum_{k=1}^{m_j}[F(\mathbf{W}(\mathbf{x_k};\mathbf{p}+\mathbf{\Delta p}))]^2 + \\ &\lambda\sum_{j=1}^{n}\sum_{k=2}^{m_j}[F(\mathbf{W}(\mathbf{x_k};\mathbf{p}+\mathbf{\Delta p})) - \\ &F(\mathbf{W}(\mathbf{x_{k-1}};\mathbf{p}+\mathbf{\Delta p}))]^2 \end{aligned} \tag{8}$$

and updates $\mathbf{p}$ as:

$$\mathbf{p} \leftarrow \mathbf{p} + \mathbf{\Delta p}. \tag{9}$$

Minimizing (8) and updating with (9) are iterated until the estimate of $\mathbf{p}$ converges. We terminate the iteration when $|\sum_{j=1}^{n}\sum_{k=1}^{m_j}[F(\mathbf{W}(\mathbf{x_k};\mathbf{p}+\mathbf{\Delta p}))] - \sum_{j=1}^{n}\sum_{k=1}^{m_j}[F(\mathbf{W}(\mathbf{x_k};\mathbf{p}))]| < \epsilon$ or the number of iterations reaches a predefined maximum of 30.

In particular, we minimize expression (8) and compute $\mathbf{\Delta p}$ using Gauss-Newton gradient descent method as follows:

$$\frac{\partial C(\mathbf{\Delta p})}{\partial \mathbf{\Delta p}} = 0 \tag{10}$$

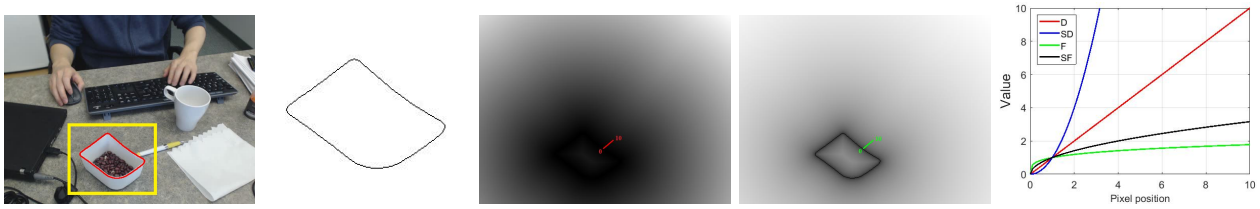
$$\mathbf{\Delta p} = A^{-1}\mathbf{b} \tag{11}$$

$$A = \sum_{j=1}^{n}\sum_{k=1}^{m_j}\mathbf{J_1^k}^T\mathbf{J_1^k} + \lambda\sum_{j=1}^{n}\sum_{k=2}^{m_j}\mathbf{J_2^k}^T\mathbf{J_2^k} \tag{12}$$

$$\mathbf{J_1^k} = \nabla\mathbf{F_k}\frac{\partial\mathbf{W}}{\partial\mathbf{p_k}} \tag{13}$$

$$\mathbf{J_2^k} = \nabla\mathbf{F_k}\frac{\partial\mathbf{W}}{\partial\mathbf{p_k}} - \nabla\mathbf{F_{k-1}}\frac{\partial\mathbf{W}}{\partial\mathbf{p_{k-1}}} \tag{14}$$

$$\begin{aligned} \mathbf{b} = &\sum_{j=1}^{n}\sum_{k=1}^{m_j}\mathbf{J_1^k}^{\mathrm{T}}F(\mathbf{W}(\mathbf{x_k};\mathbf{p}) + \\ &\lambda\sum_{j=1}^{n}\sum_{k=2}^{m_j}\mathbf{J_2^k}^{\mathrm{T}}[F(\mathbf{W}(\mathbf{x_k};\mathbf{p})) - F(\mathbf{W}(\mathbf{x_{k-1}};\mathbf{p}))]. \end{aligned} \tag{15}$$



(a) video frame (b) edge template (c) distance map (d) feature map (e) profile of lines in (c) (d)

Fig. 4. Feature map. In (e), the red line D corresponds to the red line in (c), $SD = D^2$, the green line F corresponds to the green line in (d), $SF = F^2$.

Algorithm 1 Optimization

Pre-compute:

1) Warp template edge pixel $\mathbf{x_{tp}}$ to initialize $\mathbf{x_{tp_i}} = \mathbf{W}(\mathbf{x_{tp}}; \mathbf{p_{i-1}})$ at frame i
2) Generate the feature map $\mathbf{F}$ based on $\mathbf{x_{tp_i}}$
3) Evaluate the gradient $\bigtriangledown\mathbf{F}$ of the feature map $\mathbf{F}$
4) Sample n edge pixels $\mathbf{x_{tr}}$ from the current frame

Iterate:

5) Warp target edge pixel to compute: $\mathbf{x_{tr_{itr}}} = \mathbf{W}(\mathbf{x_{tr}}; \mathbf{p_{tmp}})$
6) Evaluate the $\frac{\partial \mathbf{W}}{\partial \mathbf{p}}$ at $\mathbf{x_{tr_{itr}}}$
7) Compute $\bigtriangledown\mathbf{F_k}\frac{\partial \mathbf{W}}{\partial \mathbf{p_k}}$ and $\bigtriangledown\mathbf{F_k}\frac{\partial \mathbf{W}}{\partial \mathbf{p_k}} - \bigtriangledown\mathbf{F_{k-1}}\frac{\partial \mathbf{W}}{\partial \mathbf{p_{k-1}}}$ at $\mathbf{x^k_{tr_{itr}}}$ and $\mathbf{x^{k-1}_{tr_{itr}}}$
8) Compute (12) - (15)
9) Compute $\mathbf{\Delta p_{tmp}}$ using (11)
10) Update the parameter $\mathbf{p_{tmp}} \leftarrow \mathbf{p_{tmp}} + \mathbf{\Delta p_{tmp}}$

until $\|\Delta error\| < \epsilon$ **and** $itr > 30$

11) Update the parameter $\mathbf{p_i} \leftarrow \mathbf{inv}(\mathbf{p_{tmp}}) * \mathbf{p_{i-1}}$

The optimization process is summarized as Algorithm 1.

III. EXPERIMENTAL RESULTS

To evaluate the performance of our proposed tracking method, we test it on five newly collected video sequences and compare it's performance with other four pertinent methods[1].

A. Dataset and Evaluation Measure

We collect five video sequences (1896 frames in total). The frame size is 640×480 (width×height). As shown in Fig. 6, these to-be-tracked targets are planar rigid edge templates including box rim, cup rim, hexagon rim, ring contour and disc contour. Their motions and transformations can be described by 2D planar homography matrices. Except for edges, there are few stable and salient regional textures on these targets. In addition, partial occlusions are also introduced in some frames of every video sequence.

We quantitatively evaluate our method using the error metric defined in [14] and [15]:

$$Error = max\{\frac{E \otimes Dist_{E_{gt}}}{P_E}, \frac{E_{gt} \otimes Dist_E}{P_{E_{gt}}}\}, \qquad (16)$$

[1]video: https://youtu.be/ohnUCm-Ffc4

where E and E_{gt} are the tracked edge template and the ground truth template (manually labeled by ByLabel [18]). $Dist_E$ and $Dist_{E_{gt}}$ denote the distance transform maps of E and E_{gt}. P_E and $P_{E_{gt}}$ indicate the pixel number of E and E_{gt} respectively. $\otimes$ represents the summation of element-wise multiplication. A tracker's overall accuracy on a given sequence is measured by the success rate [2], which is defined as the ratio of frames where the tracking error $Error$ is less than a threshold of e_t pixels and the total frames.

B. Quantitative Evaluation

We compare our tracking method against the following four methods: DIST which tracks a given edge template using distance transform as template's feature map, RRC [16] which is adapted from a salient boundary grouping method, BDSP [14] which is a closed boundary tracking method via graph based line segments perceptual grouping and EFG [15] which is an edge fragments grouping based contour tracking method. To implement DIST, we adapt our method by replacing $F = \sqrt[4]{D}$ with $F = D$ and maintaining other settings. The parameters of RRC, BDSP and EFG are set to their default values as described in [16], [14] and [15].

Fig. 5 illustrates the success rate curves of these methods. Note, that our method outperforms others on the five video sequences in terms of success rate and overall average error (see, Table I).

DIST easily fails from outliers (see green edges in Fig. 6c, 6i, 6l, 6s and 6w). The outliers are usually edge pixels related to occlusions and background. The feature map of DIST (Fig. 4c and blue line in 4e) is not able to suppress the impacts of those pixels.

RRC, BDSP and EFG work well in the hexagon sequence (see Fig. 5c). Although the partial occlusions, such as those shown in Fig. 6m and 6n, produce certain tracking errors, they fail to collapse these three trackers. Because the target motion in this sequence is relatively slow and the target contour has strong saliency. However, RRC fails quickly in the other four sequences due to outliers and large area occlusions, see yellow contours in Fig. 6c 6i, 6r and 6x. Although BDSP and EFG perform better than RRC on the mug and ring sequences respectively (see Fig. 5e and 5b), they are still less competitive than our method in terms of success rates (see Fig. 5e and 5b) and overall average errors (see Table I). Experiments on sequence box (Fig. 5a) and disc (Fig. 5d) further demonstrate the robustness of our method.

TABLE I. Overall average error (pixel)

Method	Ours	DIST	RRC	BDSP	EFG
box_359	**1.72**	23.63	39.08	20.18	24.96
disc_390	**0.69**	25.05	61.89	8.48	55.84
hexagon_389	**2.27**	48.08	4.64	3.55	2.31
mug_372	**1.64**	17.30	53.61	3.15	27.06
ring_386	**1.57**	41.28	85.26	85.26	2.66
average	**1.58**	31.07	48.90	24.12	22.57

TABLE II. Average time cost of our method (ms)

Video	box	mug	hexagon	ring	disc
ave. detection	4.88	4.61	4.35	3.51	4.89
ave. splitting	2.76	2.73	4.67	2.64	3.12
ave. optimization	23.65	23.91	24.24	23.07	22.09
ave. total	**31.29**	**31.25**	**33.26**	**29.22**	**30.10**

Compared with DIST, our method uses $F = \sqrt[4]{D}$ (see green line in Fig. 4e) as the feature map, which reduces the impacts of outlier pixels greatly (see the second and third column in Tab. I). Different from grouping based methods RRC, BDSP and EFG, our method relies on the homography transformation which provides more strict constraint than saliency measure. Together with the newly proposed feature map and tracking cost, our method achieves higher accuracy (see the fourth, fifth and sixth column in Tab. I) and more robust performance to large area occlusions (see Fig. 6c and 6v).

C. Runtime

We implement our tracking method using C++ and OpenCV on a machine with a quad core 3.10 GHz Intel Core i5 processor, 16GB RAM and Ubuntu 14.04 64-bit OS. The average processing speed of our method on each of the five video sequences is illustrated in Table II. As we can see, the speed of our method mainly depends on three parts: edge detection, edge splitting and optimization. The optimization costs the most time. It is worth to note that our implementation is an unoptimized version. The average total time cost of each frame is about 30 ms, which means our method is acceptable in real-time applications.

IV. CONCLUSIONS

We proposed a novel method for rigid planar edge template tracking using a feature map derived from a distance transform on image edges. The feature map is defined as the fourth root of the distance transform. This suppresses the impacts of outlier pixels and improves robustness greatly. Instead of conducting optimization using independent edge pixels directly, our method detects edge fragments and filtered some obvious outliers via simple thresholds. To guarantee real-time performance, a certain number (100) of edge pixels are sampled along these retained edge fragments uniformly and sequentially for the optimization. We evaluated the performance of our method on real-world video sequences and compared it to other tracking methods. The results demonstrated that our method performs better than others in terms of robustness and accuracy. The real-time speed of our method is also validated. The results show that our method is promising in real-time computer vision and robotics related applications, such as Augmented Reality (AR), robot localization and visual servoing.

REFERENCES

[1] Y. Wu, J. Lim, and M. Yang, "Object tracking benchmark," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1834–1848, 2015.

[2] A. Singh and M. Jägersand, "Modular tracking framework: A fast library for high precision tracking," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2017, Vancouver, BC, Canada, September 24-28, 2017*, 2017, pp. 3785–3790.

[3] M. Kristan and et al., "The visual object tracking VOT2016 challenge results," in *Computer Vision - ECCV 2016 Workshops - Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part II*, 2016, pp. 777–823.

[4] G. Borgefors, "Distance transformations in digital images," *Computer Vision, Graphics, and Image Processing*, vol. 34, no. 3, pp. 344–371, 1986.

[5] ——, "Hierarchical chamfer matching: A parametric edge matching algorithm," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 10, no. 6, pp. 849–865, 1988.

[6] T. Ma, X. Yang, and L. J. Latecki, "Boosting chamfer matching by learning chamfer distance normalization," in *Computer Vision - ECCV 2010 - 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part V*, 2010, pp. 450–463.

[7] P. F. Felzenszwalb and D. P. Huttenlocher, "Distance transforms of sampled functions," *Theory of Computing*, vol. 8, no. 1, pp. 415–428, 2012.

[8] A. Thayananthan, B. Stenger, P. H. S. Torr, and R. Cipolla, "Shape context and chamfer matching in cluttered scenes," in *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2003), 16-22 June 2003, Madison, WI, USA*, 2003, pp. 127–133.

[9] H. Li, T. Shen, and X. Huang, "Approximately global optimization for robust alignment of generalized shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 6, pp. 1116–1131, 2011.

[10] D. Holz, A. E. Ichim, F. Tombari, R. B. Rusu, and S. Behnke, "Registration with the point cloud library: A modular framework for aligning in 3-d," *IEEE Robot. Automat. Mag.*, vol. 22, no. 4, pp. 110–124, 2015.

[11] A. Hofhauser, C. Steger, and N. Navab, "Edge-based template matching and tracking for perspectively distorted planar objects," in *Advances in Visual Computing, 4th International Symposium, ISVC 2008, Las Vegas, NV, USA, December 1-3, 2008. Proceedings, Part I*, 2008, pp. 35–44.

[12] S. Baker and I. Matthews, "Lucas-kanade 20 years on: A unifying framework," *International journal of computer vision*, vol. 56, no. 3, pp. 221–255, 2004.

[13] S. Holzer, S. Hinterstoisser, S. Ilic, and N. Navab, "Distance transform templates for object detection and pose estimation," in *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, 2009, pp. 1177–1184.

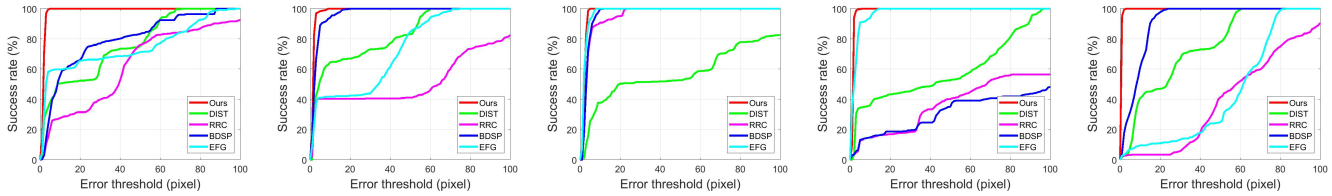
[14] X. Qin, S. He, C. P. Quintero, A. Singh, M. Dehghan, and M. Jägersand, "Real-time salient closed boundary tracking via line segments perceptual grouping," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2017, Vancouver, BC, Canada, September 24-28, 2017*, 2017, pp. 4284–4289.

[15] X. Qin, S. He, Z. Zhang, M. Dehghan, and M. Jägersand, "Real-time salient closed boundary tracking using perceptual grouping and shape priors," in *28th British Machine Vision Conference, BMVC 2017, London, UK, September 4-7, 2017*, 2017.

[16] J. S. Stahl and S. Wang, "Edge grouping combining boundary and region information," *IEEE Trans. Image Processing*, vol. 16, no. 10, pp. 2590–2606, 2007.

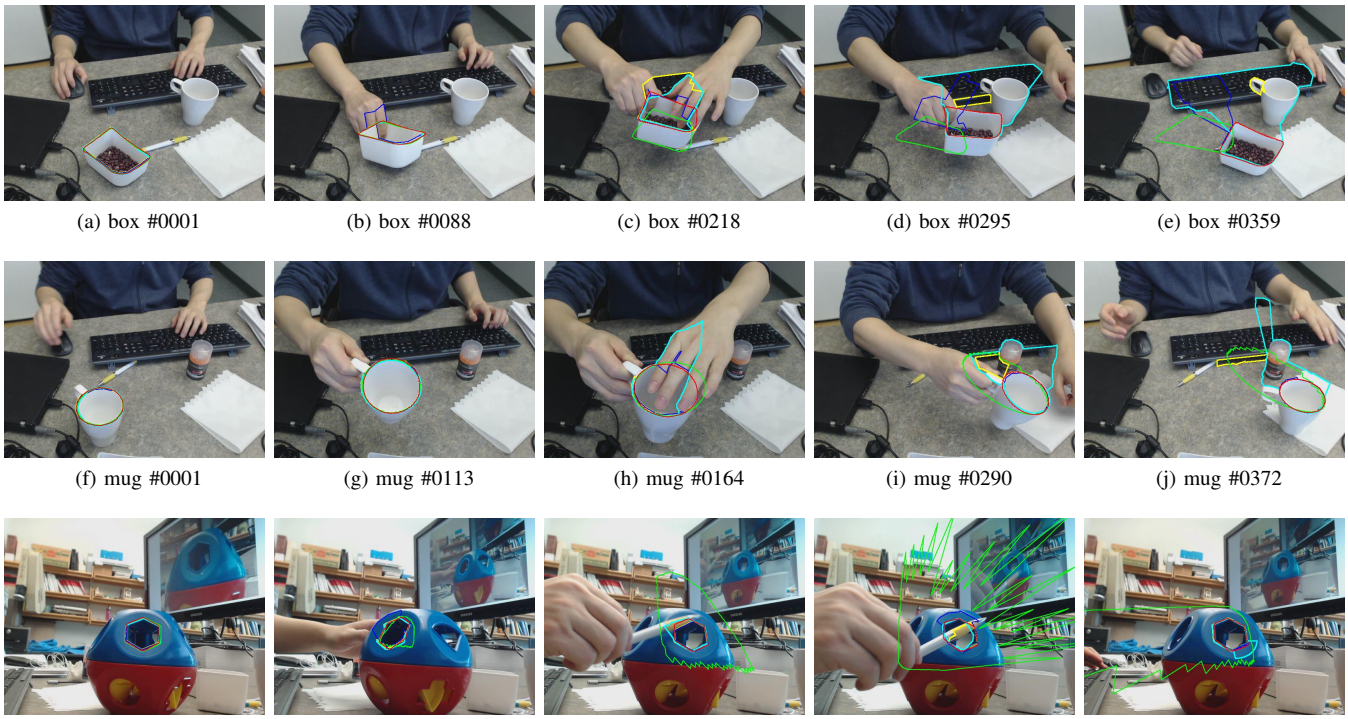
[17] C. Topal and C. Akinlar, "Edge drawing: A combined real-time edge and segment detector," *J. Visual Communication and Image Representation*, vol. 23, no. 6, pp. 862–872, 2012.

[18] X. Qin, S. He, Z. V. Zhang, M. Dehghan, and M. Jägersand, "Bylabel: A boundary based semi-automatic image annotation tool," in *2018 IEEE Winter Conference on Applications of Computer Vision, WACV 2018, Lake Tahoe, NV, USA, March 12-15, 2018*, 2018, pp. 1804–1813.



(a) box_359 (b) mug_372 (c) hexagon_389 (d) ring_386 (e) disc_390

Fig. 5. Success rate. The number after the underscore is the frame number of each sequence, e.g. (a) box_359 means that the box sequence has 359 frames.



(a) box #0001 (b) box #0088 (c) box #0218 (d) box #0295 (e) box #0359

(f) mug #0001 (g) mug #0113 (h) mug #0164 (i) mug #0290 (j) mug #0372

(k) hexagon #0001 (l) hexagon #0080 (m) hexagon #0266 (n) hexagon #0331 (o) mug #0389

(p) ring #0001 (q) ring #0057 (r) ring #0068 (s) ring #0267 (t) ring #0386

(u) disc #0001 (v) disc #0055 (w) disc #0206 (x) disc #0290 (y) disc #0390

Ours DIST RRC BDSP EFG

Fig. 6. Tracking results of sampled frames.