Vis Comput (2013) 29:861–870
DOI 10.1007/s00371-013-0847-8

ORIGINAL ARTICLE

SimLocator: robust locator of similar objects in images

Yan Kong · Weiming Dong · Xing Mei · Xiaopeng Zhang · Jean-Claude Paul

Published online: 11 June 2013

Abstract Similar objects commonly appear in natural images, and locating and cutting out these objects can be tedious when using classical interactive image segmentation methods. In this paper, we propose *SimLocator*, a robust method oriented to locate and cut out similar objects with minimum user interaction. After extracting an arbitrary object template from the input image, candidate locations of similar objects are roughly detected by distinguishing the shape and color features of each image. A novel optimization method is then introduced to select accurate locations from the two sets of candidates. Additionally, a matting-based method is used to improve the results and to ensure that all similar objects are located in the image. Finally, a method based on alpha matting is utilized to extract the precise object contours. To ensure the performance of the matting operation, this work has developed a new method for foreground extraction. Experiments show that *SimLocator* is more robust and more convenient to use compared to other more advanced repetition detection and interactive image segmentation methods, in terms of locating similar objects in images.

Keywords Similar objects · Object descriptor · Object matching · Stable locations · Contour extraction

Y. Kong · W. Dong (✉) · X. Mei · X. Zhang
LIAMA-NLPR, Institute of Automation, Chinese Academy of Sciences, Beijing, China
e-mail: wmdong@nlpr.ia.ac.cn

X. Mei
e-mail: xmei@nlpr.ia.ac.cn

X. Zhang
e-mail: xpzhang@nlpr.ia.ac.cn

J.-C. Paul
Project CAD, INRIA, Paris, France

1 Introduction

Similar image elements commonly appear in both natural and artificial scenes, which provide multiple and non-local records of the same data. These similar objects may be randomly scattered in an image, and their spatial distribution and appearance variance reinforce the image's visual effect. Moreover, using identical or similar objects that are repeated in a pattern is a standard graphic design concept [13], being a prominent compositional feature in many photographs. Therefore, locating similar objects in an image is useful in understanding the aesthetics and mechanism of a particular scene in the real world [23]. Locating and extracting such kind of redundancy accurately and efficiently can also be exploited to consolidate image edits in a non-local fashion [7].

Previous repeated elements locating or cutout methods used solely geometric [7] or color [10] similarity as criteria for matching. However, these methods fail to detect accurately similar objects with obvious variance in illumination, outer shape, inner textures, or colors (e.g., Fig. 1). Interactive image segmentation systems, such as *GrabCut* [21], *Lazy Snapping* [16] and *Paint Selection* [17], are also not suitable for segmenting similar objects because excessive manual operations may be needed when the number of objects is large, or when these objects overlap each other.

In this paper, we present a new method, namely, *SimLocator*, to analyze similar patterns in an image. After segmenting an object from the input interactively, we incorporate the shape and color features into a template matching framework to identify possible similar object locations within the scene. We then develop a joint searching scheme to determine high confidence locations within the two candidate sets, and utilize this information to calculate the final object locations. In some cases, the result can be improved further by a matting-based method to obtain a more complete locating result. After obtaining the object locations,

(a) SimLocator (b) RepFinder [7] (c) Dual-Bound [22] (d) RepSnapping [10]

Fig. 1 Comparison of different methods. Results show that the *SimLocator* method outperforms *RepFinder* and Dual-Bound in both accuracy (detecting the correct object locations) and completeness (finding all objects). *RepSnapping* can cut out the foreground from the scene, but cannot find the location and contour of each pomegranate

a new method to obtain the precise object regions is utilized to refine the contours of the objects and cut them out from the scene. Thus, our algorithm can locate similar objects that may be dramatically different in colors, textures, or even shapes, rather than the very approximately same patterns in *RepFinder* [7].

2 Related work

Various object detection methods that utilize color-, texture-, or shape-based features have been previously presented. Leung and Malik [14] proposed a graph-based algorithm to extract repeated units by growing elements using a graph. Berg et al. [5] achieved recognition by means of deformable shape matching and identification of correspondences among feature points. Ahuja and Todorovic [2] detected natural texels by searching within a segmentation tree. Unfortunately, this method is too slow for interactive applications, and is limited to examples imaged from a viewing direction, which is nearly along the surface normal. Usually, local feature descriptors, such as shape contexts [4], SIFT [18] and SURF [3], are used for object detection, and these factors can reliably match different views of an object or a scene, but cannot capture high-level scene structure. Pauly et al. [20] presented a method to discover regular or repeated geometric structures in 3D shapes. Automatic color-based affine covariant region detectors, such as MSCR [9], are suitable for detecting simple objects located in a smooth background. However, stochastic object distributions, overlapping, or subtle shape and color variations render the above methods unsuitable.

Given that the same or similar objects appear in a set of images, co-segmentation methods can segment a particular object from the set of images simultaneously. Joulin et al. [11] combined bottom-up image segmentation tools within a discriminative clustering framework to assign all images with the labels of either foreground or background. Meng et al. [19] formulated the co-segmentation problem as the shortest path problem based on local region similarities and saliency models. Kim et al. [12] solved the co-segmentation problem by using hierarchical clustering. Rather than segmenting similar objects from multiple images, the method presented in the current paper focuses on locating similar elements in a single image.

Only a few studies have focused on the detection of general repeated or similar objects. Cheng et al. [7] presented a user-assisted approach in identifying approximately repeated scene elements in an image. Using boundary band matching to locate possible elements and employing active contours to obtain object boundaries, this efficient system is convenient for reconstructing scene structures. Nevertheless, as their method relies on similarity of boundary maps, illumination and inner pattern variations limit the accuracy of their results. On the other hand, the prerequirement of foreground matting also limits the practical usage of *RepFinder*, as separating the foreground from the image through simple user interactions can be difficult in many cases. Schweitzer et al. [22] presented a fast template matching technique that uses both lower and upper bounds on the match measure to compute for the K best matches. However, their method, which employs the Walsh transform based matching algorithm that is very sensitive to inner texture and illumination variations, is not fit for the detection of similar objects in a natural image. Moreover, the requirement of user-indicated number of objects also limits the application of this method. Huang et al. [10] presented a graph-based method to cut out repeated elements from a scene. The limitation of their method is its strong dependence on very similar colors of objects. Furthermore, their method experiences difficulties in detecting accurate location and contours of each object, especially for partially occluded objects. *To the best of our knowledge, a fully automatic method that can detect similar arbitrary objects from a single natural image has not been proposed.*

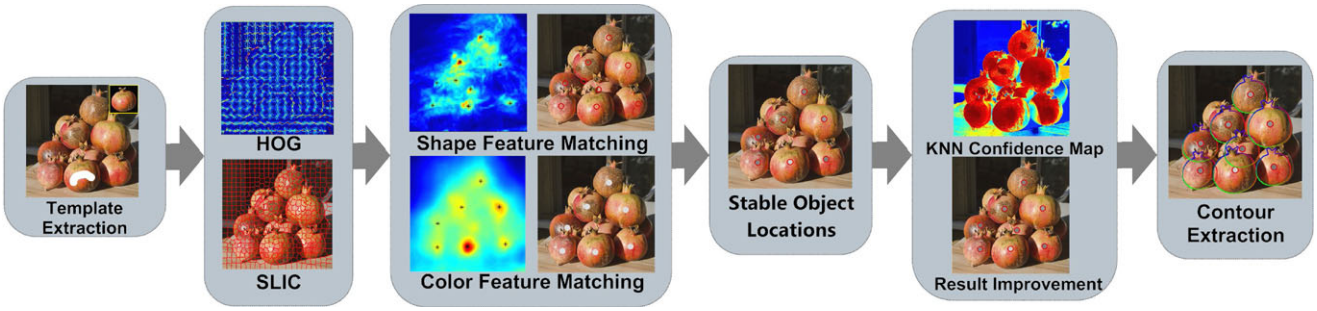


Fig. 2 Algorithm workflow. User scribbles indicate an object template. We use Histogram of Oriented Gradients (HOG) as the shape feature descriptor and SLIC (Simple Linear Iterative Clustering) to generate the color feature descriptor (histogram-based). Similar object instances are located by our joint features matching method. Matting-based methods are developed to improve the locating result further and extract precise object contours

(a) Objects locating by HOG (b) Objects locating by color (c) Stable object locations (d) RepFinder

Fig. 3 We separately use HOG and color descriptor to locate the objects, and then find stable locations for the final result

3 Overview

Our method employs an interactive framework to extract similar objects in a particular image. The algorithm workflow is shown in Fig. 2.

In the object detection stage, our system allows users to select a sample of objects that have multiple instances in the input image. Those objects may be subject to deformation, overlap, illumination influence, and appearance variation. Shape and color templates of user-selected object type(s) are extracted automatically. Our system then uses template matching to obtain two sets of candidate object instances by separately using the shape and color descriptor. A joint optimization scheme is developed to decide the final locating result.

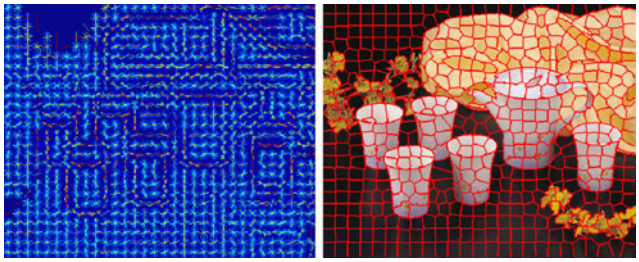
In boundary extraction stage, we use alpha matting to obtain a precise object region. Thus, in order to get a good matting result, we introduce a trimap foreground extraction algorithm to obtain better foreground.

4 Similar object locating

We use *PaintSelection* [17] on the input image to extract an instance from the similar objects. In terms of the selection of a feasible feature to use in the template matching process, different types of features may achieve dramatically different matching results (Fig. 3). Usually, building a stable template matching system with a single feature for similar object locating is difficult because of the objects' vivid inner textures, self-occlusion, or color/lighting variations. Therefore, in order to make the object location more robust and accurate, we use two features in the template matching process and separately obtain candidate object locations for further optimization.

4.1 Object feature descriptors

For its desirable performance in a variety of tasks, its speed, robustness, adaptability to sliding window search, and popularity in the community, HOG template descriptor [8] is integrated in the proposed method as a shape descriptor to detect the possible similar object locations. For each image window, the input image is divided into small spatial regions ("cells"), and for each cell N-bins local 1-D HOG is accumulated. In our experiments, we use $N_b^H = 9$ bins and set the cell size as 8×8. The orientation bins are spaced over $0°$ to $180°$. Each pixel in one cell is voted to its histogram based on gradient direction and magnitude. However, in contrast with the method in [8], a normalized local histogram is not used over a larger spatial region ("block") because normalization will decrease the separating capacity of the HOG descriptor in our template matching process. Figure 4(a) shows an example of HOG features extracted from an input image.



(a) HOG (b) SLIC segmentation

Fig. 4 Shape (HOG) and color (super-pixel histogram) features used in our template-matching algorithm

To enhance the robustness of our algorithm in complex environments, a color feature descriptor is developed to perform a separate locating process. We first segment both the template and the input image by SLIC [1]. Subsequently, we use a three-channel color histogram to represent the distribution of colors in each super-pixel. In our experiments, all colors are represented in HSV color space, and each channel is divided into $N_b^C = 32$ bins. Figure 4(b) shows an image segmentation example using SLIC method.

4.2 Object locating by matching

Similar objects are accurately located using object descriptors. For template $\mathbf{T}$, let $\mathcal{T} = \{p_m | m = 1, \ldots, N_\mathbf{T}\}$ denote the set of feature points (HOG or color) in the template, where $N_\mathbf{T}$ is the number of elements in $\mathbf{T}$. To find similar objects efficiently, the template in the input image at each possible location is matched by scaling and rotating the image according to pre-set discrete intervals. In all our experiments, we pre-compute the object template in the discrete space of seven discrete angles $\{-120°, -45°, -10°, 0°, 10°, 45°, 120°\}$ and five scales $\{0.4, 0.8, 1.0, 1.2, 1.6\}$. We mainly employ histogram intersection to calculate the matching scores at each candidate location by separately using HOG and color descriptor. Two score maps are used to record the HOG and color matching scores at each location. Moreover, the feature point set (HOG or color) of the current sub-window $\mathbf{w}$ is denoted as $\mathcal{W} = \{q_n | n = 1, \ldots, N_\mathbf{w}\}$, where $N_\mathbf{w}$ is the number of elements in $\mathbf{w}$. Thus, for a rotated and scaled template T_i at a candidate location j, we formulate equations to calculate the HOG matching score S_H and the color matching score S_C as:

$$S_H(T_i, w_j) = \frac{1}{N_{T_i}^H} \sum_{\substack{p_m \in \mathcal{T}^H, \\ q_n \in \mathcal{W}^H}} \frac{\sum_{k=1}^{N_b^H} \min(T_{k,p_m}^H, w_{k,q_n}^H)}{\sum_{k=1}^{N_b^H} w_{k,q_n}^H} \tag{1}$$

$$S_C(T_i, w_j) = \frac{1}{N_{T_i}^C} \sum_{\substack{p_m \in \mathcal{T}^C, \\ q_n \in \mathcal{W}^C}} \frac{\sum_{k=1}^{N_b^C} \min(T_{k,p_m}^C, w_{k,q_n}^C)}{\sum_{k=1}^{N_b^C} w_{k,q_n}^C}, \tag{2}$$

where T_{k,p_m}^H and S_{k,q_n}^H are the corresponding HOG histograms of points p_m and q_n, and T_{k,p_m}^C and S_{k,q_n}^C are the corresponding color histograms, respectively. Both scores are normalized to [0,1].

After scanning the entire input image with scale and rotation variations, we choose the maximum score at each possible location as the final score and then record this value in the score map. Then, in both score maps, each local maximum matching score that is larger than $\eta = 0.5$ is chosen as a candidate object location. We ignore a local maximum if another local maximum with a larger matching score is seen within the distance threshold d. In our experiments, we set d to half of the bounding box circumcircle radius of the template. We take $\mathcal{P}_H = \{X_i | i = 1, \ldots, N_H\}$ to denote the set of candidate object locations acquired from the HOG score map, and $\mathcal{P}_C = \{Y_j | j = 1, \ldots, N_C\}$ to denote the set acquired from the color score map. N_H and N_C are the numbers of elements in $\mathcal{P}_H$ and $\mathcal{P}_C$.

As shown in parts (a) and (b) of Fig. 3, the locating result obtained by using the shape and color descriptors may be evidently different due to the complex visual variations of the objects. Different features are sensitive to different variations. Moreover, wrong locations may also be included in the candidate sets because of the disturbance of the background ((a) and (b) of Fig. 3). Therefore, to obtain accurate object locations, high-confidence locations must be determined first. For the candidates in the two sets, we separately calculate the Euclidean distance of each candidate pair (X_i, Y_j), and use the following metric to include the feasible pairs (stable locations) into a new set $\mathcal{P}_S$, which is formulated as follows:

$$\begin{aligned} \mathcal{P}_S = \Big\{ (X_i', Y_j') | X_i' &= \arg\min_i D(X_i, Y_j'), \\ Y_j' &= \arg\min_j D(X_i', Y_j), \\ D(X_i', Y_j') &\le D_{\max} \Big\} \end{aligned} \tag{3}$$

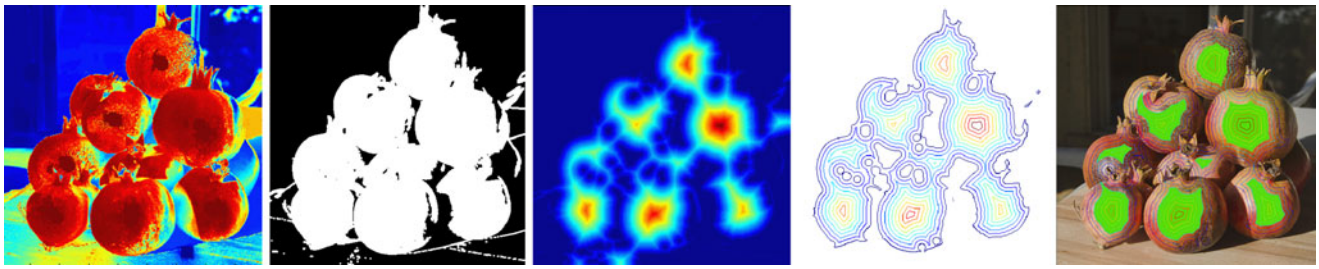
where $D(X_i, Y_j)$ is the Euclidean distance between X_i and Y_j, $X_i \in \mathcal{P}_H$, $Y_j \in \mathcal{P}_C$. We set $D_{\max}$ to half of the bounding box circumcircle radius of the template. If N_p denotes the number of stable locations in $\mathcal{P}_S$, we then define the stable confidence values of HOG and color descriptors as $c_H = \frac{N_p}{N_H}$ and $c_C = \frac{N_p}{N_C}$, respectively. Thus, the coordinates of the stable locations are calculated as follows:

$$Z_k = \beta \cdot X_i' + (1.0 - \beta) \cdot Y_j', \quad k = 1, \ldots, N_p, \tag{4}$$

where $\beta = \frac{c_H}{c_H + c_C}$. As shown in Fig. 3(c), we use the coor-

(a) Locating by HOG (b) Locating by color (c) Stable locations (d) KNN map (e) Improved by matting

Fig. 5 Optimizing the locating result by stable locations and KNN matting. Wrong locations can be removed in this process



(a) S_{alpha} (b) Step 1 (c) Step 2 (d) Step 3 (e) Step 4

Fig. 6 Workflow of Algorithm 1. (**a**) S_{alpha} is calculated from the KNN matting operation in Sect. 4.3. (**b**) Binary map generated from S_{alpha}. (**c**) Distance map Map_{dist}. (**d**) *Contour map* used for constructing a contour tree. (**e**) *Green areas* are the foreground areas found by Algorithm 1

dinates of the stable locations as the final location of similar objects.

4.3 Result improvement by matting

In most cases in our experiments, all the similar objects can be found accurately by using the above locating method. However, a few objects may sometimes be lost during the screening process of using the metric in Eq. (3). Certain objects may appear to have apparent shape or color differences compared with the template (but still belong to the same category) because of some environmental or artificial conditions. Therefore, to enhance the effectiveness of our algorithm, we combine the S_H and S_C as S_{Combine}, and then use KNN matting [6] to complete the result set. With $\mathcal{P}_S$, we can use the score map, which is more similar to the template, as the main component of S_{Combine} and define it as follows:

$$\alpha_H = \frac{N_p}{N_H}, \qquad \alpha_C = \frac{N_p}{N_C},$$
$$S_{\text{Combine}} = \frac{\alpha_H * S_H + \alpha_C * S_C}{\alpha_H + \alpha_C}, \tag{5}$$

where N_p is the number of stable object locations in $\mathcal{P}$. Thus, we sample randomly around the locations of $\mathcal{P}$ as the foreground locations, and likewise, sample randomly where scores are lower than 0.1 as background locations. Using this information, we can obtain an alpha score map S_{alpha} by KNN matting, which describes the confidence of each pixel. Finally, for each local peak score in S_{Combine}, we choose the locations in which the local average alpha score is larger than 0.9 as the final similar object locations. As shown in Fig. 5, all the similar objects are located after the matting process.

5 Contour extraction

With the information on similar object locations, we can employ alpha matting [15] to obtain the precise region of each location and then extract its outer contour. In using alpha matting, we first need to construct a trimap for each object to specify the foreground, background, and unknown regions. However, alpha matting may not perform as desired if the small area around the object location is used as foreground, as such information is too limited to extract a precise object region. A new method (Algorithm 1) solves this problem by extracting the foreground area for an object trimap, as illustrated in Fig. 6. With the confidence score given by S_{alpha} (calculated from the KNN matting operation in Sect. 4.3), we can estimate a foreground area around each object location. Thus, we treated the area determined by Algorithm 1 as the foreground, the area out of the template window as the background, and the area in the middle as the unknown region.

However, if the background or the inner texture is complex, alpha matting will fail to extract precise object contours. S_{tmpl} denotes the area covered by the template contour in one object location, and A_{matting}, the area extracted by alpha matting. If $\frac{\|A_{\text{matting}} - A_{\text{tmpl}}\|}{A_{\text{tmpl}}} > 0.3$, we use an active

Algorithm 1: Trimap Foreground Extract

Input: S_{alpha}, template area A_{tmpl}, candidate locations set P

Output: Foreground area of each candidate location

1. Convert S_{alpha} to a binaries map B_{alpha} using threshold 0.8, then dilate B_{alpha} using 3×3 template in order to fill small holes;
2. Calculate a distance map Map_{dist};

for *each pixel p in* B_{alpha} **do**
  $Map_{\text{dist}}[p] =$ the Euclidean distance between p and the nearest zero pixel;

3. Build a contour tree of Map_{dist}. Set the lower contour as the root and the higher contour as the leaf;
4. Let L_{area} as candidate foreground area;

for *each local maximum point p in* Map_{dist} **do**
  a. Find the local maximum point q which the Euclidean distance between p and q is smaller than $\sqrt{A_{\text{tmpl}}}$, put p into set Q;
  b. R is a contour;
  if $Q == \emptyset$ **then**
    Let R as the largest contour which $area(R) < A_{\text{tmpl}}$ and p in the area of R;
  else
    Let $R =$ largest contour;
    for *each point q in set* Q **do**
      (1) Let F as the lowest common ancestor between p and q in contour tree;
      (2) Let U as the largest contour which smaller than S_{tmpl} and contain p;
      (3) Let $R = U$;
  Add R to L_{area};

5. For the candidate locations p, the foreground area is $L_{\text{area}} \cap TemplateWindow(p)$;

contour-based model in [7] to refine the object contour further. In most cases, the contour extracted by alpha matting is sufficient, and is usually better than that by pure active contour method for the final object segmentation. As shown in Fig. 7, the object contours extracted by our method are more precise and smooth than the ones by only using active contour method.

6 Results and discussion

Our interactive system is targeted toward high level image pattern analysis. In the entire workflow, the cutout of the template is the only manual operation. We have implemented our algorithm in C++ on a computer with Intel Core i7 CPU at 3.9 GHz, 8 GB RAM, and Geforce GTX 670. Our CUDA-accelerated template matching plus the searching of stable locations typically takes 0.6 seconds to process a 900×900 image under seven rotations and five scalings. For our locating improvement and automatic object contour refinement through matting, the computation time depends on the image size, but usually consumes 2.0 to 4.5 seconds, thereby guaranteeing that the entire workflow of the system runs at interactive rates.

(a) Active Contour Only (b) Our result

Fig. 7 Object contour extraction. (**a**) Object contour is extracted only using active contour. (**b**) Object contour is extracted using our method. The two leftmost occluded pandas are extracted through active contour, whereas the others are extracted by alpha matting

To validate the effectiveness of our *SimLocator*, we have tested it on a variety of input images with similar scene elements. Then, we discuss the experiments used to evaluate the performance of *SimLocator* qualitatively and to draw a comparison with other state-of-the-art repetition detection methods. The key ingredients of *SimLocator* are the similarity measurements S_H (Eq. (1)) and S_C (Eq. (2)), and the stable location selection metric $\mathcal{P}_S$ (Eq. (3)). In the experiments conducted in this paper, we use both shape and color information. More feature descriptors, such as Gabor texture and SIFT/SURF descriptor, can also be incorporated into our method.

6.1 Comparison

We compare *SimLocator* with the state-of-the-art interactive repetition detection methods *RepFinder* [7], *DualBound* [22] and *RepSnapping* [10]. To attain a fair comparison, we also run *RepFinder* on the original image without segmenting the entire foreground first, as in their paper. In fact, for many examples, segmenting the foreground by using a few strokes is difficult, and thus we treat this pre-processing requirement as a strong limitation for a similar objects detection system. By contrast, our algorithm can accurately and efficiently locate similar objects directly on the original image, which allows for more practical uses than *RepFinder*. Figure 8 shows the results of the comparison. Our method is more robust than the other template-

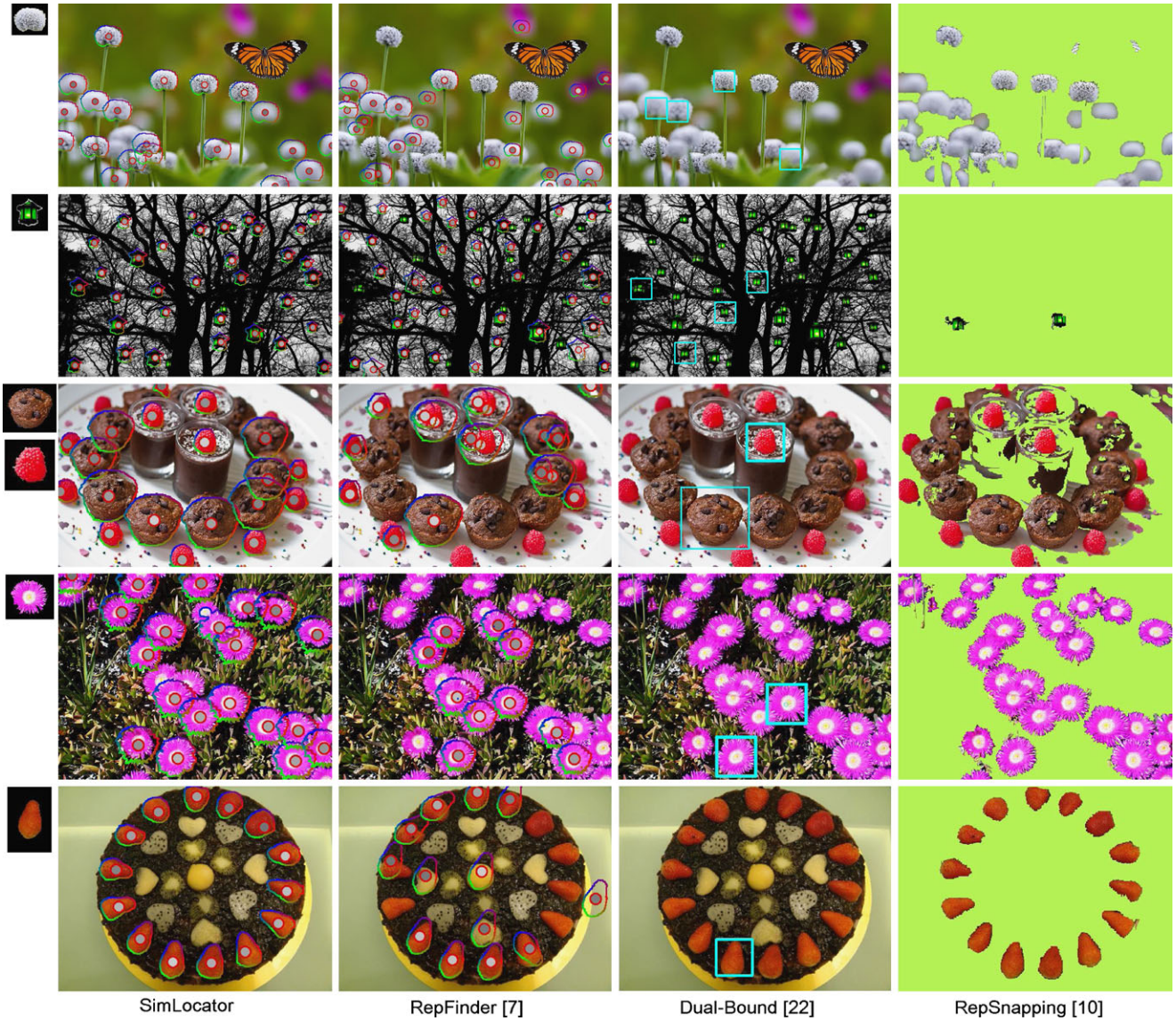


Fig. 8 Comparison of different methods locating similar objects. Templates are shown in the *leftmost column*

matching-based methods *RepFinder* and *Dual-Bound* in terms of finding object locations in a scene completely and accurately. The shape matching algorithm used in *RepFinder* is sensitive to the defocused effect (dandelions), moderate shape changes (chocolate cakes and strawberries), and complex textural backgrounds (green lanterns and purple flowers). In our algorithm, however, the integration of the color feature and stable location searching scheme effectively solve those problems. On the other hand, *Dual-Bound* is somewhat too accurate for a matching method. In most cases in our experiments, *Dual-Bound* can only find the template itself, which is not suitable for locating similar objects in natural images. Finally, *RepSnapping* can simultaneously cut out almost all the repeated elements with limited user interactions. However, the method is very sensitive to color variations (green lanterns), and is also unable to acquire information on individual object locations. Figure 9 illustrates more comparison results. *SimLocator* is more robust than *RepFinder* when the image has a perspective effect (meatballs and strawberries), 3D transformation (flowers), or severe shape deformation (fried balls and strawberries).

6.2 Template selection

In our experiments, the selection of template does not largely affect the setting of parameters and the accuracy of the detection result, if the user does not choose an object that is too different from the other objects. As shown in Fig. 10, our algorithm is robust when using variant templates. In this respect, *SimLocator* outperforms *RepFinder* [7] for both locating operations.

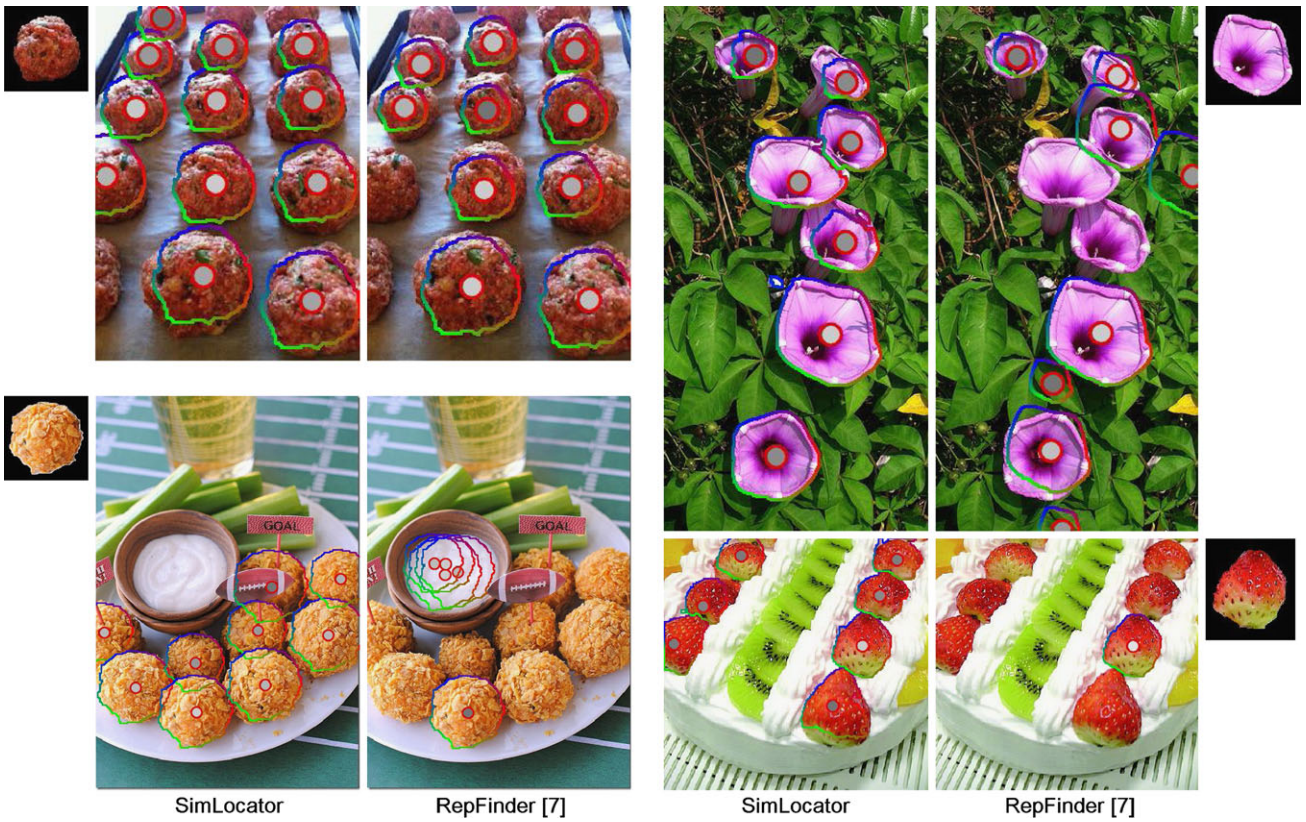


Fig. 9 Comparison of our method with *RepFinder*. Templates are placed beside the image samples

(a) SimLocator 1 (b) RepFinder 1 (c) SimLocator 2 (d) RepFinder 2

Fig. 10 Our algorithm is robust when users choose two templates that have apparently different inner textures

6.3 Limitations

SimLocator provides an efficient computational framework to locate similar elements in an image. Our algorithm is robust to intra-class appearance variations, illumination changes, and complex background, as well as certain levels of deformations and occlusions. However, like all other locating approaches, our algorithm fails in some cases, especially in images with 3D transformations, severe occlusion, or dense clusters, which may lead to some errors, as shown in the two examples of failed applications in Fig. 11. Moreover, the object contours may not be precise because of the influence of background gradients. In these cases, we can integrate more feature descriptors in template matching or specify more strokes to classify the objects from the background manually.

7 Conclusion and future work

We have developed a novel framework called *SimLocator* for locating and extracting similar elements in an image scene with very little user interaction. Unlike previous methods that only considered single features in matching, we incorporate shape and color information in the computation of our dense correspondences across objects, thus making our method more robust than other state-of-the-art repeat/similar pattern analysis systems.

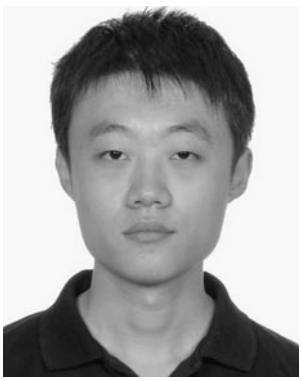
Fig. 11 Examples of failed runs. The school of fish in the left image has a dense cluster and severe occlusion. One cup in the right image suffers from 3D transformation and has a semantically different appearance from the template and the other objects

In the future, we plan to test more feature descriptors in the matching process, for example using Gabor texture descriptors to help to locate objects with similar inner textures but different color patterns. Developing new applications that can benefit from the information of images with similar content is another direction. Moreover, automatic similar object detection and extraction remains a very challenging research direction that merit further investigation.

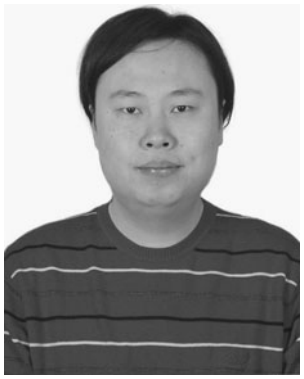
Acknowledgements We thank anonymous reviewers for their valuable input. We thank Fuzhang Wu for making some Dual-Bound results. We thank the flickr members who kindly share their images under Creative Commons License: Bestfriend (tea cups), RenateEurope (purple flowers), acaffery (balloons), Luko Gecko (fish school), Bold Huang (petunia), Swami Stream (chocolate cake). We thank the users of 500px.com who have shared their images through public domain: Ricky Marek (pomegranates), Nitin Prabhudesai (dandelions), Joram Huyben (green lanterns). We also thank the users of pinterest.com who have put the media in their spaces: Gus's Mom (panda cakes), Rhonda E. Peterson (meatballs), Cindy Loo (fried balls). The two fruit cake images are borrowed from [10]. This work is supported by National Natural Science Foundation of China under project Nos. 61172104, 61271430, 61201402 and 61202324, by Beijing Natural Science Foundation (Content Aware Image Synthesis and its Applications, No. 4112061), and by SRF for ROCS, SEM.

References

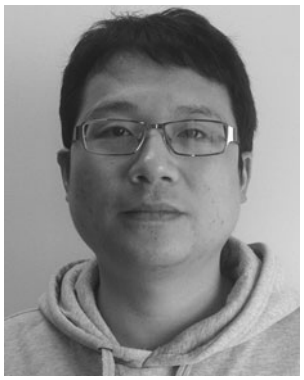
1. Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Susstrunk, S.: SLIC superpixels compared to state-of-the-art superpixel methods. IEEE Trans. Pattern Anal. Mach. Intell. **34**(11), 2274–2282 (2012)
2. Ahuja, N., Todorovic, S.: Extracting texels in 2.1d natural textures. In: IEEE International Conference on Computer Vision, pp. 1–8. IEEE Computer Society, Los Alamitos (2007)
3. Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-up robust features (surf). Comput. Vis. Image Underst. **110**(3), 346–359 (2008)
4. Belongie, S., Malik, J., Puzicha, J.: Shape matching and object recognition using shape contexts. IEEE Trans. Pattern Anal. Mach. Intell. **24**(4), 509–522 (2002)
5. Berg, A.C., Berg, T.L., Malik, J.: Shape matching and object recognition using low distortion correspondences. In: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 1, CVPR'05, pp. 26–33. IEEE Computer Society, Washington (2005)
6. Chen, Q., Li, D., Tang, C.K.: KNN matting. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 869–876 (2012)
7. Cheng, M.M., Zhang, F.L., Mitra, N.J., Huang, X., Hu, S.M.: RepFinder: finding approximately repeated scene elements for image editing. ACM Trans. Graph. **29**(4), 83:1–83:8 (2010)
8. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005, vol. 1, pp. 886–893. IEEE, New York (2005)
9. Forssén, P.E.: Maximally stable colour regions for recognition and matching. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–8 (2007)
10. Huang, H., Zhang, L., Zhang, H.C.: RepSnapping: efficient image cutout for repeated scene elements. Comput. Graph. Forum **30**(7), 2059–2066 (2011)
11. Joulin, A., Bach, F., Ponce, J.: Discriminative clustering for image co-segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR2010), pp. 1943–1950 (2010)
12. Kim, E., Li, H., Huang, X.: A hierarchical image clustering cosegmentation framework. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 686–693 (2012)
13. Krages, B.: Photography: The Art of Composition. Allworth Press, New York (2005)
14. Leung, T.K., Malik, J.: Detecting, localizing and grouping repeated scene elements from an image. In: Proceedings of the 4th European Conference on Computer. ECCV'96, vol. I, pp. 546–555. Springer, London (1996)
15. Levin, A., Lischinski, D., Weiss, Y.: A closed-form solution to natural image matting. IEEE Trans. Pattern Anal. Mach. Intell. **30**(2), 228–242 (2008)
16. Li, Y., Sun, J., Tang, C.K., Shum, H.Y.: Lazy snapping. ACM Trans. Graph. **23**(3), 303–308 (2004)
17. Liu, J., Sun, J., Shum, H.Y.: Paint selection. ACM Trans. Graph. **28**(3), 69:1–69:7 (2009)
18. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vis. **60**, 91–110 (2004)
19. Meng, F., Li, H., Liu, G., Ngan, K.N.: Object co-segmentation based on shortest path algorithm and saliency model. IEEE Trans. Multimed. **14**(5), 1429–1441 (2012)
20. Pauly, M., Mitra, N.J., Wallner, J., Pottmann, H., Guibas, L.J.: Discovering structural regularity in 3d geometry. ACM Trans. Graph. **27**(3), 43:1–43:11 (2008)
21. Rother, C., Kolmogorov, V., Blake, A.: "grabcut": interactive foreground extraction using iterated graph cuts. ACM Trans. Graph. **23**, 309–314 (2004)
22. Schweitzer, H., Deng, R., Anderson, R.F.: A dual-bound algorithm for very fast and exact template matching. IEEE Trans. Pattern Anal. Mach. Intell. **33**(3), 459–470 (2011)
23. Thompson, D.W.: On Growth and Form. Cambridge University Press, Cambridge (1992)



Yan Kong is a PhD candidate in the Sino-French Laboratory (LIAMA) and National Laboratory of Pattern Recognition (NLPR) at Institute of Automation, Chinese Academy of Sciences. He received his BSc in Computer Science in 2011 from Beijing Jiaotong University, PR China. His research interests include image synthesis and image processing.



Weiming Dong is an Associate Professor in the Sino-French Laboratory (LIAMA) and National Laboratory of Pattern Recognition (NLPR) at Institute of Automation, Chinese Academy of Sciences. He received his BSc and MSc degrees in Computer Science in 2001 and 2004, both from Tsinghua University, PR China. He received his PhD in Computer Science from the University of Henri Poincaré Nancy 1, France, in 2007. His research interests include image synthesis and realistic rendering. Weiming Dong is a member of ACM and IEEE.



Xing Mei is an Assistant Professor in the Sino-French Laboratory (LIAMA) and National Laboratory of Pattern Recognition (NLPR) at Institute of Automation, Chinese Academy of Sciences (CASIA). He received his BSc degree in Electronic Engineering in 2003 from the University of Science and Technology of China (USTC). He received his PhD degree in 2009 from CASIA. His research interests include image processing, computer vision and computer graphics. Xing Mei is a member of IEEE.

Xiaopeng Zhang received his MSc degree in Mathematics from Northwest University in 1987, and the PhD degree in Computer Science from Institute of Software, Chinese Academy of Sciences (CAS), in 1999. He is a Professor in the Sino-French Laboratory (LIAMA) and (National Laboratory of Pattern Recognition) at Institute of Automation, CAS. His main research interests are computer graphics and pattern recognition. He received the National Scientific and Technological Progress Prize (Second Class) in 2004. Xiaopeng Zhang is a member of ACM and IEEE.

Jean-Claude Paul is a director of research at INRIA and a professor at Tsinghua University, Beijing, China. He received his PhD degree in Mathematics from University of Paris XI and graduated in architecture design from the French National School of Fine Arts (ENSBA) in 1976. In 1995, he obtained the Academie des Sciences Prize and the Academie des Beaux Arts Prize, for both his artistic and scientific work. His research interests include realistic rendering, geometry processing and curves and surfaces theory.