2016 23rd International Conference on Pattern Recognition (ICPR)
Cancún Center, Cancún, México, December 4-8, 2016

# Interactive Region Segmentation for Manga

Kota Ito, Yusuke Matsui, Toshihiko Yamasaki, and Kiyoharu Aizawa
The University of Tokyo

***Abstract*—Manga (Japanese comics) are popular all over the world, and are created digitally. In this paper, we propose an interactive segmentation method tailored for manga. The proposed method enables annotators to select areas in manga efficiently. Our experimental results showed that the proposed framework works better than Adobe Photoshop CC, which is the most widely used commercial image editing software.**

## I. Introduction

Manga (Japanese comics) are an important aspect of the Japanese popular culture, and they are read widely in many countries. Recently, manga are issued not only in print but also electronically. Several researches on manga image processing have appeared, e.g., reference-based colorization [1], production of stylistic manga layouts [2], and so on. In research in the field of computer vision, public datasets with annotations have played an important role. Some large-scale datasets, including ImageNet [3], have been widely used by researchers. Recently, datasets with detailed annotation, such as PASCAL-Context dataset [4] and MSCOCO [5], were made available. These datasets allow many methods to be compared to each other. In addition, several technologies using machine learning have been developed. For manga image processing, a large-scale dataset called Manga109 [6], which consists of 109 comic books, is used. Although this dataset includes a large number of manga pages, it does not include detailed annotations. Annotating meta data would make the Manga109 dataset more useful. However, manual annotation is a laborious and time-consuming task. A way to alleviate the burden of such a task is to use an interactive assistant tool for region selection.

Many methods have been proposed for interactive segmentation, and these methods can be broadly classified into two: region-based segmentation and contour-based segmentation. Region-based segmentation handles the segmentation problem by adopting a pixel-wise approach as in [7]. This method requires users to scribble strokes on an image. A label (e.g., foreground or background) is then assigned to each pixel on the basis of these scribbles and color (textures) of the image. In contour-based segmentation, the contour of target regions is obtained. Representative examples are Snake [8] and Live wire [9]. Currently, contour-based methods are mainly studied in the field of medical image segmentation. Because region-based methods make use of colors and textures, these methods are not effective for manga segmentation where manga usually do not contain colors. Therefore, we focus on the contour-based approach. Usually, the target of contour-based methods is naturalistic images, which have ambiguous contours in general. Unlike naturalistic images, manga images have distinctive contours, and almost all meaningful regions are enclosed by them. Therefore, the purpose of our method is to extract contour lines of a target region.

In this paper, we propose a contour-based interactive segmentation method that focuses on line drawings. To extract an objective subset of contour lines by user interaction, we express a line drawing in a graph structure. Contour lines are represented as edges of the graph, where small gaps are automatically filled by edges. We extract both a subgraph and a region enclosed by the subgraph. The targets of our method are grayscale raster images which have some textures. We conducted a user study to compare our method to Adobe Photoshop CC [10]. Experimental results show that our method is superior to some of the region segmentation tools of Photoshop.

There are some labeling methods for hand-drawing sketches [11], [12]. However, those methods assume that target strokes are clearly discreminated from background and assign labels to strokes' pixels only.

## II. Method

Our proposed method consists of three steps, (1) line drawings extraction from manga images, (2) construction of graph from line drawings, and (3) extracting a subgraph by user interaction. First, we explain the method of graph construction from line drawings in Section II-A. Next, we show a method of extracting a subgraph in Section II-B. Because manga consists of line drawings and screentones (printed textures), directly extracting a region from manga is not appropriate. Therefore, finally in Section II-C, we explain how to apply our method to a manga image that contains screentones. Fig. 1 shows the workflow of the proposed method.

### A. Graph construction

Given an input line drawing (Fig. 1a), we first extract a graph structure (Fig. 1b) from it. To construct a graph structure, we sample black pixels from a line drawing. These sampled pixels are used as nodes for a graph. Generally, a line of line drawings has some thickness. In order to preserve visual structure, nodes must be sampled from the center line of the line. We show an algorithm for extracting nodes from line drawings in Alg. 1. $I$ is a binarized line drawing, where values of background are $0$ and those of lines are $1$. $DT()$ is distance transform (DT) operation. The resultant $I_{DT}$ is an image whose size is the same as the size of $I$, and the value of each pixel is the distance between the pixel and the closest white pixel in the original image $I$. Let us focus a line in the original image, and consider corresponding pixels in a slice (which is perpendicular to the line direction) of

978-1-5090-4846-5/16/$31.00 ©2016 IEEE

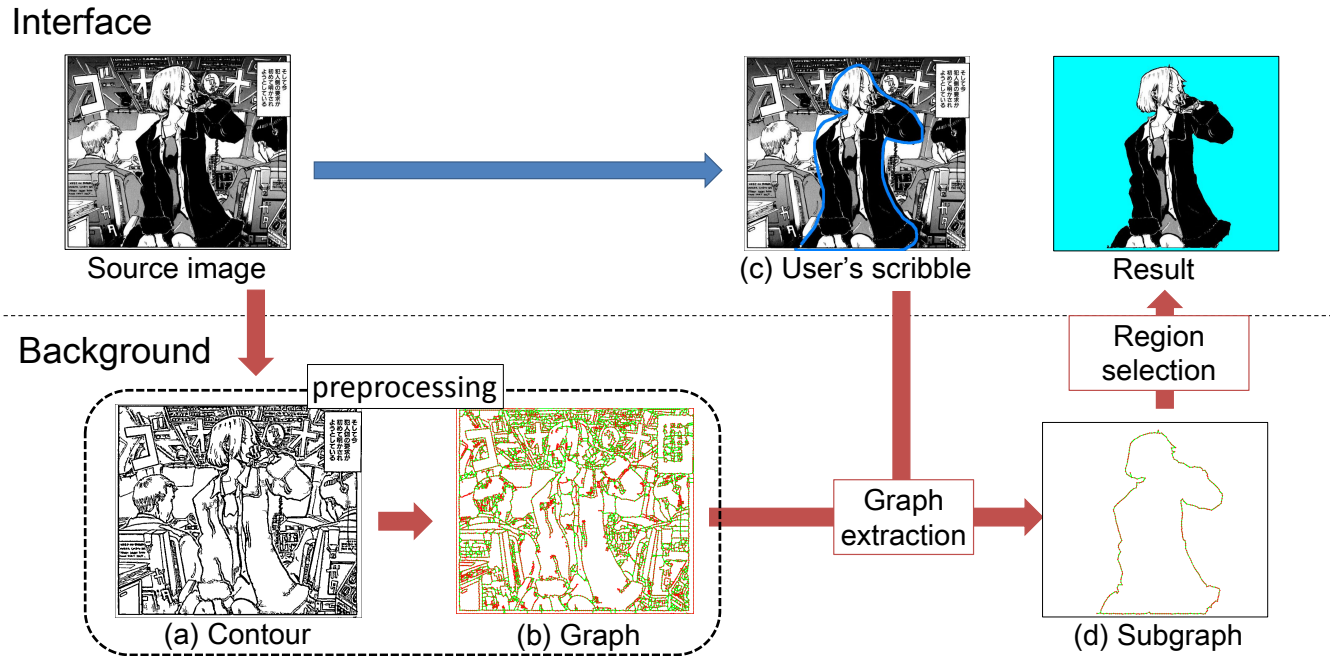



Fig. 1. Workflow: (a) Extracted line drawings based on [13]. (b) Graph expression of manga. Red dots are nodes and green lines are edges. (c) An example of user's scribble. (d) Extracted subgraph based on user's scribble. © Takashi Ki

**Algorithm 1** Node sampling.

```
1:  Input: I                          // Input binary line drawing
2:  Output: N                         // Sampled pixels (graph nodes)
3:  I_DT ← DT(I)                      // Distance transformed image
4:  V ← φ                             // Local maximum pixels
5:  for p do
6:    if I_DT(p) is the local maximum then
7:      V ← V ∪ p
8:    end if
9:  end for
10: N ← φ
11: while p_top ← pop(V) do
12:   N ← N ∪ p_top
13:   V ← V \ {p | p ∈ V, d(p, p_top) < 2I_DT(p_top)}
14: end while
```

the line in $I_{DT}$. The value of the center line of contour line should take the local maximum; hence, we sample such pixels. In our implementation, this is achieved by simply picking a pixel whose value is higher than those of surrounding pixels. This sampling is denoted in L5-L9, where $p$ iterates all pixels in $I_{DT}$. $V$ is a set of the local maximum pixels. Finally, we further subsample nodes from $V$ with a constant interval (L11-L14), where $\mathrm{pop}(V)$ picks up a pixel from $V$.

After node sampling, we span edges among these nodes. First, edges are spanned among contiguous nodes. Here, "contiguous" means that nodes are connected directly in the original contour lines. To judge this connection, we cluster the pixels belonging to contour lines to the nodes by watershed clustering algorithm. If some pixels (which belong to node A) are adjacent to some pixels (which belong to node B), then we define that nodes A and B are contiguous. In such cases, we span an edge of weight 1.0 between these nodes. Hereafter, we call the graph constructed by this process a simple adjacent graph.

Although a meaningful region tends to be enclosed by contour lines, it is not necessarily closed. There might be some small gaps on the contour lines, and humans can recognize the contour with small gaps as one closed contour. Because we would like to represent a contour line as a graph, such small gaps should be filled. To fill such gaps, we span additional edges from end nodes contiguous with only one node. We define three conditions represented by Eq. (1) - Eq. (3). $n_0$ is an end node and $n_1$ is a node contiguous with $n_0$. $d_1()$ is L1 norm and $p_n$ is the position of node $n$. $d_g()$ is a distance on a current graph, i.e., the sum of the weights of the shortest path between two input nodes. Eq. (1) means that $n$ is near $n_0$ in terms of Euclidean distance. Eq. (2) means that $n$ is located on the extension of the end point $n_0$. $V_{nm}$ is a vector extending from node $n$ to node $m$. Eq. (3) means that $n$ is far from $n_0$ on the current graph. If some node $n$ meets these conditions, an edge of weight $\frac{d_1(n_0,n)}{DT(n_0)}$ is spanned among $n$ and $n_0$. Fig. 1(b) shows an example of a constructed graph. Red dots represent nodes and green lines are edges.

$$d_1(p_n, p_{n_0}) < thresh_d \tag{1}$$

$$d_1(p_{n_0}, p_n)\sin\theta_{V_{n_1 n_0} V_{n_0 n}} < thresh_\theta \tag{2}$$

$$d_g(n_0, n) > \frac{d_1(p_{n_0}, p_n)}{DT(p_{n_0})} \tag{3}$$

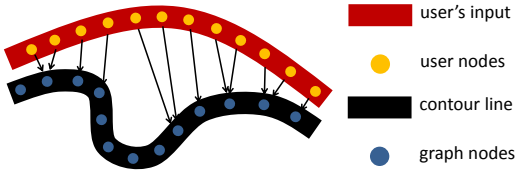



Fig. 2. Conceptual image of subgraph selection.

### B. Interactive segmentation

After the graph (Fig. 1b) is constructed automatically from the input image, a user interactively select a region. Here, the user draws a scribble that approximately encloses the region (Fig. 1c). Based on the contour given by the user, a subgraph (Fig. 1d) in the graph is then extracted. A conceptual image of our extraction approach is shown in Fig. 2. Note that the contour given by the user is also represented as a graph. The problem is then defined as a node assignment problem. First, the user's input is divided into some nodes. Then, each user node is assigned to a graph node. Finally, paths between these graph nodes are calculated, where the order of nodes are decided by the order of user nodes. The nodes and edges of the computed path are extracted as a final subgraph.

We formulate this assignment problem as an energy minimization problem. We minimize the energy represented by Eq. (4).

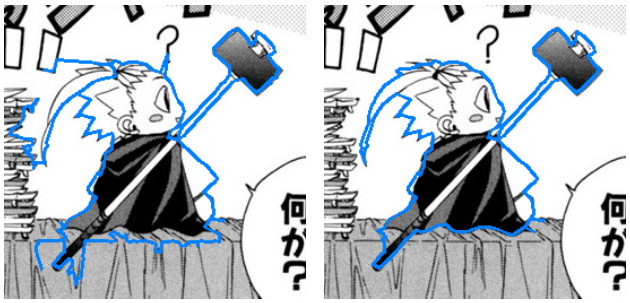
$$E(X) = \sum_i \left( E_u(i) + \lambda E_h(i) \right), \tag{4}$$

$$E_u(i) = d_1(p_{x_i}, p_i), \tag{5}$$

$$E_h(i) = \sum_{j=1}^{n_{adj}} d_g(x_i, x_{i+j}), \tag{6}$$

where $X$ is a set of $x_i$, and $x_i$ is a graph node to which $i$th user node is assigned. $p_{x_i}$ is the position of $x_i$th graph node, and $p_i$ is the position of $i$th user node. $E_u(i)$ is a unitary term, and tries to move a user node closer to its assigned graph node. $E_h(i)$ is a smoothing term, and tries to keep the selected graph nodes closer on the graph. $n_{adj}$ is a parameter that controls how many adjacent nodes are considered. The effect of this parameter is shown in Fig. 3. Fig. 3(a) shows the result when $n_{adj}$ is set as one and Fig. 3(b) shows the result when $n_{adj}$ is set as six. The same user scribble was given in both the cases. While discontinuous contour lines are selected when $n_{adj}$ is one, reasonably continuous contour lines are selected when $n_{adj}$ is six. In this work, $\lambda$ is set as one and $n_{adj}$ is set as six. The energy can be minimized by using the alpha expansion graphcut algorithm [14]. For the initial input, the user's scribble is regarded as a cycle, meaning that a starting point and an end point are connected.

After Eq. (4) is minimized, an initial subgraph is extracted. Sometimes, this initial subgraph is not appropriate; some parts of extracted contours might be wrong. We show how to modify such wrong parts. Users can modify wrong parts by giving additional scribbles, which select a right contour partially. A new subgraph is extracted based on the user's additional input and wrong parts of the initial subgraph are replaced with the new subgraph. It works on the same manner as that of initial scribbles. In the case of modification, the unitary term $E_u(i)$ in Eq. (4) is modified as Eq. (7).



(a) $n_{adj} = 1$ (b) $n_{adj} = 6$

Fig. 3. The effect of parameter $n_{adj}$. © Yuki Kobayashi

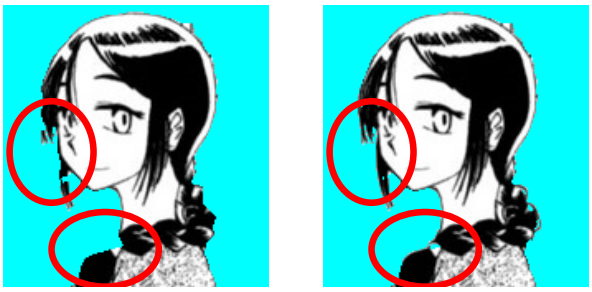
$$\hat{E}_u(i) = \begin{cases} E_u(i) + d_g(n_{nearest}, x_i) & (i \in S) \\ E_u(i) & (otherwise), \end{cases} \tag{7}$$

where $n_{nearest}$ is a node in currently extracted subgraph, and is the nearest node of endpoint of user's scribble. $S$ is the set of user nodes that are close to the endpoints of the user's scribble. In our implementation, user nodes that are connected by less than three edges are selected as $S$. An additional term $d_g(n_{nearest}, x_i)$ tries to assign user nodes to graph nodes that are close to current extracted nodes. This modified energy is minimized and a part of current subgraph is replaced by the result.

Once the subgraph is extracted, the target region should be selected. Normally, an image is divided by a polygon created by connecting the nodes of the subgraph. When a contour map is constructed, thin black filled areas remain as a line. Because graph nodes are center of lines, cutting by a polygon does not tend to be appropriate at regions where a curvature is high or a stroke thickness is bold as shown in Fig. 4(a). Therefore, we provide an additional cutting protocol for handling such thin areas. If users are not satisfied with the results for thin areas (Fig. 4(a)), they can additionally trace the areas by simply scribbling. Then, the additional cutting protocol is applied, and the areas are selected (Fig. 4(b)). In this cutting protocol, the pixels that are clustered traced subgraph nodes are also regarded as the segmented area.

### C. Application to manga

Manga images consist of line drawings and screentones (printed texture patterns) as shown in Fig. 5. Therefore, we should extract line drawings from manga images (Fig. 1a). There is a filter for separating screentones and line drawings of manga [13]. This method adjusts parameters of two different filters and merges their outputs appropriately. See [13] for more detail. We slightly improve this filter and add postprocessing.



(a) The result by polygon (b) The result with additional cutting protocol

Fig. 4. The effect of additional cutting protocol. © Satoshi Mizukami

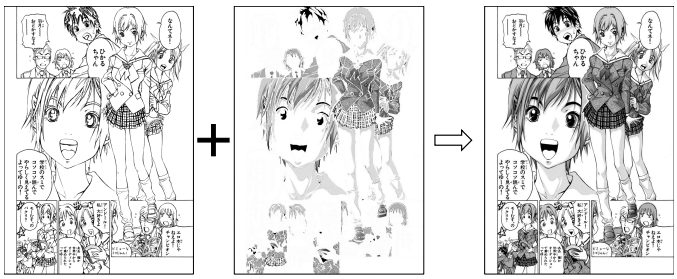



Fig. 5. Manga construction. Manga images consist of line drawings layer and screentones layer. © Ryusei Deguchi

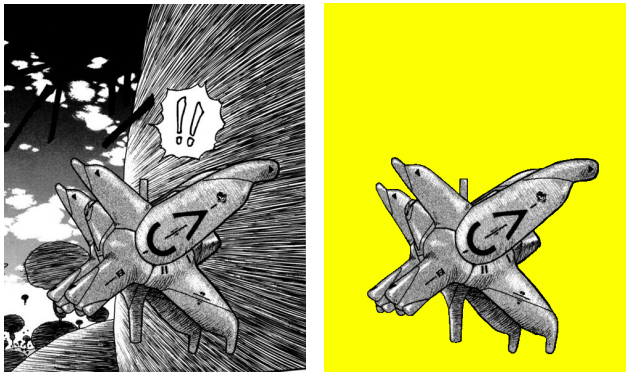
The previous filter sometimes regards a pixel which must be white as a black pixel. Therefore, we make a base mask by using Sauvola's binarization method [15]. This mask defines pixels which can be black and we merge the base mask result of the previous filter. Additionally, we improve the scale invariance of this filter. We estimate a line thickness of a manga image from the base mask and determine some parameters of filters according to this thickness. By doing so, invariance against the change of scale is enhanced. Finally, we restore a part of lines based on line connectivity.

## III. Experiment

We conducted a user study to compare our method to Adobe Photoshop CC [10], which is the most widely used commercial image editing software. We selected a magnetic lasso tool and a quick selection tool, both of which are functions of Photoshop, as comparative methods. Photoshop parameters were set as shown in Table I. Smart Radius, which is the range of automatic edge detection, was set as 0 px because smart edge detection did not work well in the case of line drawings. The experiment was conducted according to the following procedure. We selected eight images from the Manga109 dataset [6]. For each source image (Fig. 6(a)), we created ground truth results of segmentation in advance by the polygonal lasso tool of Photoshop as shown in Fig. 6(b). For each tool, the user study was conducted as follows. First, we asked the participants to practice the use of the tool in seven minutes. Next, for each image, both a source image and a ground truth image were displayed to the participants until they memorized the ground truth. Then, participants segmented an image in one minute. Images were presented in a random order. A source image and a ground truth image were displayed while the participants worked. We evaluated the accuracy of the results by a criterion represented by Eq. (8).

TABLE I
Photoshop parameters

| | |
|---|---|
| Smart Radius | 0 px |
| Smooth | 0 |
| Feather | 0 px |
| Contrast | 100% |
| Shift Edge | 0% |
| Anti Alias | disable |



(a) soure image (b) ground truth

Fig. 6. An example of a source image and a ground truth image. © Masaki Kato

$$overlap = \frac{|I_{method} \bigwedge I_{gt}|}{|I_{method} \bigvee I_{gt}|} \tag{8}$$

where $I_{method}$ is a segmented region by each tool and $I_{gt}$ is a ground truth segmentation. This is a standard criterion for evaluating the accuracy of object detection [16].

### A. Comparison of the proposed method with the quick selection tool

We recruited seven people as participants in the experiment to compare our method with the quick selection tool. Fig. 7 shows the results of the accuracy achieved by the two methods. Quantitatively, the proposed method was superior to the quick selection tool in almost all the cases. Examples of error maps are shown in Fig. 11, which represent where the participants tended to fail while carrying out segmentation. Brighter pixels mean that more participants failed to segment. The error map is constructed by stacking the seven error maps from the seven participants. With our proposed method, the outermost contours did not tend to be selected. This is because regions were selected by polygons that consist of nodes of the selected subgraph. The pixels located outside of the polygons cannot be segmented. On the other hand, strokes themselves did not tend to be selected in the case of the quick selection tool.

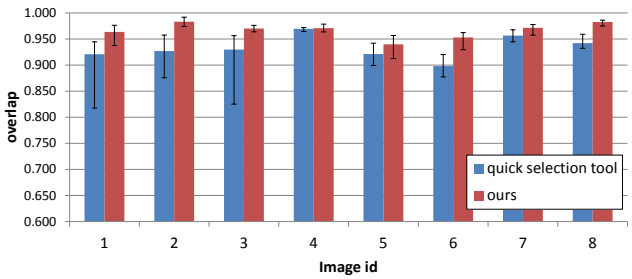



Fig. 7. The result of comparison with quick selection tool.

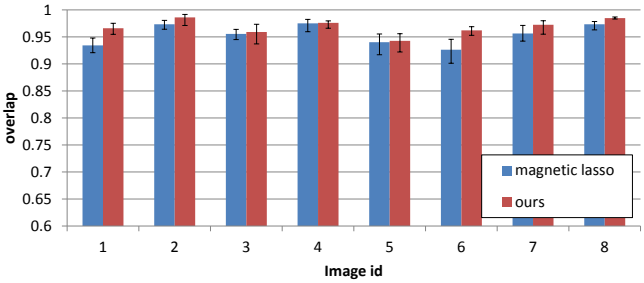



Fig. 9. The result of comparison with magnetic lasso tool.

(a) original image. (b) result of quick selection tool. (c) result of our method.

Fig. 8. The example of difference between our method and quick selection tool.

(a) original image. (b) result of magnetic lasso tool. (c) result of our method.

Fig. 10. The example of difference between our method and magnetic lasso tool.

This is because the quick selection tool is a region-based method. In the case of line drawings, contour lines have some thickness and they are regarded as regions. The example of this difference is shown in Fig. 8. As can be seen, the contour lines cannot be selected by the quick selection tool. Therefore, when using the quick selection tool, contour lines need to be selected carefully.

In addition, big blocks were selected or not selected by mistake occasionally. Pixels that were included in the same closed region or that had similar texture tended to be selected by once operation of the quick selection tool. Many areas of line drawings are simple closed regions without texture or with similar texture. Therefore, the result obtained using the quick selection tool is unstable; an unnecessarily large area might be selected unpredictably even if the user slightly fails to select the region.

### *B. Comparison of the proposed method with the magnetic lasso tool*

We recruited six people as participants in the experiment carried out to compare the proposed method with the magnetic lasso tool. Quantitative result is shown in Fig. 9. The result shows that a contour-based method is better suited to region selection of manga images. Though the proposed method seems slightly better than the magnetic lasso tool in terms of the value of accuracy, the difference can be clearly seen in images 1, 6, and 8. Our method worked well for images that have a relatively large number of lines or complicated curves. The error maps are shown in Fig. 12. The overall trend of our system is the same as when it was compared with the quick selection tool. The magnetic lasso tool did not tend to work well for contours surrounded by strong edges with high curvatures, as shown in Fig. 10. Actually, the proposed method also has the same trend in the first extracted subgraph. However, the correction step is applied to modify the wrong result, as described in (Section II-B).

These results show that our proposed segmentation method is better than the quick selection tool and the magnetic lasso tool of Adobe Photoshop CC. The limitation of the proposed method is the computational cost of preprocessing. Given an input image, we need to construct the graph in advance, and this step takes about ten seconds in our implementation. However, we argue that the cost is not a serious drawback because the runtime of interactive segmentation is usually longer than that of preprocessing.

## IV. Conclusion

We proposed a method of interactive segmentation for manga images. Our method represents manga images as a graph structure and extracts a subgraph by user interaction. We have verified that our proposed method is more effective than commercial image editing tools.

Becaus we express a line drawing in a graph structure, the segmentation result of our method is limited by this graph. Our method cannot select the region that is not enclosed by any subgraphs.

## References

[1] K. Sato, Y. Matsui, T. Yamasaki, and K. Aizawa, "Reference-based manga colorization by graph correspondence using quadratic programming," in *SIGGRAPH Asia 2014 Technical Briefs*, ser. SA '14. ACM, 2014, pp. 15:1–15:4.

[2] Y. Cao, R. Lau, and A. B. Chan, "Look over here: Attention-directing composition of manga elements," *ACM Transactions on Graphics (Proc. of SIGGRAPH 2014)*, vol. 33, 2014.

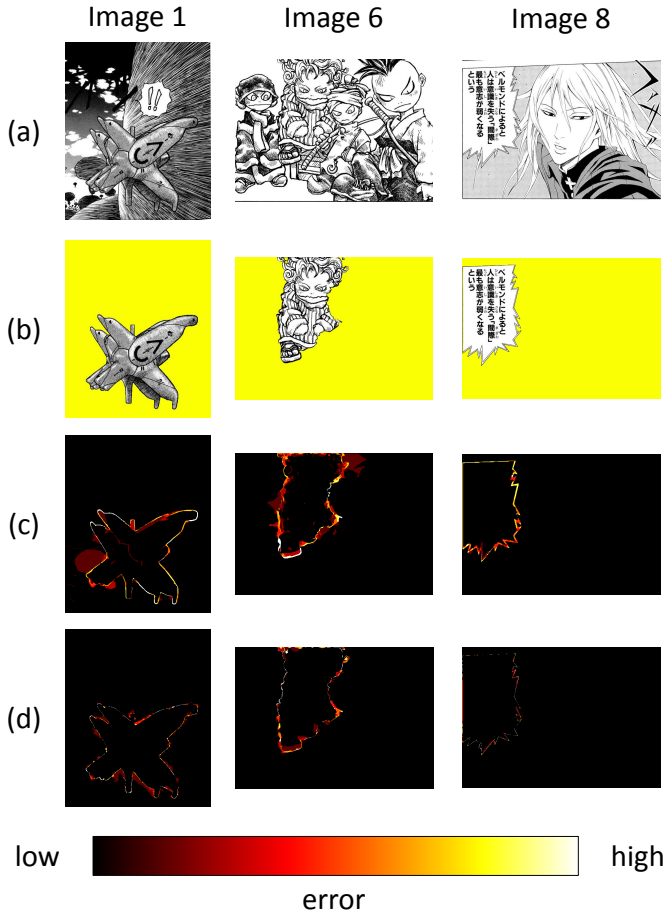



Fig. 11. Error map of comparison with quick selection tool: (a) original image, (b) ground truth, (c) error map of quick selection tool, and (d) error map of our method. © Masaki Kato, Shouei Ishioka, Shuji Takeya.

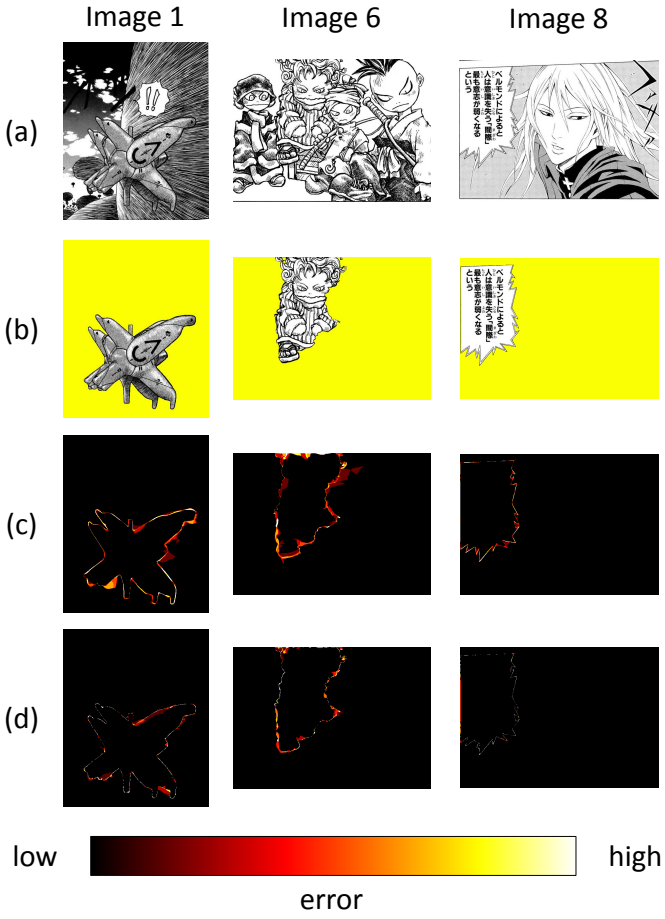



Fig. 12. Error map of comparison with magnetic lasso tool. (a) original image, (b) ground truth, (c) error map of magnetic lasso tool, and (d) error map of our method. © Masaki Kato, Shouei Ishioka, Shuji Takeya.


[3] J. Deng, W. Dong, R. Socher, L. J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, June 2009, pp. 248–255.

[4] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille, "The role of context for object detection and semantic segmentation in the wild," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[5] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, *Computer Vision – ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*. Cham: Springer International Publishing, 2014, ch. Microsoft COCO: Common Objects in Context, pp. 740–755. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-10602-1_48

[6] Y. Matsui, K. Ito, Y. Aramaki, T. Yamasaki, and K. Aizawa, "Sketch-based manga retrieval using manga109 dataset," *CoRR*, vol. abs/1510.04389, 2015. [Online]. Available: http://arxiv.org/abs/1510.04389

[7] C. Nieuwenhuis and D. Cremers, "Spatially varying color distributions for interactive multilabel segmentation." *PAMI*, pp. 1234–1247, 2013.

[8] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331.

[9] E. N. Mortensen and W. A. Barrett, "Intelligent scissors for image composition," in *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '95. ACM, 1995, pp. 191–198.

[10] "Adobe photoshop cc," https://www.adobe.com/jp/products/photoshop.htm.

[11] G. Noris, D. Skora, A. Shamir, S. Coros, B. Whited, M. Simmons, A. Hornung, M. Gross, and R. Sumner, "Smart scribbles for sketch segmentation," *Computer Graphics Forum*, vol. 31, no. 8, pp. 2516–2527, 2012.

[12] Z. Huang, H. Fu, and R. W. H. Lau, "Data-driven segmentation and labeling of freehand sketches," *ACM Trans. Graph.*, vol. 33, no. 6, pp. 175:1–175:10, Nov. 2014. [Online]. Available: http://doi.acm.org/10.1145/2661229.2661280

[13] K. Ito, Y. Matsui, T. Yamasaki, and K. Aizawa, "Separation og manga line dtawings and screentones." in *Eurographics short paper*. ACM, 2015.

[14] Y. Boykov and R. Zabih, "What energy functions can be minimized via graph cuts?." *PAMI*, vol. 26, no. 2, pp. 147–159, 2004.

[15] J. Sauvola, T. Seppanen, S. Haapakoski, and M. Pietikainen, "Adaptive document binarization," in *Document Analysis and Recognition, 1997., Proceedings of the Fourth International Conference on*, vol. 1, Aug 1997, pp. 147–152 vol.1.

[16] M. Everingham, S. Eslami, L. V. Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *IJCV*, pp. 1–39, 2014.