Multimed Tools Appl (2014) 69:171–197
DOI 10.1007/s11042-012-1241-7

Automatic comic page segmentation based on polygon detection

Luyuan Li · Yongtao Wang · Zhi Tang · Liangcai Gao

Published online: 27 September 2012

Abstract Comic page segmentation aims to automatically decompose scanned comic images into storyboards (frames), which is the key technique to produce digital comic documents that are suitable for reading on mobile devices. In this paper, we propose a novel method for comic page segmentation by finding the quadrilateral enclosing box of each storyboard. We first acquire the edge image of the input comic image, and then extract line segments with a heuristic line segment detection algorithm. We perform line clustering to further merge the overlapped line segments and remove the redundancy line segments. Finally, we perform another round of line clustering and post-processing to compose the obtained line segments into complete quadrilateral enclosing boxes of the storyboards. The proposed method is tested on 2,237 comic images from 12 different printed comic series, and the experimental results demonstrate that our method is effective for comic image segmentation and outperforms the existing methods.

Keywords Comic image segmentation · Polygon detection · Line clustering · Line segment detection · Integral image

1 Introduction

Comics are very popular among people from all over the world. Besides reading the printed version of comic books, more and more people start reading the scanned copies of the printed ones on their mobile devices such as cell phones or tablet PCs. However, due to the

L. Li · Y. Wang (✉) · Z. Tang · L. Gao
Institute of Computer Science & Technology, Peking University, Beijing, China
e-mail: wyt@pku.edu.cn

L. Li
e-mail: liluyuan@pku.edu.cn

L. Gao
e-mail: gaoliangcai@pku.edu.cn

Z. Tang (✉)
State Key Laboratory of Digital Publishing Technology, Beijing, China
e-mail: tangzhi@pku.edu.cn

screen size and other hardware limitations of these mobile devices, the content of a single page of comics is larger than a mobile device can hold. An intuitive solution to solve this problem is splitting the page of comics into many smaller units and viewing them sequentially on the mobile device according to the reading orders of them [19]. For example, Fig. 1 (a) illustrates a conventional comic page, which consists of several disjoint polygon regions, namely the storyboards (or frames, similarly hereinafter). These storyboards are much smaller than the original page, thus each of them can be better displayed on the small screen of the mobile devices. If we know the structural information of this page, that is, we know how to split the page into storyboards and identify the reading order of them, we can better view this page storyboard by storyboard on the small screen of the mobile devices as illustrated in Fig. 1(b).

Unfortunately, existing electronic comic documents are usually in the image format without structural information of each page. Thus, we have to additionally extract the structural information of each comic page and two subtasks will be involved: (1) splitting the page into storyboards (namely comic page segmentation), (2) identifying the reading orders of them. The topic of this paper is the first subtask, that is, to address the comic page segmentation problem.

Comic page segmentation can be intuitively classified to the field of document image segmentation. However, existing document image segmentation methods are usually specifically designed for text documents such as newspapers [18] and scientific articles [12]. These methods are not suitable for comic page segmentation as a common comic page mainly consist of line drawings while the basic elements in text documents are characters which are aligned horizontally or vertically.

To the best of our knowledge, Tanaka et al.'s work [15] is the first one to address comic page segmentation. They first detect the division lines between storyboards using density gradient, and then iteratively divide the comic image horizontally and vertically with these lines to obtain the final segmentation result. The method is

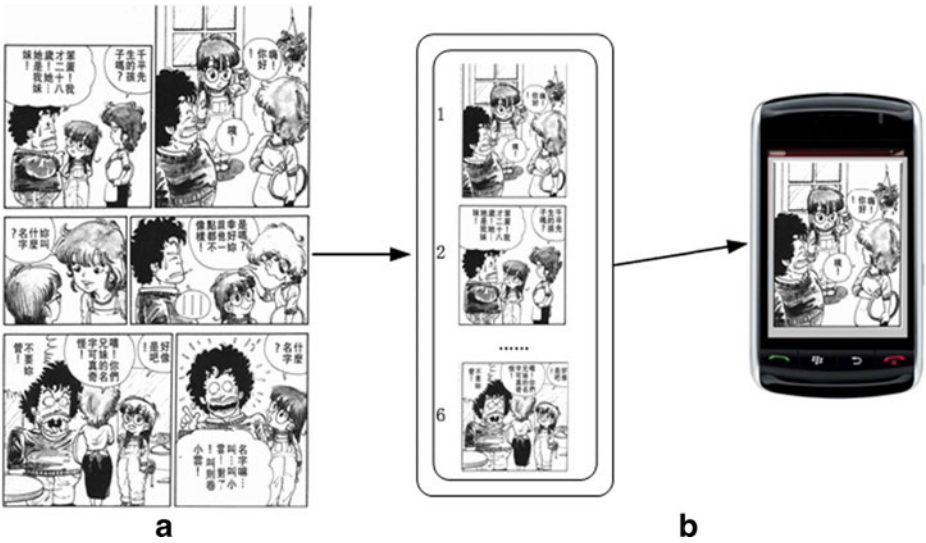


Fig. 1 **a** A conventional comic image example (source: title Dr. Slump, author Akira Toriyama, publisher Shueisha Inc., Volume 1 p.16); **b** One example of the application of frame segmentation into mobile devices

further refined by Yusuke [10], which weights the difference between the contour and other pixels during the division line detection process. Ishii et al. [11] extend Tanaka's work by utilizing the results of frame separation and frame corner detection. This method assumes that the orientation of the division line equals to that of the frame border line. Therefore, after obtaining the division lines, they find the frame corners and link those corners whose connecting line parallels with the corresponding division line. Arai [1] et al. proposes another method for comic content extraction. They define comic images in which no overlap balloons and comic art exists as flat comics and extract these flat comics using modified connected component labeling algorithm. For those overlapped frames, they are separated by division line detection process. The drawbacks of these methods lie in two major aspects. First, the frame within the comic image should be cut out without any blank margins, but the recursive line division process still can't handle the blank margin very well (see Fig. 2(a)). Second, the above method can't process image with more complex layout patterns especially when comic balloons or others objects are frequently drawn over scene frames (see Fig. 2(b)).

In this paper, we propose a novel page segmentation method for comic images, which can handle comic pages with more complex geometric layout patterns. The core idea of this method is to detect the approximate quadrilateral polygon enclosing the content of each storyboard, which is based on our observation that almost all the frames in the comic image can be circumscribed by quadrilateral polygons (see Fig. 3). To achieve this goal, we firstly extract the borderlines of the storyboards (i.e. line segment detection) and then compose the detected borderlines into quadrilateral polygons. This method can handle comic images with both complex geometric layout patterns and large blank margins. The experimental results demonstrate that our method outperforms existing algorithms.

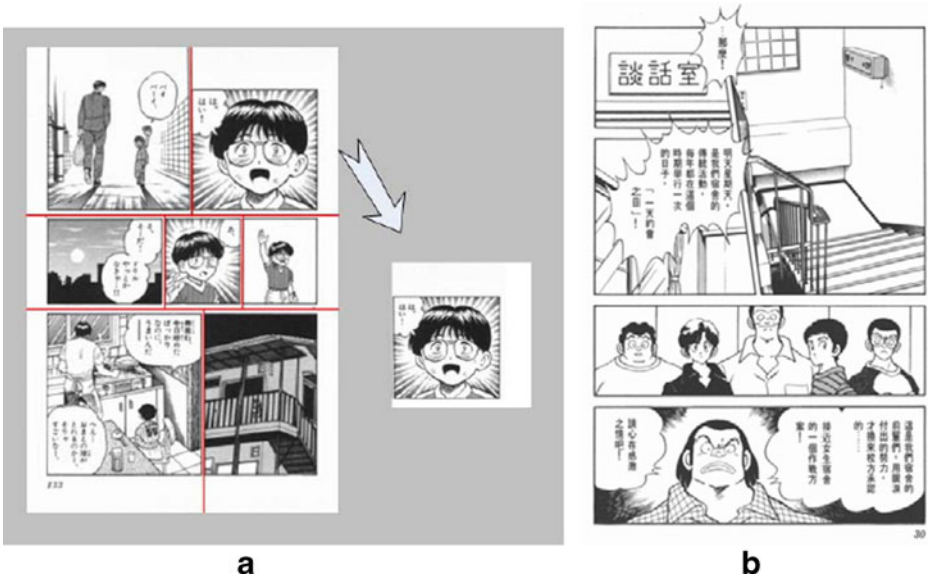


Fig. 2 **a** The division line process can't deal with the blank margin well. (source: title MAJOR, publisher Shogakukan Inc.,Volume 1 p.133); **b** A comic page with more complex layout pattern. (source: title ROUGH, author Mitsuru Adachi, publisher Shogakukan Inc., Volume 2 p.30)

Fig. 3 **a**, **b**, **c** are examples of storyboards cut out from different comic pages which are enclosed by arbitrary quadrilateral polygons. (source: title ROUGH, author Mitsuru Adachi, publisher Shogakukan Inc., Volume 3 **a** p.156; **b** p.155; **c** p. 170)

The rest of the paper is organized as follows. The work flow of our comic image segmentation method is introduced in section 2. The proposed specific line segment detection and polygon detection algorithms are described in section 3 and 4 respectively. The parameter discussion and experimental results on variety sets of comic strip documents are reported in section 5. In section 6, we make our conclusions and discuss future works related to this subject.

2 Work flow overview

Our method aims to find the quadrilateral enclosing box of each storyboard and thus achieves comic page segmentation, which consists of two major stages. The first stage is to find line segments in the edge image and the second stage is to compose these detected line segments into the complete enclosing boxes. The work flow diagram of our proposed method is shown in Fig. 4 and every step within the two stages is listed below.

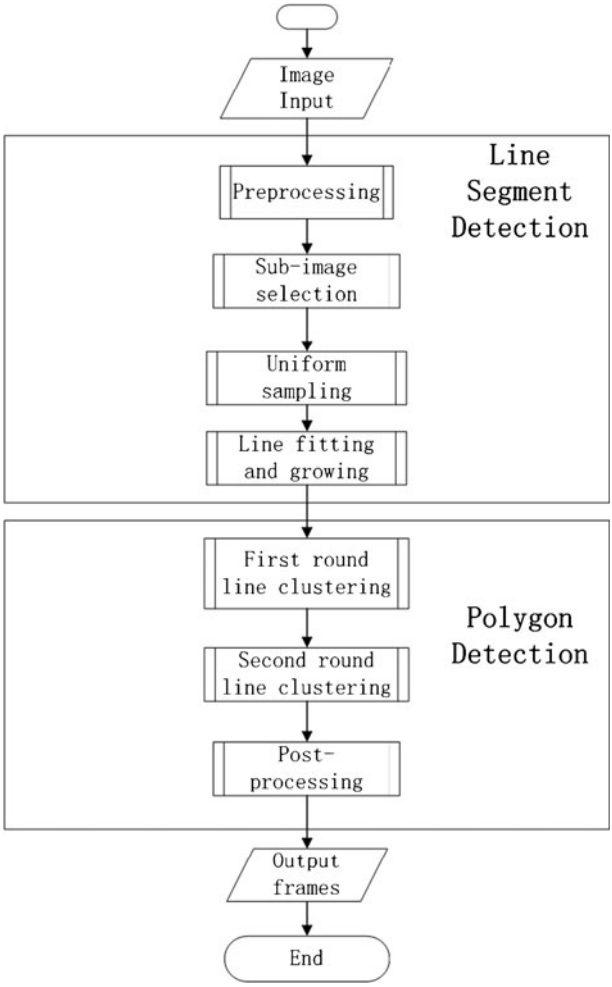


Fig. 4 The workflow chart of our proposed method

2.1 Line segment detection

The first stage of our method is to find straight line segments in the edge image. The process of finding line segments consists of the following four steps:

Step 1: **Preprocessing:** the first step of line segment detection is to acquire the edge image while calculating and storing information of the edge pixels in advance.

Step 2: **Sub-image selection:** for each pixel within the image, a sub-image of different aspect ratio is tested, and the sub-images that are more likely to contain a line segment are selected out.

Step 3: **Uniform sampling**: a large number of qualified sub-image candidates may be obtained from the previous step, so only samples of them will be further processed.

Step 4: **Line fitting and growing**: for each sampled sub-image, a short line segment is fitted, and then it grows into a longer one.

2.2 Polygon detection

The detected line segments are then clustered to form the enclosing box. In this stage, a two-rounds clustering method is used. The first round is to filter out redundant lines and the other is to compose the whole enclosing box. In the end, several post-processing methods are applied.

Step 1: The first round of line segments clustering aims to reduce the redundant lines as large numbers of sub-image are selected in the previous step. The lines that overlap along the parallel direction (as we define) should be merged together. Then the connect component search is used to cluster them together.

Step 2: The second round of line segments clustering aims to compose these detected lines into storyboard enclosing boxes to achieve the comic strip segmentation. We define several rules for deciding whether two lines belong to the same storyboard. Connect component search is also used in this step.

Step 3: As storyboard are often truncated by extruding objects such as characters and text balloons, the clustered line segments will not always form a complete enclosing box. The post-processing step aims to find the complete enclosing boxes from the fragmentary clustered lines.

3 Line segment detection

Line segment detection is one of the basic tasks of computer vision and many line detection algorithms have been proposed, in which Hough transform based ones [2, 5, 9, 13] and gradient based ones [3, 7] are commonly used. In this paper, we propose a heuristic algorithm to detect straight line segments in comic document image. This is not to assert that our line segment detection algorithm is superior to any other algorithms in any circumstances. Our line segment detection algorithm is just specially designed for our polygon detection based comic segmentation work flow.

In fact, we compare our algorithm with a state of the art line segment detection algorithm "LSD" [7] on some comic images. Figure 5 shows an example of the line segment detection results of LSD and our method, and one can see that the borderlines of the storyboards are clearly extracted out by both methods. The runtime of our line segment detection algorithm is about 1.2 s for a common comic image of 2,000×2,000 in size, which is slightly shorter than that of LSD (about 1.4 s).

3.1 Preprocessing

Given an image of a comic page, the first step is to obtain the edge image of it. Any edge detector which can provide clear and strong edge images is suitable for our proposed line segment detection algorithm. In our work, we use the Canny operator [4]. Figure 6(a) demonstrates that the Canny operator works well on comic images, where one can see that the boundaries of the storyboards and contours of some other objects are extracted out. The OpenCV library provides the implementation of Canny operator and we use 100 for the smaller threshold and 200 for the larger one, as it provides clear edge images.

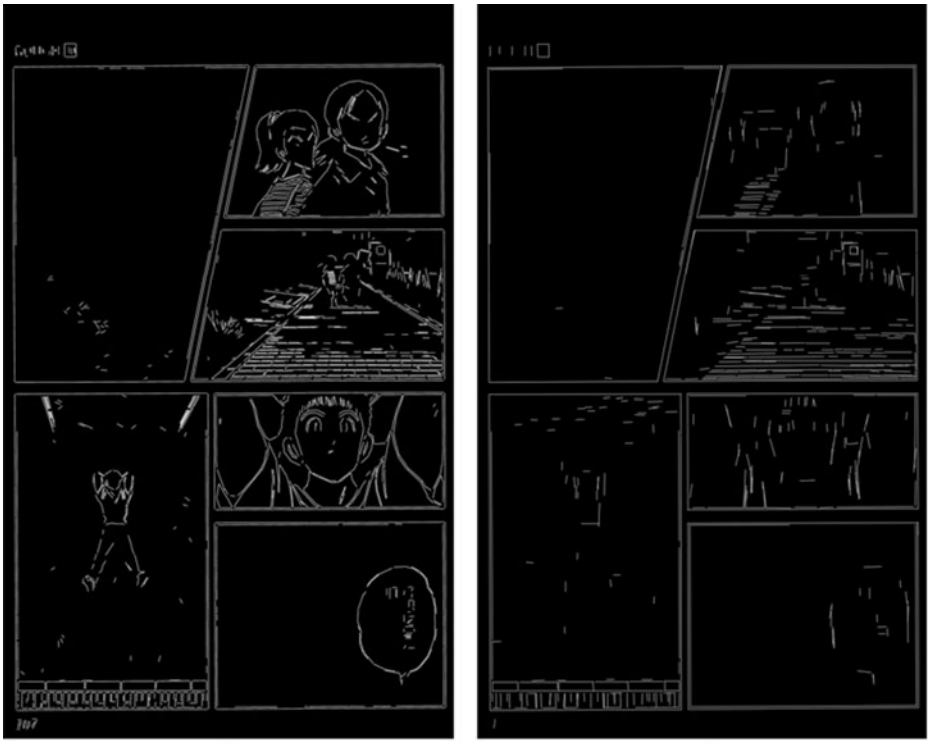


Fig. 5 **a** is the line segment result of LSD; **b** is the line segment result of our method and we can see the borderlines of the frames are clearly extracted out. (source of the input image: title ROUGH, author Mitsuru Adachi, publisher Shogakukan Inc., Volume 10 p.107)

3.2 Line candidate selection

Based on our observation on a large number of comic edge images, there always exist several small areas around each frame boarder, in which the edge points exclusively belong to the frame border lines (for example, please see Fig. 6(b)). Thus, instead of utilizing the overall edge information, we select candidate line segment from a small block of the edge image termed as sub-image. The statistical information of edge points in a sub-image is used to decide whether this sub-image contains line segment candidate. For a given sub-image I with n edge points $p_i = \{x_i, y_i\}^T$, $i=1, \ldots, n$, we calculate the covariance matrix M of these edge points:

$$M = \begin{bmatrix} \sigma_{x^2} & \sigma_{xy} \\ \sigma_{xy} & \sigma_{y^2} \end{bmatrix} = \frac{1}{n} \sum_{i=1}^{n} \begin{bmatrix} (x_i - \mu_x)^2 & (x_i - \mu_x)\left(y_i - \mu_y\right) \\ (x_i - \mu_x)\left(y_i - \mu_y\right) & \left(y_i - \mu_y\right)^2 \end{bmatrix}, \quad (1)$$

where

$$\mu_x = \frac{1}{n} \sum_{i=1}^{n} x_i, \mu_y = \frac{1}{n} \sum_{i=1}^{n} y_i.$$

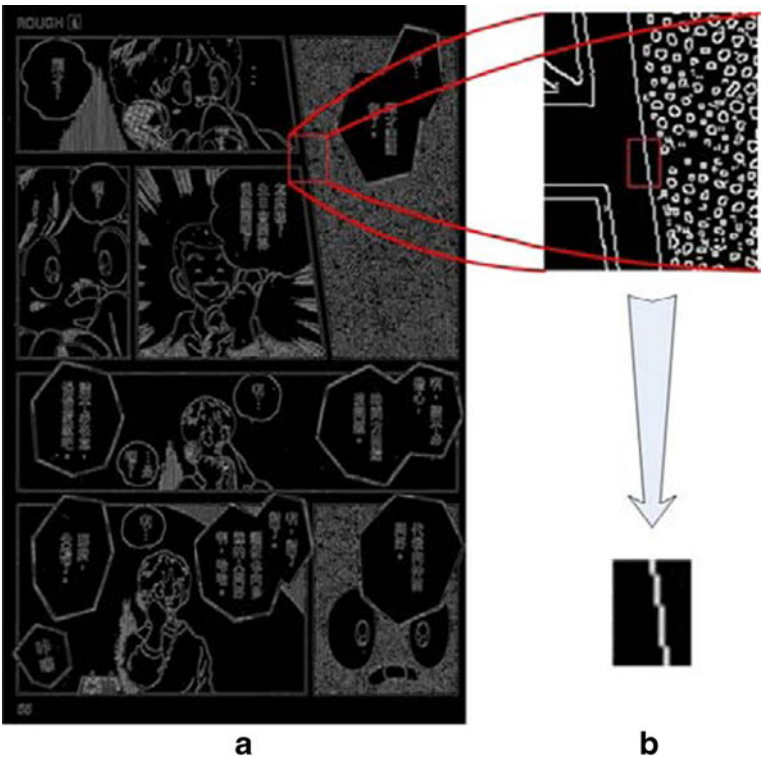


Fig. 6 **a** The result image after Canny operator operation; **b** The small area around the frame border in which the pixels belong exclusively to the frame lines. (source of the input image: title ROUGH, author Mitsuru Adachi, publisher Shogakukan Inc., Volume 6 p.65)

In theory, when we use an ellipse to approximate the edge points in sub-image I, the length of the major axis and the length of the minor axis are equals to $c\sqrt{\alpha}, c\sqrt{\beta}$, where α, β denote the larger and the smaller eigenvalue of the covariance matrix M respectively, and c is a constant [16]. Intuitively, the larger value of the ratio $c\sqrt{\alpha}$ to $c\sqrt{\beta}$ (i.e. $\sqrt{\alpha/\beta}$) implies the ellipse are more similar to a line segment. The most extreme case is that, for a horizontal or vertical line segment with one pixel thickness, β is equal to zero, so that $\sqrt{\alpha/\beta}$ approaches infinity. In this light, we use the ratio of α to β as a criterion to select the sub-images containing line segment candidates.

Using the method that Lowe [14] proposed, we can avoid the direct computation of α, β. It is achieved by calculating the sum of the two eigenvalues from the trace of M and their product from the determinant, which is:

$$Tr(M) = \sigma_{x^2} + \sigma_{y^2} = \alpha + \beta; \tag{2}$$

$$Det(M) = \sigma_{x^2}\sigma_{y^2} - \left(\sigma_{xy}\right)^2 = \alpha\beta. \tag{3}$$

The value r is supposed to be the ratio between the larger eigenvalue and the smaller one and thus $\alpha = r\beta$, $r \geq 1$. Therefore:

$$\frac{Tr(M)^2}{Det(M)} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r}. \tag{4}$$

The function $(r+1)^2/r$ is at the minimum value when $r=1$ and it increases with r when $r>1$. In contrast to the Lowe's method, which tried to eliminate poorly defined peak in the difference-of-Gaussian function, we try to filter out those sub-images which the ratio is too small to indicate the possibility of containing line segments. We only need to check:

$$\frac{Tr(M)^2}{Det(M)} > t, \tag{5}$$

where t is a ratio threshold. Moreover, it also occurs that the determinant is equal to zero which makes the inequality above meaningless. We can assign an infinitive positive value for it or change the formula into the following final format:

$$Tr(M)^2 > tDet(M). \tag{6}$$

We can easily compute the above inequality with a few floating point operations. In our experiment, the value chosen for t will be discussed in section 5, which can eliminate those sub-images from which a short line segment cannot be fitted.

3.3 Sub-image division

The previous section deals with the problem of sub-image selection, but the how to acquire the sub-images from the edge images and decide the size of the sub-image are still problems. In our experiments, we choose three different sizes (20×5, 5×20, 20×20) of sub-image which can better detect horizontal lines, vertically lines and lines of other directions respectively (as shown in Fig. 7).

As for the sub-image acquirement, an intuitive and direct approach is to partition the edge image into grid meshes of the same size (as shown in Fig. 8(a)). However, omissions occur when line segments from different storyboard are put into the same sub-image (as shown in Fig. 8(b)). Therefore a finer-grain acquirement is need. In our work, for each pixel of the comic image, taking this pixel as the left up corner, we acquire a $m \times n$ window as the sub-image unless the window is out of the image boundary.

As mentioned above, three sub-images of different size are generated for one pixel in the edge image from different iterations. Exhaustive as it may seem, it still have a problem: the high computation cost. Almost every pixel within an edge image of approximately 2,000× 2,000 in size will generate three sub-images and then be analyze and selected. We solve this problem by pre-computing edge pixel information and stored in the integral image.

As illustrated in Fig. 9, the integral image [17] is defined as:

$$IntMap(x,y) = \sum_{x'<x,y'<y} I(x',y'), \tag{7}$$

where $IntMap(x, y)$ is the integral image value at the point (x, y), and $I(x', y')$ is an involved two dimensional array to be integrated. One of the recurrent formulas for computing the integral image can be:

$$IntMap(x,y) = IntMap(x-1,y) + IntMap(x,y-1) - IntMap(x-1,y-1) + I(x,y). \tag{8}$$

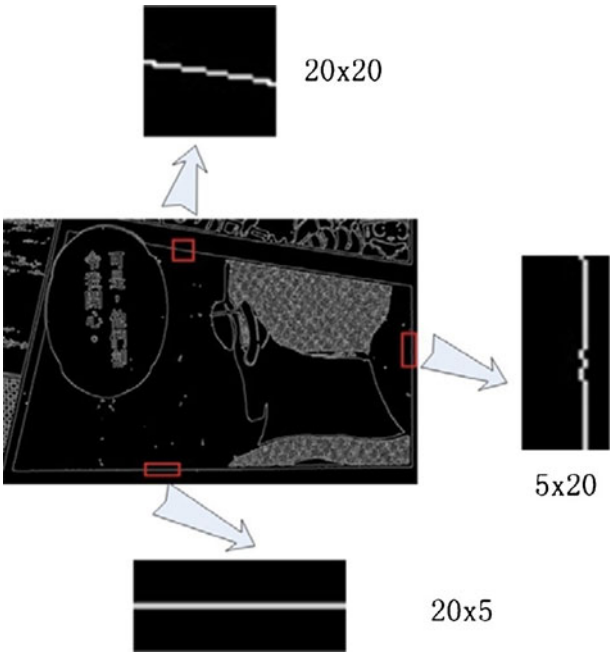


Fig. 7 An example of three different sizes of sub image (20×20, 5×20, and 20×5 which are cut from the source image) are selected for different orientation of the lines on the frame borders. (source of the input image: title ROUGH, author Mitsuru Adachi, publisher Shogakukan Inc., Volume 3 p.147)

Any rectangular sum can be computed by referring to four integral image values as shown in Fig. 10. In our case, the dense calculation of the covariant matrix M for each sub-image can be accelerated by reformulating the formula (1) as:

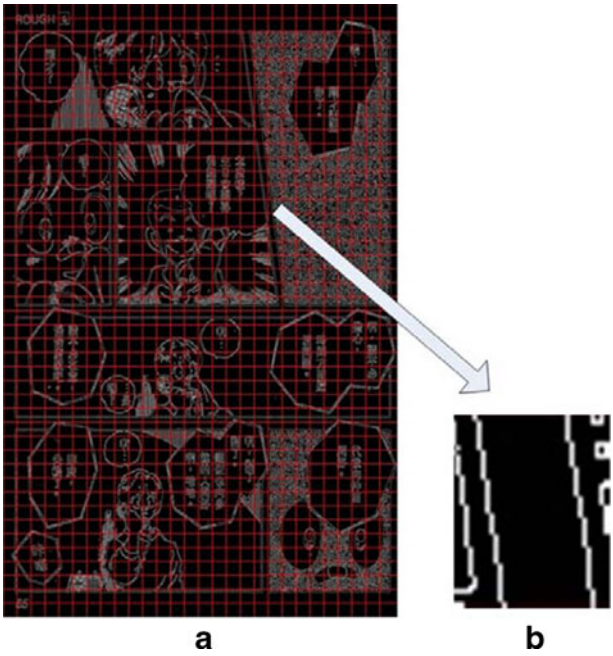


Fig. 8 **a** Result of the direct partition of a comic edge image into grid meshes; **b** Such partition will inevitably put lines from different frames together. (The source image is the same as that of Fig. 6(a))

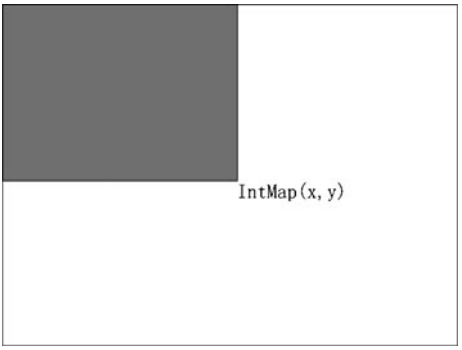


Fig. 9 The value of the integral image at point (x, y) is the sum of all the pixels above and to the left

$$\begin{aligned}
\sigma_{x^2} &= \frac{1}{n}\sum_{i=1}^{n}(x_i-\mu_x)^2 = \frac{1}{n}\left(\sum_{i=1}^{n}x_i^2 - 2n\mu_x + n\mu_x^2\right) = \frac{1}{n}\sum_{i=1}^{n}x_i^2 - \frac{1}{n^2}\left(\sum_{i=1}^{n}x_i\right)^2,\\
\sigma_{y^2} &= \frac{1}{n}\sum_{i=1}^{n}y_i^2 - \frac{1}{n^2}\left(\sum_{i=1}^{n}y_i\right)^2,\\
\sigma_{xy} &= \frac{1}{n}\sum_{i=1}^{n}x_iy_i - \frac{1}{n^2}\sum_{i=1}^{n}x_i\sum_{i=1}^{n}y_i.
\end{aligned} \tag{9}$$

For an edge image $I(x, y)$, we calculate the integral images corresponding to six dummy images to reduce the calculation of the terms $\sum_{i=1}^{n}x_i, \sum_{i=1}^{n}y_i, \sum_{i=1}^{n}x_iy_i, \sum_{i=1}^{n}x_i^2, \sum_{i=1}^{n}y_i^2$ and n respectively. The six dummy images $I_i\,(x, y)$, i=1, … 6 are:

$$\begin{aligned}
I_1(x,y) &= \begin{cases} x & if \quad I(x,y)=255 \\ 0 & if \quad I(x,y)=0 \end{cases}, \quad I_2(x,y) = \begin{cases} y & if \quad I(x,y)=255 \\ 0 & if \quad I(x,y)=0 \end{cases},\\
I_3(x,y) &= \begin{cases} xy & if \quad I(x,y)=255 \\ 0 & if \quad I(x,y)=0 \end{cases}, \quad I_4(x,y) = \begin{cases} x^2 & if \quad I(x,y)=255 \\ 0 & if \quad I(x,y)=0 \end{cases},\\
I_5(x,y) &= \begin{cases} y^2 & if \quad I(x,y)=255 \\ 0 & if \quad I(x,y)=0 \end{cases}, \quad I_6(x,y) = \begin{cases} 1 & I(x,y)=255 \\ 0 & I(x,y)=0 \end{cases}.
\end{aligned} \tag{10}$$

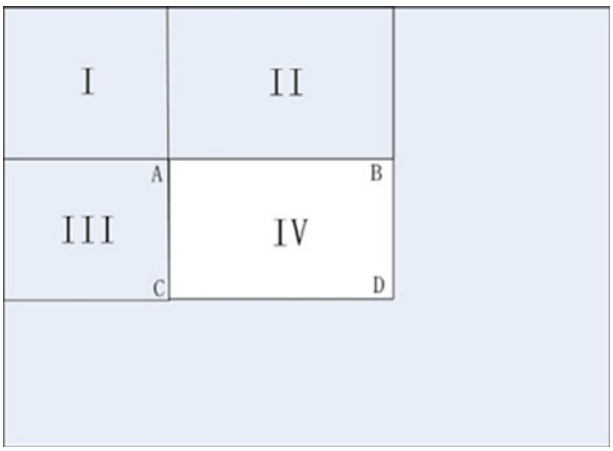


Fig. 10 The sum of the pixel value inside rectangle IV can be computed with four other references to the integral image. The value of the integral image at point A is the sum of the pixels in rectangle I. The value at B is I+II, with I+III at C and at location D is I+II+III+IV. Therefore, the sum within rectangle IV can be computed as D + A – (B + C)

3.4 Uniform sampling

In our experiment, the edge image is iterated thrice by different size of sub-images as mentioned in the previous section. Each pixel is assigned with the value $Tr(M)^2/Det(M)$ calculated from the corresponding sub-image. The pixels with value larger than the threshold t are marked as the potential seeds points. Figure 11(a) shows an example with 5×20 sub-image size, in which the white pixels are the obtained potential seeds points. One can see that most of the content within the storyboard are filtered out. Although the selected points are all good candidates, the number of candidates (which ranges from 1×10^4 to 1×10^5 though our experimentation) will be too large for the rest steps of processing. Therefore, a second round of selection is needed. In order to reduce the number of line candidate, we sample the seed points uniformly by dividing the marked image into grid meshes of 5×5 in size (as shown in Fig. 11(b)) and finding the corresponding maximum value in each mesh.

3.5 Line fitting

After the point with maximum ratio value in each grid mesh is selected, a short line segment in the corresponding sub-image should be estimated and fitted from the edge points within the sub-image. In our experiment, we use a total least square method [6]. In this model, in order to fit a straight line $ax + by + c = 0$ from a sequence of n measurements, (x_i, y_i), a maximum-likelihood solution is obtained by maximizing the following expression:

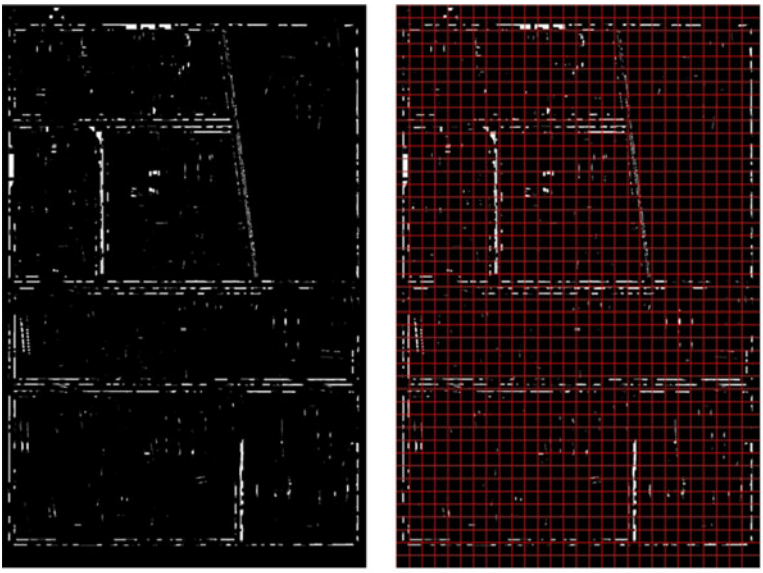


Fig. 11 **a** the candidate points whose corresponding sub-images contain short line segments **b** these candidate points are reduced by uniformly sampling using 5×5 grid meshes. (The source image is the same as that of Fig. 6(a))

$$-\frac{\sum_{i=1}^{n}(ax_i+by_i+c)^2}{2\sigma^2}+C, \tag{11}$$

where σ is the variance of the Gaussian noise, a, b, c are line parameters. The problem is substituted by the eigenvalue problem where the estimated matrix is

$$\begin{bmatrix} \frac{1}{n}\sum_{i=1}^{n}x_i^2-\frac{1}{n^2}\sum_{i=1}^{n}x_i\sum_{i=1}^{n}x_i & \frac{1}{n}\sum_{i=1}^{n}x_iy_i-\frac{1}{n^2}\sum_{i=1}^{n}x_i\sum_{i=1}^{n}y_i \\ \frac{1}{n}\sum_{i=1}^{n}x_iy_i-\frac{1}{n^2}\sum_{i=1}^{n}x_i\sum_{i=1}^{n}y_i & \frac{1}{n}\sum_{i=1}^{n}y_i^2-\frac{1}{n^2}\sum_{i=1}^{n}x_i\sum_{i=1}^{n}y_i \end{bmatrix}\begin{bmatrix} a \\ b \end{bmatrix}=\mu\begin{bmatrix} a \\ b \end{bmatrix}. \tag{12}$$

Combining with formula (1), the above formula can be presented as:

$$M\begin{bmatrix} a \\ b \end{bmatrix}=\mu\begin{bmatrix} a \\ b \end{bmatrix}. \tag{13}$$

As we can see, the line fitting process unifies with the ellipse approximation of the edge points in section 3.2. The first eigenvector of M is the major axis of the approximated ellipse which is also the orientation of the estimated straight line and the second eigenvector of M is the minor axis of the approximated ellipse which is the normal vector of the estimated line (i.e. $[a,\ b]^T$). Therefore the line can be approximated. The two values of the second eigenvector are the estimated values of a, b respectively. The values of the estimate matrix in formula (12) can also be calculated using the integral image mentioned in 3.4 which can speed up the line fitting process.

3.6 Line growing

After each short line segments are estimated from the sub-image, we employ a line growing method to extend the line segments to find more complete lines which enclose the story-board. The growing algorithm is inspired by the line-support regions growing method proposed in [7]. For each short line segments $ax+by+c=0$, our version first finds the starting point ending points and classifies the lines into horizontal-based and vertical-based by the slope factor of the lines. For the horizontal-based lines, they are growing at the two directions along the x-axis until no pixels exist in the region (x', y'), where $\left|y'+\frac{a}{b}x+\frac{c}{b}\right|<\varsigma$. For the vertical –based lines, a similar method is applied. The process is repeated until all the candidate short line segments have grown and most enclosing lines of the storyboards are completely extracted. However, there are several very short line segments inside the storyboards which don't 'grow' too long in this process. Therefore, they can be discarded by line segment length threshold. Algorithm 1 gives the complete details and Fig. 12 shows several results of the detected lines of our line segment detection algorithm.

Algorithm 1. LINE GROW

Input: An edge image I ; a starting short line segment with starting point $A(x_1, y_1)$ and ending point $B(x_2, y_2)$; angle θ with which the line intersect with x-axis; a distance tolerance ς

Output: An extended starting point $A(x'_1, y'_1)$ and ending point $B(x'_2, y'_2)$

IF $\theta <= 45^\circ$ *or* $\theta >= 135^\circ$ **THEN**

do $x'_1 \leftarrow x_1$;

$x'_2 \leftarrow x_2$;

repeat

$x'_1 \leftarrow x'_1 - 1$

until no edge pixel exists s.t. $| y + \frac{a}{b} x'_1 + \frac{c}{b} | < \varsigma$

repeat

$x'_2 \leftarrow x'_2 + 1$

until no edge pixel exist s.t. $| y + \frac{a}{b} x'_2 + \frac{c}{b} | < \varsigma$

$y'_1 \leftarrow -\frac{a}{b} x'_1 - \frac{c}{b}$

$y'_2 \leftarrow -\frac{a}{b} x'_2 - \frac{c}{b}$

end

ELSE

$y'_1 \leftarrow y_1$

$y'_2 \leftarrow y_2$

repeat

$y'_1 \leftarrow y'_1 - 1$

until no edge pixel exists s.t. $| x + \frac{by'_1 + c}{a} | < \varsigma$

repeat

$y'_2 \leftarrow y'_2 + 1$

until no edge pixel exists s.t. $| x + \frac{y'_2 + c}{a} | < \varsigma$

$x'_1 \leftarrow -\frac{y'_1 + c}{a}$

$x'_2 \leftarrow -\frac{y'_2 + c}{a}$

ENDIF

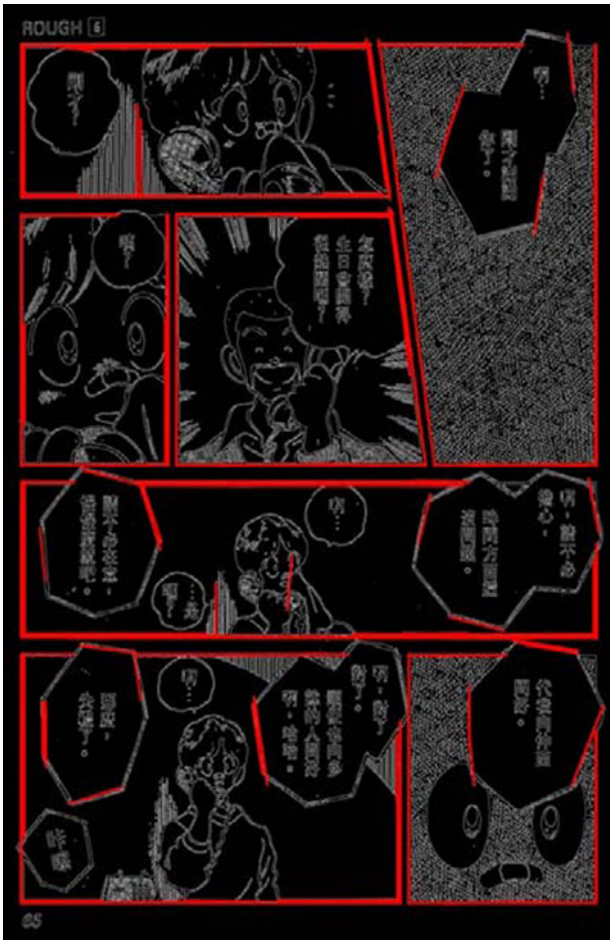


Fig. 12 The line segment detection result after the line growing process. (The source image is the same as that of Fig. 6(a))

4 Polygon detection

As described in Section 1, our proposed comic page segmentation method is based on polygon detection. Given a set of detected line segments, the next step is to combine these line segments into polygons which represent enclosing boxes of the storyboards. Based on the line segment detection results, we further develop a polygon detection algorithm to extract the enclosing box of each storyboard.

4.1 First round of clustering

Through our experimentation, about 1,000 to 10,000 line segments are detected for a common comic image. In order to reduce the number of the line segments, we merge some redundant lines by performing the first round of line clustering.

We check every pair of lines to see if they can be merged together, in other words, to see if they can be assigned to the same cluster. Let l_1 and l_2 denote two line segments; they can be merged together if the following conditions are satisfied:

(1) l_1 and l_2 are parallel to each other;
(2) l_1 and l_2 are overlapped with each other along the parallel direction;
(3) the distance between l_1 and l_2 is less than a threshold.

To check condition (1), we calculate the angle difference between the two lines. To check the condition (2), we project the two end points of l_1 onto l_2 direction vertically. If one or two of the projected points on l_2 direction are within l_2, these two lines overlap with each other along the parallel direction. For the condition (3), the distance between the two parallel lines ($ax + by + c_1 = 0, ax + by + c_2 = 0 \, where \, a^2 + b^2 = 1$) can be calculated by $|c_1 - c_2|$. An example showing two lines can be merged together is shown in Fig. 13.

Using the preceding criteria that determine whether each pair of line segments can be merged as the "connectivity" between the line segments, we perform connected components search to cluster the line segments. After clustering, we further fit a single line for each cluster. Instead of using all the points of the lines, the two end points are used to estimate the fitted line. In this process, we still use the total least square method as mentioned in Section 3.5. In our experiment, the number of detected lines is significantly reduced after first round line segments clustering: the total of more than 1,000 detected lines are reduced to about 100 for a common comic image.

4.2 Second round of clustering

At the point, we obtain the all the complete none-redundant straight line segments in the comic image. The following step is to combine these line segments into the full enclosing boxes to achieve the comic image segmentation.

The way to perform the second round clustering is very similar to that of the first round, we still use connect component search to cluster the line segments. However, the connectivity between two lines is defined by the distance between their end points. For a pair of lines (namely AB, CD), there four distances between their end point which are AC, AD, BC, BD. If either one of the combination four values between the end points smaller than a threshold (in our case, 10), they probably belong to the same storyboards. However, as storyboards sometimes are arranged compactly within the image (see Fig. 14), only use distance as the criteria to cluster the lines will inevitably put the lines from different storyboard together.

To solve this problem, we add an additional verification process to check local connectivity between the end points of the two line segments. For each pair of end points whose distance is below 10 pixels, a small square area with the side length of 10 is acquired. Within this small area, we check the 8-connctivity of the two end points in the edge image. If they connect with each other, the two line segments belong to the same storyboard (see Fig. 15

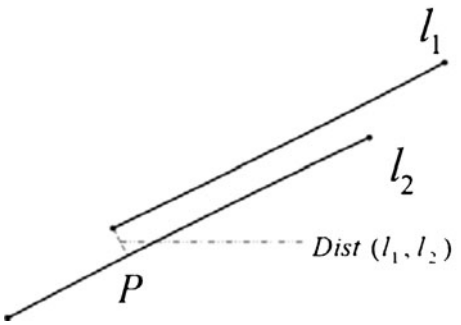


Fig. 13 $l_1 // l_2$. The left end point of l_1 are projected vertically on l_2 (i.e. P). P is on the line segment l_2. If the distance between l_1 and l_2 is below certain threshold, they should be merged together

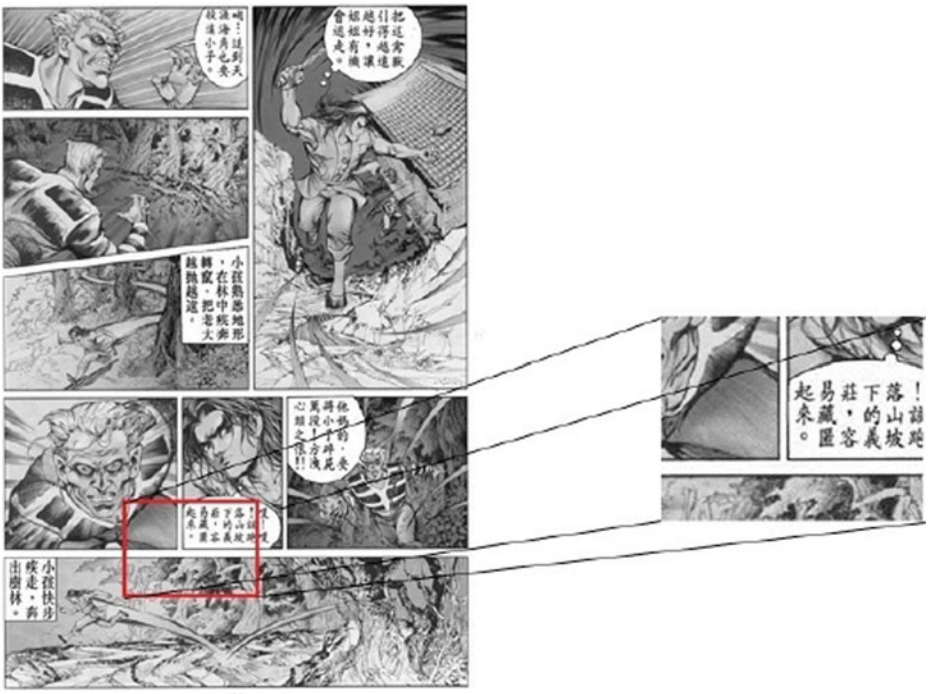


Fig. 14 The storyboard arrangement sometimes is very compact, which makes the lines from different storyboard very close. As a result, only use end point distance is not enough for checking their connectivity. (source: title YiYongMen (Hongkong), author Yulang Huang, Volume 1 p.28)

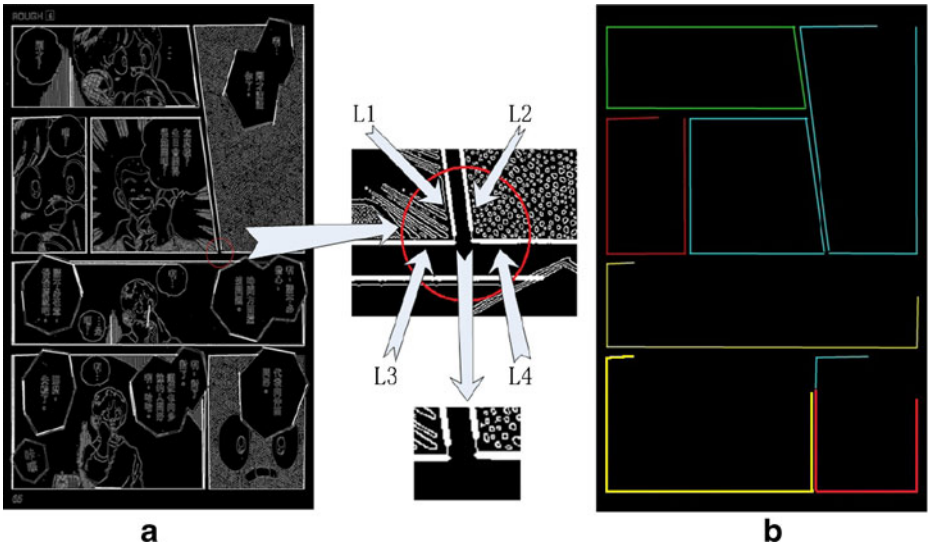


Fig. 15 **a** For every pair of line segments, if their endpoints are within 10 pixels, we check their local connectivity. L1, L2, L3, L4 are very close to each other, but connectivity constraints permit only L1 connect with L3, L2 connect with L4. **b** After the second round of clustering, detected line segments are composed and form approximate enclosing box of the storyboard (the box on the left up is complete while the box on the left down of the image is incomplete). (The source image is the same as that of Fig.6 (a))

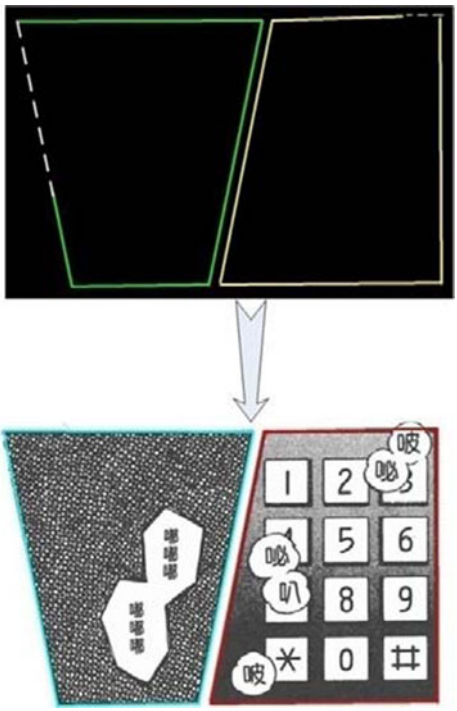


Fig. 16 The cluster with four or five line segments can form the complete enclosing box by finding the intersect of two lines which have one endpoint without connecting with other lines. (source: title ROUGH, author Mitsuru Adachi, publisher Shogakukan Inc., Volume 6 p.64)

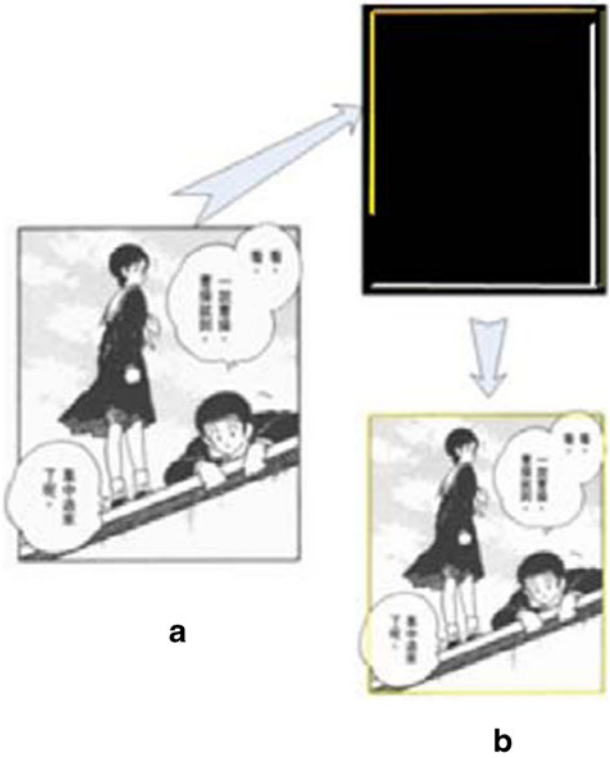


Fig. 17 **a** Line clusters are separated with obtruding objects. **b** For each pair of line cluster of two or three lines, they are checked if they can be merged together. (source: title ROUGH, author Mitsuru Adachi, publisher Shogakukan Inc., Volume 6 p.28)

(a)). Obviously, we can skip this local connectivity verification process if two end points are not globally connected. We previously establish connected component labels of each edge points with a very efficient algorithm [8] and store them in a label array. Thus, we can check whether two end points are globally connected or not very quickly by referring to the label array.

To sum up, if the above two criterion are met, the connectivity between the two lines is true, that is to say, they belong to the same enclosing box of the frame. After this round of line clustering, the detected line segments are formed into either complete or incomplete enclosing box of the comic frames (as shown in Fig. 15(b)).

4.3 Post processing

The segmentation results of the previous step are not always satisfying all the time, especially for the comic image in which many extruding objects and text balloons exist. Under such circumstances, the enclosing boxes generated from the previous step are always fragmentary (see Fig. 15(b)). Therefore, a post processing stage is needed to get the complete enclosing box. We present several rules for constructing the complete enclosing box of the storyboard which is listed below.

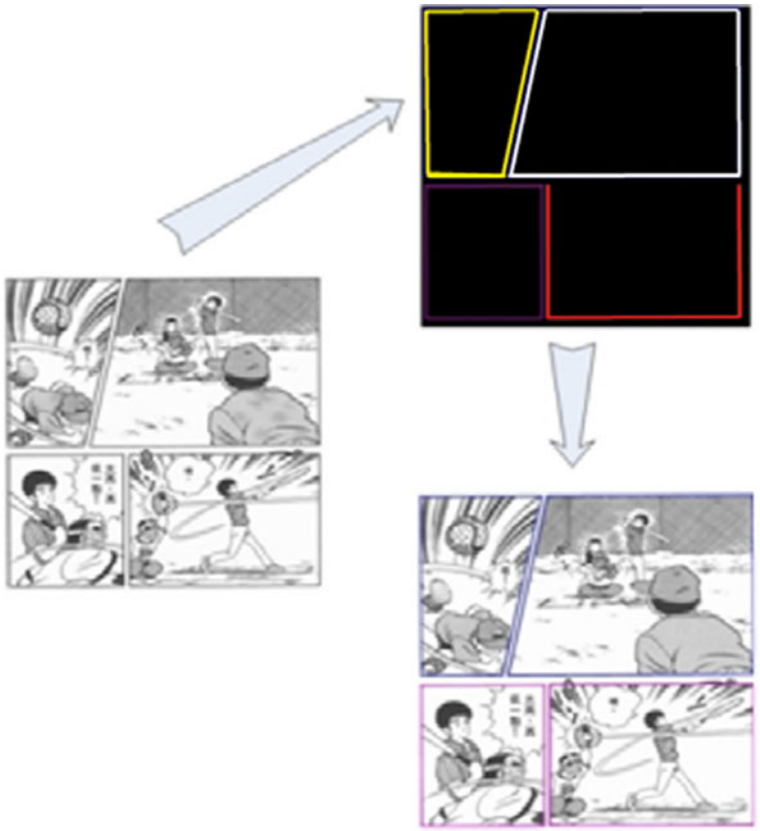


Fig. 18 For the cluster with three line segments, if they can't be merged with other cluster, we link the two end points that don't connect with other line segments. (source: title ROUGH, author Mitsuru Adachi, publisher Shogakukan Inc., Volume 9 p.139)

(1) If there are four of five line segments in one cluster and the enclosing box are incomplete, we find intersect of the two lines in the cluster whose either one of the end point is not connected to the other lines. Then the complete enclosing box is formed (see Fig. 16)

(2) For the clusters with two or three line segments, they may originate from the same storyboard as the obtruding objects tear them apart (see Fig. 17(a)). Therefore, we should merge these clusters. To achieve this goal, we check every pairs of clusters with two or three line segments. To be more specific, given two clusters C_A, C_B, we find the bounding box of them $Bound_{AB}$ (i.e. the minimal rectangle that contains the two clusters) and bounding boxes of each one ($Bound_A$ $Bound_B$ respectively). The two clusters can be merged together if the following two rules are satisfied:

Rule 1: the span of the bounding box $Bound_{AB}$ on the x-axis, y-axis is smaller than half of the width and height of the image respectively.

Rule 2: the overlap area between the bounding box of the two clusters and the bounding box of each cluster ($Overlap$ ($Bound_{AB}$, $Bound_A$) $Overlap$

Fig. 19 Our final segmentation result, the comic image is partitioned into several disjoint quadrilateral regions. (The source image is the same as that of the Fig. 6(a))

Table 1 The threshold list of our approach with their minimal, maximum values and incremental units

Name	Function	Minimal value	Maximum value	Unit
LineDist	Line distance threshold for the first round of line clustering	2	20	2
EndptDist	End point distance threshold for the second round of line clustering	2	20	2
SampleSize	The side length of the uniform sampling grid mesh	2	15	1
LocalArea	The side length of the local connectivity search square area	5	50	5
GrowPara	Tolerance value in line grow algorithm	1	10	1
EngenRatio	The ratio t in sub-image analysis	300	600	20

($Bound_{AB}$, $Bound_B$)) take up more than a certain percentage (in our case, we set the percentage to 0.75) of area of the bounding box of each of the two clusters. See formula (14):

$$\begin{matrix} Overlap(Bound_{AB}, Bound_A)/S(Bound_A) > 0.75 \\ Overlap(Bound_{AB}, Bound_B)/S(Bound_B) > 0.75 \end{matrix}. \quad (14)$$

As a result, all the clusters that belong to the same storyboard are merged together (see Fig. 17(b)).

(3) For the cluster which can't be merged with any other line clusters, the two end point which are not connected to any other lines are linked together to form a complete enclosing box (see Fig. 18).

After the post-processing, the comic image is segmented and divided into several disjoint areas (i.e. the storyboards). Figure 19 shows our final result for the sample comic image.

5 Experiments

In this section, we introduce our experiment (which are divided into two parts) to show the performance of the proposed comic page segmentation method, including parameter settings, evaluation criteria, experimental results and comparison with other available methods. We collect 200 pages of comic images as training data and 2,237 pages as testing data, which come from 12 different comics published from Japan, China mainland and Hong Kong. These comic images are of different genres including campus, chivalrous conduct, youth and girls. We use Microsoft .NET framework with C++ as the programming language and OPENCV library.

Table 2 The parameter set obtained from the training process

Threshold	Value	Threshold	Value
LineDist	2	LocalArea	10
EndptDist	10	GrowPara	2
SampleSize	5	EngenRatio	400

Table 3 Comparison of our experimental results with Tanaka's [15] and Arai's [1] methods for the Dragon Ball Volume 42 comic image sources

The result type	Tanaka's method	Arai's method	Our method
Correct	195/89 %	218/92 %	223/94 %
Failure	22	19	14/6 %
Not tested	20	0	0
Tested pages	217	237	237
Total pages	237	237	237

5.1 Parameter settings

As stated above, our comic page segmentation algorithm includes a set of thresholds, which are crucial for the final segmentation results. They are listed in Table 1. In order to find the best parameter settings for the comic page segmentation, we select 200 comic images from the 12 different comic series as training data for tuning the thresholds. The correct segmentation results of the training data are marked by manually pointing out the four vertices of each frame. The evaluation criterion between the experimental segmentation results and ground truth data is based on the coordinates of the vertices of the detected frames (similar to that of the [10]). If the distance between the all four detected vertices of a frame and the corresponding corners in the ground truth data (which is marked manually) is less than K pixels. It falls into the correct detection category. Otherwise, it's an incorrect segmentation of the frame. In our experiment, we set the K value to 10 pixels based on our experience, and we find this method is consistent with human's visual perception.

The experiment examination criteria include the precision P rate (see formula 15) and recall R rate (see formula 16) of the frame detection:

Fig. 20 Figure (**a**), (**b**) and (**c**) show the type of simple, complex and hard comic images respectively. (source: **a** title ROUGH, author Mitsuru Adachi, publisher Shogakukan Inc.,Volume 1 p.92 **b** title Dragon Ball, author Akira Toriyama, publisher Shueisha Inc. Volume 1 p.154 **c** title LuDingJi, author Louis Cha Leung Yung, Chapter 29 p.22)

Table 4 Comic dataset information

Cartoon type	Quantity	Image size
Simple	750	1000×1200 (approx.)
Complex	1050	
Hard	200	

$$P = S/N, \tag{15}$$

$$R = S/G, \tag{16}$$

where S denotes the number of the correctly detected frames; N is the number of the frame detected and G is the number of the correct frames to be detected. Therefore, we use the F measure as the scoring function and thus maximize the following expression:

$$F = \frac{2PR}{P+R} \tag{17}$$

We change the value of each threshold from its minimal to maximum by incrementing one unit of the corresponding value (in Table 1), and find the setting that maximize the F value which is shown in Table 2. Then, these setting are kept unchanged in all of our experiments.

5.2 Experiment 1

Two hundred thirty-seven of the 2,237 pages are from the comic "Dragon Balls" Volume 42 which are used as experimental data in Tanaka's [15] and Arai [1]'s works. In order to compare our method with these two methods, we first test our method on these comic page images.

The evaluation criterion is the number of the correct segmented pages rather than the frames, which is used in [15] and [1]. To be more specific, if there exist one or more incorrect frame segmentation within one page, the total segmentation of that page fails which we term "Failure"; otherwise, the segmentation result is "Correct". The experimental result is shown in Table 3 below.

As we can see, our method performs better than both Tanaka's and Arai's method. We successfully segment five pages more than Arai does. Moreover, we further check the failure cases, and find that for those unsuccessfully segmented images, most of them contain only one or two incorrect segmented frames.

Table 5 Experimental result for three different categories of comic images; G refers to the number of fames to be detected; N refers to the number of detected frames; S refers to the number of correctly detected frame and Page refers to the number of correctly segmented comic images

Comic type	Quantity	G	N	S	Page
Simple	750	3938	3936	3936	748/99.73 %
Complex	1050	5575	5540	5137	899/85.61 %
Hard	200	989	865	722	62/31 %

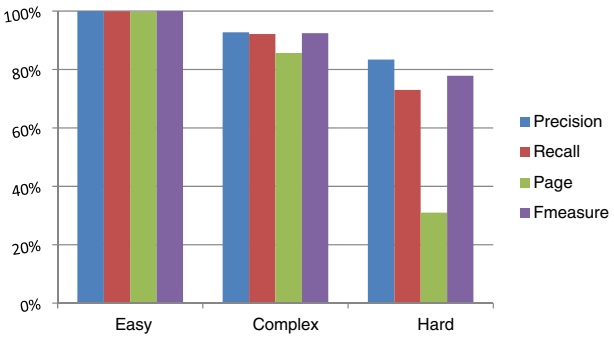


Fig. 21 The precision, recall, accuracy and page rate of the three different kinds of comic images; again, the page that contains one or more frame errors is considered a failure

5.3 Experiment 2

The rest of the comic images in our data set can be categorized into three class based the complexity of the comic layout patterns: simple, complex and hard. If there are no text balloons or other objects overlapping on the frame borders and all the frames are rectangles, such page is a simple page. For the page with contents occasionally obtrude the frame lines or some of the frames are irregular quadrilateral polygons, we put them into the complex category. For those comics whose layout patterns are extremely complicated which even people can't not fully-analyze, their type is "hard". Examples of three different kinds of comic images are shown in Fig. 20. The quantity of each class of comic in our data set is decided based on our observation on a huge number of comic images from which we can roughly decided the approximate ratio of the quantity of each type of the comic images. The details of the comic data set are shown in Table 4.

We extend our experiment examination criteria in the second part by including the precision P rate (see formula 15), recall R rate (see formula 16) and F measure (see formula 17) of the frame detection and Table 5 shows our experimental results.

The precision, recall, accuracy rate and F measure of each type of the comic image are shown in Fig. 21.

Fig. 22 The frame example whose border lines are severely fragmented. (source: title Dr. Slump, author Akira Toriyama, publisher Shueisha Inc., left image Volume 2 p.95; right image Volume 3 p.192)

The results show that the segmentation results for the easy mode of the comic images are nearly perfect as these images contain only rectangles without any objects obtruding from the borderline. More than 85% of the complex images are successfully segmented and those failed pages contain only one or two frame segmentation errors which are indicated by the precision and recall rate. Although the page segmentation rate for the hard images falls sharply into 31 % which results from the extremely complicated layout pattern, the frame segmentation rate is optimistic. The average processing time of our method is 11 s per image and the time can be reduced by exploiting the parallelism manifest in our algorithm.

In sum, the first part of experiment indicates a better segmentation result compared with Tanaka's and Arai's by using the same data set. The second part of the experiment works on three different classes of data sets and achieves perfect segmentation for simple comic image, satisfying segmentation for complex comic image and comparatively lower rate for the hard type of comic image which we will be working on in the future.

6 Conclusion and discussion

In this paper, we present a novel method for automatically extracting scene frames from comic images. The first step of our method is to detecting straight line segments from the image, which includes preprocessing, sub-image division, uniform sampling and line growing. The second step is the polygon detection aiming at compose the detect lines into quadrilateral polygons. We achieved this by using two rounds of line clustering and post-processing stages. The first part of our experiment indicates a better segmentation result compared with Tanaka's and Arai's by using the same data set. The second part of the experiment works on three different classes of data sets and achieves perfect segmentation for simple comic image, satisfying segmentation for complex comic image and comparatively lower rate for the hard type of comic image. Our future work is to make those failed image to be segmented correctly, for example, for the frame border lines that are severely fragmented by obtruding drawings (see Fig. 22), we will apply more post processing rules to compose these fragmented lines together.

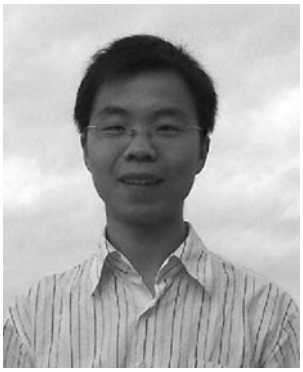
Acknowledgments This work is supported by National Basic Research Program of China, also Named "973 Program" (No. 2010CB735908).

References

1. Arai K, Tolle H (2010) Automatic e-comic content adaptation. Int J Ubiquit Comput 1(1):1–11
2. Ballard DH (1981) Generalizing the Hough transform to detect arbitrary shapes. Pattern Recognit 13(2):111–122
3. Burns JB, Hanson AR, Riseman EM (1986) Extracting straight lines. IEEE Trans Pattern Anal Mach Intell 8(4):425–455
4. Canny J (1986) A computational approach to edge detection. IEEE Trans Pattern Anal Mach Intell 8(6):679–698
5. Chung KL, Lin ZW, Huang ST, Huang YH, Liao HYM (2010) New orientation-based elimination approach for accurate line-detection. Pattern Recognit 31:11–19
6. Forsyth DA, Ponce J (2002) Computer vision: a modern approach, 1st edn. Prentice Hall 467–490
7. Gioi RG, Jakubowicz J, Morel JM, Randall G (2010) LSD: a fast line segment detector with a false detection control. IEEE Trans Pattern Anal Mach Intell 32(4):722–732
8. Grana C, Borghesani D, Cucchiara R (2010) Optimized block-based connected components labeling with decision trees. IEEE Trans Image Process 19(6):1596–1609
9. Ho CT, Chen LH (1996) A high-speed algorithm for line detection. Pattern Recognit Lett 17:467–473
10. In Y, Oie T, Higuchi M, Kawasaki S et al (2010) Fast frame decomposition and sorting by contour tracing for mobile phone comic images. Proc. International Conference on Visualization, imaging and simulation (VIS), Wisconsin, 2010:23–28

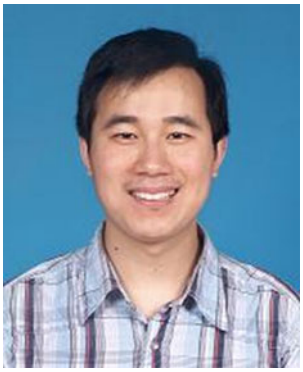
11. Ishii D, Watanabe H (2010) A study on frame position detection of digitized comic images. Workshop on Picture Coding and Image Processing (PCSJ), Nagoya, 2010:124–125
12. Jain AK, Yu B (1998) Document representation and its application to page decomposition. IEEE Trans Pattern Anal Mach Intell 20(12):294–308
13. Lo RC, Tsai WH (1995) Gray-scale Hough transform for thick line detection in gray-scale images. Pattern Recognit 28(5):647–661
14. Lowe DG (2004) Distinctive image features from scale-invariant keypoints. Int J Comput Vis 60(2):91–110
15. Tanaka T, Shoji K, Toyama F, Miyamichi J (2007) Layout analysis of tree-structured scene frames in comic images. Proc. International Joint Conferences on Artificial Intelligence (IJCAI), Hyderabad, January 2007: 2885–2890
16. Theodoridis S, Koutroumbas K (2008) Pattern recognition, 4th edn. Academic Press 20–50
17. Viola P, Jones MJ (2004) Robust real-time face detection. Int J Comput Vis 57(2):137–154
18. Xi J, Hu J, Wu L (2002) Page segmentation of Chinese newspapers. Pattern Recognit 35 (12):2695–2704
19. Yamada M, Budiarto R, Endoo M, Miyazaki S (2004) Comic image decomposition for reading comics on cellular phones. IEICE Trans Inf Syst E87-D(6):1370–1376

Luyuan Li received his B.Eng degree from Tianjin University in 2010, China. He is now a PhD candidate at the Institute of Computer Science and Technology, Peking University.



Yongtao Wang received his Ph.D. degree of pattern recognition and intelligent system in 2009 from Huazhong University of Science and Technology (HUST), China. During 2010, He was a research scientist of the Temasek Laboratories, Nanyang Technological University, Singapore. He is currently an assistant professor of Institute of Computer Science& Technology, Peking University, China. His research interests involve document image analysis and understanding, wide baseline matching, and motion segmentation.

Zhi Tang received his PhD degree of computer science in 1995 from Peking University, China. He is now a professor at the Institute of Computer Science & Technology, Peking University and the director of the State Key Laboratory of Digital Publishing Technology. His research interest majorly includes document analysis and understanding, digital rights management.



Liangcai Gao received his PhD degree of computer science in 2010 from Peking University, China. He is now an assistant professor at the Institute of Computer Science & Technology, Peking University. His research interest mainly includes document analysis and understanding, metadata management.