Offline Text and Non-text Segmentation for Hand-Drawn Diagrams

Buntita Pravalpruk(✉) and Matthew M. Dailey

Computer Science and Information Management, Asian Institute of Technology, Klong Luang, Pathumtani, Thailand
buntita@gmail.com, mdailey@ait.asia
http://www.cs.ait.ac.th/

Abstract. Writing and drawing are basic forms of human communication. Handwritten and hand-drawn documents are often used at initial stages of a project. For storage and later usage, handwritten documents are often converted into a digital format with a graphics program. Drawing with a computer in many cases requires skill and more time than less formal handwritten drawings. Even when people have experience in computer drawing and are familiar with the application, it takes time. Automatic conversion of images of hand-drawn diagrams into a digital graphic format file could save time in the design process. One of early critical tasks in hand-drawn diagram interpretation is segmentation of the diagram into text and non-text components. In this paper, we compare two approaches for offline text and non-text segmentation of contours in an image. We describe the feature extraction and classification processes. Our methods obtain 82–86 % accuracy. Future work will explore the application of these techniques in a complete diagram interpretation system.

Keywords: Image recognition · Hand-drawn diagram · Hand-written diagram · Text and non-text classification

1 Introduction

Writing is a simple and natural way for people to take notes and exchange information, exchange thoughts and feelings, express ideas, and plan. Handwritten notes often contain pictures or diagrams in addition to the associated text. Such notes contain a trove of useful information that may take many forms, including doctors' summaries of patient interviews, business analysts' depictions of business processes, engineers' descriptions of computer system designs, architects' sketches of a building design, and so on.

Handwritten and hand-drawn documents are very common at the initial stage of a design or brainstorming process. However, oftentimes, rough initial handwritten documents must be converted into a clean digital format with a graphics program or other tool. Unfortunately, producing formal diagrams with a computer usually requires more skill and more time than drawing less formal

© Springer International Publishing Switzerland 2016
R. Booth and M.-L. Zhang (Eds.): PRICAI 2016, LNAI 9810, pp. 380–392, 2016.
DOI: 10.1007/978-3-319-42911-3_32

diagrams on paper. Even when the person transcribing the document is skilled, the conversion process wastes precious time, making the prospect of automatic conversion attractive. For this reason, we are interested in automatic conversion of hand-drawn documents into digital format after they are scanned or photographed.

There are generally three main steps in automated hand-drawn document interpretation. First, the document should be divided into text and non-text regions, because diagrams and text require different strategies for interpretation. Second, in text regions, each letter or digit can be detected and classified (Freitas et al. 2007; Kara and Stahovich 2007; Lauer et al. 2007; Zhong et al. 2010), then we can form words and numbers from the letters and digits. Meanwhile, in non-text regions, we can detect the lines and curves comprising objects (Hammond and Davis 2006; Stahovich 2004), and if the document is known to be structured (i.e., a representative of a specific type of document such as a UML or electrical circuit diagram), the lines and curves can be given semantic interpretations. In a final step, we can combine the results of the two flows, redraw the document consistently and accurately in terms of the recognized text segments and objects, and provide the semantic interpretation of the diagram to downstream processes.

In this paper, we focus on the first step, namely text and non-text segmentation. There are two general approaches to the problem, depending on the type of input device used to acquire the diagram. The first type of device is a sketchpad or tablet that records the timing and sequence of each pen/finger stroke. There have been many methods proposed to deal with such *online* documents by researchers, focusing on either the segmentation step (Waranusast et al. 2009) or the interpretation step (Costagliola et al. 2006; Hammond and Davis 2007; Kara and Stahovich 2007). One area in which successful methods have been demonstrated is in flowchart interpretation (Bresler et al. 2013; Lemaitre et al. 2013).

However, when working with the second type of devices, scanners and cameras, we are faced with the task of interpreting *offline* documents, simple raster images without stroke or timing information. This problem is more difficult, because letters, digits, lines, and curves must be inferred before they can be classified then recognized. There has been some work in this area, for example, on the use of pattern mapping and statistics to detect shapes in offline flowchart diagrams (Wu et al. 2015). Successful segmentation and interpretation of more complex kinds of diagrams, however, would require more sophisticated methods.

We tackle the problem of offline document segmentation by first detecting contours in the input image then applying sequence analysis methods to the result. In the first stage, we classify individual contours as to whether they are text or non-text objects, and then we analyze relationships between neighboring contours to classify them into the same group. Exploiting relationships between neighboring contours enables improved interpretation.

We have implemented two approaches to the problem of offline text and non-text segmentation and, as a case study, have tested the two methods on a collection of hand-drawn diagrams in the Business Process Model and Notation

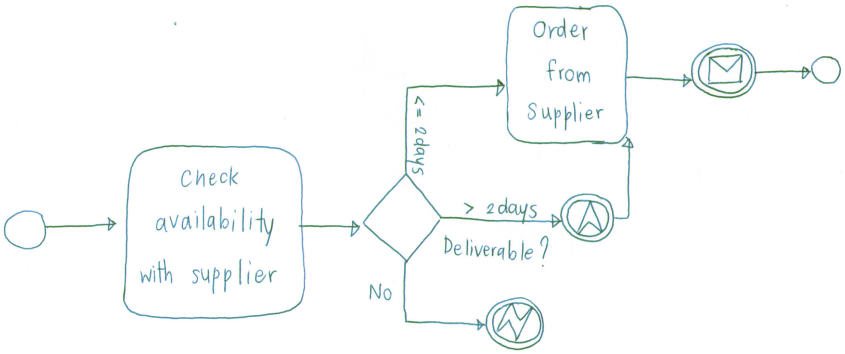

Fig. 1. An example diagram in Business Process Model and Notation (BPMN) format.

(BPMN) format. An example diagram is shown in Fig. 1. In an experiment with 120 diagrams, we obtain 80–84 % accuracy on a per-pixel basis. In the rest of this paper, we present the methodology, experiments, and analysis of the experimental results in more detail.

2 Method

Our method performs the following steps (Fig. 2):

1. Acquire input image I.
2. Binarize I to obtain binary image I_b.
3. Find the connected components $C_1, C_2, \ldots$ in I_b. Let N be the number of connected components found.
4. Convert each connected component C_i to a sequence S_i consisting of the *turns* made along the *outer contour* of C_i. We use turn codes S (straight), L (left), and R (right).
5. Classify each sequence code S_i using two alternative classifiers:
   (a) *Turn ratio classifier*: simple threshold on the ratio of straight elements to turn elements in the sequence.
   (b) *Bayes maximum a posteriori (MAP) classifier*: determine the class (text or non-text) with the highest conditional probability: $P(\text{text} \mid S_i)$ or $P(\text{non-text} \mid S_i)$.
6. Output the result of each classifier (labels y_i^B for the MAP classifier and labels y_i^R for the ratio classifier) to a database.

2.1 Binarization

For binarization, we use the standard Otsu algorithm (Otsu 1979) implemented by OpenCV.

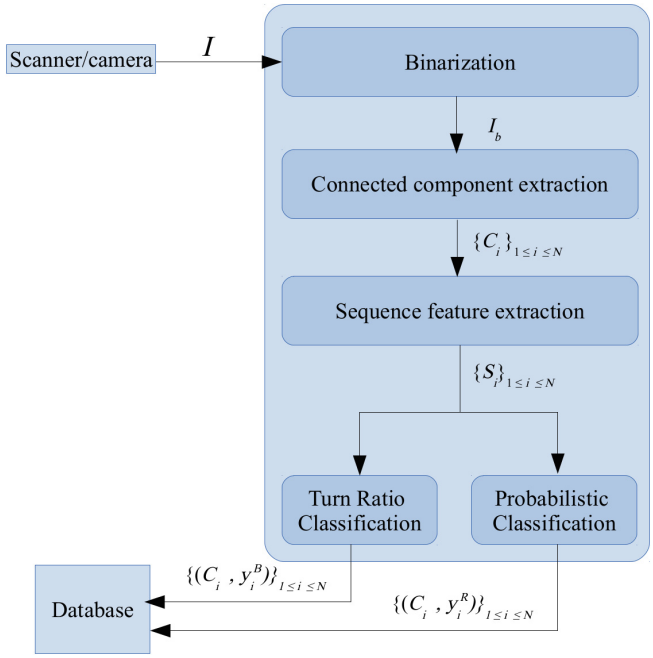

Fig. 2. System overview.

2.2 Contour Conversion

We find the connected components $C_i, 1 \le i \le N$ of I_b. Currently, we assume each contour is entirely text or non-text without any overlap. This assumption will be relaxed in future research. We then find the outer contour of each connected component using the contour tracing algorithm of Suzuki and Abe (Suzuki and Abe 1985) as implemented in OpenCV (OpenCV Dev Team 2016).

2.3 Sequence Feature Extraction

Once we find the contour for a given connected component C_i, we trace the contour, converting each pixel into a feature indicating whether, relative to the previous pixel in the chain, the current pixel represents a straight forward motion or a turn to the left or right. For each connected component C_i we obtain a sequence S_i of turn features S (straight), L (left turn), and R (right turn).

See Fig. 3 for the coding of pixels in the contour sequence as S, L, and R. Note that the encoding is ambiguous (the original contour cannot be re-created from the turn feature sequence), but the encoding performs well in classification, as we shall see in the results.

For straight moves, assume we are given the position of the middle point of three contour pixels as (x, y). There are three cases that should be classified

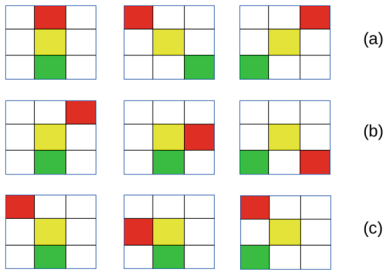

Fig. 3. Examples of turn types. In every diagram, the green pixel is the previous pixel in the contour, the yellow pixel is the pixel being considered, and the red pixel is the next pixel in the contour. (a) Straight examples. (b) Right turn examples. (c) Left turn examples. (Color figure online)

as straight: (1) the three points have the same position in x with all different positions of y, or the same position in y and all different positions in x; (2) the first point has both x position and y position less than the position of the middle point and the last point has both x position and y position greater than those of the middle point, or vice versa; (3) the first point has an x position greater than that of the middle point and a y position less than that of the middle point, while the last point has a x position less than the x position of the middle point and a y position greater than that of the middle point, or vice versa. If the sequence of three consecutive pixels around a target pixel fall into one of these three cases, the pixel is considered straight and assigned a label S.

For right turns, there are four cases: (1) the first point has less position of y than that of the last point and both of them have the same or less position of x than that of the middle point. (2) the first point has greater position of y than that of the last point and both of them have the same or greater position of x than that of the middle point. (3) the first point has less position of x than that of the last point and both of them have the same or less position of y than that of the middle point. (4) the first point has greater position of x than that of the last point and both of them have the same or greater position of y than that of the middle point. If the sequence of three consecutive pixels around a target pixel fall into one of these three cases, the pixel is considered right turn and assigned a label R.

For left turns, there are also four cases: (1) the first point has greater position of y than that of the last point and both of them have the same or less position of x than that of the middle point. (2) the first point has less position of y than that of the last point and both of them have the same or greater position of x than that of the middle point. (3) the first point has greater position of x than that of the last point and both of them have the same or less position of y than that of the middle point. (4) the first point has less position of x than that of the last point and both of them have the same or greater position of y than that of the middle point. If the sequence of three consecutive pixels around a target

pixel fall into one of these three cases, the pixel is considered right turn and assigned a label L.

On the hypothesis that second-order relationships between contour turns could be useful for contour classification, we also consider pairs of consecutive turns as features. After converting contour i into turn sequence S_i, we can obtain a sequence of second-order turn features left turn after left turn (LL), right turn after left turn (RL), straight after left turn (SL), left turn after right turn (LR), right turn after right turn (RR), straight after right turn (SR), left turns after straight (LS), right turn after straight (RS) and straight after straight (SS). We use the consecutive turn counts in the probabilistic classifier described below.

2.4 Sequence Feature Classification

Most classification methods require a fixed vector of inputs and produce a binary or multi-class output. For sequential data such as the turn sequences S_i, whose length are not pre-determined, we need to convert the arbitrary-length data sequence into a fixed size vector of summary features.

For the turn ratio classifier, we do this by simply counting the number of turn features in a given sequence, e.g., for left turn features in turn sequence S_i, we find the number of L features in S_i, written $N_i^L = |S_i|_L$, and similarly to obtain N_i^R and N_i^S.

For the probabilistic classifier, as will be seen in the next section, we use a first-order Markov model that requires counts of the consecutive turn features, i.e., N_i^{LL}, N_i^{LR}, N_i^{LS}, N_i^{SL}, N_i^{SR}, N_i^{SS}, N_i^{RL}, N_i^{RR}, and N_i^{RS}. When we sum these features over all sequences in a training set to obtain the total frequency of each type of consecutive turn, we denote the result $N^{SL} = \sum_i N_i^{SL}$, and similarly for the other consecutive turn types.

Turn Ratio Calculation. Based on the hypothesis that contours of non-text elements of a drawing will tend to have longer straight strokes than text elements, we can expect that the ratio of turn sequence elements to straight sequence elements, i.e., $R_i = (N_i^R + N_i^L)/N_i^S$, should be relatively small when C_i is a text element and relatively large when C_i is a non-text element.

Classifying elements as text or non-text based on the feature R_i, then, merely requires a threshold above which we classify the element as text and below which we classify the element as non-text. We use a straightforward optimization over a training set to find the best threshold.

Markov Chain Processing. For the probabilistic classifier, we attempt to find posterior probabilities $P(\text{text} \mid S_i)$ and $P(\text{non-text} \mid S_i)$, and answer with the class having the highest posterior. From Bayes' rule, we have

$$P(\text{text} \mid S_i) = \frac{P(S_i \mid \text{text})P(\text{text})}{P(S_i)}. \tag{1}$$

Expanding the sequence S_i into its tokens and ignoring the constant denominator, we have

$$P(\text{text} \mid S_i) \propto P(S_{i,1}, S_{i,2}, \ldots, S_{i,|S_i|} \mid \text{text}) P(\text{text}). \tag{2}$$

We can use the chain rule for the joint conditional probability and the first-order Markov assumption to obtain

$$P(\text{text} \mid S_i) \propto P(\text{text}) \prod_{j=1}^{|S_i|} P(S_{i,j} \mid S_{i,1} \cdots S_{i,j-1}, \text{text}) \tag{3}$$

$$\approx P(\text{text}) \prod_{j=1}^{|S_i|} P(S_{i,j} \mid S_{i,j-1}, \text{text}). \tag{4}$$

The probabilities $P(S_{i,j} \mid S_{i,j-1}, \text{text})$ merely express the frequency with which a turn or straight element of the contour sequence follows a turn or straight element in a text element contour. For each type of element (text and non-text), there are nine such probabilities that are easily estimated based on turn count features. For example, $P(S_{i,j} = \text{S} \mid S_{i,j-1} = \text{L}, \text{text})$ is simply $N^{LS}/(N^{RS} + N^{SS} + N^{LS})$, or more simply, N^{LS}/N^{S}.

As an example, see Fig. 4 for a visualization of the conditional probabilities we observed for text and non-text contours in a training set consisting of BPMN diagram elements. Confirming the basic hypothesis, we can observe that $P(S_{i,j} = \text{S} \mid S_{i,j-1} = \text{S}, \text{non-text}) > P(S_{i,j} = \text{S} \mid S_{i,j-1} = \text{S}, \text{text})$, i.e., $0.852 > 0.645$.

To classify a given sequence, we simply compare the posterior probabilities. If $P(\text{text} \mid S_i) > P(\text{non-text} \mid S_i)$, we conclude that the sequence is text; otherwise we conclude it is non-text.

3 Experimental Design

To empirically evaluate the methods described in Sect. 2, we collected 15 example BPMN 2.0 process specifications in XML format from the OMG Website[1] and imported them into Signavio Process Editor.[2] Signavio could only successfully convert nine of 15 process specifications, so we exported those nine diagrams to PDF then printed them. We had 200 secondary school and undergraduate students in Thailand copy the diagrams, then we scanned each diagram using a desktop scanner at 2400 dpi. The result was 600 hand-drawn BPMN diagrams as raw bitmaps. We selected 120 of these diagrams for ground truth annotation at random.

To annotate the hand-drawn diagrams with ground truth information, we developed a simple application able to binarize and extract contours from the bitmaps. The application highlights each connected component of the diagram

[1] http://www.omg.org/spec/BPMN/20100602/.
[2] https://editor.signavio.com/.

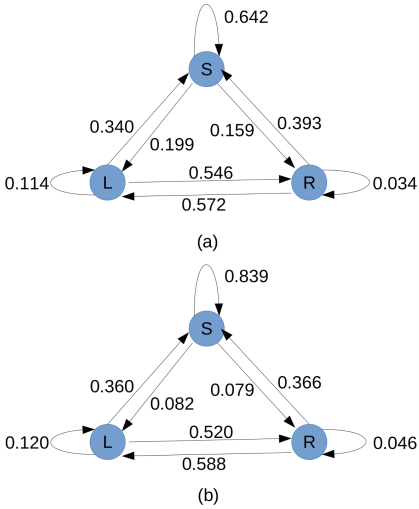

Fig. 4. Example Markov chains for turn sequences for text and non-text contours, calculated from a training set of 96 BPMN diagrams. (a) Markov chain for text contours. (b) Markov chain for non-text contours. Non-text contours tend to have more S elements than do text contours.

in turn for the user to annotate. The user identifies each contour as *text*, *non-text*, or *noise*. *Text* contours are those that contain letters or parts of letters without connection to anything other than letters. *Non-text* contours are those that have no connection to any letter, that contain just lines, curves, and other shapes meaningful for the overall diagram. An example of a partial annotation of a diagram is shown in Fig. 5. Contours without any meaning to the diagram are classified as *noise*. We used colors to identify contour types as shown in the figure.

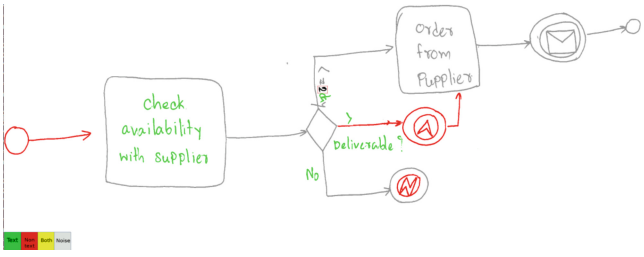

Fig. 5. Example partially annotated hand-drawn BPMN diagram. Text contours are marked green. Non-text contours are marked red. Contours that contain both text and non-text are marked with yellow and split into contours that can be classified into one type. (Color figure online)

When a contour is marked as containing both text and non-text, the annotator is asked to annotate the text and non-text parts of the contour using a paint-like tool. An example is shown in Fig. 6.

Fig. 6. Example annotation of text and non-text for a contour containing both. Non-text parts are painted black, text parts are painted red, and overlapping parts are painted yellow. (Color figure online)

For convenience, we also preprocess each contour to get the turn counts during the annotation process. (In an online system, of course, the processing would be done online as soon as the diagram is loaded.) As described in Sect. 2, we classify each pixel along the contour of each connected component as either *straight*, a *right turn*, or a *left turn* (refer again to Fig. 3), then we count the number of pixels falling into each type of turn and consecutive turn category (the quantities N_i^S, N_i^L, N_i^R, N_i^{SS}, N_i^{LS}, N_i^{RS}, N_i^{SL}, N_i^{LL}, N_i^{RL}, N_i^{SR}, N_i^{LR}, and N_i^{RR} introduced in Sect. 2). The result of the process is a repository of ground truth files in YAML format, for example as shown in Fig. 7.

```
%YAML:1.0
file: "2"
collectDate: "Wed Jul  8 14:55:30 2015\n"
contours:
   - { type:Noise, xmin:177, ymin:233, xmax:177, ymax:233, leftcount:1,
       rightcount:0, straightcount:0, LR:1, LL:0, LS:0, RL:0, RR:0, RS:0,
       SL:0, SR:0, SS:0, total:1 }
   - { type:Noise, xmin:190, ymin:253, xmax:190, ymax:253, leftcount:1,
       rightcount:0, straightcount:0, LR:1, LL:0, LS:0, RL:0, RR:0, RS:0,
       SL:0, SR:0, SS:0, total:1 }
   - { type:Nontext, xmin:389, ymin:101, xmax:430, ymax:204, leftcount:75,
       rightcount:72, straightcount:150, LR:7, LL:36, LS:32, RL:37, RR:2,
       RS:33, SL:31, SR:34, SS:85, total:297,
       GT:"/home/anita/Desktop/Thesis program/outputfile/2/2_contour91.bmp" }
   - { type:Noise, xmin:463, ymin:520, xmax:469, ymax:521, leftcount:4,
       rightcount:2, straightcount:7, LR:1, LL:2, LS:1, RL:2, RR:0, RS:0,
       SL:1, SR:0, SS:6, total:13 }
```

Fig. 7. Example ground truth data file.

At analysis time, we use the contour and turn counts for classification. To evaluate the two classifiers, we split the 120 images into five partitions for leave-one-out five-fold cross validation. For each of the five splits of the data into test

and train sets, we used the training set to find the best threshold for the turn ratio classifier, and we calculated the probabilities $P(S_{i,j} \mid S_{i,j-1}, \text{text})$ over all text contours in the training set and the probabilities $P(S_{i,j} \mid S_{i,j-1}, \text{non-text})$ over all non-text contours in the training set. For example, for the probabilities $P(S_{i,j} \mid S_{i,j-1} = \text{S}, \text{text})$ we simply calculate the ratios N_t^{SS}/N_t^S, N_t^{LS}/N_t^S and N_t^{RS}/N_t^S, where the counts are taken over all text contours in the training set. These probabilities are then used to evalute, for each contour i in the test set, $P(\text{text} \mid S_i)$ and $P(\text{non-text} \mid S_i)$. Repeating for all five splits, we obtain test results over the entire 120-image data set.

4 Experimental Results

After collecting and processing the data as described in the previous section, we obtained the following results for turn ratio classification and maximum a posteriori classification.

4.1 Experiment 1 (Turn Ratio Classification)

The results of turn ratio classification from five-fold cross validation are shown in Table 1.

Table 1. Experiment 1 results. Accuracy of turn ratio classification for training set and test set, for each partition of the data for five-fold cross validation, on a per-pixel and per-contour basis. Column 2 shows the turn ratio threshold that is optimal for per-pixel classification on that partition's training set.

Cross validation partition	Optimal ratio	Train set accuracy per pixel	Test set accuracy per pixel	Train set accuracy per contour	Test set accuracy per contour
1	0.779	83.68 %	86.41 %	71.05 %	71.14 %
2	0.778	84.42 %	82.92 %	71.83 %	67.87 %
3	0.784	83.61 %	86.04 %	70.06 %	73.91 %
4	0.775	84.28 %	83.42 %	71.30 %	71.11 %
5	0.779	85.03 %	81.42 %	70.98 %	71.35 %
Average	0.779	84.204 %	84.042 %	71.044 %	71.076 %
S.D	0.003	0.5218	1.904	0.575	1.918

The turn ratio method is surprisingly accurate. Similar results on the training and test sets indicate that the data set is sufficiently large to provide a stable estimate of the single turn ratio threshold parameter.

4.2 Experiment 2 (Probabilistic Classification)

The results of probabilistic classification from five-fold cross validation are shown in Table 2.

Again surprisingly, the per-pixel accuracy of the MAP classifier is approximately 1.4 % less accurate than the turn ratio classifier. The turn ratio classifier was tuned to obtain the best possible per-pixel accuracy on the training set, whereas the probabilistic classifier is trained to provide an estimate of $P(\text{text} \mid S_i)$ on a per-contour basis. Since the probabilistic method is approximately 1.2 % more accurate than the turn ratio threshold classifier on a per-contour basis, optimizing the threshold on $P(\text{text} \mid S_i)$ on a per-pixel basis would improve results. This would roughly correspond to adjusting the Bayes risk for each contour based on the size of the contour. When we perform this small optimization, we obtain an average test set per-pixel accuracy of 85.6 %.

Table 2. Experiment 2 results. Accuracy of probabilistic classification for training set and test set, for each partition of the data for five-fold cross validation, on a per-pixel and per-contour basis.

Cross validation partition	Train set accuracy by pixels	Test set accuracy by pixels	Train set accuracy by contours	Test set accuracy by contours
1	82.05 %	84.87 %	72.44 %	74.28 %
2	83.38 %	81.31 %	72.66 %	67.68 %
3	82.19 %	84.04 %	72.12 %	74.27 %
4	82.59 %	82.73 %	72.99 %	71.50 %
5	83.65 %	80.23 %	71.99 %	73.89 %
Average	82.772 %	82.636 %	72.44 %	72.324 %
S.D	0.638	1.704	0.362	2.542

4.3 Discussion

The BPMN diagram data set generally contains many short text contours and fewer long non-text contours. The proportion of text pixels overall is approximately 30.16 %, and the proportion of non-text pixels is approximately 69.84 %, whereas the proportion of text contours is 81.48 %, while the proportion of non-text contours is 18.52 %. Turn ratio classification yields an accuracy of 84.0 %, whereas the probabilistic classifier yields an accuracy of 82.6 % or 85.6 % depending on its objective function. Looking more closely at these results, we found that the two classifiers agreed about 94 % of the time, with 81 % of the contours overall classified correctly by both methods and 13 % incorrectly classified by both methods.

Figure 8 shows examples of contours that are correct classified by both methods and examples of contours on which the two classifiers disagree. Non-text contours with long straight lines or smooth curves are generally classified correctly,

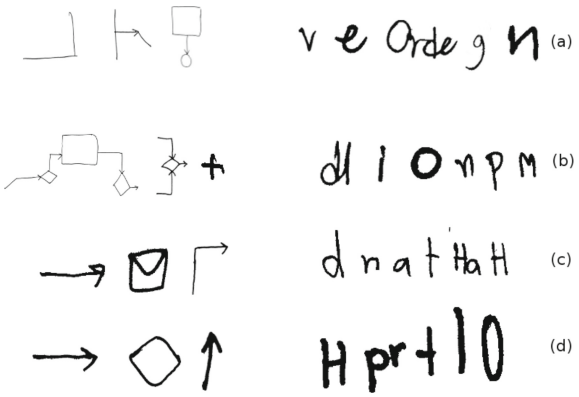

Fig. 8. Example contours classification example. (a) Contours that are correctly classified by both methods. (b) Contours that correctly classified by the turn ratio classifier but incorrectly classified by the probabilistic classifier. (c) Contours that are correctly classified by the probabilistic classifier but incorrectly classified by the turn ratio classifier. (d) Contours that are incorrectly classified by both methods.

and non-text contours forming circles or shorter straight lines with turns are likely to be classified as text. Overall, non-text contours contain longer straight lines than do text contours, so the probability of moving to the S state of the Markov chain are higher in the non-text model than in the text model. While this works well overall, it does mean that text contours with longer, straighter segments, such as *l*, *d*, *p*, and *H*, are likely to be classified as non-text contours.

The data set was collected in Thailand from students who may write English letters differently than native English speakers, so model parameters may need re-estimation for different data sets. Binarization might also require adaptation to different scanners or drawing styles.

5 Conclusion

This paper has presented a method for text and non-text segmentation of static hand-written drawings that works without requiring stroke information. Although the methods are relatively simple, the results on BPMN diagrams are quite accurate, in the 82 % to 86 % range. The method could be improved with a richer feature set and a binary classifier such as the support vector machine (SVM) (Burges 1998; Niu and Suen 2012; Yang et al. 2013). The segmenter described here will make an excellent starting point for a complete diagram digitization system. The fact that such accurate results can be obtained for BPMN diagrams, which describe executable business processes, raises the tantalizing prospect of bootstrapping business automation software development processes with a simple drawing.

References

Burges, C.J.C.: A tutorial on support vector machines for pattern recognition. Data Min. Knowl. Disc. **2**, 121–167 (1998)

Lemaitre, A., Carton, C., Couasnon, B.: Fusion of statistical and structural information for flowchart recognition. In: 2013 12th International Conference on Document Analysis and Recognition, pp. 1210–1214. IEEE (2013)

Costagliola, G., Deufemia, V., Risi, M.: A multi-layer parsing strategy for on-line recognition of hand-drawn diagrams. In: IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2006, pp. 103–110. IEEE Computer Society, September 2006

Freitas, C.O.A., Oliveira, L.S., Bortolozzi, F., Aires, S.B.K.: Handwritten character recognition using nonsymmetrical perceptual zoning. Int. J. Pattern Recogn. Artif. Intell. **21**(01), 135–155 (2007)

Hammond, T., Davis, R.: Tahuti: a geometrical sketch recognition system for UML class diagrams. In: ACM SIGGRAPH 2006 Courses, SIGGRAPH 2006. ACM (2006)

Hammond, T., Davis, R.: Ladder, a sketching language for user interface developers. In: ACM SIGGRAPH 2007 Courses, SIGGRAPH 2007. ACM (2007)

Kara, L.B., Stahovich, T.F.: Hierarchical parsing and recognition of hand-sketched diagrams. In: ACM SIGGRAPH (2007)

Lauer, F., Suen, C.Y., Bloch, G.: A trainable feature extractor for handwritten digit recognition. Pattern Recogn. **40**(6), 1816–1824 (2007)

Bresler, M., Prua, D., Hlavác, V.: Modeling flowchart structure recognition as a max-sum problem. In: 2013 12th International Conference on Document Analysis and Recognition, pp. 1215–1219. IEEE (2013)

Niu, X.-X., Suen, C.Y.: A novel hybrid CNN-SVM classifier for recognizing handwritten digits. Pattern Recogn. **45**(4), 1318–1325 (2012)

OpenCV Dev Team. Finding contours in your image (2016). http://docs.opencv.org/2.4/doc/tutorials/imgproc/shapedescriptors/find_contours/find_contours.html

Otsu, N.: A threshold selection method from gray-level histograms. IEEE Trans. Syst. Man Cybern. **9**(1), 62–66 (1979)

Stahovich, T.F.: Segmentation of pen strokes using pen speed. In: Proceedings of the AAAI Fall Symposium on Making Pen-Based Interaction Intelligent and Natural, pp. 21–24 (2004)

Suzuki, S., Abe, K.: Topological structural analysis of digitized binary images by border following. Comput. Vis. Graph. Image Process. **30**(1), 32–46 (1985)

Waranusast, R., Haddawy, P., Dailey, M.: Segmentation of text and non-text in on-line handwritten patient record based on spatio-temporal analysis. In: Combi, C., Shahar, Y., Abu-Hanna, A. (eds.) AIME 2009. LNCS, vol. 5651, pp. 345–354. Springer, Heidelberg (2009)

Wu, J., Wang, C., Zhang, L., Rui, Y.: Offline sketch parsing via shapeness estimation. In: Proceedings of the 24th International Conference on Artificial Intelligence, IJCAI 2015, pp. 1200–1206. AAAI Press (2015)

Yang, X., Qiaozhen, Y., He, L., Guo, T.: The one-against-all partition based binary tree support vector machine algorithms for multi-class classification. Neurocomputing **113**, 1–7 (2013)

Zhong, C., Ding, Y., Fu, J.: Handwritten character recognition based on 13-point feature of skeleton and self-organizing competition network. In: Proceedings of the 2010 International Conference on Intelligent Computation Technology and Automation, ICICTA 2010, vol. 02, pp. 414–417. IEEE Computer Society, Washington, D.C. (2010)