A Novel Method for Flash Character Localization

Jie Liu, Shuwu Zhang, Heping Li, Wuyi Yang
Institute of Automation, Chinese Academy of Sciences, Beijing, China
{jliu, swzhang, hpli, wyyang}@hitic.ia.ac.cn

Abstract

It has been a challenge to locate character in flash due to the variations of character in size, color and style. Besides, multilingual text in flash brings more difficulties to character localization. In this paper, a novel character localization method based on connected component analysis is proposed for locating character in flash. The proposed method first clusters the color to separate the color image into homogeneous color layers. Then a heuristic connected component merging algorithm is performed to merge some connected components which satisfy the characteristic of character in each color layer on the premise of non-merging mistakes. Next, for effective character localization, the features of characters are extracted by connected component clustering, and then most characters will be located according to these features. Finally, a new noise removal method is performed to remove all non-characters connected components, and then all characters are located. The experimental results show that the proposed method can locate flash character effectively.

1. Introduction

Flash has already grabbed everyone's attention as the hot thing in Web Ads. There is a significant need to extract and analyse the text in flash, for Web Ads monitoring and retrieval. In flash on Chinese Internet, there are not only many Chinese characters but also some English characters and Arabic numbers. However, locating multilingual text is still difficult. In addition, the variations of character in size, color, and style make the problem of automatic flash character localization extremely challenging.

Currently, there are several related works on character localization [1-6]. These methods can be broadly classified into two classes: texture-based method and connected-component-based method. The first one usually uses texture analysis algorithms such as Gabor filtering [1], spatial variance [2] or wavelet transform [3] to locate text regions. The second one [4-6] usually assumes that text is represented with a uniform color. Therefore, they first quantize the color space of the input image into color bins by a clustering procedure, and then they analyze the connected components and extract characters for each color class. These methods most hold assumptions on invariance of character size, color, font style, and language. Therefore they are not suitable for flash text.

The width, height and aspect ratio of noises vary irregularly while those of characters are relatively stable. If some characters have been located, these features of characters would be extracted by connected components clustering and other characters would be located according these features. Based on this idea, a new connected-component-based character locating method is proposed in this paper. First, color clustering based on mean-shift algorithm is performed to separate the color image into homogeneous color layers. Second, a novel two-phase character localization method extracts reliable features of characters and locates most characters according to these features in each color layer, which is composed of heuristic connected component merging algorithm (HCCM) and connected component merging algorithm based on adaptive feature (CCMAF). Finally, a new strategy based on histogram of stroke width is introduced for noise removal, and then all characters are located. The proposed method is capable of locating the character in flash, and is robust for variations of character in size, color and style, as well as multilingual text. The experiment verifies the effectivity of the algorithm.

The rest of this paper is organized as follows: In section 2, the detail of the method is described. In section 3, experimental results are presented and discussed. Finally, we draw conclusions in section 4.

2. Detail of the algorithm

2.1. Color clustering

Assuming character strokes of the same text have similar color, the characters can be detected in each color plane. Therefore, a color clustering method is first performed to decompose a color image into several binary planes. Each binary plane represents an individual color cluster. In this paper, we apply the mean shift method for color clustering [7].

2.2. Heuristic connected component merging algorithm (HCCM)

In each color plane, connected component analysis is performed to generate bounding boxes of the connected component (CC). Then the HCCM algorithm merges some connected components (CCs) which satisfy the characteristic of character. This phase generates sufficient characters in order that connected component merging algorithm based on adaptive feature (CCMAF) can cluster CCs and extract features of characters.

HCCM algorithm correctly merges close enough CCs according to the characteristic of Chinese characters and ensures that English characters and Arabic numbers are not falsely merged with other CCs.

As we known, the aspect ratio of Chinese characters is nearly equal to 1. Taking the italic characters into account, the width and height of Chinese character should meet formula (1):

$$\frac{width}{height} < 1.2 \quad \text{and} \quad \frac{height}{width} < 1.2 \tag{1}$$

Algorithm description:

Initializing a connected component list (CC-list). For each CC, run *Step 1* to *Step 2* infinitely until the CC number of CC-list remains unchanged.

Step 1: Choosing a CC *i* from CC-list;

Step 2: Searching next CC *j* in CC-list, if the two CCs satisfy any one of the follow conditions, their bounding boxes are removed from CC-list and merged to produce a new bounding box surrounding them, then the new CC will be inserted into CC-list;

(a) $$\frac{Area(i \cap j)}{\min(Area(i), Area(j))} > 0.5 \tag{2}$$

where *Area(i)* and *Area(j)* are the areas of the two bounding boxes of CC *i* and CC *j* respectively. $\min(Area(i), Area(j))$ is of the smaller of them, and $Area(i \cap j)$ is their overlapping area.

(b) $\frac{W(i \cup j)}{H(i \cup j)} < 1.2$ and $\frac{H(i \cup j)}{W(i \cup j)} < 1.2$ and $Dis(i, j) < T_{dis}$ (3)

where $W(i \cup j)$ and $H(i \cup j)$ are width and height of the new bigger bounding box surrounding CC *i* and CC *j*, respectively. *Dis(i,j)* is the distance between their centers, and T_{dis} is a adaptive threshold which is proportional to the minimum value of width and height of CC *i* and CC *j*. It is defined as:

$$T_{dis} = \min(W(i), H(i), W(j), H(j)) \times a \tag{4}$$

where *a* is a coefficient, and it is set at 2.3 in my experiment.

2.3. Connected component merging algorithm based on adaptive feather (CCMAF)

After HCCM, all CCs can be classified as the following classes: Chinese characters, strokes of Chinese characters, English characters, Arabic numbers and noises. Then we cluster CCs according to the feature which are the ratio of width to height, width and height. The features of noises vary irregularly while the features of characters are relatively stable. Thus, if the CCs belong to a cluster are adequate, these CCs will be supposed as characters and this cluster will be regarded as a significant cluster. Then other potential characters can be located according to the features of these significant clusters. For locating narrow characters, such as "—", we need to do some special treatments. If a narrow CC cannot compose a character with other CCs while it satisfies the width of a significant feature, it will be accepted as narrow character. Similarly, the algorithm can locate slim characters, such as "1" or "I". The overall process can be demonstrated as follow steps:

Algorithm description:

Definition:

$W(i)$ and $H(i)$ are defined as width and height of the bounding box of CC *i* respectively, while FW and FH are width and height of the bounding box of the current feature respectively. $WHRatio(\bigcup_{i \in PC-list} CC(i))$ denotes the ratio of width to height of bounding box surrounding all CCs in PC-list and *FWHRatio* denotes the ratio of width to height of current feature. k_1 and k_2 both denote degree of closeness. $1 \le k_1 \le 1.1$, $0 \le k_4 < 0.2$.

Initialization:

Step 1: Initializing a connected component list (CC-list).

Step 2: After feature clustering of CCs, if CCs in a cluster are more than a threshold *T*, the cluster will be regarded as a significant cluster. In our experiment, *T* is proportional to the number of CCs. Then the features of all significant clusters will be extracted. The feature is defined as: width of the bounding box of CC, height of the bounding box of CC and the ratio of width to height of the bounding box of CC.

Step 3: Removing all CCs satisfied the features of all significant clusters from CC-list.

Step 4: These features will be sorted in order of the ratio of width to height, from small to high. Then with reference to each feature, the potential characters can be located by the flowing steps:

Main:

Step 1: Choosing a feature; set the point of origin: left-top point of the image.

Step 2: If CC-list is not empty, a CC *i* is chosen from CC- list, which is nearest to the point of origin, then the process will move to *step 3*. Otherwise the process will move to *step 7*.

Step 3: If the follow condition (5) is satisfied, the CC *i* will be inserted into a temp potential character list (TPC-list), then the point of origin will be reset at the center of CC *i*, then the process will move to *step 4*. Otherwise the CC *i* will be erased from CC-list and inserted into a remaining list and the process will move to *step 2*;

$$\frac{W(i)}{FW} < k_1 \text{ and } \frac{H(i)}{FH} < k_1 \tag{5}$$

Step 4: If the follow condition (6) is satisfied:

$$\frac{W(\bigcup_{j \in TPC-list} CC(j))}{FW} < k_1 \text{ and } \frac{H(\bigcup_{j \in TPC-list} CC(j))}{FH} < k_1 \tag{6}$$

then the CC *i* will be registered as a part of a whole potential character and be inserted into a potential character list(PC-list) and be erased from CC-list, then the process will move to *step 2*. Otherwise, the process will move to *Step 5*.

Step 5: if the follow condition (7) is satisfied:

$$\left| WHRatio(\bigcup_{i \in PC-list} CC(i)) - FWHRatio \right| < k_2 \tag{7}$$

then all CCs in PC-list will be merged to produce a new bounding box and the new CC will be inserted into the character list, then TPC-list and PC-list will be cleared and the process will move to *step 2*. Otherwise, the process will move to *Step 6*.

***Step 6*:** If PC-list is empty, the process will move to *Step 2*. Otherwise, the first CC *j* will be chosen and removed from PC-list and TPC-list. If it satisfies the follow condition (8), it will be inserted into the character list, or else it will be inserted into the remaining list. Finally, the process will move to *Step 4*.

$$\frac{W(j)}{H(j)} < T_{s\lim} \text{ or } \frac{W(j)}{H(j)} > T_{narrow} \tag{8}$$

where $T_{s\lim}$ and T_{narrow} denote the thresholds that a character can be regarded as a slim or narrow character, respectively. In our experiment, $T_{s\lim} = 0.4$, $T_{narrow} = 4$.

Step 7: If the condition (7) is satisfied, all CCs in PC-list will be merged to produce a new bounding box and the new CC will be inserted into the character list, then TPC-list and PC-list will be cleared, then the process will move to *Step 1*. Otherwise the process will move to *Step 8*.

Step 8: If PC-list is empty, the process will move to *Step 9*. Otherwise, the first CC *j* will be chosen and removed from PC-list. And then if the condition (8) is satisfied for CC *j*, it will be inserted into the character list, or else, it will be inserted into the remaining list. Finally, the process will return to *Step 8*.

Step 9: All CCs in the remaining list are removed and combined to CC-list, then the process move to *step 1*.

2.4. Noise removal based on histogram of stroke width

In the process of CCMAF, all Chinese characters and some non-Chinese characters are located. Then the reminders of connected components can be English characters, Arabic numerals or noises. Therefore, we merely remove all non-character connected components. Then all characters are located.

In general, the width of character stroke is rough stable, while that of noise is not stable. According to this characteristic, a novel character verification method is proposed.

For each CC, the run length in vertical direction of the stroke is used to build the vertical run length histogram. Similarity, the horizontal run length histogram is built. Then we cluster the frequency of vertical run length and horizontal run length respectively.

A CC will be regarded as a character, if it satisfies the two conditions:

(a) The frequencies of the peaks of both histograms for a CC are adequate;

(b) The width of the peak of the vertical run length histogram is not much greater or less than that of the peak of the horizontal run length histogram.

3. Experimental results

The proposed method is designed primarily to locate texts in flash, these flashes are downloaded from: http://www.sohu.com and http://www.sina.com.cn. Characters in flash usually vary in size, color and style. Besides, there usually are multilingual texts in flashes. Our database contains 206 test images which are extracted from these flashes. There are 1724 characters in these images, including Chinese characters, English characters and Arabic numbers. In Figure 1 (a) ~ (d) , we compare our approach with Wang's work[5].

(a) (b) (c) (d)

Figure 1. Comparisons with Wang's work. Top: our results. Bottom: results by Wang's work

In Figure 1(a), there are some small English characters and Arabic numbers besides Chinese characters. Most of characters can be located by our method due to the ability of handling multilingual text. Although the size of characters varies in Figure 1(a) and Figure 1(b), good results are still achieved since our algorithm is robust for variation of character size. Figure 1(c) includes Chinese characters, Arabic numbers and punctuations. These punctuations do not affect the proposed method and correct locations of all characters are found. In addition, in Figure 1(c) and Figure 1(d), it should be noted that our method can distinguish between the dash and Chinese character "一". Two dashes are discarded since their widths do not satisfy any features of significant CC clusters.

In table 1, we use recall and precision to compare the performance of character location with Wang's work[5]. Recall is the percentage of the correct locations in all labeled examples. Precision is the percentage of the correct character locations in the all detected locations. The comparison indicates that our method significantly outperforms Wang's work for the images from flash.

Table 1. Performance comparison for character location

	Recall (%)	Precision (%)
Our method	82.36	93.35
Wang's method	53.36	67.55

4. Conclusions

In this paper, a novel method is proposed to locate flash character. Our contributions mainly lie in two-phase character localization algorithm: Guaranteeing no merging-error, HCCM is first used to merge some CCs, which satisfy the characteristic of character, as much as possible. Then, all CCs are clustered and features of characters are extracted, and then most characters are located according to these features. In addition, we also present a novel noise removal method to filter out all non-character CCs. The experimental results show that the proposed method is robust for the variations of character in size, style and color. Besides, it can also handle multilingual text.

References

[1] Jain A, Bhattacharjee S. Text segmentation using Gabor filters for automatic document processing. *Machine Vision and Applications*,5(3):169-84, 1992.

[2] Wu V, Manmatha R, Riseman EM. TextFinder: An automatic system to detect and recognize text in images. *Ieee Transactions on Pattern Analysis and Machine Intelligence*,21(11):1224-9, 1999.

[3] Li HP, Doermann D, Kia O. Automatic text detection and tracking in digital video. *Ieee Transactions on Image Processing*,9(1):147-56, 2000.

[4] Jain AK, Yu B. Automatic text location in images and video frames. *Pattern Recognition*,31(12):2055-76, 1998.

[5] Wang K, Kangas JA. Character location in scene images from digital camera. *Pattern Recognition*,36(10):2287-99, 2003.

[6] Mariano VY, Kasturi R. Locating uniform-colored text in video frames. *15th International Conference on Pattern Recognition, Vol 4, Proceedings - Applications, Robotics Systems and Architectures*:539-42, 2000.

[7] Comaniciu D, Meer P. Robust analysis of feature spaces: color image segmentation. *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*:750, 1997.