LexisNexis® Academic

Print Request: Current Document: 21
Time Of Request: Friday, February 09, 2018 02:17:01 EST
Send To:

MEGADEAL, ACADEMIC UNIVERSE
MASTER'S COLLEGE
LIBRARY
SAN JOSE, CA 00000

Terms: (autonomous vehicle)

Source: IET Computer Vision
Project ID:

21 of 31 DOCUMENTS

IET Computer Vision

August 2015

Text detection and recognition in natural scene with edge analysis

BYLINE: ChongYu; YonghongSongsongyh@mail.xjtu.edu.cn; QuanMeng; YuanlinZhang; YangLiu

SECTION: Pg. 603 - 613 Vol. 9 No. 4 1751-9632

LENGTH: 6080 words

1

Introduction

With the great popularity of digital devices like smartphones, text detection in natural scenes is receiving increasing attention and finds many applications in various fields, including visual impairment assistance, tourist assistance, content-based image retrieval and unmanned ground **vehicle** navigation. However, detecting and recognising text from a complex background is a very difficult problem, because of the variations of scale, font, colour, lighting and shadow [1, 2]. In recent years, many approaches [3, 4] have been proposed and achieved promising results. The detection methods can be classified into two categories: texture-based methods and region-based methods.

Texture-based methods [5, 6] are based on the observation that text in images is usually different in texture from the background. In these methods, the image is generally scanned in multi-scales using a sliding window. Then, texture analysis is applied, such as discrete cosine transform, Fourier transform, wavelet coefficients, spatial variance and Gabor filters are used to obtain texture information.

Region-based methods [7, 8] are mainly bottom-up approaches. In these methods, pixels exhibiting similarities in colour and intensity are grouped as connected components (CCs) firstly. Then, non-text CCs are filtered out from the candidate character components using CC analysis.

In this paper, we propose a text detection method based on two steps of edge analysis: candidate edge combination and edge classification. We observed that in a complex background situation, the text edge is often connected with the background edge and that a character usually has several edges. On the basis of this observation, we propose to use the idea of over-segmentation and region merging to obtain candidate edges. Firstly, the whole edge is divided into small edge segments. It is an undeniable fact that the text is different from other objects in two aspects: its nearly constant stroke width and constant colour. Therefore, neighbouring edge segments are merged if they have a similar stroke width and colour. Through the candidate edge combination step, each character is described by one edge segment set, and we call these sets as candidate boundaries. In order to filter out non-text boundaries from the background, candidate boundaries are merged into chains firstly. Then, character-based features and chain-based features are extracted, followed by classification based on these features. After texts are detected, an end-to-end text recognition method is also proposed. The method first uses the edge recombination step to obtain the candidate boundaries. Then the candidate boundaries are merged into chains and grey image patches are extracted based on the location information of each boundary. A classifier and histogram of gradient (HOG) feature are used to recognise each grey image patch. Fig. 1 is the overview of the proposed framework. Colour Figures for this paper are available online.

Fig. 1 Overview of the proposed framework of text detection and recognition
a Source image
b Segmentation points detection result
Each segmentation point is described by the asterisk
c Edge merging: one character is described by one edge segment set
d Text chain aggregation, and each chain is described by one colour in the map
e Edge classification result
f Detection result

g Source image
h Candidate text regions, marked by the rectangle
i Recognition result

2

Related work

To detect text in natural scene and video frames, the information of edges has been used in many published methods [9-11]. The method proposed by Lee *et al.* [12] is a region-based method, which uses edges as constraint terms in the image segmentation step. Firstly, a confidence map is constructed using text localise to estimate the probability of a pixel being text. Then, a modified K-means clustering algorithm is adopted to find dominant colours in the computed candidate text regions. Finally, an edge constraint is utilised to force the group of text colours and background colours into different clusters.

Epshtein *et al.* [10] use a local image operator, namely stroke width transform (SWT), which transfers an input image into a grey-scale map. The value of a pixel indicates the estimated width of stroke. Then the output of the SWT is grouped into letter candidates and CC analysis methods are used to filter out background CCs. The method uses the edge of the SWT computation stage, and uses the idea of parallel edges. However, the method is sensitive to noise and blurry images, since it depends on a successful edge detection. In addition, the method transfers SWT into CCs, which may decrease the precision of detection.

Another algorithm that uses the idea of parallel edges is presented in [11]. The method considers a closed boundary as a character candidate, and the unclosed boundaries are regarded as non-text boundaries, which are erased from the edge map directly. However, because of the complex background, some character edges may connect to background edges and become unclosed.

The proposed detection method improves upon Epstein's method in several aspects. First, to overcome the sensitivity of noise and blur of SWT, we use edges to detect text directly instead of transferring the edge into CCs and detecting text based on CC analysis. Second, to calculate more precise stroke width, we create different definitions of stroke width to Epstein's work. In our method, we compute the width per pixel of the most likely stroke containing the pixel. Moreover, in our method, we set stroke width as the attribute of edge segments, rather than attribute of pixels. In order to obtain the precise stroke width, we use an energy function to search the corresponding point of each edge pixel based on the gradient image, and then we use the stroke width of the whole edge to modify the result of corresponding point. Through the two-step search, we can find a reliable corresponding point, despite the noise and blurry of images.

Our method is also different from the method proposed by Zhang, which regards the character as closed boundaries. This could have trouble because many character edges may be unclosed when adhered to background edges. Inspired by the work of over-segmentation and region merging, we proposed an edge analysis method which divides character edges into small segments and then merges them based on their spatial relationship, similarities of colours and stroke width. Through these steps, one character can be described by an edge segment set, and then regarded as a unit for analysis.

As for text recognition, Wang *et al.* [13] proposed an end-to-end text recognition system. They use a sliding window classification and HOG feature to detect characters of an image. Then they perform word detection using a Pictorial Structures framework, treating the characters as parts of a word. Finally, detected words are re-scored using features based on the global layout and determined by non-maximal suppression.

Our recognition method uses edge recombination and candidate boundary links to obtain the location of each candidate character, instead of using a sliding window. Through this step we only need to compute limited candidate windows. In the recognition step, the lexicon is not used, which means our method can deal with more general scenes.

3

Candidate edge recognition

In order to describe a character with edges, we propose a candidate edge recombination method. We first separate the whole edge into edge segments and then merge these edge segments again based on their colour and stroke attributes. After this step, character edges can be separated from background edges, even in the situations of complex background, like shadow and reflecting light. The details of our method follow.

3.1

Edge segmentation

The text consists of strokes, and a stroke is composed by two parallel edges. Thus, the first step of our algorithm is edge extraction using the Canny edge operator [14]. A salient problem is that text edges could be connected with surrounding backgrounds, especially in the images with complex backgrounds.

By observation, we found that the edge can be separated by two kinds of points: the intersections of two edges and the points where edge directions are very different in a local neighbourhood. The latter is precisely the definition of corner point. Therefore, we use the Harris corner detection method to find the corner points and then the edge is segmented into small edge segments by intersections and corner points. These edge segments are similar with superpixels in over-segmentation. Fig. 2 is an example of edge segmentation.

Fig. 2 Example of edge segmentation
a Original image
In order to show the result in details, we only show the part marked by the rectangle
b Edge image
c Segmentation points marked by the asterisk
d Segmentation result
One colour means one edge segment

3.2

Edge merging

This step is the most important step. Firstly, we build a neighbourhood map based on the spatial relationships of edge segments. As we know, text usually has similar colour and stroke width. Thus, colour and stroke width of each edge segment are used to determine whether two neighbour edge segments should be merged. After this step, each character is described by an edge segment set, namely character boundary. The following describes the merging strategy in detail.

In order to merge two neighbour edge segments, we first compute the similarity between each neighbour edge segment using the following formula (1)

$$s = \begin{cases} \exp\left(-\dfrac{\mathrm{abs}(\mathrm{sw}_1 - \mathrm{sw}_2)}{\mathrm{sw}_1 + \mathrm{sw}_2}\right) - \mathrm{abs}(\mathrm{colour}_1 - \mathrm{colour}_2) & \text{if } v > \lambda \\ \exp(-\alpha \cdot \mathrm{abs}(\mathrm{colour}_1 - \mathrm{colour}_2)) & \text{otherwise} \end{cases}$$

where sw_1 and sw_2 are the stroke width of two neighbour edge segments, and colour_1 and colour_2 are the colour. We will describe them in the following section. *v* can be calculated in (4) and *[#x3bb]* is a threshold to measure whether the stroke width of the two edge segments is credible. In our experiment, we found *[#x3bb]* set to 0.8 get the best performance. Then, a greedy hierarchical clustering method is adopted to merge the neighbour edge segments. In edge segments clustering, the edge segment pair with the largest similarity calculated by (1) is first merged as a new segment. Then, the new segment takes the place of the edge segment pair and makes the next clustering step with the remaining segments. The clustering operation ends when none of the edge segment pairs should be merged (i.e. the similarity is below a given threshold).

However, some characters are composed by many edge segments, and there is no neighbour relationship between them. Thus, after edge merging based on neighbour map, we need to determinate whether two segments are in the relationship of surrounding, such as character A in Fig. 3*b*.

Fig. 3 Example of edge merging
a Edge segments before merging
Each colour means one segment
b First merging result
We can see that the character 'a' is described by two edges, because they do not have neighbour relationship
c Second merging result

By analysis, we find that most corresponding points of the inner edge segment are projected on the outline edge segment. The colour difference between the inner edge segment and the outline edge segment is very small. Therefore, we use this information to judge whether two edge segments should be merged if they are in the relationship of surrounding. An example is shown in Fig. 3*c*.

3.3

Stroke width

As we know, one important attribute of text is that a character is composed by two parallel edges and characters in a text line have similar stroke width. We first define the corresponding point of an edge point as a point which appears on the other parallel edge and the gradient orientation difference between these two points is approximate to *[#x3c0]*. The distance between the two points can be used to measure the stroke width of the character.

To compute the corresponding point of each edge point, we use a two-step search method based on energy functions in gradient direction ray. Using the energy function to compute the corresponding point instead of the first valid pixel on the search ray [10], we can avoid taking background edges between strokes as corresponding points (Fig. 4). In addition, because of the light reflection, some corresponding points do not exist. In such situations, we discard unreliable point with another function, (4), even if it is found on the search ray. Thus, the proposed method can overcome the effect of background cluster and light reflection.

Fig. 4 Example of corresponding point detection
a Edge image
We compute the corresponding points of the edge points marked by red colour
b First detection result
corresponding points are marked by green colour and the corresponding relationships are marked by blue lines
c Modified result

The first step is similar to the method proposed by Epshtein *et al.* [10]. For each edge point, we search along the gradient direction and each edge point in the ray is recorded. The energy of these recorded points is computed by the following formula (2)

$$\mathbf{Energy} = \exp(-\theta_d) \cdot \exp(-\mathbf{Grad}_d) \cdot \exp(-t)$$

where $[\#x3b8]_d$ = abs(abs($[\#x3b8]_p$-$[\#x3b8]_q$) -*[#x3c0]*), which is used to measure whether the gradient orientation difference between the original edge point $[\#x3b8]_p$ and the candidate corresponding points $[\#x3b8]_q$ is approximate to *[#x3c0]*. The second term is used to measure the gradient magnitude difference. The last term is a constraint term. If there are two parallel edges in the ray direction, we select the point that appears in the first one as corresponding point and t is related to the number of edges between the original point p and the candidate point q. Then, the point which has the maximum energy value is selected as the candidate corresponding point.

However, there may be some excess edges in the middle of two parallel edges, which are usually caused by illumination. The excess edge may be parallel to the stroke edge. To eliminate the effect of this situation, we regard an edge segment as a unit, and compute the average stroke width. As a character edge segment should have similar stroke width, the expected position of each corresponding point should be in the distance of average stroke width. We use this information to modify the corresponding point. The energy function is described as below (3)

$$\mathbf{Energy} = \exp(-\theta_d) \cdot \exp(-\mathbf{Grad}_d) \cdot \exp\left(-\frac{(S_{\mathrm{w}} - \mu_{\mathrm{sw}})^2}{\mu_{\mathrm{sw}}}\right)$$

where the first two terms are the same as (1) and the last term is also a constraint term that supposes that the corresponding point is in the position of mean stroke width.

S_{w} is the stroke width of the edge point and $[\#x3bc]_{\mathrm{sw}}$ is the average stroke width of the edge segment. Fig. 4 is an example of corresponding point detection. We use corresponding points to compute the stroke width of an edge segment. However, the edge segments in the corner do not have enough corresponding points, so the stroke widths computed based on the corresponding points are unreliable. Some backgrounds do not have the attribute of parallel edges. The stroke width of these edges is also unreliable. Thus, the first step is to judge whether a stroke width of an edge is reliable using the following formula (4)

$$w_i = \exp\left(-\frac{1}{2\mathbf{sigm}^2}(d_i - d_m)^2\right) \quad w_i = \frac{w_i}{\sum_{i=1}^{N} w_i}$$

$$\mathbf{distr} = \sum_{i=1}^{N} (w_i(d_i - d_m))/d_m$$

$$v = \exp(-\mathbf{distr}/2) \cdot (1 - (\mathbf{occupy} - 1)^2)$$

where distr is used to measure the distribution of stroke width of an edge segment. We know that the text has similar stroke width, thus the smaller distr is, the more credible the stroke width. d_i is the distance between the edge point i and its corresponding point, and d_m is the mean distance between edge points and their corresponding points in edge segment. sigm is used to revise the weight factor w_i. In

our experiment, we set sigm to the median value of d_i. occupy is used to measure the percentage of edge points that have a corresponding point. As the stroke width is not believable in the corner position described in [10], a weight factor w_i is selected to eliminate the effect of this outlier stroke width, which is inspired by the work of [15]. When a stroke width is far from the mean stroke width, it has a lesser contribution to the final result. v is the value to measure whether this stroke width is believable. The bigger v is, the more likely this edge segment is an edge of stroke, which means it has a parallel edge.

3.4

Colour description

Another important attribute of text is that it usually has similar colour. To get the colour of edge segment, we should solve two problems:
Which points in the image are related to an edge segment.
What is the confidence of these related points to describe the colour of an edge segment.On the basis of observation and experiments, we find that the bigger the gradient magnitude, the less likely its colour can be used to describe the edge segment. As the gradient magnitude describes the colour difference between a point and its neighbours and texts usually have uniform colour, only the points with small gradient magnitude can describe the true colour of an edge segment. Thus, we use the colour of all related points with weights to describe the colour of an edge (5)

$$\mathbf{colour} = \frac{\sum w_i \cdot c_i}{\sum w_i}$$

where $w_i = \exp(-\mathrm{mag}_i)$ is used to describe the confidence of related points to describe the colour of edge segment, mag_i is the gradient magnitude and c_i is the colour of a related point.

4

Edge classification

After obtaining the candidate edges, we should determine whether they are text or not. Firstly, we link the candidate boundaries together. Then, boundary features and chain features are extracted to determine the class of a chain, such as in Fig. 5.

Fig. 5 Example of edge classification
a Chain link result; one colour means one chain
b Analysis result based on chain

4.1

Candidate link

In order to increase the reliability of text verification and remove randomly scattered noise, we group candidate boundaries into chains and consider text chain as the elementary analysis unit.

An important cue to candidate boundaries grouping is that text in a group is expected to have similar colour, stroke width, letter height and width. Therefore, we use these criteria to merge separated candidate boundaries into pairs firstly.

4.1.1

Spatial position

The distance between two candidate boundaries should be smaller than twice the biggest scale (6)

$$D(x_1,\ x_2) < 2\max(h_1 + w_1,\ h_2 + w_2)$$

where $D(x_1, x_2)$ computes the distance between the boundaries x_1 and x_2, h_i and w_i are the height and width of the ith boundaries.

4.1.2

Stroke width similarity

The stroke width of two boundaries should be similar (7)

$$\max(\mathrm{sw}_1/\mathrm{sw}_2,\ \mathrm{sw}_1/\mathrm{sw}_2) < 2\max(\mathrm{sw}_1,\ \mathrm{sw}_2)$$

where sw_i is the mean stroke width of the *i*th boundary.

4.1.3

Colour difference

Characters in a line often have similar colour. Thus, we use the Euclidean distance of each channel to compute the colour difference between two boundaries. HSI colour space is used in this part (8)

$$\sum_{\mathrm{channel}_k} \|C_{1,\mathrm{channel}_k} - C_{2,\mathrm{channel}_k}\| < 0.2$$

where

$C_{i,\mathrm{channel}_k}$

is the average colour value in the *k*th channel of the *i*th boundary. In our experiment the threshold is set to 0.2 to get the best performance.

4.1.4

Scale

Characters in a line should have similar scale (9)

$$\max(h_1/h_2,\ w_1/w_2) < 2$$

Through the four criteria above, boundaries in an image are grouped into boundary pairs. Then the pairs are aggregated into character chains based on the pair orientation and the number of boundaries [9].

A candidate boundary might belong to multiple chains, since arbitrary orientation is considered in chain aggregation stage. However, in reality, a character can only belong to a single chain. To eliminate the ambiguous situation, the probability of the ascription is computed for each candidate boundary using the size variation and direction bias (10)

$$p = \exp(-\mathrm{std}(\mathrm{scale}) - \mathrm{std}(\mathrm{o}))$$

where scale describes the scale of a chain and o describes the orientation of all pairs in a chain.

Then the chain with the biggest probability is selected as the chain of the candidate edge.

4.2

Character feature

In order to determine whether a candidate boundary is a character or not, we proposed several features based on the boundary directly.

4.2.1

Stroke width variation

This feature is computed using (4). A character usually has a constant stroke width and most backgrounds do not have this attribute. This feature can eliminate many backgrounds edges, such as foliage.

4.2.2

Axial ratio

One obvious feature of characters is that the ratio between height and width is in a range. Thus, this feature is used to eliminate some background edges.

4.2.3

Corresponding point occupy

Since text is composed by stroke, most edge points have corresponding points. Thus, the proportion of edge points that have corresponding points is useful.

4.2.4

Orientation difference

As described before, a character consists of two parallels. Thus, the orientation difference between an edge point and its corresponding point is approximately opposite. Thus, the average orientation difference between edge points and their corresponding points is used to describe character edges.

4.2.5

Stroke width scale

This feature is described by two values: the ratio between stroke width and candidate edge width, and the ratio between stroke width and candidate edge height. It can be used to eliminate background edges, such as barrier and bricks.

4.3

Chain feature

4.3.1

Histogram of gradient

The HOG feature [16, 17] can describe the boundary of an object very well. We use the location of the chain to extract the RGB image from the original image. Then, we transfer the RGB image into grey image and resize it to 24 × 72 pixels. The resized image is divided into 14 patches, such as in Fig. 6. The HOG of each patch is calculated as the HOG feature. In order to eliminate the effects of illumination, the HOG feature is normalised by the STD of intensity.

Fig. 6 Block regions of HOG feature, including ten single local region and four overlapped local regions (2-3, 3-4, 2-3-4, 6-7)

4.3.2

Structure similarity

Text usually has similar structures, but false positives usually have almost the same structure or diverse structure. We first compute the HOG of each region as in Fig. 7. Then, the similarity between structures of a text line is (11)

$$s = \sum_{i,j} \chi^2(S_{c_i},\ S_{c_j})$$

where

S_{c_i}

is the structure similarity of *i*th boundaries.

Fig. 7 Structure description

4.3.3

Mean probability

For each boundary in a text line, we can get the probability of text by putting the boundary features into a random forest classifier [25]. The probability is computed from the votes. Then, we use the mean probability to measure the probability of a text line for the chain.

4.4

Chain analysis

After obtaining the text line and line features, we train another random forest classifier to determine whether a candidate chain is text or non-text. For each training image, the boundaries of ground truth are used as positive samples and all other boundaries are selected as negative samples. For example, in Fig. 5, the set of boundaries of the word 'Famous' is a positive sample and some boundaries selected from the background make up a negative sample. Then, we extract chain features of each sample as described above and put these features into a random forest classifier for training. After training, when the classifier predicts that a candidate text line is text, we output the location of this candidate directly.

5

End-to-end recognition

We also proposed an end-to-end system of natural scene text recognition. In the text detection system, each candidate boundary is determined, whether it is text or not, through chain linking, character feature, chain feature and chain analysis. In the end-to-end recognition system, candidate boundaries are merged into chains, then they are recognised directly.

Firstly, we use the edge recombination method proposed in the detection system to get the candidate boundaries. As text in a natural scene rarely appears as the form of a solo letter, candidate boundaries are linked into the chain; the details are described in Section 4. Then, the HOG feature is extracted to describe each candidate boundary and a chain is considered as a unit to determine whether it is a text line.

The HOG feature has demonstrated its advantages in many machine learning problems [16, 17] and it can also describe character very well. Thus, we use the grey-image channel of the extracted boundary and resize it to 100 × 100 pixels. Then, the HOG feature is computed with eight bins and nine orientations. A random forest classifier is trained to determine whether a boundary is a character or not and if it is a character, which character it is. The output of the classifier is a number from 0 to 62.0 indicates that the boundary is not a character; 1-62 indicate the characters a-z, A-Z and 0-9. All training samples are extracted from the ICDAR 2011 training set. Fig. 8 gives some examples.

Fig. 8 Positive and negative samples extracted from the ICDAR 2011 database
Top: the positive samples extracted from the ICDAR 2011 database
Bottom: the negative samples also extracted from the ICDAR 2011 database
All samples are resized into 100 × 100 pixel patches

However, some background boundaries have the same property as character boundaries, so it is difficult to identify whether it is a character or not. As described above, text in natural scene appears as text line. So we take a text line as a unit, and if most boundaries in a candidate text line are classified as characters, the candidate text line could be considered as text, or the candidate text line is discarded directly. When a candidate text line is regarded as text, we should determine which character each boundary is. If a boundary is regarded as a character, which means the output of the random forest is 1-62, the boundary is labelled as the corresponding character directly. If a boundary is regarded as not-text in a text line, we select the character with the second biggest probability based on the votes of random forest as the label, Fig. 9 is an example of this modification.

Fig. 9 Direct results of random forest classifier and results after modification
Top: direct results of random forest classifier
Bottom: results after modification
We can see that the characters r and d are regarded as non-text characters
Through modification, the system can recognise them correctly

6

Experiment

6.1

Detection

In order to evaluate the effectiveness of the proposed text detection method, we experiment on three public databases [13, 18, 26]: ICDAR 2003 database, ICDAR 2011 database and Street View Text (SVT) database. All learning steps are trained by the ICDAR 2011 training database. To train the candidate edge classification, we select [#x223c]1000 examples of character edge segments and 4000 examples of non-character edge segments. In the HOG feature training stage, we select the text in the training database as positive sample directly and negative samples are randomly selected from an image, as long as the selected region is not overlapped with the ground truth of this image. Fig. 10 shows some output results.

Fig. 10 Detection results of ICDAR 2011 database
Detected results are marked by rectangle
We can see that this algorithm is robust to reflections, complex background, variations of character size

The ICDAR 2003 public database contains 249 images with various sizes, colours, fonts and complex backgrounds. The same evaluation criteria method as ICDAR 2003 Robust Reading Competition is adopted to evaluate the algorithm. The precision is 0.79, recall is 0.62 and *f*-score is 0.69.

Epshtein *et al.* [10] also use edge and stroke width to detect text, and their precision is 0.73, recall is 0.60 and *f*-score is 0.66. As their method extracts stroke width from edge directly, they cannot extract stroke in the case of light reflection and some complex backgrounds. Table 1 shows that our method gives competitive results compared with ICDAR 2003 Robust Reading Competition.
Table 1 Text detection results on the ICDAR 2003 database

Algorithm	Recall	Precision	*f*
proposed	0.62	0.79	0.69
Hinnerk Becker [19]	0.67	0.62	0.62
Chen and Yuille [5]	0.60	0.60	0.58
Qiang Zhu [19]	0.40	0.33	0.33
Kim *et al*. [20]	0.28	0.22	0.22
Ezaki *et al*. [21]	0.36	0.18	0.22

The ICDAR 2011 public database contains 255 images with different sizes, colours, fonts and illuminations. We use the same evaluation criteria method and the same parameters in ICDAR 2011 Robust Reading Competition [18] to evaluate our algorithm.

Our algorithm achieves precision of 78.44%, recall of 63.49% and *f*-measure of 70.18% (Table 2). The Neumann's method gets the best result, which is a new method using an end-to-end text localisation method. Our precision is a little worse than Kim's method, while recall is better, and the *f*-score is close to it. Our result is better than the other methods besides Neumann's and Kim's. To evaluate the performance of proposed corresponding point detection method quantitatively, we design an experiment based on the suggestion of reviewer. We use the corresponding point detection method in SWT of Epstein to replace our corresponding point detection method, and the other parts remain. The precision is 0.8, the recall is 0.47 and the *f*-score is 0.6. The *f*-score decreases to 0.1. The results show that our corresponding point detection method is more robust.
Table 2 Text detection results on ICDAR 2011 database, compared with other methods

Algorithm	Recall	Precision	*f*
proposed	0.63	0.78	0.70
Neumann's method [22]	0.68	0.85	0.75
Kim's method [23]	0.62	0.83	0.71
Yi's method [24]	0.58	0.67	0.62
TH-TexLoc System [18]	0.58	0.67	0.62
TDM_IACS [18]	0.54	0.64	0.58

We also test our method on a more difficult database, namely the SVT database, which contains 249 images harvested from Google Street View. Image text in this database exhibits high variability and often has low resolution. Fig. 11 shows some output results in this database with proposed algorithm. The results show that the proposed algorithm is robust to font, size, variety orientation and illumination.

Fig. 11 Detection results of SVT database
We can find that this algorithm is robust to different font, blur and illumination

In order to compare with other algorithms, we also evaluate the results with the evaluation criteria in ICDAR 2011 dataset. The criteria method [27] uses two parameters (t_p, t_r) to evaluate the precision and the recall. When the area precision is bigger than t_p, and area recall is bigger than t_r, we think the detected result is true and the target is detected. However, in an experiment we found that the ground truth of this database is different from ICDAR 2011, as only some words are annotated. The successfully detected words which are not annotated are counted as false positives. So, the precision cannot be taken into account in this sense. Besides, almost all of the ground truths are bigger than the corresponding text area. Thus, even if we detect the text successfully, the area recall could be smaller than t_r. In such situations, the successfully detected result is also considered as a wrong result when calculating recall and precision. Fig. 12 are some examples of the mentioned situations.

Fig. 12 Ground truth and detected result in SVT database
Ground truth is marked by dark rectangles, and detected results are marked by light rectangles
From the figure, we can see that the algorithm has detected the text precisely
When the parameter of tr is 0.8, the bigger the ground truth is, the precision and recall are closer to zero
Another problem we can find is that the ground truth only labelled part of the text in the database, which causes low precision

Owing to the problem about the SVT database mentioned above, we use different parameters (t_p is from 0.4 to 0.8 and t_r is from 0.5 to 0.8) to evaluate our algorithm. Table 3 shows the results of this algorithm applied on the SVT database with different parameters. When t_p is 0.4 and t_r is 0.8, the precision, recall and *f*-score are 24.97, 37.17 and 29.87%, respectively.
Table 3 Text detection results on SVT database with different parameters

*t*p	*t*r											
	0.5			0.6			0.7			0.8		
Precision	Recall	*f*	Precision	Recall	*f*	Precision	Recall	*f*	Precision	Recall	*f*	
0.4	39.70	52.21	45.14	37.38	49.42	42.57	33.29	44.57	38.12	24.97	37.17	29.87
0.5	39.45	52.01	44.87	36.58	48.57	41.73	32.28	43.39	37.02	23.53	36.25	28.54
0.6	39.12	51.60	44.50	35.83	47.70	40.92	31.32	42.25	35.98	22.70	35.16	27.59
0.7	38.75	51.07	44.06	35.12	46.81	40.13	30.44	41.14	34.99	21.25	34.39	26.27
0.8	38.28	50.37	43.50	34.40	45.88	39.32	29.58	40.03	34.02	19.16	32.85	24.20

Fig. 13 illustrates the results changed with different parameters. We can see that the results are stable with the parameter t_p. However, the results decrease with the parameter t_r because the detection text is smaller than the ground truth. When t_r grows, the area recall will be smaller than t_r, and the detected regions are thought as non-text regions, even if the algorithm has detected the text correctly.

Fig. 13 Change in result with parameters tp and tr
Top: the result changes with the parameter tp
Bottom: the result changes with the parameter tr

The method in [28], evaluates its method on this database too, and has 32.9% recall and 19.1% precision on default parameters (t_p = 0.4, t_r = 0.8). To compare with our result more clearly, we extract the result under the same evaluation criteria with different methods in Table 4. We can see that our algorithm achieves a better result both on recall and precision compared to [28]. Another method that used this database is the work of Wang *et al.* [13], but they used different evaluation criteria. To compare with their method, we also test our algorithm with their evaluation criteria: a bounding box is counted as a match if it overlaps a ground truth bounding box by more than 50%. In this case, our method achieves 0.40 precision, 0.52 recall and 0.45 *f*-score, which is better than [13] (*f*-score is 0.38).
Table 4 Text detection results on SVT database, compared with different methods using corresponding evaluation criteria

Algorithm	Recall	Precision	*f*	Algorithm	Recall	Precision	*f*
proposed	0.37	0.25	0.30	proposed	0.52	0.40	0.45
Neumann's method [26]	0.33	0.19	0.24	Wang's method [13]	?	?	0.38

6.2

Recognition

In order to evaluate the end-to-end system, we run our methods on the ICDAR 2011 database. This experimental section is divided into two parts: text detection and character recognition. The end-to-end system recognises characters in an image directly. To evaluate the performance of detection, we link the recognised characters into chains, and compare the locations of chains with ground truth using the ICDAR 2011 evaluation criterion. To evaluate the recognition results, we also use recall, precision and *f*-score. When a word is

detected correctly and all characters are recognised correctly, we consider that this word is recognised correctly. Some other character recognition methods [29, 30] use recognition accuracy to evaluate the recognition results, which operate directly on the ground truth. These two evaluation methods both work; we choose the previous one as it shows the actual recognition result of an end-to-end system.

In the random forest training stage, we select the positive samples from the ICDAR 2011 database directly. The negative samples are also selected randomly from this database, as long as the regions have no intersection with text regions. We use 4669 positive samples of 62 characters in total and 1000 negative samples.

The precision, recall and *f*-measure of the detection results are 86.44, 59.14 and 68.68%. Compared with the proposed detection method, the end-to-end system has better precision, because the recognition system recognises characters directly and can easily distinguish text from the background. The recall is worse than the detection method, because the detection method does not need to recognise characters so that some small characters can be detected as text correctly. The precision, recall and *f*-measure of the recognition results are 34.77, 24.74 and 28.91%. Fig. 14 shows some examples of recognition results.

Fig. 14 Recognition results on the ICDAR 2011 database

Both the text detection and recognition methods are implemented in MATLAB. The average processing time to detect and recognise a scene text image of ICDAR database is about 5 s on an Intel Core i5-2400 CPU 3.10 GHz processor. By implementing on C++ and using parallel processing, the implementation could be faster.

7

Conclusion

In this work, we presented a text detection and recognition approach by analysing the edges of an image. In order to distinguish the text edges from the background, segmentation points are detected firstly. Then, neighbour edges are merged by analysing their stroke width and colour similarity. Finally, edge-based features are extracted to determine whether these edges are text. In the end-to-end recognition system, we use the candidate boundary computation method proposed in the text detection method to obtain candidate boundaries. Then, HOG feature and random forest are used to recognise the characters. Experimental results on different databases demonstrate that the proposed approach compares favourably with other state-of-the-art methods.

LOAD-DATE: July 30, 2015

LANGUAGE: ENGLISH

BIBLIOGRAPHY:

REFERENCES

1 Liang J., Doermann D., Li H.: 'Camera-based analysis of text and documents: a survey', Int. J. Doc. Anal. Recognit., 2005, 7, (2-3), pp. 84-104 (doi: 10.1007/s10032-004-0138-z)

2 Jung K., Kim K., Jain A.: 'Text information extraction in images and video: a survey', Pattern Recognit., 2004, 37, (5), pp. 977-997 (doi: 10.1016/j.patcog.2003.10.012)

3 Lucas S.M., Panaretos A., Sosa L., Tang A., Wong S., Yong R.: 'ICDAR 2003 robust reading competitions'. ICDAR 2003, 2003, pp. 682

4 Lucas S.M.: 'Text locating competition results'. Proc. Third Int. Conf. on Document Analysis and Recognition, 2005, vol. 0, pp. 80-85

5 Chen X., Yuille A.L.: 'Detecting and reading text in natural scenes'. CVPR, 2004, vol. 2, pp. 366-377

6 Ye Q., Huang Q., Gao W., Zhao D.: 'Fast and robust text detection in images and video frames', Image Vis. Comput., 2005, 23, pp. 565-57 (doi: 10.1016/j.imavis.2005.01.004)

7 Zhang J., Kasturi R.: 'Extraction of text objects in video documents: recent progress'. Proc Eighth IAPR Workshop on Document Analysis Systems (DAS08), Nara, Japan, 2008, pp. 1-13

8 Jain A., Yu B.: 'Automatic text location in images and video frames', Pattern Recognit., 1988, 31, (12), pp. 2005-2076

9 Yao C., Bai X., Liu W., Ma Y., Tu Z.: 'Detecting texts of arbitrary orientations in natural images'. CVPR, 2012

10 Epshtein B., Ofek E., Wexler Y.: 'Detecting text in natural scenes with stroke width transform'. CVPR, 2010

11 Zhang J., Kasturi R.: 'Character energy and link energy-based text extraction in scene images'. ACCV 2010, November 2010 (LNCS, 6495), vol. II, pp. 832-844

12 Lee S., Cho M., Jung K., Kim J.: 'Scene text extraction with edge constraint and text collinearity'. ICPR, 2010

13 Wang K., Babenko B., Belongie S.: 'End-to-end scene text recognition'. ICCV 2011, 2011

14 Canny J.: 'A computational approach to edge detection', IEEE Trans. Pattern Anal. Mach. Intell., 1986, 8, pp. 679-698 (doi: 10.1109/TPAMI.1986.4767851)

15 Perazzi F., Krahenbuhl P., Pritch Y., Hornung A.: 'Salient filters: contrast based filtering for salient region detection'. CVPR, 2012

16 Dalal N., Triggs B.: 'Histograms of oriented gradients for human detection'. CVPR, 2005

17 Pan Y.-F., Hou X., Liu C.-L.: 'A robust system to detect and localize text in natural scene images'. DAS, 2008

18 Shahab A., Shafait F., Dengel A.: 'ICDAR 2011 robust reading competition challenges 2: reading text in scene images'. ICDAR 2011, 2011, pp. 1491-1496

19 Lucas S.: 'Icdar 2005 text locating competition results'. ICDAR, 2005

20 Kim J.S., Park S.C., Kim S.H.: 'Text locating from natural scene images using image intensities'. ICDAR, 2005

21 Ezaki N., Kiyota K., Minh B.T., Bulacu M.: 'Improved text-detection methods for a camera-based text reading system for blind persons'. ICDAR, 2005

22 Neumann L., Matas J.: 'On combining multiple segmentations in scene text recognition'. ICDAR, 2013

23 Koo H.I., Kim D.H.: 'Scene text detection via connected component clustering and nontext filtering', IEEE Trans. Image Process., 2013, 22, (6), pp. 2296-2305 (doi: 10.1109/TIP.2013.2249082)

24 Yi C., Tian Y.: 'Text string detection from natural scenes by structure-based partition and grouping', IEEE Trans. Image Process., 2011, 20, (9), pp. 2594-2605 (doi: 10.1109/TIP.2011.2126586)

25 Breiman L.: 'Random forest', Mach. Learn., 2001, 45, (1), pp. 5-32 (doi: 10.1023/A:1010933404324)

26 Wang K., Belongie S.: 'Word spotting in the wild'. ECCV, 2010

27 Wolf C., Jolion J.-M.: 'Object count/area graphs for the evaluation of object detection and segmentation algorithms', Int. J. Doc. Anal. Recognit., 2006, 8, pp. 280-296 (doi: 10.1007/s10032-006-0014-0)

28 Neumann L., Matas J.: 'Real-time scene text localization and recognition'. CVPR, 2012

29 Shi C., Wang C., Xiao B., Zhang Y.: 'Scene text recognition using part-based tree-structured character detection'. CVPR, 2013

30 Mishra A., Alahari K., Jawahar C.V.: 'Top-down and bottom-up cues for scene text recognition'. CVPR, 2012

PUBLICATION-TYPE: Magazine

---- End of Request ----
Print Request: Current Document: 21
Time Of Request: Friday, February 09, 2018 02:17:01 EST