Detecting and Segmenting Text from Natural Scenes with 2-Stage Classification

Renjie Jiang, Feihu Qi, Li Xu, Guorong Wu
Department of Computer Science and Technology, Shanghai Jiao Tong University
Shanghai, China
{blizard1982, fhqi, nathan.xu, grwu}@sjtu.edu.cn

Abstract

This paper proposes a novel learning-based approach for detecting and segmenting text from scene images. First, the input image is decomposed into a list of Connected-Components (CCs) by color clustering algorithm. Then all the CCs including text CCs and non-text CCs are verified by a 2-stage classification module, where most of non-text CCs are discarded by cascade classifier and the remaining CCs are further verified by SVM. All the accepted CCs are output to generate result image. Experiments have been taken on a lot of images with different nature scenes and show satisfactory performance of our proposed method.

1. Introduction

Text is ubiquitous in our daily life on road signs, bill boards, shop names, menus, labels and so on. Extracting and recognizing text has a promising future in information retrieval, auto-driving system, aiding visually impaired people or abroad travelers. Since OCR could only deal with text in simple background, text extraction is the preliminary procedure for recognition in variant scenes. If we can detect and segment text from natural scene images, it will be very helpful for many important applications.

Many studies have been researched in this field. These approaches can be divided into two categories: texture-based and region-based.

Region-based method utilized the properties of the color or gray scale in a text region or their differences with the corresponding properties of the background. These methods can be further divided into two sub-approaches: connected component (CC)-based and edge based. These two approaches work in a bottom-up fashion: first, they identifying sub-structures in pictures, such as CCs [1][2] or edges [3][4][5], and then merge these sub-structures to mark bounding boxes for text by heuristic rules[1][5] or learning-based rules such as neural networks, Markov Random Field[2], etc.

Texture-based methods use the observation that text in images has distinct textural properties that distinguish them from the background. The techniques based on FFT [6], Gabor [7], wavelet [8], spatial variance [9][10], etc. can be used to detect the textural properties of a text region in an image. In the stage of verification, heuristic rules, neural network [6], SVM [10] and conditional random field [11] are popular.

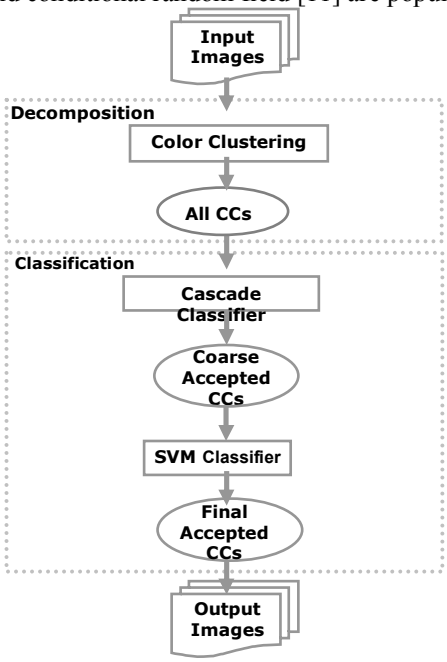


Figure 1. Architecture of our approach

In this paper, we propose a novel CC-based algorithm whose architecture is shown in Fig. 1. First, the input image is decomposed into a lot of CCs by color clustering algorithm including text CCs and non-text CCs. To segment text from background, our purpose is to eliminate non-text CCs but preserve text CCs. Then the problem of segmentation turns out to be the problem of classification. The most significant

Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications (ISDA'06)
0-7695-2528-8/06 $20.00

difference between our method and others is that we build a 2-stage classification module, in which all the CCs are verified by a cascade classifier and a SVM. The cascade classifier is combined by a series of weak classifiers. Most apparently non-text CCs are discarded as early as possible to save a great deal of computation. SVM concentrates on CCs accepted by the cascade and does further verification. Only those accepted by both cascade classifier and SVM are output in final result. This combination of weak and strong classifiers guarantees the effectiveness and efficiency of our approach.

The paper is organized as follows: Section 2 describes the decomposition algorithm. Section 3 shows the features used to classification. Section 4 presents the details of 2-stage classification. Section 5 discusses the experiment and performance. Finally, section 6 summarizes the contributions and future works.

2. Image decomposition

The performance of a CC based method depends on the quality of its CCs. Different from many traditional methods which use bit-drop [1] to group pixels into CCs, our approach detects CCs directly in 24-bit RGB color space. However, due to illumination variation and noise, pixels in same character can not be in same color. In order to group them into one CC, our clustering algorithm has tolerance of color variation.

In this paper, each CC stores a map of pixels belonging to it and average color of these pixels. The Euclidean distance between two points in 24-bit RGB space is measured to decide the connectivity. We make the assumption that P1 is a grouped pixel of CC1, and P2 is an ungrouped neighboring pixel of P1. P1 and P2 are connected only if two conditions (Eq.1) and (Eq.2) are met:

$$\sqrt{(r_1-r_2)^2+(g_1-g_2)^2+(b_1-b_2)^2} < tNeighbour \quad (1)$$

$$\sqrt{(r_c-r_2)^2+(g_c-g_2)^2+(b_c-b_2)^2} < tCenter \quad (2)$$

$$(r_c, g_c, b_c) = \frac{1}{n}\sum_{i=1}^{n}(r_i, g_i, b_i) \quad (3)$$

Where

tNeighbour and *tCenter* are predetermined thresholds.

(r_1, g_1, b_1) and (r_2, g_2, b_2) are color of P1 and P2 respectively.

(r_c, g_c, b_c) is the central color of CC1, namely the average color of all pixels in CC1 (Eq.3).

n is the number of pixels in CC1.

If P2 is connected to P1, we will cluster P2 into CC1 and update the central color (r_c, g_c, b_c) .Else a new CC is found and its central color is set. Then the clustering algorithm will proceed until all pixels have been clustered. Figure 2 displays the result of decomposition. In this figure, the original image (a) is clustered in color space, and the clustered result image (b) is composed by thousands of CCs.

Figure 2. Clustering result: (a)original image, (b)clustered image

3. Text features

As we all know, the quality of classification depends on the quality of features. Therefore, the selection of text features is important for the following processing. Before we can discuss the details of 2-stage classification, please pay our attention on text features.

Totally 15 features are developed to discriminate text CCs from non-text CCs in our method. All these features can be divided into 5 categories: geometric features, shape regularity features, edge features, stroke features and spatial coherence features.

3.1. Geometric features

The first category is geometric features. They are used to measure the basic properties of CCs such as size, width, height, and aspect ratio. They are easy to calculate and helpful to discard a large number of apparently non-text CCs quickly.

Feature *AreaRatio* measures the size of the CC (Eq.4), where $area(\cdot)$ is the area of input CC's bounding box. It will discard many non-text CCs which are too big or too small. Since a large portion of CCs are small non-text CCs, *AreaRatio* helps reject them as early as possible.

$$AreaRatio = \frac{area(CC)}{area(Pic)} \quad (4)$$

Feature *LengthRatioL* (Eq.5) and *LengthRatioS* (Eq.6) measure the length of the CC, where *w* and *h* are CC's width and height, *PicW* and *PicH* are picture's width and height. They will discard CCs which are too long or too short for input picture.

$$LengthRatioL = \max\{\frac{w}{PicW}, \frac{h}{PicH}\} \quad (5)$$

Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications (ISDA'06)
0-7695-2528-8/06 $20.00 © 2006

$$LengthRatioS = \min\{\frac{w}{PicW}, \frac{h}{PicH}\} \quad (6)$$

Feature *AspectRatio* measures the aspect ratio of input CC and reject CCs which are too slim or too flat (Eq.7).

$$AspectRatio = \min\{\frac{w}{h}, \frac{h}{w}\} \quad (7)$$

3.2. Shape regularity features

Text CCs always posses more regular shape than non-text CCs. Based on this observation, we propose 5 shape regularity features for further exploiting the difference between text CCs and non-text CCs, such as occupy ratio, number of holes, compactness, roughness of contour, and minimum border distance.

Feature *OccupyRatio* measures the occupy ratio on input CC (Eq.8), where $|\cdot|$ counts the pixels belonging to the CC.

$$OccupyRatio = \frac{|CC|}{area(CC)} \quad (8)$$

Feature *Holes* counts the holes in input CC (Eq.9), where $imholes(\cdot)$ is morphological holes counting. Usually text CCs have no more than 6 holes, so CCs with too many holes will be rejected.

$$Holes = imholes(CC) \quad (9)$$

Feature *ContourRoughness* measures the roughness of input CC's contour (Eq.10), where $open(\cdot, strel)$ is morphological open operation. It rejects CCs whose contours are too rough.

$$ContourRoughness = \frac{|CC - open(CC, 2\times 2)|}{|CC|} \quad (10)$$

Feature *Compactness* divides the area of CC's bounding box by the square of CC's perimeter (Eq.11), where $contour(\cdot)$ gets all contour pixels of input CC. Because of the presence of strokes, text CCs have long perimeter and get low response in this feature.

$$Compactness = \frac{area(CC)}{|contour(CC)|^2} \quad (11)$$

Feature *BorderDist* measures the minimum distance of input CC to the picture's border (Eq.12), where *(tlx,tly)* is the position of CC's top-left corner and *(brx,bry)* is the position of bottom-right corner. On the ground that important texts are always in the center of picture, we discard CCs which are too close to the picture border.

$$BorderDist = \min\{tlx, tly, PicW - brx, PicH - bry\} \quad (12)$$

3.3. Edge features

Edge features are the intrinsic properties of characters. Two edge features are proposed in this paper: *EdgeContrast* and *EdgeAngleSym*.

Feature *EdgeContrast* measures the contrast between character and background (Eq.13). We make the reasonable assumption that most text CCs are highly closed by strong edges. By counting contour pixels with strong edge, we can calculate strong edge pixel ratio to get *EdgeContrast*. Where $contour(\cdot)$ gets CC's contour pixels and $edge(\cdot)$ is the Canny binary edge map of input picture.

$$EdgeContrast = \frac{contour(CC) \cap edge(Picture)}{contour(CC)} \quad (13)$$

Another edge feature *EdgeAngleSym* is inspired by statistics on CCs' edge angle [9]. We find there is a strong symmetry in angle distribution of text CCs' contour pixels. While for non-text CCs, this symmetry doesn't exist. Then we use feature *EdgeAngleSym* to measure CCs' angle symmetry degree (Eq.14), where $A(\theta)$ is number of CC's contour pixels whose angles are θ:

$$EdgeAngleSym = \sum_{\theta=0}^{\pi} |A(\theta) - A(\theta + \pi)| \quad (14)$$

3.4. Stroke features

Stroke features also expose the essence of text. A character is composed by strokes with small and uniform width. Though one character is decomposed into several text CCs in our approach, the strokes are not decomposed. Usually, a text CC consists of one or more connected strokes. Two features *StrokeWidthMean* and *StrokeWidthDev* exploit the 'smallness' and 'uniformity' of stroke respectively. Stroke features help discriminate text CCs from non-text CCs effectively, but they are computed with high cost.

Feature *StrokeWidthMean* measures the average width of strokes in input CC (Eq.15). It guarantees the 'smallness' of stroke. Feature *StrokeWidthDev* measures the deviation of stroke width in input CC (Eq.16). It rejects CCs without uniform stroke width.

$$StrokeWidthMean = mean(strokeWidth(skeleton(CC))) \quad (15)$$

$$StrokeWidthDev = \frac{dev(strokeWidth(skeleton(CC)))}{mean(strokeWidth(skeleton(CC)))} \quad (16)$$

Where

$mean(\cdot)$ and $dev(\cdot)$ calculate mean and deviation respectively .

$skeleton(\cdot)$ is morphological skeleton operation.

Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications (ISDA'06)
0-7695-2528-8/06 $20.00 © 2006

$strokeWidth(\cdot)$ stands for the shortest distance between the pixel on skeleton to the pixels outside CC.

3.5. Spatial coherence features

All features described above are based on single CC. However, other features can be extracted from spatial relationship between multi CCs, for improving the performance of classification. These features include *BackgroundInfo* (Eq.17) and *Cohesion* (Eq.18):

$$BackgroundInfo = \frac{area(background(CC))}{area(Pic)} \quad (17)$$

$$Cohesion = \frac{area\left(dilate(CC, 5\times5)\right)}{area(\text{Pic})} \quad (18)$$

Where

$dilate(\cdot, strel)$ is morphological dilation operation.

$background(\cdot)$ is the smallest CC which contains the input CC.

4. Classification

In this paper, the proposed classification is divided into two stages: coarse classification and precise classification. The former is implemented by a cascade classifier and the latter is by a SVM.

4.1 Cascade classifier

In the stage of coarse classification, a cascade classifier is employed to discard apparently non-text CCs. The cascade classifier consists of a series of weak classifiers, each concentrates on one feature mentioned in section 3. Figure 3 illustrates the structure of cascade.

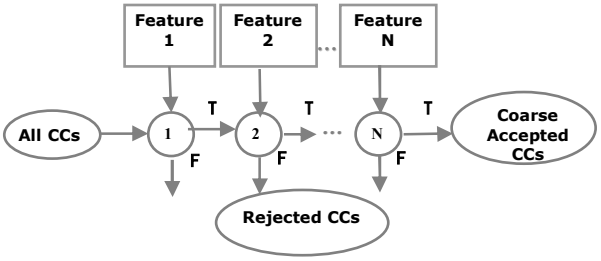


Figure 3. Structure of cascade classifier

In the paper, a weak classifier is composed by a feature and two thresholds: one upper threshold and one lower threshold (Eq.19). For each input CC, the weak classifier measures the feature and makes the decision whether the CC is text or not.

$$h_i(x) = \begin{cases} 1, & \text{if } \theta L_i < f_i(x) < \theta U_i \\ 0, & \text{otherwise} \end{cases} \quad (19)$$

Where

x is the input CC.

f_i is the feature of ith weak classifier.

θL_i is the lower threshold of ith weak classifier.

θU_i is the upper threshold of ith weak classifier.

h_i is the decision of ith weak classifier. If the decision is 1, the CC is regarded as text. Otherwise, the CC is considered as non-text.

At the beginning, all CCs extracted in the decomposition step are feed into the first weak classifier. It measures certain feature on CCs one by one and categorizes them into positive or negative. The negative CCs are considered as non-text and rejected immediately. For positive CCs, similar processing is repeated in following weak classifiers until the end of the cascade. In the coarse result after this cascade, about 90% non-text CCs are discarded and almost all the text CCs are preserved. Figure 4 demonstrates the result of the cascade.

(a) **(b)**

Figure 4. Effect of cascade classifier: (a) input clustered image, (b) result of cascade classifier

Which is more important, all features used by SVM for precise classification are prepared in the cascade. Though SVM has satisfactory discriminating ability, its tolerance of feature absence is poor. So we must calculate all features for all input CCs before SVM can classify them. Without the cascade, the system would be quite computationally exhaustive.

Due to the advantage of cascade, there is no need to calculate all 15 features for all CCs. We make a statistics on 500 testing images and the advantage of cascade is shown in figure 5. Number of non-text CCs are decreasing after each weak classifier. Since most non-texts are rejected in the early stage of cascade, a great deal of meaningless computation is reduced. The cascade classifier helps accelerate the processing greatly.

Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications (ISDA'06)
0-7695-2528-8/06 $20.00 © 2006

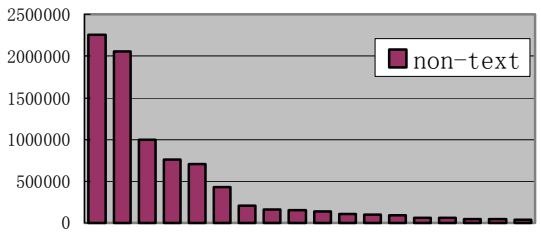


Figure 5. Cascade process for non-text CCs

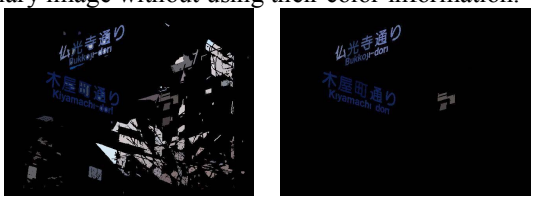
K. Zhu et al [12] propose a cascade training method. In the paper, we still follow this training scheme to train our cascade classifier. Table 1 shows the performance of the cascade. We retain text CCs as many as possible. Though there are a lot of non-text CCs in the intermediate result, SVM in next stage will be capable to filter them out.

Table 1. Cascade training and testing

	Pics	Input CCs		Output CCs	
		text	non -text	text	non -text
Train	100	3,554	556,962	3,452	9,774
Test	500	14,804	2,257,346	14,053	39,689

4.2 SVM

The stage of precise classification is implemented by SVM to do further verification on intermediate result of previous coarse classification. In our approach, with the help of cascade classifier, all 15 features are prepared with relatively small computational cost. All CCs which passed through the cascade are input into the SVM with normalized features, and only those accepted by SVM are considered as texts. Figure 6 shows classification result of SVM. Accepted CCs are output to form the result binary image without using their color information.



(a) **(b)**

Figure 6. SVM classification: (a) input image of cascade result, (b) SVM result

In the paper, we train SVM on a subset of training CC base instead of on the whole set, in order to make the SVM more suitable for 2-stage classification. We treat text CCs and non-text CCs differently. All text CCs are retained while we put non-text CCs into the cascade classifier. And those misclassified by the cascade as positive are merged with all text CCs to form the training set for SVM. Because we require the distribution of training CCs to be close to what the SVM will meet in practical application. Table 2 shows the performance of SVM, where most non-text CCs have been discarded.

Table 2. SVM training and testing

	Pics	Input CCs		Output CCs	
		text	non -text	text	non -text
Train	100	3,452	9,774	3,187	298
Test	500	14,053	39,689	13,197	1,114

5. Experiments

Taking into account that text CCs are usually large while false-accepted non-text CCs are small, it's unfair to directly compare the number of finally accepted text CCs and non-text CCs. Instead, we employ a strict pixel-wise evaluation method which compares output image with ground-truth image shown in Fig. 7.

(a) **(b)** **(c)**

Figure 7. Evaluation detail: (a) original image, (b) result binary image, (c) ground truth image

In figure 7, the result image and ground-truth image are binary images where text pixels are labeled as true in black color. To assure the soundness of evaluation, all the ground-truth images are labeled manually. The details of evaluation are shown as follows (Eq.20):

$$hit = area\left(Result \,\&\, GroundTruth\right)$$
$$error = area\left(Result \,\&\, \overline{GroundTruth}\right) \quad (20)$$
$$miss = area\left(\overline{Result} \,\&\, GroundTruth\right)$$

$$precision = \frac{hit}{hit + error} \qquad recall = \frac{hit}{hit + miss}$$

Table 3. Performance of the system

	Pics	Precision	Recall
Train Set	100	92.13%	94.57%
Test Set	500	90.35%	94.81%
Method[12]	500	88.90%	97.50%

Our system is implemented on Pentium4 3GHz with VC++ 6.0. We build a testing base containing 500 scene images (640*480) with variation on language, font, size, color, skew angle, illumination and surface. The average processing time is less than 1second and the detail of performance is shown in table 3. Compared with [12], our method has higher precision but lower recall. Some of results are shown in figure 8.

Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications (ISDA'06)
0-7695-2528-8/06 $20.00 © 2006

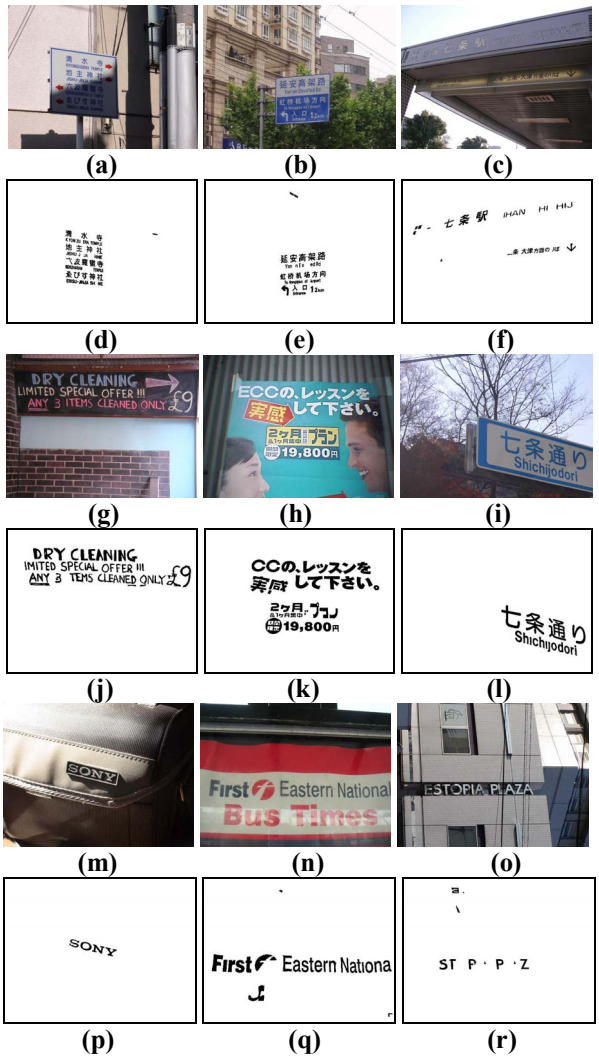


Figure 8. Examples: (a)~(c) (g)~(i) (m)~(o) original images, (d)~(f) (j)~(l) (p)~(r) final result images.

6. Conclusion

This paper presents a novel approach on scene text detection and segmentation. The method is learning-based and robust for various text size, font, language, color, skew angle and surface (Fig.8 (d)~(f),(j)~(l),(p)). But for text with specular reflectance, the result is unsatisfactory (Fig.8 (q)(r)). Further work will proceed on improvement of our approach.

Acknowledgment

The authors would like to thank reviewers for their helpful comments on this paper. This work was performed at Computer Vision Laboratory, SJTU and was supported by OMRON under PVS project.

References

[1] K.Q. Wang, J.A. Kangas, "Character Location in Scene Images from Digital Camera", *Pattern Recognition* 36, 2003，pp.2287-2299.

[2] D.Q. Zhang, F.H. Chang, "Learning to Detect Scene Text Using a Higher-order MRF with Belief Propagation", Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, June, 2004, pp.101-107.

[3] K.C. Kim, H.R. Byun, Y.J. Song, Y.W. Choi, S.Y. Chi, K.K. Kim, Y.K. Chung, "Scene Text Extraction in Natural Scene Images using Hierarchical Feature Combining and Verification", Proc. of ICPR, August, 2004, vol.2, pp.679-682.

[4] C. Liu, C. Wang, R. Dai, "Text Detection in Images Based on Unsupervised Classification of Edge-based Features", Proc. of ICDAR 2005.

[5] M.R. Lyu, J. Song, M. Cai, "A Comprehensive Method for Multilingual Video Text Detection, Localization, and Extraction", IEEE Trans. on Circuits and System for Video Technology, 2005, vol.15, no.2, pp243-255.

[6] B.T. Chun, Y. Bae, T.Y. Kim, "Automatic text extraction in Digital Videos Using FFT and Neural Network", Proc. of IEEE International Fuzzy Systems Conference, 1999, vol.2, pp.1112-1115.

[7] D. Chen, K. Shearer, H. Bourlard, "Text Enhancement with Asymmetric Alter for Video OCR", Proc. of ICIAR 2001, pp.192-197.

[8] W. Mao, F. Chung, K. Lanm, W. Siu, "Hybrid Chinese / English Text Detection in Images and Video Frames", Proc.of ICPR 2002, vol.3, pp.1015-1018.

[9] P. Clark, M. Mirmehdi., "Finding Text Regions Using Localized Measures", Proc. of the 11th British Machine Vision Conference, 2000, pp.675-684.

[10] K.I. Kim, K. Jung, J.H. Kim "Texture-Based Approach for Text Detection in Images Using Support Vector Machines and Continuously Adaptive Mean Shift Algorithm", IEEE Trans. on Pattern Analysis and Machine Intelligence, Dec.2003, vol.25, no.12, pp.1631-1639.

[11] J. Weinman, A. Hanson, A. McCallum, "Sign Detection in Natural Images with Conditional Random Fields", Proc. of IEEE International Workshop on Machine Learning for Signal Processing, Brazil, Sep.2004, pp.549-558.

[12] K. Zhu, F. Qi, R. Jiang, L. Xu, "Using Adaboost to Detect and Segment Characters from Natural Scenes", Proc. of CBDAR 2005, pp.52-59.