Available online at www.sciencedirect.com
ScienceDirect
Signal Processing 87 (2007) 2649–2658
SIGNAL PROCESSING
www.elsevier.com/locate/sigpro

An MRF model-based approach to the detection of rectangular shape objects in color images

Yangxing Liu*, Takeshi Ikenaga, Satoshi Goto

Graduate School of Information, Production and Systems, Waseda University, Japan

Received 16 January 2006; received in revised form 17 March 2007; accepted 25 April 2007
Available online 10 May 2007

Abstract

Rectangular shape object detection in color images is a critical step of many image recognition systems. However, there are few reports on this matter. In this paper, we proposed a hierarchical approach, which combines a global contour-based line segment detection algorithm and an Markov random field (MRF) model, to extract rectangular shape objects from real color images. Firstly, we use an elaborate edge detection algorithm to obtain image edge map and accurate edge pixel gradient information (magnitude and direction). Then line segments are extracted from the edge map and some neighboring parallel segments are merged into a single line segment. Finally all segments lying on the boundary of unknown rectangular shape objects are labeled via an MRF model built on line segments. Experimental results show that our method is robust in locating multiple rectangular shape objects simultaneously with respect to different size, orientation and color.

Keywords: Edge detection; Line segment detection; MRF model; Randomized Hough transform; Rectangle detection

1. Introduction

Object detection is a topic of great importance for many recognition systems. Rectangular shape object detection is one of the basic tasks of computer vision and is especially important in some applications such as man-made object detection [1], which frequently has shapes with rectangular edges, for example some traffic signs and building roofs.

Some papers about the rectangular object detection from binary or grey-level images have been published [2–5]. However, nowadays, color image has definitely supplanted monochromatic or grey information with increasing speed and decreasing costs of computation. Color information permits a more complete representation of images and a more reliable segmentation of them. So extracting rectangular shape objects from color images, which may contain multiple complex objects, is becoming a significant operation in the application of computer vision.

In this paper, a novel rectangular shape object detection method targeted towards being robust with respect to diverse kinds of rectangular object appearances, including object size, orientation, and color, is presented. The method is significantly different from conventional rectangular object detection techniques, which directly estimate the five parameters (center location (two parameters),

*Corresponding author.
E-mail address: lyx@ruri.waseda.jp (Y. Liu).

0165-1684/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.sigpro.2007.04.018

length, width and orientation) of an arbitrarily oriented rectangle to extract rectangular objects with high time complexity and space requirement. The basic idea of our algorithm is to use an MRF model built on detected line segments in image edge map to label certain line segments, which lie on the boundary of rectangular objects. First, with an elaborate edge detection algorithm based on differential geometry, we obtain image edge map and precise gradient direction of each edge pixel. Second we link edge pixels with similar orientation into straight line segments and group neighboring parallel line segments into a single line segment. Eventually, an MRF model is built to label line segments belonging to different rectangular shape objects. The randomness is used to model the uncertainty in the assignment of the labels.

The rest of this paper is organized as follows. Section 2 gives an overview of previous work on shape-based object detection. Section 3 describes our edge detection algorithm. The line segment detection and combination algorithm is discussed in Section 4. Section 5 discusses our MRF model and its application to rectangle detection. Experimental results are explained in Section 6. Conclusion remarks are given in last section.

2. Related work

Shape-based object detection is one of the difficult tasks in computer vision systems. Although it has been studied for dozens of years, an accurate and high performance approach is still a great challenge today.

Many methods have been previously used to extract geometric primitives from image data. The most popular methods are variations on the Hough transform (HT) [6]. Conventional HT is very time-consuming and expensive in memory, especially for rectangle detection, which has five unknown parameters to be estimated. The randomized Hough transform (RHT) [7] was originally designed mainly for analytical curve extraction, such as line, circle, etc. Its application to direct rectangle detection is still a time-consuming process. Another great problem affecting the RHT is that its performance is poor when the image is noisy or complex.

In [8], a rectangle detection algorithm based on line primitive is proposed to extract fixed orientation rectangles which are composed of horizontal and vertical lines in binary images. Zhu et al. [9] propose rectangular HT to detect rectangular particles in cry-electron microscopy images. But this algorithm can only detect rectangles with the same dimensions in the image and the dimension of rectangle must be known. Jung et al. [2] propose a windowed HT algorithm to detect rectangles from grey-level images. But this algorithm is also time-consuming and may include many false alarms in detection result.

Roth et al. [10] proved that the shape detection problem can be formulated in terms of an optimization problem, so the genetic algorithm is proposed to solve this problem [11,12]. However, it is only applied to binary image and its optimization process is time-consuming. Moreover, it does not go far enough to detect multi-object simultaneously.

3. Image edge extraction

It is necessary for our algorithm to detect image edges in a preprocessing step, even though the edge detection is not the focus of this paper.

First, we use an accurate isotropic edge detector based on multi-dimensional gradient analysis to obtain edge information, which tends to be robust under changes in illumination or related camera parameters and greatly reduces the detection time complexity. Because the edge detector we adopted is different from most existing isotropic edge detectors, which cannot provide accurate edge direction information [13], basic ideas are presented below to illustrate the edge detector. The idea about the edge detector has been described in [14].

3.1. Edge detector

Given a color image, the difference vector DV in rgb color space induced by moving an infinitesimal step in the image plane in the direction $\{\mathrm{d}x, \mathrm{d}y\}$ is: $\mathrm{DV} = (\mathrm{d}x\ \mathrm{d}y)J_c^{\mathrm{T}}$,

$$J_c = \begin{bmatrix} \partial r/\partial x & \partial r/\partial y \\ \partial g/\partial x & \partial g/\partial y \\ \partial b/\partial x & \partial b/\partial y \end{bmatrix},$$

where J_c is the Jacobian matrix of the image, r_x, r_y, g_x, g_y, b_x and b_y, are the six first order derivatives of the three color channels with respect to the two coordinates x and y.

The Euclidean squared magnitude of DV is

$$\mathrm{DV}^2 = (\mathrm{d}x\ \mathrm{d}y)M_c(\mathrm{d}x\ \mathrm{d}y)^{\mathrm{T}},$$

where $M_c = J_c^{\mathrm{T}}J_c = \begin{bmatrix} M_{xx} & M_{xy} \\ M_{xy} & M_{yy} \end{bmatrix}$,

$$M_{xx} = (r_x)^2 + (g_x)^2 + (b_x)^2,$$

$$M_{xy} = r_x r_y + g_x g_y + b_x b_y,$$

$$M_{yy} = (r_y)^2 + (g_y)^2 + (b_y)^2$$

and DV^2 is a measure of the rate of change of the image in the direction of $\{dx, dy\}$. Maximizing this magnitude is an eigenvalue problem. We can obtain the magnitude extremum in the direction of the eigenvector of the matrix M_c and the extremum value is the corresponding eigenvalue. The trace of M_c, $((r_x)^2 + (g_x)^2 + (b_x)^2 + (r_y)^2 + (g_y)^2 + (b_y)^2)$ is evaluated as a measure of the joint channel gradient intensity.

The larger eigenvalue of M_c is

$$V = (\sqrt{(M_{xx} + M_{yy})^2 - 4 \times (M_{xx} \times M_{yy} - M_{xy}^2)} + M_{xx} + M_{yy})/2 \quad (1)$$

and its corresponding eigenvector is $\{M_{xy}, V - M_{xx}\}$. The gradient direction angle is defined by eigenvector:

$$\theta = \arctan\left(\frac{V - M_{xx}}{M_{xy}}\right). \quad (2)$$

The square root of the larger eigenvalue and its corresponding eigenvector direction are the equivalents of the gradient magnitude and gradient direction at any given point. So we can get precise gradient magnitude and direction of each pixel (i,j) by computing corresponding eigenvalue $V(i,j)$ and eigenvector direction $\theta(i,j)$ with Eqs. (1) and (2).

3.2. Edge detection algorithm

Based on the detector [14] in previous section, we exploit image pixel gradient direction, magnitude, spatial information and region property to obtain image edge. A selection-and-verification scheme is proposed to finish this task. First we will select edge pixel candidates according to image pixel eigenvector direction and eigenvalue magnitude. Then spatial information and region-based analysis are integrated to verify all candidate edge pixels.

(1) Candidate edge pixel selection: First of all, the matrix M_c is computed for each pixel on the image and we will get a series of eigenvalues V and eigenvectors E for all pixels.

Edge pixels are those points with local maximum gradient magnitude in their gradient direction. So a pixel is kept as a candidate edge pixel only if it has a larger gradient magnitude than that of its neighbor located in the direction closest to its gradient direction, i.e., the eigenvalue of an edge pixel must be greater than that of both two neighboring points, which are closest to its eigenvector direction. We illustrate our idea more clearly with Fig. 1. The arrow line shows the gradient direction of an image pixel (i,j) corresponding to its eigenvector direction. Because the gradient direction is closest to pixel $(i+1, j-1)$ and pixel $(i-1, j+1)$, the pixel (i,j) will be classified as candidate to be edge pixel only if $V(i,j) > V(i+1, j-1)$ and $V(i,j) > V(i-1, j+1)$. Otherwise, it is classified as a non-edge pixel.

(2) Edge pixel verification: Furthermore, to avoid detecting false edge pixels, such as noise points, following two operations incorporating spatial information and region-based analysis are performed.

- First if a potential edge point has no potential edge point in its eight neighboring points, it must be treated as noise and removed from edge point collection.
- Second we also noticed that the gradient magnitude variance of two adjacent points in the same region is not apparent. So if the corresponding eigenvalue difference between a candidate edge pixel and its eight neighboring labeled edge pixels is always very large, the candidate edge pixel will not be selected as an edge pixel. Inequality (3) is defined to evaluate the difference:

$$\left(\sum_{0}^{\mathrm{Num}(i,j)-1} |V(i,j) - V(i+p, j+q)| - \sum_{0}^{\mathrm{Num}(i,j)-1} V(i+p, j+q)\right) > 0. \quad (3)$$

Given a candidate edge pixel (i,j), $\mathrm{Num}(i,j)$ is the count of its neighboring labeled edge pixel

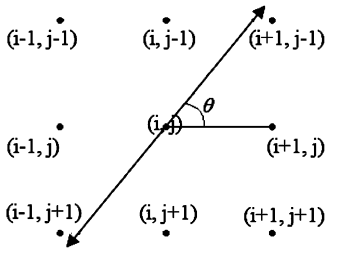

Fig. 1. Pixel (i,j) gradient direction.

$(i+p, j+q)$ and $\mathrm{Num}(i,j)$ value could be $1,\ldots,7$, with p and $q \in \{-1,0,1\}$. If (3) is true, the pixel (i,j) will be labeled as a non-edge point.

After removing false edge points, we can provide more accurate edge information for next line segment detection and reduce the number of subsequent searches.

4. Line segment detection and combination

After we get the edge map of a color image, we need to link neighboring edge pixels with similar orientation to a set of straight line segments by tracing the pixels on the edge to prepare data for later rectangle detection. Due to effect of noise and illumination, the edges of some objects are incomplete and do not make up one continuous line. So it is necessary to merge some neighboring parallel line segments into a single line segment.

4.1. Line segment detection algorithm

In this stage, we group neighboring edge pixels with similar orientation into straight line segments.

Because edge pixels are an unordered list of points in two dimensional Cartesian space, like *HT* we first map each pixel (x,y) into a curve $\rho = x\cos\vartheta + y\sin\vartheta$ in $\rho-\vartheta$ plane, where ρ is the normal distance of the desired line from the origin and ϑ is the angle that the normal to the line makes with positive x-axis.

Although HT can find infinite straight lines on which the image edges lie, a straight line in an image has two ends and HT [16] does not find end-points. So we will store the two end pixel coordinates corresponding to a point (ρ_0,ϑ_0) in $\rho-\vartheta$ plane, which can be mapped into a straight line $\rho_0 = x\cos\vartheta_0 + y\sin\vartheta_0$ in $x-y$ plane. Furthermore, edge pixel orientation obtained in previous stage can greatly reduce computation burden because each edge pixel only votes for one bin in the accumulator array with ϑ fixed by the gradient direction.

To fit a straight line to a set of data points, we quantize ρ and ϑ with $\delta\rho = 1$ and $\delta\vartheta = 2$, which are sufficiently accurate for line segment detection task. Then following steps are performed to detect line segments from the edge map.

- Scan the image from left to right and from top to bottom.
- For each edge pixel (i,j), we compute its polar coordinate $\rho(i,j) = i\times\cos\theta(i,j) + j\times\sin\theta(i,j)$ in $\rho-\vartheta$ plane, where $\theta(i,j)$ is its gradient direction angle. After this, we will group (i,j) according to following two cases:
  - If no line segment lying along the perpendicular direction defined by $\theta(i,j)$ is connected with (i,j), then a new line segment is generated with its start and end labeled with (i,j), else
  - the starting and ending pixels of the line segment, which has the same orientation and connects with (i,j), are updated by computing the Manhattan distance (The Manhattan distance between two points is measured along axes at right angles. Given two pixels $p1(x1,y1)$ and $p2(x2,y2)$, the Manhattan distance between $p1$ and $p2$ is $|x1-x2|+|y1-y2|$.) between two ends and (i,j).

After completing the scan of the whole image, all detected line segments are output to the next step for further processing.

4.2. Line segment combination

Due to effect of noise or illumination variation, line segments detected in former procedure may be highly fragmented and a combination process is necessary. Moreover, line segment combination will reduce the time complexity of later rectangle boundary detection process. Therefore, we merge some straight line segments into a single segment, whose length and orientation are derived from the contributing segments, according to several perceptual grouping criteria.

- First, the merged line segments must have the same orientations.
- Second, the distance among line segments should be very small (within 2 pixel distance).
- Third, parallel line segments should not overlap a significant portion when they are projected in the direction perpendicular to the line.

With criteria described above, some neighboring parallel segments are merged into a single segment. If a line segment is longer than the minimum length, it will be added into the output list to prepare data for subsequent rectangle detection.

5. Rectangle detection algorithm

After obtaining all line segments in image edge map, we classify certain line segments, which are

boundaries of different rectangular shape objects existing in image, through an MRF model.

MRF [15] has been demonstrated to be successfully applied to many image processing tasks. Because the probability of a line segment being one rectangle side mainly depends on that of its neighbors, we can build an MRF model to label line segments being rectangle sides or not. Detailed information about our MRF model is discussed below.

Let $L = \{l_1, l_2, \ldots, l_n\}$ be the line segment set we obtained in previous process. The set of sites $d = \{1, 2, \ldots, n\}$ indexes L. Let $F = \{F_1, F_2, \ldots, F_n\}$ be a family of random variables, in which each random variable F_i takes its value from 0 to 1 and indicates whether a line segment l_i lies on the boundary of a rectangular shape object or not. When $F_i = 1$, l_i is regarded as one rectangle side.

Because neighborhood selection is an important issue in MRF, we will first introduce the neighborhood system used in our algorithm.

5.1. Neighborhood system

Let $N(l_i)$ be the set of all line segments in L that are neighbors of l_i such that $l_i \notin N(l_i)$ and if $l_j \in N(l_i)$, then $l_i \in N(l_j)$.

In our algorithm, one line segment l_j will be regarded as one neighborhood of l_i only if it meets the following three requirements:

- First, l_j must be parallel or perpendicular to l_i, which is determined by the prominent geometry feature of rectangles. Let $H(i,j)$ be a measure of spatial relations between l_i and l_j. $H(i,j) = 1$ indicates that l_i and l_j are perpendicular to each other while $H(i,j) = 0$ indicates that l_i and l_j are parallel to each other.
- Second, the distance $D(i,j)$ between l_i and l_j should not be too large or be very small.

  If $H(i,j) = 0$, $D(i,j)$ equals to the length of the segment perpendicular to one line from one point lying on another line. Then $D(i,j)$ should be smaller than the maximum possible rectangle size MRS (the largest value of MRS can be image size) and greater than the minimum possible rectangle size MIS in image, i.e., $D(i,j) < \mathrm{MRS}$ and $D(i,j) > \mathrm{MIS}$.

  If $H(i,j) = 1$, $D(i,j)$ equals to the minimum distance between each end of one line segment and another line segment. In this case, $D(i,j)$ should be smaller than the half length of each line segment, i.e., $D(i,j) < \min(\mathrm{LEN}_i, \mathrm{LEN}_j)/2$, where LEN_i is the length of a line segment l_i.

  In Fig. 2, l_1 and l_2 are one parallel line pair and perpendicular to l_3. $D(1,2)$ is the distance between l_1 and l_2, $D(2,3)$ is the distance between l_2 and l_3.
- Third, if l_i and l_j are parallel to each other, they should overlap a significant portion when projected in direction perpendicular to the line. When the amount of overlap between two line segments exceeds 60% of each line length, one line segment will have chance to be a neighbor to the other one. We can illustrate our idea more clearly with Fig. 2.

  In Fig. 2, we project l_1 onto l_2 yielding segment AB. Only if $|AB| > (0.6 \times \max(\mathrm{LEN}_1, \mathrm{LEN}_2))$, then l_2 will be accepted as one neighbor of l_1.

With the neighborhood system defined above, the field F can be assumed to be an MRF with its local characteristics.

For

$$i \in d,\ P(F_i | F_j, j \in d, j \neq i) = P(F_i | F_j, j \in N(l_i)).$$

5.2. Labeling line segments

Let $f = \{f_1, f_2, \ldots, f_n\}$ be a configuration of F, i.e., $\{F_1 = f_1, \ldots, F_n = f_n\}$ and $N = \{N(l_i) | \forall i \in d\}$ be the collections of line segments neighboring to one line segment. Then we can calculate the posterior probability following Gibbs distribution as follows:

$$P(F = f | L) = Z^{-1} \exp[-E(f)/T], \tag{4}$$

where T is the temperature which is assumed to be one unless, otherwise stated, Z is the normalization factor and $E(f)$ is the posterior energy

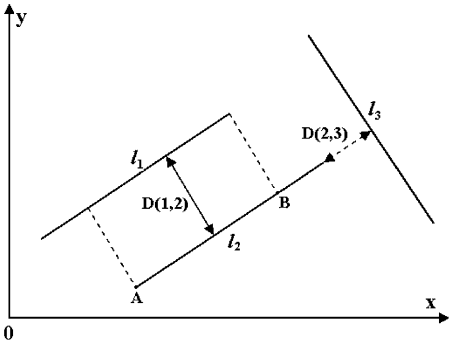

Fig. 2. Illustration of distance and overlapping area among line segments.

function.

$$E(f) = U(f)/T + U(l|f),$$

where $U(f)$ and $U(l|f)$ are the prior energy and the likelihood energy, respectively.

Now, we can define the rectangle side extraction as the following optimization problem.

For a given L, arg max $P(f_i|l_i)$ for each l_i, which is also equivalently found by

arg min $E(f)$.

Minimize the energy function will maximize the probability defined by Eq. (4).

Maximizing the probability defined by Eq. (4) will give the maximum posterior estimate of potential rectangles' sides in image edge map. The energy function E of our model is minimized while each f_i is converged to either 0 or 1. E consists of the following four terms:

$$E_1 = \alpha_1 \sum_{i=1}^{n} \sum_{j \in N(l_i)} f_i f_j D(i,j), \tag{5}$$

$$E_2 = \alpha_2 \sum_{i=1}^{n} \left(\frac{f_i}{\mathrm{LEN}_i} \times \mathrm{AVGLEN} \right), \tag{6}$$

$$E_3 = -\alpha_3 \sum_{i=1}^{n} (f_i \ln f_i + (1 - f_i) \ln(1 - f_i)), \tag{7}$$

$$E_4 = \alpha_4 \sum_{i=1}^{n} \sum_{j \in N(l_i)} f_i f_j \left\{ H(i,j) \times \min_{k \in (N(l_i) \cap N(l_j))} \frac{D(k,j) + D(k,i)}{f_k} + (1 - H(i,j)) \times \min_{k \in (N(l_i) \cap N(l_j))} \frac{D(k,j) + D(k,i)}{f_k H(k,j)} \right\}, \tag{8}$$

where α_1, α_2, α_3 and α_4 are positive constants, which control the contributions of the individual energy term to the value of the energy function. AVGLEN is the average length of line segment set L:

$$\mathrm{AVGLEN} = \frac{1}{n} \sum_{i=1}^{n} \mathrm{LEN}_i.$$

E_1 supports grouping of line segments that are close to each other. E_2 favors long line segments. E_3 is the entropy of the configuration $\{f_i\}$ hence pushes the $\{f_i\}$ to either 0 or 1. E_4 favors two neighboring line segments, both of which are close to one of their mutual neighbor line segments.

The $\{f_i\}$ are all initialized to 0.5 and the energy is gradually reduced by using a gradient descent algorithm with a learning rate scheduling scheme:

$$\{f_i\}_{t+1} = \{f_i\}_t - \eta_t \nabla E, \tag{9}$$

$$\eta_{t+1} = \eta_t - \beta, \tag{10}$$

where η is a positive step-size parameter, which influences convergence rate and final labeling result, β is a small constant, which makes the step size linearly decreased at each iteration, and ∇E is the gradient of E.

Upon minimizing the energy, we can select certain line segments, which are boundary segments of unknown rectangular shape objects in image, with its label parameter $f_i \approx 1$ from L. Then four sides of the same rectangle will be grouped together by the spatial information and geometrical relations among line segments, which can be obtained from our neighborhood system.

Since one rectangle is uniquely determined by four linked line segments, we built our neighborhood system, which fully exploits this fact. Given two line segments, only when they are close to each other and parallel or perpendicular to each other, they have the possibility to become neighborhoods.

After we labeled those line segments, which lie on the boundaries of unknown rectangular objects, we intend to search their neighboring line segments to form integrated rectangles. Given one line segment l, which is labeled as one side of unknown rectangular objects, we will select three additional labeled line segments, in which one line segment is parallel to l and two others are perpendicular to l, from its neighborhoods to form a rectangle. Furthermore, those three labeled line segments must be the neighborhoods of each other.

With the scheme described above, we can locate rectangular objects in color images by grouping four labeled line segments together.

6. Experimental results

In order to evaluate the actual performance of our proposed algorithm, we implemented the algorithm in C++ language under Windows-XP on an EPSON Endeavor MT7000 PC equipped with Intel PIV 2.4 GHz processor and 1 GB RAM.

Our image database contains 513 real color images which include variant rectangular shape objects. The resolution of test images varies from 73×42 to 3072×2048 pixels. Rectangular object appearance varies with different size, orientation

and colors. The size of rectangular objects in images ranges from 7 to 1605 pixels.

6.1. Choice of parameters

During the experimentation, considering tradeoff between cost and performance, we set different values for parameter MRS (maximum rectangle size), MIS (minimum rectangle size) and η_0 (initial learning rate), which makes our algorithm more flexible.

Both MRS and MIS influence the time complexity and detection accuracy of our algorithm. So we change both parameters adaptively according to variant image size such that

$$\mathrm{MRS} = \max(\text{image width}, \text{image height}),$$

$$\mathrm{MIS} = \max(5, \min(\text{image width}, \text{image height})/100),$$

which can provide satisfactory accuracy in most cases.

η_0 is another important parameter of our rectangle detection algorithm. If η_0 is too small, our algorithm will take a long time to converge. If η_0 is too large, our algorithm may oscillate and the detection result becomes unstable. So η_0 is set to different value according to variant image complexity such that

$$\eta_0 = \frac{\text{number of line segments in image}}{(\text{image width}) \times (\text{image height})}.$$

In addition, the setting of the four coefficients: α_1, α_2, α_3 and α_4 affects the efficiency and accuracy of our detection result. So we have done a large amount of experiments to figure out which parameter setting is the best one for our algorithm. We examined different settings of those four parameters with 100 images selected from 513 test images, which have variant complexity. From experimental results we discovered that the following relationship is best suited for our algorithm:

$$\alpha_2 = \alpha_1/5, \quad \alpha_3 = 2 \times \alpha_1, \quad \alpha_4 = 2.5 \times \alpha_1.$$

6.2. Detection evaluation

For measuring accuracy, false alarm rate and detection rate are calculated to evaluate our algorithm performance. Detection rate is defined as follows:

$$[\text{Detection rate}] = \frac{\text{number of detected true rectangular objects}}{\text{total number of rectangular objects in images}}.$$

False alarm rate is evaluated as follows:

$$[\text{False alarm rate}] = \frac{\text{number of false alarms}}{\text{number of detected true rectangular objects} + \text{number of false alarms}}.$$

Due to limited space, only one test image with 770 × 889 pixels is illustrated in Fig. 3 (note that the image has been shrunk). Fig. 4 shows the image edge map obtained by using our edge extraction algorithm discussed in Section 3, which has 86,116 edge pixels. The precise gradient direction of each edge pixel computed from Eq. (2) is represented with short green line in Fig. 5. In Fig. 6, we depict all line segments detected from the image edge map with

Fig. 3. Initial color image.

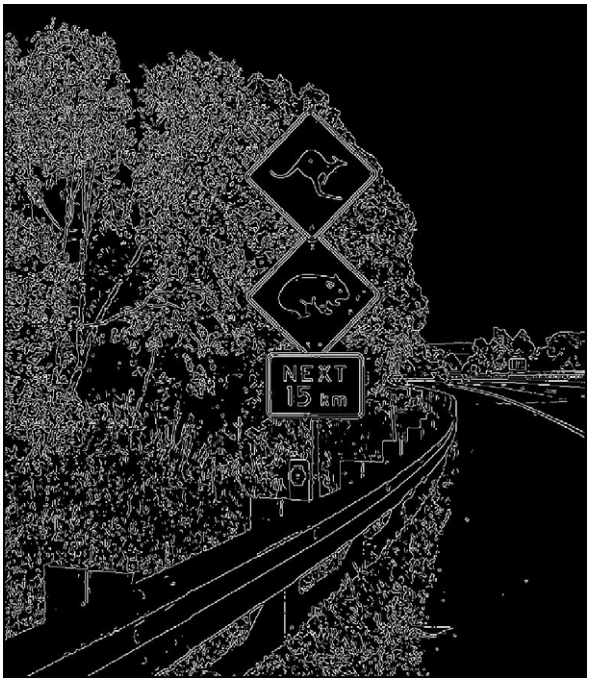

Fig. 4. Edge image of the tested image.

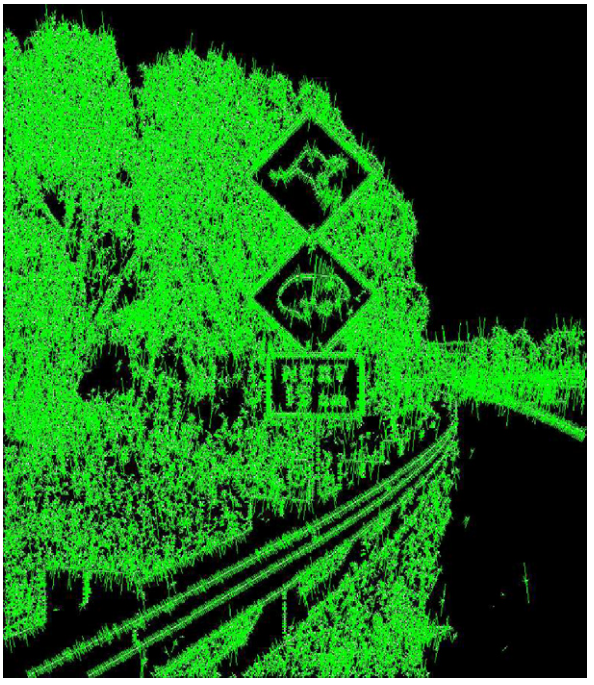

Fig. 5. Eigenvector direction map.

red line segments, whose number is 9055. Then rectangular shape objects are detected via an MRF model built on line segments and the detection result is shown in Fig. 7, where blue line segments

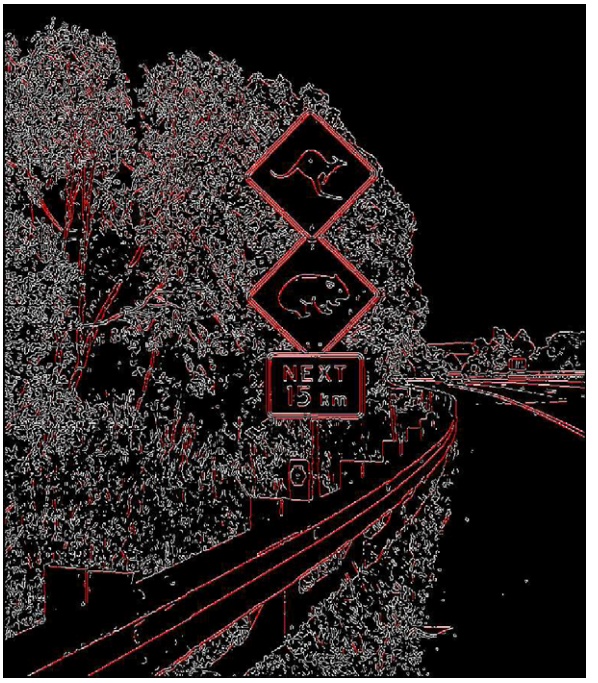

Fig. 6. Line segment detection result.

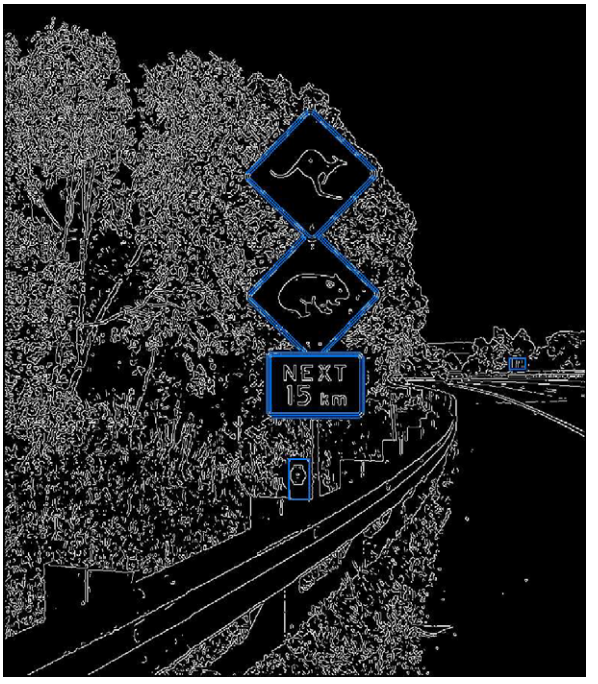

Fig. 7. Rectangular object detection result with MRS = 889, MIS = 9 and $\eta_0 = 0.0132$.

represent the rectangular shape object boundary. As it can be noticed, all rectangular shape objects contained in Fig. 3 were successfully detected (the

parameters used wereMRS = 889, MIS = 9 and $\delta_0 = 0.0132$). The total processing time of this image is 5.890 s. The edge detection process takes only 0.515 s and the line segment detection and combination process consumes 0.859 s. More than 75% of the total execution time is consumed by rectangle detection task.

Compared with most previous rectangle detection algorithms, like [2,9], which directly estimate the five parameters of an arbitrarily oriented rectangle with high time complexity and space requirement, our algorithm explores the merit of MRF to allow a global optimization problem to be simplified and solved locally, whereby the computation cost is minimized.

Table 1 summarizes the computation time of our algorithm and RHT on 513 real color images, from which we can see that our algorithm is much more efficient than RHT. The detection result achieved by our algorithm and RHT is summarized in Table 2, which shows the evidence that our algorithm has much higher detection rate and lower false alarm rate than RHT. To make the comparison more impressive, we depict the detection result of Table 2 with one graph (Fig. 8).

But we also noticed that one problem associated with our algorithm is that when rectangular shape object boundary in color images is blurred with cast shadowed by other objects, it is difficult to get precise edge information and locate such object accurately. So we hope to integrate color constancy technique to our algorithm to make the rectangular shape object detection approach more robust in future.

7. Conclusions

In this paper, we regard the rectangle detection task as an optimization problem and propose an MRF model-based algorithm to extract multiple rectangular shape objects simultaneously from color images.

The resulting algorithm has demonstrated high accuracy and efficiency with real color images containing multiple rectangular shape objects with different size, orientation and color. Furthermore,

Table 1
Average execution time (s) of our algorithm and RHT

N_S	N_N	N_R	T_E (s)	T_L (s)	T_R (s)	T_H (s)
$<380 \times 330$	53	42	0.051	0.092	0.719	8.329
$<650 \times 480$	78	97	0.155	0.219	3.562	31.728
$<960 \times 1280$	130	396	0.422	0.897	6.891	70.129
$<1786 \times 1680$	220	364	1.376	1.953	14.187	201.325
$<3072 \times 2048$	32	36	3.756	5.382	35.012	519.431

N_S: image size (pixels); N_N: number of images; N_R: number of rectangular objects in images; T_E: average execution time of our edge extraction; T_L: average execution time of our line segment detection; T_R: average execution time of our rectangle detection; T_H: average execution time of rectangle detection by using RHT, which is the average of 15 trials.

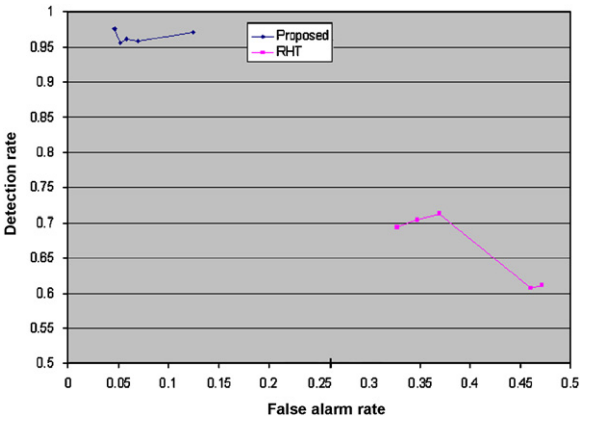

Fig. 8. Detection result.

Table 2
Rectangular object detection result of our algorithm and RHT

N_S	N_N	False alarm rate		Detection rate	
		Proposed (%)	RHT (%)	Proposed (%)	RHT (%)
$<380 \times 330$	53	4.7	34.8	97.6	70.4
$<650 \times 480$	78	7.0	47.2	95.9	61.2
$<960 \times 1280$	130	5.9	32.8	96.2	69.4
$<1786 \times 1680$	220	5.2	37.0	95.6	71.3
$<3072 \times 2048$	32	12.5	46.1	97.2	60.7

with different selection of neighborhood system, our algorithm can also be extended to detect some objects with low-degree polynomial curve boundary, such as parallelogram shape objects.

References

[1] S. Noronha, R. Nevatia, Detection and modeling of buildings from multiple aerial images, IEEE Trans. Pattern Anal. Machine Intell. 23 (5) (2001) 501–518.

[2] C.R. Jung, R. Schramm, Rectangle detection based on a windowed Hough transform, in: Proceedings of the 17th Brazilian Symposium on Computer Graphics and Image Processing, October 2004, pp. 113–120.

[3] C. Herley, Recursive method to extract rectangular objects from scans, in: Proceedings of the International Conference on Image Processing, vol. 3, September 2003, pp. 989–992.

[4] L. Hopwood, W. Miller, A. George, Parallel implementation of the Hough transform for the extraction of rectangular objects, in: Bringing Together Education, Science and Technology, Proceedings of the IEEE Southeastcon '96, April 1996, pp. 261–264.

[5] C. Lin, R. Nevatia, Building detection and description from a single intensity image, Comput. Vision Image Understanding 72 (2) (1998) 101–121.

[6] J. lllingworth, J. Kittler, A survey of the Hough transform, Comput. Vision Graph. Image Process. 44 (1) (1988) 87–116.

[7] H. Kalviainen, P. Hirvonen, An extension to the randomized Hough transform exploiting connectivity, Pattern Recognition Lett. (1997) 77–85.

[8] X. Li, D. Doermann, W.-G. Oh, W. Gao, A robust method for unknown forms analysis, in: Proceedings of the Fifth International Conference on Document Analysis and Recognition, September 1999, pp. 531–534.

[9] Y. Zhu, B. Carragher, F. Mouche, C. Potter, Automatic particle detection through efficient hough transforms, IEEE Trans. Med. Imaging 22 (9) (2003) 1053–1062.

[10] G. Roth, M.D. Levine, Extracting geometric primitives, Comput. Vision Graph. Image Process. Image Understanding 58 (1) (1993) 1–22.

[11] E. Lutton, P. Martinez, A genetic algorithm for the detection of 2D geometric primitive in images, in: Proceedings of the 12th International Conference on Pattern Recognition, vol. 1, October 1994, pp. 526–528.

[12] G. Roth, M.D. Levine, Geometric primitive extraction using a genetic algorithm, IEEE Trans. Pattern Anal. Machine Intell. 16 (9) (1994) 901–905.

[13] J. Fan, D.K.Y. Yau, A.K. Elmagarmid, W.G. Aref, Automatic image segmentation by integrating color-edge extraction and seeded region growing, IEEE Trans. Image Process. 10 (10) (2001) 1454–1466.

[14] H.-C. Lee, D.R. Cok, Detecting boundaries in a vector field, IEEE Trans. Signal Process. 39 (5) (1991) 1181–1194.

[15] X. Wang, H. Wang, Evolutionary optimization with Markov random field prior, IEEE Trans. Evol. Comput. 8 (6) (2004) 567–579.

[16] P.R. Thrift, S.M. Dunn, Approximating point-set images by line segments using a variation of the Hough transform, Comput. Vision Graph. Image Process. 21 (1983) 383–394.