Perspective rectification for mobile phone camera-based documents using a hybrid approach to vanishing point detection

Xu-Cheng Yin Jun Sun Satoshi Naoi
Fujitsu R&D Center Co. Ltd, Beijing, China
{xuchengyin; sunjun; naoi}@cn.fujitsu.com

Yusaku Fujii Katsuhito Fujimoto
Fujitsu Laboratories Ltd, Kawasaki, Japan
{fujii.yusaku; fujimoto.kat}@jp.fujitsu.com

Abstract

Documents captured by a mobile phone camera often have perspective distortions. In this paper, a fast and robust method for rectifying such perspective documents is presented. Previous methods for perspective document correction are usually based on vanishing point detection. However, most of these methods are either slow or unstable. Our proposed method is fast and robust with the following features: (1) robust detection of text baselines and character tilt orientations by heuristic rules and statistical analysis; (2) quick detection of vanishing point candidates by clustering and voting; and (3) precise and efficient detection of the final vanishing points using a hybrid approach, which combines the results from clustering and projection analysis. Our method is evaluated with more than 400 images including paper documents, signboards and posters. The image acceptance rate is more than 98% with an average speed of about 100ms.

1. Introduction

Recently, camera-based document analysis becomes a hot research field [6][10]. With widespread usage of the cheap digital ***cam***era built-in the ***mobile*** phone (***MobileCam*** in abbreviation thereafter) in people's daily life, the demand for simple, instantaneous capture of document images emerges. Different from the traditional scanned image, lots of the MobileCam-based document images have perspective distortions (see Fig. 6(a)(b)). Consequently, rectifying MobileCam-based perspective document images becomes an important issue.

As a general computer vision problem, most perspective correction methods rely on vanishing point detection. And these methods involve extracting multiple lines and their intersections, or using texture and frequency knowledge [4][11]. In document analysis, there are also various works on correction of perspective documents captured by general digital cameras [2][3][7][8][9][12][13]. Many of these methods use document boundaries and text lines to vote and detect vanishing points. And other methods take advantage of information of text lines and character tilt orientations.

We divided the methods of vanishing point detection into two sub groups: direct approaches and indirect approaches. The direct approaches directly analyze and calculate on image pixels, such as projection analysis from a perspective view for horizontal vanishing point detection [2]. These approaches have rather good precisions. But the search space in such approaches is an infinite 2D space, and a full or partial search of the space is computationally expensive, even impossible. The indirect approaches convert the original space into a clue space, and search the vanishing point in that new small space. Most of the indirect approaches involve extracting multiple straight or illusory lines[1] and voting vanishing points by model fitting. To calculate the horizontal vanishing point, some methods explicitly fit a line bundle in the linear clue feature space [8][9]. Some researchers use the spacing between adjacent horizontal lines of text to vote the vertical vanishing point [2][8]. Lu et al. use character stroke boundaries and tip points to correct perspective distortions based on multiple fuzzy sets and morphological operators [7]. Liang et al. use the equal text line spacing property to calculate and vote vanishing points, and they suggest their method can be applied on mobile devices [12][13]. These indirect approaches are time efficient. However, the model fitting in these methods are sensitive.

Moreover, in MobileCam-based document analysis, the captured images are always in low resolution with blurring, and the captured text contains often a partial portion of the whole document.

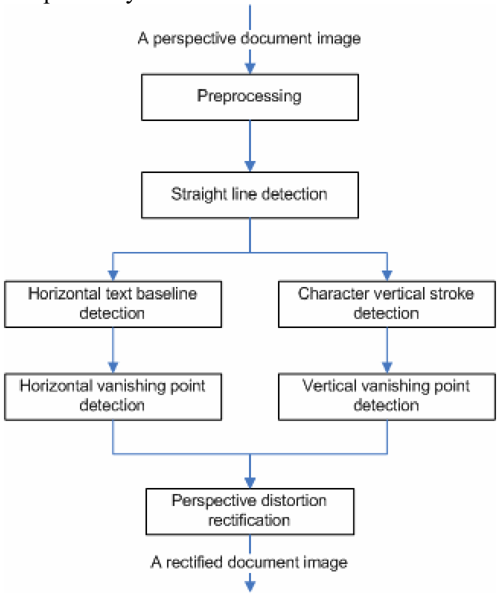
In conclusion, there are two main challenges for rectifying MobileCam-based perspective documents. First, the rectifying engine should be highly precise with fast

[1] The straight and illusory lines used in this paper are similar to the hard and illusory lines respectively termed in [9].

speed. The above methods can't cover the two issues well at the same time. Second, a MobileCam-based image is usually a partial portion of a whole document with few document boundaries. But many traditional methods mainly rely on document boundaries for correction.

Therefore, the traditional direct and indirect approaches have limitations for practical situations. To solve the above problems aiming at a practical MobileCam application, we focus on a fast and robust method for rectifying general perspective documents. First, we propose a hybrid approach for robust real-time vanishing point detection by integrating the direct and indirect approaches efficiently. As for the second challenge, we utilize horizontal text baselines and character tilt orientations as illusory lines to vote and compute vanishing points. Since the character strokes are used in line detection, paper boundaries are not necessary.

Our rectifying method is described in Fig. 1, where preprocessing includes grayscale image conversion, thresholding, and edge calculation, et al. The straight lines, horizontal text baselines, and character vertical strokes all are extracted by connected component analysis, heuristic rules and statistical analysis. Then, the hybrid approach clusters and detects the horizontal and vertical vanishing points respectively.



A perspective document image
Preprocessing
Straight line detection
Horizontal text baseline detection
Character vertical stroke detection
Horizontal vanishing point detection
Vertical vanishing point detection
Perspective distortion rectification
A rectified document image

Fig. 1. The flowchart of our method for perspective rectification.

The main contributions of this paper are as follows. The first contribution is a hybrid approach to vanishing point detection, which combines a greedy "indirect" method with a high-quality pixel-based "direct" method. The fast indirect method gives a set of candidates, which are refined afterwards using a direct approach. And the decision is made by linearly combining these two methods. The second contribution is the clustering learning for vanishing point candidates in the indirect method, which fast selects potential vanishing point candidates in the intersection point distribution from all straight and illusory lines. The third contribution is a robust and fast perspective distortion image rectifier, which is a working system for applications of mobile phone camera-based document images.

The remainder of this paper is organized as follows. Section 2 introduces the basic principle of our method. Section 3 explains how to extract the lines and the strokes in detail. In Section 4, we show the strategy of vanishing point detection. Section 5 is the experiments and result analysis. Finally we conclude the paper in Section 6.

2. Basic principle for vanishing point detection

The vanishing point (horizontal or vertical) in a 2D space can be described as (v_x, v_y), where v_x and v_y are the X and Y coordinates respectively in the 2D Euclidean space,

$$\Re^2 = \{(x,y) \mid -\infty < x < +\infty, -\infty < y < +\infty\}, \qquad (1)$$

where (0,0) is the center point of the image. In general, the vanishing point does not lie within the image itself.

Generally speaking, vanishing point detection is to find a proper point according to an optimization process in the image plane. That is to say,

$$(v_x, v_y) = \underset{(x,y)\in\Re^2}{\arg\max}\, f(x,y),$$

where $f(x,y)$ is the profit function for the optimization.

For the direct approaches for vanishing point detection, the search space is $\Re^2$ (equation (1)). Obviously, search in such a space is computationally expensive.

In this paper, we propose a novel and hybrid approach for vanishing point detection. Our approach first votes and clusters line intersections into vanishing point candidates. This process belongs to an indirect approach. Then projection analysis from perspective views on these candidates is performed, which is a direct approach. The vanishing point is obtained by an optimization of a function based on the previous two steps. The function can be expressed as following:

$$g(x,y) = G(f_{indirect}(x,y), f_{direct}(x,y)), \qquad (2)$$

where $f_{indirect}(x,y)$ and $f_{direct}(x,y)$ are the profit functions for the indirect and direct approaches respectively.

For vanishing point detection, first, we locate all straight and illusory lines. Then calculate all intersections for every line pair. The set of the resulting intersection points is

$$Set(Pt) = \{(x_1,y_1),(x_2,y_2),\ldots,(x_{N_{PT}},y_{N_{PT}})\},$$

where N_{PT} is the number of intersections. These points are partitioned into several groups by a clustering algorithm. Therefore, we get a new space $\boldsymbol{S}$,

$$S = \{S_i \mid S_i \subset \Re^2, i = 1,\ldots,N\}.$$

The center point of each cluster is regarded as a typical representation of its sub region, which is

$$C = \{c_i \mid c_i \in S_i, i = 1,\ldots,N\}. \qquad (3)$$

Rather than searching on the whole space in $\Re^2$, we search on the representative point set in ***C*** for speed up. Since the point set in ***C*** is representative enough, the searched maximum in ***C*** is a good approximation to the global maximum.

Consequently, the final resulting vanishing point is given by

$$(v_x, v_y) = \underset{(x,y)\in C}{\arg\max}\, g(x,y).$$

where *C* is defined in Equation (3).

Now, we perform a direct approach (e.g., projection analysis from a perspective view) on the new search space. Compared ***C*** in equation (3) with $\Re^2$ in equation (1), the search space of our hybrid approach is just composed by several points, which is much smaller than that of the direct approaches. Hence, it is time efficient. The above idea is used in horizontal and vertical vanishing point detection.

If the number of detected lines is enough, sufficient line intersections will be generated. And the true vanishing point will be embedded in these intersections with a high probability. Different from the conventional methods, our method finds the line candidates not only by paper boundaries lines but also by text baselines and the nature-born long character strokes in the text content. Therefore, the robustness of our method is improved.

3. Line and stroke detection and selection

When a document is lined up in a horizontal direction, we call it a horizontal document. Each text row has a clue horizontal direction. Our method uses document boundaries and text rows to detect the horizontal vanishing point. However, in the vertical direction of a horizontal document, there will be no text columns for vertical clues. Similar with the method described in [7], we extract the vertical character strokes as illusory vertical lines to detect a stable vertical vanishing point.

In Section 3.1, straight line detectin includes document boundaries and other straight lines. And text baseline detection is described in Section 3.2. Character tilt orientations are detected in Section 3.3.

3.1. Straight line detection

Line detection is solved with well-known conventional techniques. A lot of related work for such geometric problems is proposed, such as RANSAC-based methods, Least-Square-based methods, and Hough transform-based methods [14]. In this paper, in order to perform in a more efficient way, our line detection algorithms are based on edge information, connected component analysis, heuristic rules, and statistical analysis.

First, the input image is down-sampled. Then the edge is extracted by Canny edge detector [1]. Connected component analysis is used to find long connected components, which are merged in the horizontal or vertical direction according to shape and size information.

The merged connected components are regarded as line candidates. Given a connected component C_i, its corresponding line[2], LC_i, is fitted by the Least-Square algorithm.

The distance from one point (x,y) in C_i to the fitted line is

$$DIS_i(x,y) = \frac{|a_i y + b_i x + c|}{\sqrt{a^2 + b^2}}.$$

And we can get

$$f(LC_i) = \begin{cases} 1 & Len(LC_i) > len\_thres,\ N_{LC_i} > n\_thres\_line \\ 0 & otherwise \end{cases},$$

where

$$P_{LC_i}(x,y) = N(DIS_i(x,y), \mu_{line}, \sigma_{line}),$$

$$I_{LC_i}(x,y) = \begin{cases} 1 & P_{LC_i}(x,y) > p\_thres\_line \\ 0 & otherwise \end{cases},$$

and

$$N_{LC_i} = \sum_{(x,y)\in C_i} I_{LC_i}(x,y).$$

In the above equation, $Len(C_i)$ is the length of C_i, and $N(x,\mu,\sigma)$ is a Gaussian distribution for *LC* with mean μ and standard deviation σ. And μ_{line} and σ_{line} have been determined experimentally from different images.

If $f(LC_i)$ is equal to 1, then C_i will be a straight line.

Horizontal and vertical line detection and selection can be performed by the above steps respectively.

3.2. Horizontal text line smearing and detection

This process is based on a binarized image. We use a Block-Otsu algorithm for image binarization. After connected component analysis on the binary image, character candidates are selected by component size and shape analysis. Then, they are merged into horizontal text lines by a smearing algorithm derived from [5]. Finally, the horizontal direction of each smeared text line is computed.

The above procedure sometimes will produce smeared blocks that include more than one horizontal text lines because of perspective distortions. Therefore, we use a robust line direction detection method which is described in following.

First, we estimate the shape and size of each smeared text lines. Through vertical projection, we can obtain the upper and lower contour points of each smeared text line respectively. The upper contour points are

$$\{(x_1, y_1^U), (x_2, y_2^U), \ldots, (x_N, y_N^U)\},$$

where *N* is the width of the smeared text line. The lower contour points are

$$\{(x_1, y_1^L), (x_2, y_2^L), \ldots, (x_N, y_N^L)\}.$$

The average distance between each upper contour point and its corresponding lower contour point, *contour_thres*, is

[2] In this paper, the equation of lines is described as $ay+bx+c=0$.

then calculated.

If the distance of one contour point is less than *contour_thres*, then it is discarded. The reserved contour points are

$$Set(U) = \{(x_1, y_1^U), (x_2, y_2^U), \dots, (x_M, y_M^U)\}$$

And

$$Set(L) = \{(x_1, y_1^L), (x_2, y_2^L), \dots, (x_M, y_M^L)\},$$

where M is the number of the reserved contour points. And the middle points of the above contour points are

$$Set(C) = \{(x_1, (y_1^U + y_1^L)/2), (x_2, (y_2^U + y_2^L)/2), \dots, (x_M, (y_M^U + y_M^L)/2)\}.$$

Three lines are fitted by the Least-Square algorithm according to the above upper, lower and middle contour points respectively: "the upper baseline", "the lower baseline", and "the center baseline".

We select a smeared line as a real horizontal line when

$$cross\_angle(U, L) < angle\_thres$$

And

$$ave\_height(U, L) < height\_thres,$$

where U and L represent the upper and lower baselines respectively, *cross_angle* is the cross angle between two lines, and *ave_height* is the average height between the upper and lower baselines. Both *angle_thres* and *height_thres* are thresholds. And the horizontal direction of one text line is the direction of "the center baseline".

3.3. Character vertical stroke extraction

In many situations, vertical clues are scarce. When an image is a partial portion of a whole document, there may be few or even no straight vertical lines. But character tilt orientations can be regarded as clue directions and the character vertical strokes can be used as vertical clues. However, these vertical clues are not stable. Though vertical stroke detection is solved with several conventional techniques [7], our method is rather efficient for MobileCam-based document analysis. Different from the multiple fuzzy sets used in [7], our method extracts a stable vertical stroke set by heuristic rules and statistical analysis.

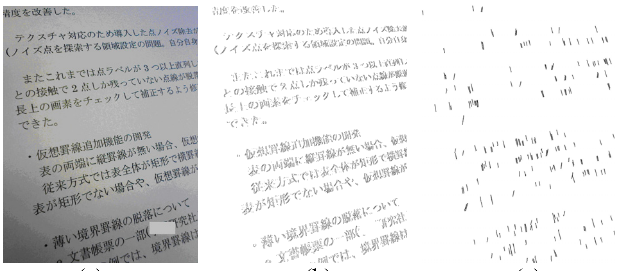

(a) (b) (c)

Fig. 2. Vertical character stroke detection: (a) captured image, (b) edge image, (c) vertical strokes detected.

The character vertical stroke extraction is also based on the edge image of the document. Fig. 2(a) is the original perspective document, and Fig. 2(b) shows an example edge image. After connected components analysis on the vertical edge image, some long-stroke-like connected components are selected.

These selected connected components include vertical, horizontal, and slant direction. On the assumption that the perspective angle in the vertical direction is less than *45* degree, we select vertical stroke candidates by a simple rule: if the height of a connected component is much longer than its width, it is a vertical stroke candidate. In Chinese, Japanese and English characters and texts, there are many curve strokes, such as "八", we remove these curve candidates by detecting the "straightness" of the stroke which is similar to the one described in Section 3.1.

Given a connected component C_i, its fitted line LC_i, and the distance from one point (x,y) in C_i to LC_i is $DIS_i(x,y)$, there is

$$f(LC_i) = \begin{cases} 1 & N_{LC_i} > n\_thres\_stroke \\ 0 & otherwise \end{cases},$$

where,

$$P_{LC_i}(x, y) = N(DIS_i(x, y), \mu_{stroke}, \sigma_{stroke}),$$

$$I_{LC_i}(x, y) = \begin{cases} 1 & P_{LC_i}(x,y) > p\_thres\_stroke \\ 0 & otherwise \end{cases},$$

and

$$N_{LC_i} = \sum_{(x,y) \in C_i} I_{LC_i}(x, y).$$

In the above equation, $N(x, \mu, \sigma)$ is a Gaussian distribution for LC with mean μ and standard deviation σ. And μ_{stroke} and σ_{stroke} are also determined experimentally. Because there are some noise and curves, we use the above steps to measure straightness of detected lines. That is, if one line is straight enough, it can be taken as a real line.

If $f(LC_i)$ is equal to 1, then C_i will be a vertical stroke. In order to detect straight vertical strokes, *p_thres_stroke* takes a high value (e.g., near to 1), and *n_thres_stroke* is near to the number of pixels in this component. The resulting vertical strokes of one document (Fig. 2(a)) are shown in Fig. 2(c).

Since in Chinese, Japanese and English texts, most slant strokes are curve strokes, after the above processing, the real vertical strokes are obtained. Consequently, the vertical vanishing point detection will be very robust.

4. Vanishing point detection by a hybrid approach

We use a hybrid approach for vanishing point detection. After the line intersections are calculated by line pairs, the intersection points are partitioned by clustering algorithm, and typical points are selected as reliable vanishing point candidates. This process can be viewed as an indirect approach. Next, a direct approach is performed by projection analysis from perspective views onto these point candidates. Finally, results of both approaches are linearly combined. The optimal candidate is selected as the final vanishing point.

4.1. Clustering for vanishing point detection

Without loss of generality, we describe the clustering based method for locating the horizontal vanishing point.

All horizontal lines (including straight lines and smeared lines detected in Section 3) are

$$Set(Line) = \{(a_1, b_1, c_1), ..., (a_N, b_N, c_N)\},$$

where N is the number of all horizontal lines.

As we know, two lines will produce an intersection point. As a result, there are $N_P = N \times (N-1)/2$ intersections which are possible candidates of the horizontal vanishing point. These intersections are

$$Set(Pt) = \{(x_1, y_1), ..., (x_{N_P}, y_{N_P})\}.$$

After checking the distribution of line intersections, we discover that these intersections are located in the 2D space with one or more groups with high density. It is natural to partition these points into groups by clustering. A sample of the distribution of intersections for a horizontal vanishing point is described in Fig. 3.

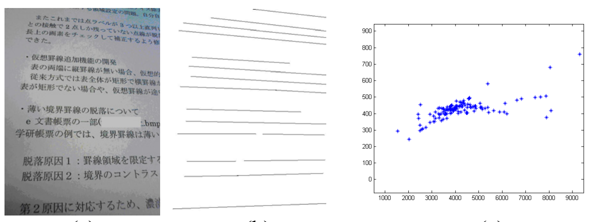


Fig. 3. Intersection distribution for a horizontal vanishing point: (a) captured image, (b) horizontal lines, (c) point distribution.

Our clustering space is 2D Euclidean space, and the similarity measure of two points is the Euclidean distance,

$$d(x_i, x_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2},$$

where (x_i, y_i) is the feature vector (a point in the space).

The k-means clustering algorithm is a rather robust clustering algorithm, which is also proved from our initial experiences. The number of clusters in k-means is specified by the following simple rule:

$$N_{cluster} = \max(\lceil \ln(N_P) \rceil,\ 10).$$

4.2. Vanishing point detection and selection

For horizontal vanishing point detection, the first task is to locate several vanishing point candidates based on clustering introduced in Section 4.1. After clustering, we will obtain $N_{cluster}$ clusters. Each cluster is composed by some intersection points. The number of points in each cluster is

$$Set(Num) = \{N_1, N_2, ..., N_{N_{cluster}}\}.$$

And the series of centers of all clusters is

$$X_C = \{x_1^c, x_2^c, ..., x_{N_{cluster}}^c\},$$

where the center of each cluster is the average of all the intersection points in this cluster.

In our method, these centers are candidates of the horizontal vanishing point. And each candidate has a weight from clustering. The weight is given by

$$w_i^c = N_i / \sum_{i=1}^{N_{cluster}} N_i.$$

Each of these weights can be regarded as the profit function for the indirect approach. that is

$$f_{indirect}(x_i, y_i) = w_i^c(x_i, y_i).$$

In order to get a more stable vanishing point, we use a direct approach to refine vanishing point candidates in the above search space.

As shown in [2], for a perspectively skewed target, the bins represent angular slices projected from $\boldsymbol{H}(x,y)$, and mapping from an image pixel $\boldsymbol{p}$ for a bin $B_i(x,y)$ is as follows:

$$i(p, H) = \frac{1}{2}N + N\frac{\angle(H, H-p)}{\Delta\theta} \qquad (4)$$

where $\angle(H, H-p)$ is the angle between pixel $\boldsymbol{p}$ and the center of the image, relative to the vanishing point $\boldsymbol{H}(x,y)$, and $\Delta\theta$ is the size of the angular range within the document is contained, again relative to $\boldsymbol{H}(x,y)$. $\Delta\theta$ is obtained from

$$\Delta\theta = \angle(T_L, T_R)$$

where T_L and T_R are the two points on the bounding circle whose tangents pass through $\boldsymbol{H}(x,y)$. These are shown in Fig. 4.

For each cluster center, the above projection analysis is performed, and the derivative-squared-sum of the projection profiles from a perspective view is calculated,

$$f'_{direct}(c_i(x,y)) = \sum_{j=1}^{N_B-1}(B_{j+1}(x,y) - B_j(x,y))^2. \qquad (5)$$

where $B(x,y)$ is a projection profile with a vanishing point candidate $\boldsymbol{H}(x,y)$, and N_B is the number of projection bins. This is the profit function for the direct approach.

For a computational convenience, the used profit is changed to a coefficient by

$$f_{direct}(c_i(x,y)) = \frac{f'_{direct}(c_i(x,y))}{\sum_{i=1}^{N} f'_{direct}(c_i(x,y))}.$$

Then according to equation (2), we combine these two profits in a linear way,

$$g(x_i, y_i) = \alpha f_{indirect}(x_i, y_i) + \beta f_{direct}(x_i, y_i).$$

where $\alpha + \beta = 1$. In our experiments, $\alpha = \beta = 0.5$.

The resulting horizontal vanishing point is given by

$$(v_x, v_y) = \arg\max_{(x_i, y_i) \in X_C} g(x_i, y_i).$$

The last step is to confirm the resulting vanishing point. Our rejection strategy is that the derivative-squared-sum of the true vanishing point will be larger than values of other points. The derivative-squared-sum of the resulting vanishing point is $f'_{direct}(v_x, v_y)$, which is calculated by equation (5). The unchanged horizontal vanishing point is $(-\infty,0)$. And the derivative-squared-sum of it is $f'_{direct}(-\infty,0)$. If the following condition is satisfied, then the final horizontal vanishing point is (v_x, v_y):

$$f'_{direct}(v_x, v_y) > (1+\varepsilon) f'_{direct}(-\infty,0),$$

where $0 < \varepsilon < 1$, and in our method, $\varepsilon = 0.1$. Otherwise, we take a rejection strategy, and the final vanishing point will be $(-\infty,0)$.

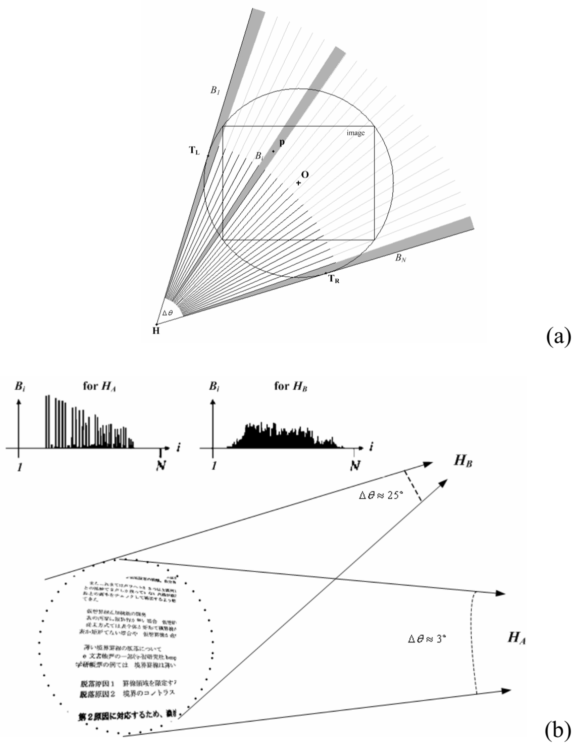


Fig. 4. Analysis of projection profiles from a perspective view: (a) generating projection profiles from ***H***(x,y), (b) the winning projection profiles from the vanishing point ***H***$_A$ and a poor example from ***H***$_B$ (derived from [2]).

The way for the vertical vanishing point detection and selection is similar to the horizontal vanishing point. In vertical vanishing point detection, we use a projection analysis method which is slightly different from the one used in [2].

In character segmentation, vertical white-space often serves to separate different characters. In a similar way, given a vertical vanishing point candidate ***V***(x,y), the bins represent angular slices projection from ***V***(x,y) of each text row. Similar with equation (4), mapping from an image pixel ***p*** in the *kth* text row for the bin $B_i^k(x,y)$ is as follows:

$$i(p,V,k) = \frac{1}{2} N_k + N_k \frac{\angle(V, V-p)}{\Delta\theta_k},$$

where N_k is the number of bins on the *kth* text row. And $\angle(V, V-p)$ is the angle between ***p*** and the center of the image, relative to the vanishing point ***V***(x,y), and $\Delta\theta_k$ is the size of the angular range within the *kth* text row is contained. Then, the profit function of the optimization becomes

$$f_{direct}(x,y) = \sum_{k=1}^{K} \sum_{i=1}^{N_k} (I_i^k(x,y)),$$

where $I_i^k(x,y) = 1$ or 0, and K is the number of text rows. If $B_i^k(x,y)$=0, then $I_i^k(x,y)$=1; otherwise, $I_i^k(x,y)$=0.

Consequently, a candidate with a maximum number of white columns of all rows from perspective views is the vertical vanishing point.

5. Experiments

The rectification transform of our system is performed as follows. Given the horizontal and vertical vanishing points, we can recovery documents with fronto-parallel views of the perspectively skew document. For perspective transformation on a *2D* plane, the transformation relation is

$$\begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & 1 \end{bmatrix} \begin{bmatrix} x_u \\ y_u \\ 1 \end{bmatrix}$$

where (x_u, y_u) is the rectified (undistorted) image coordinate, and (x_d, y_d) is the original (distorted) image coordinate.

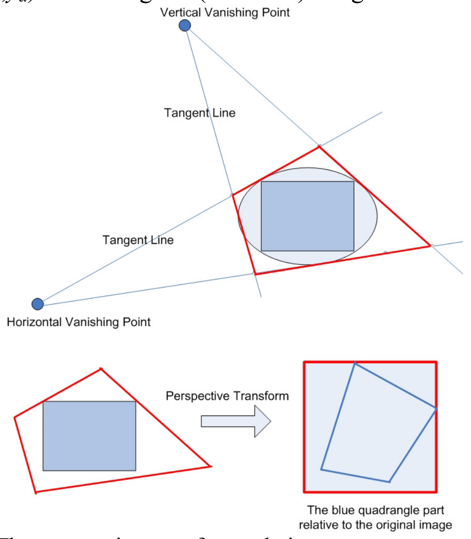

Vertical Vanishing Point
Tangent Line
Tangent Line
Horizontal Vanishing Point
Perspective Transform
The blue quadrangle part relative to the original image

Fig. 5. The perspective transform relation.

Given the horizontal and vertical vanishing points on the image plane, we can calculate a convex quadrangle on the image plane which is according to a rectangle in the rectified image plane. A versatile method for detecting such a convex quadrangle is described in Fig. 5. In Fig. 5, the ellipse is the circum-ellipse of the image rectangle.

The aspect ratio of the result image is decided as follows. The average length of the top and bottom sides of the quadrangle is the width of the result image, and the average length of the left and right sides of the quadrangle is the height of the result image.

The experiment database is *418* test samples captured by several mobile phone cameras. These images are in RGB color format with a 1280×960 resolution. More than *90%* of the images have perspective distortions, and other images have weak perspective distortions. Some samples are shown in Fig. 6.

Given a resulting vanishing point, VP, the relative distance from the ground truth VP_t is calculated. If

$$\frac{|VP - VP_t|}{|VP_t|} < T_{VP}, \tag{6}$$

then VP is regarded as a correct vanishing point. In our system, the ground truth vanishing points are calculated from the manually marked horizontal and vertical lines. When the difference in Equation 6 is less than the threshold (T_{VP}=*1/20*), then there is no seemingly perspective distortion. It is also a conclusion from our experiments.

We divide our rectified images into five groups: (1) "**HIT**", successful for perspective rectification in both horizontal and vertical directions; (2) "**HHIT**", successful in the horizontal direction; (3) "**VHIT**", successful in the vertical direction; (4) "**REJ**", the rectified image is the same as the original image; (5) "**ERR**", represents rectifying with wrong perspective angles.

We compared our method (**M1**) to other four methods: **M2** doesn't use character vertical strokes for vertical vanishing point detection; **M3** uses the indirect approach based on clustering only to detect vanishing points; **M4** uses model fitting in [9] instead of clustering; and **M5** uses a sequential correction style (horizontal direction correction then vertical direction correction) to compare the speed with our method. The accuracy results are described in Fig. 7. And some rectified images with front-parallel views of our method are shown in Fig. 6. And the resulting image is the inner rectangle area of the detected perspective quadrangle.

As shown in Fig. 6, test samples include many different types. There are even some street signs and non-document images. The fraction of these non-document images is about *20%*. The "**REJ**" rate of our method is *26.32%*, which is mainly caused by too large distortions. For a mobile phone with some proper interactive GUIs, users may accept the results of **HIT**, **HHIT**, **VHIT**, and **REJ** because the resulting image from these has a much better quality than (or a same quality as) the originally captured image. In this way, the acceptance rate of our method is *98.33%*.

Compared with M2, our method (M1) improves the "HIT" groups by 11.48%. This shows that character vertical strokes are very useful to detect the vertical vanishing point for documents without vertical boundaries. Compared with M3, our method improves 6.94% for the "HIT" accuracy. Our hybrid approach is more adaptive and robust for vanishing point detection. Compared with M4, our method improves the "HIT" accuracy by 2.39% and decreases the processing time by 12ms, which shows that our clustering strategy is robust and fast compared to the traditional model fitting. Our method has a similar performance with M5, though M5 uses a sequential style with partial rectification.

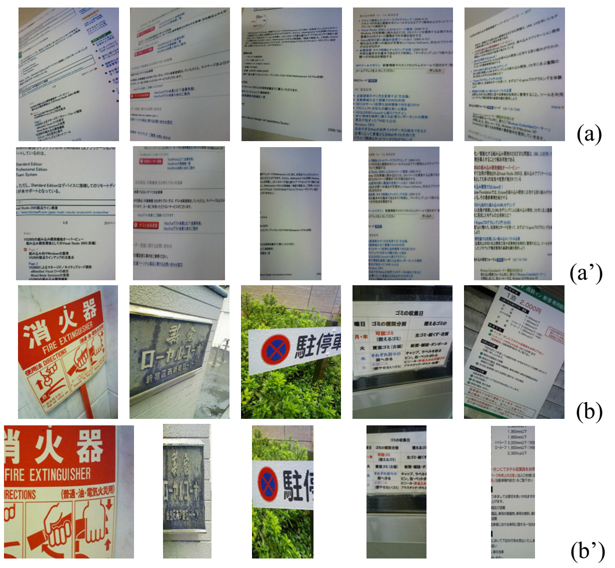


Fig. 6. Samples and the rectified images by our method (*x*' are the corresponding rectified images): (a) general documents, and (b) signboards and posters.

The processing speed is shown in Table 1, where "Time" represents the average processing time for each image without including the time for the grayscale image conversion and the final perspective transformation. Experiments are run on a DELL PC with 3GHz CPU, 2G Memory on a Windows XP OS.

Table 1. Results of the average processing time.

	M1	**M2**	**M3**	**M4**	**M5**
Time (ms)	103	72	90	115	226

As shown in Table 1, the average processing time of our method is largely less than M5, and the reduced time is 123ms. The additional time of M5 is spent in partially rectifying. We also test the direct approach by a hierarchical search for horizontal vanishing point detection in [2], which is more time consuming. For one image in test samples, the detection time is more than one second.

In conclusion, the acceptance rate of our method is more than *98%*, while the error rate is less than *2%*. And the processing time is only about *100*ms. With serious or unstable distortions, we take the rejection strategy, which may be more acceptable for a mobile user. Furthermore, with illusory lines derived from smeared text baselines and character tilt orientations, our method can rectify perspective documents without full boundaries (see Fig. 6(a)). All these show that our rectification method is fast and robust.

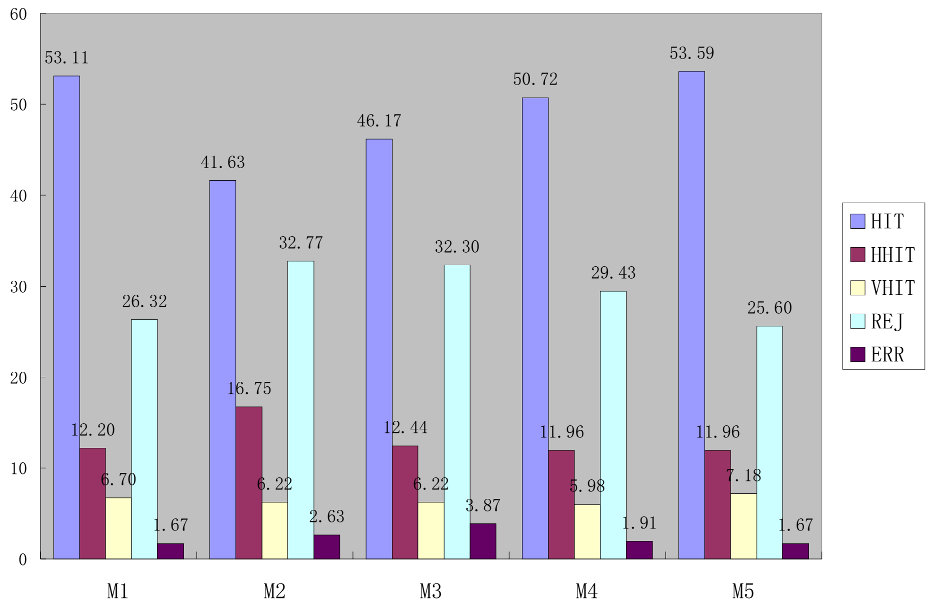


Fig. 7. Accuracy (%) of each resulting group.

6. Conclusions

Perspective rectification of MobileCam based documents faces several challenges, such as speed, robustness, non-boundary documents, etc. In this paper, we present a fast and robust method to deal with these problems. In our method, the hybrid approach for vanishing point detection combines direct and indirect approaches with high precision and fast speed. Furthermore, our method robustly detects text baselines and character tilt orientations by heuristic rules and statistical analysis, which is effective for documents without full boundaries. The experiments on different document images captured by mobile phone cameras show that our method has a good performance with an average speed of about *100*ms on a regular PC.

References

[1] J.F. Canny, “A computational approach to edge detection,” *IEEE Trans. on PAMI*, vol. 8, no. 6, pp. 679-698, 1986.

[2] P. Clark, and M. Mirmehdi, “Rectifying perspective views of text in 3D scenes using vanishing points,” *Pattern Recognition*, vol. 36, no. 11, pp. 2673-2686, 2003.

[3] C.R. Dance, “Perspective estimation for document images,” *Proceedings of SPIE Conference on Document Recognition and Retrieval IX*, pp. 244-254, 2002.

[4] W.L. Hwang, C.-S. Lu, and P.-C. Chung, “Shape from texture: estimation of planar surface orientation through the ridge surfaces of continuous Wavelet transform,” *IEEE Trans. on Image Processing*, vol. 7, no. 5, pp. 773-780, 1998.

[5] D.X. Le, G.R. Thoma, and H. Wechsler, “Automated borders detection and adaptive segmentation for binary document images,” *Proceedings of International Conference on Pattern Recognition*, vol. 3, pp. 737-741, 1996.

[6] J. Liang, D. Doermann, and H.P. Li, “Camera-based analysis of text and documents: a survey,” *International Journal on Document Analysis and Recognition*, vol. 7, no. 2-3, pp. 84-104, 2005.

[7] S.J. Lu, B.M. Chen, and C.C. Ko, “Perspective rectification of document images using fuzzy set and morphological operations,” *Image and Vision Computing*, vol. 23, no. 5, pp. 541-553, 2005.

[8] C. Monnier, V. Ablavsky, S. Holden, and M. Snorrason, “Sequential correction of perspective warp in camera-based documents,” *Proceedings of IEEE Conference on Document Analysis and Recognition*, vol. 1, pp. 394-398, 2005.

[9] M. Pilu, “Extraction of illusory linear clues in perspectively skewed documents,” *Proceedings of IEEE CVPR*, pp. 363-368, 2001.

[10] S. Pollard, and M. Pilu, “Building cameras for capturing documents,” *International Journal on Document Analysis and Recognition*, vol. 7, no. 2-3, pp. 123-137, 2005.

[11] J. Shufelt, “Performance evaluation and analysis of vanishing point detection techniques,” *IEEE Trans. on PAMI*, vol. 21, no. 3, pp. 282-288, 1999.

[12] J. Liang, D. DeMenthon, and D. Doerman, “Flattening curved documents in images,” *Proceedings of IEEE CVPR*, pp. 338-345, 2005.

[13] J. Liang, D. DeMenthon, and D. Doerman, “Camera-based document image mosaicing,” *Proceedings of International Conference on Pattern Recognition*, pp. 476-479, 2006.

[14] K.R. Castleman, *Digital Image Processing*, Prentice Hall, 1996.