Deskew for Card Image Scanning

Ilia Safonov, Ilya Kurilin
Samsung Research Center, Moscow, Russia
ilia dot safonov at samsung dot com

Abstract

Skew is a common defect of scanned images; therefore, deskew is a mandatory function of any scanning solution. Frequently users scan cards such as identity, bank, charge and business, etc. Often images of cards differ from common document images; and conventional deskew techniques do not perform well enough, especially for cards with light background. We propose a new robust skew-angle estimation algorithm based on iterative analysis of distance arrays for each column from top and bottom of extended bounding box to the nearest foreground pixel. Also rotation in situ is discussed that has minimal requirements to memory and is suitable for implementation in embedded system. Performance of proposed method corresponds to the best solutions in OCR and scanning software.

Keywords: *Deskew, Rotation in place.*

1. INTRODUCTION

Frequently consumers scan and copy various card-size documents such as postage cards and stamps, identity cards, passports, bank cards, certificates, charge cards, business cards, etc. The card image may be skewed by some angle due to inexact placing on the platen of a device. Skewed images should be corrected i.e. *deskew* has to be done. That is why deskew is mandatory function of any scanning solution and sometimes it is embedded in copying devices. Fig. 1 demonstrates deskew of a card image.

Card images have rectangular form like majority of document images, but usually card images significantly differ from conventional text document images: they have complex color background, number of text symbols is relatively small, and size of symbols can vary significantly, a lot of additional graphic elements present. That is why very often well-known deskewing methods fail.

We propose new fast and robust deskew algorithm intended for card images, but it operates well for any types of documents. Our technique is suitable for implementation in embedded systems because it has low computational complexity and requires small amount of memory due to *in situ* rotation approach.

2. RELATED WORK

There are plenty publication devoted to deskew algorithms. The key problem is reliable skew-angle estimation. A survey [4] classifies about 50 techniques in four groups: projection profile analysis, Hough or Radon transforms analysis, feature point distribution and orientation-sensitive feature analysis. All algorithms assume that an input document contains some amount of text. Recent publications [1, 3, 5, 6, 8] contain some improvements and specifications, but, in general, they use similar assumptions and approaches. A lot of modern techniques rely on text areas detection as first step and make skew-angle estimation for those areas only.

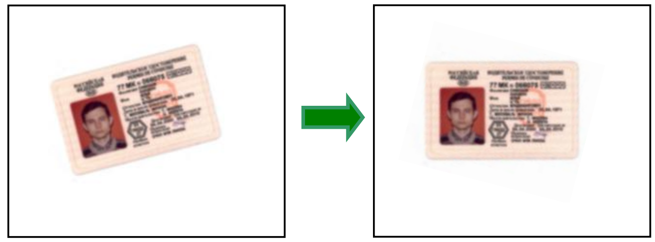

Figure 1: Example of deskew of card image.

Initially we collected test set from about 300 skewed card images and tried to apply those four general approaches to the test images. We did not do formal measurement of outcomes, but we evaluated applicability of each approach for our task only.

A straightforward solution for determining the skew-angle of a binary document image is to use a projection profile. Each element in the vertical projection profile is a count of the number of black pixels in the corresponding row of the image. This profile has the maximum energy when the text lines lie horizontally. In order to find skew-angle, the image or projection axis is rotated by a small step. Usually card image contains a lot of various objects on complex color background. It leads to noisy objects in binary image around and between text lines. Frequently maximum of projection profile energy does not correspond to zero-angle. Only about for 50% of card images skew-angle can be estimated with high precision by means of projection profile analysis.

Preliminary detection of text blocks makes outcomes a little bit better. For document segmentation we used approach described in [7]. Its performance corresponds to well-known OCR applications FineReader and ReadIRIS, but for images with complex background it is too hard to segment text areas only. In this way skew-angle is estimated well enough for about 65% of card images; and this result is not acceptable. Moreover, projection profile calculation for a great number of angles requires big computational resources. So, projection profile analysis is not applicable for our task.

Another class of technique for skew-angle detection reduces the number of operations that are performed in a projection analysis by first extracting the x-y coordinates of connected regions in an image. All subsequent computations are performed on those coordinates. Nearest objects are connected in chains or graphs. Skew-angle is calculated by means of approximation, for example by application of LSM. This technique is faster in comparison with profile analysis, but nevertheless computational complexity is high because labeling, chains and approximations are complex procedures. Skew-angle estimation outcomes are not so good too due to the same complexities.

The Hough transform is a well-known technique. In particular, it is used to detect straight lines. Techniques based on Hough transform operate well when edge detection is a preprocessing step to obtain binary image. Outer edges of cards have maximum magnitude in Hough space. Unfortunately, for cards with white background often outer edges are not detected at all. In this case inner noisy objects can lead to wrong skew-angle estimation. Fig. 2 shows two examples of improper skew-angle estimation by lines detection via Hough transform. Detected lines are marked by green. Only part of the lines is applicable for skew-angle estimation. Nevertheless for our test set the method based on Hough-transform allows to estimate properly about 95% of card images. Computational complexity of the algorithm for high-resolution image is high enough. Usually downsampled image is carried out to decrease processing time.

Figure 2: Examples of improper skew-angle estimation by lines detection via Hough transform.

Another type of approach for determining the skew angle detects the presence of local orientation-sensitive features in an image and uses their angles to vote for the skew-angle estimation. Such approach combines aspects of the feature extraction techniques and methods based on Hough transform. Several our modifications of orientation-sensitive feature analysis provided correct skew-angle estimation above 90%. Such way is promising; and we generated our own approach based on main concepts of orientation-sensitive feature analysis.

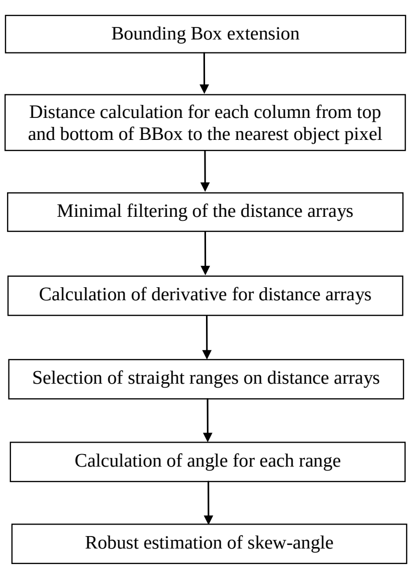

Figure 3: General scheme of skew-angle estimation.

3. SKEW-ANGLE ESTIMATION

3.1 General workflow

Let's we have card binary image BW. This image is computed by thresholding the brightness channel or edge detection. Usually combination of both techniques provides a better result. Bounding box is calculated via projection profile analysis. Fig. 3 demonstrates general scheme of skew-angle estimation.

In order to guarantee presence of image of entire card side inside bounding box we first extend bounding box:

$$ce_{\min} = \max(1, c_{\min} - (c_{\max} - c_{\min})/4),$$

$$ce_{\max} = \min(Nc, c_{\max} + (c_{\max} - c_{\min})/4),$$

$$re_{\min} = \max(1, r_{\min} - (r_{\max} - r_{\min})/4),$$

$$re_{\max} = \min(Nr, r_{\max} + (r_{\max} - r_{\min})/4),$$

where c is column index, r is row index. Nr – number of rows in image; Nc – number of columns in image, c_{min}, r_{min} are coordinates of left-top corner of initial bounding box, c_{max}, r_{max} are coordinates of right-bottom corner.

We calculate skew-angle by means of analysis of distances from top and bottom of extended bounding box to the nearest foreground pixel (see fig. 4). Such approach detects angle of outer edges when they are present. If background of card image is white then distance arrays reflects angle of inner objects. Iterative statistical estimation of angles of some number of objects allows to find skew-angle with high precision or to make decision about absence of skew or non-rectangular form of the image.

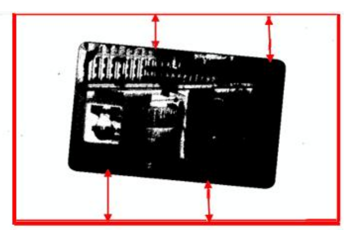

Figure 4: Distances from top and bottom of extended bounding box to the nearest foreground pixel.

3.2 Distances calculation

Arrays of distance from top and bottom of bounding box correspondingly are calculated by the formulae:

$$Dt(c) = \min((re_{\max} - re_{\min}),\ \min_{r \in [re_{\min}, re_{\max}]}(r \mid BW(r,c) = 1)),$$

$$Db(c) = \min((re_{\max} - re_{\min}),\ \max_{r \in [re_{\min}, re_{\max}]}(r \mid BW(r, ce_{\max} - c) = 1)),$$

where $c \in [ce_{\min}, ce_{\max}]$.

In order to suppress some noisy peaks we apply 1D erosion filter to both arrays:

$$Dtf(c) = \min(Dt(c-h), ..., Dt(c), ..., Dt(c+h)),$$

$$Dbf(c) = \min(Db(c-h), ..., Db(c), ..., Db(c+h)),$$

where aperture depends on scanning resolution, for 300 dpi $h = 4$.

3.3 Selection of straight line ranges

In order to detect straight ranges on distance arrays we calculate derivatives for these arrays:

$$dDtf(c) = Dtf(c+h) - Dtf(c-h),$$

$$dDbf(c) = Dbf(c+h) - Dbf(c-h).$$

Strictly speaking, dDtf and dDbf are not derivatives but finite differences. However, similarly to a lot of publications in image processing we will reference them as derivatives.

Straight line ranges are selected according to the following statements:

$$\{St(i)\} = \{\forall_{c\in[st1(i),st2(i)]} |dDtf(c)| \le k1, (st2(i) - st1(i)) \ge k2,$$
$$Dtf(st2(i)) < 3(re_{\max} - re_{\min})/4\}, i = 1..Nt,$$

$$\{Sb(j)\} = \{\forall_{c\in[sb1(j),sb2(j)]} |dDbf(c)| \le k1, (sb2(j) - sb1(j)) \ge k2,$$
$$Dbf(sb2(j)) < 3(re_{\max} - re_{\min})/4\}, j = 1..Nb,$$

where *{st1}* and *{sb1}* are coordinates of beginning of straight line ranges on distance arrays from top and bottom of bounding box correspondingly, *{st2}* and *{sb2}* are coordinates of ending of straight line ranges on distance arrays from top and bottom of bounding box, correspondingly, *Nt* and *Nb* are numbers of straight ranges on distance arrays from top and bottom of bounding box, correspondingly; constants *k1* and k2 depend on scanning resolution, for 300 dpi *k1* = 10, *k2* = 50.

Fig. 5 shows plots of distance from top array and its derivative as well as two ranges of straight lines.

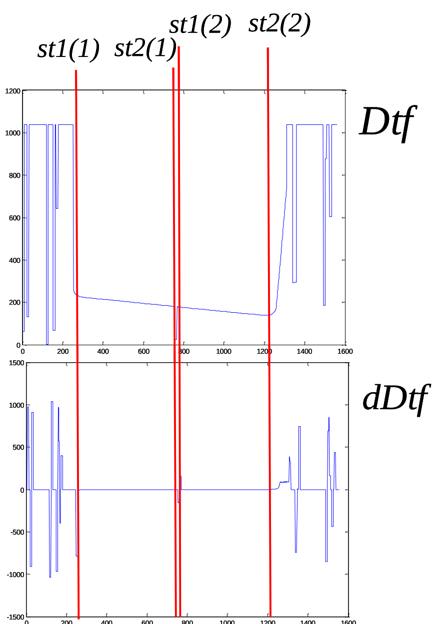

Figure 5: Distance and derivative of distance arrays.

3.4 Iterative estimation of skew-angle

For each straight line range we compute adjacent cathetus *dx*, opposite cathetus *dy* and angle *α*:

$$dx(i) = st2(i) - st1(i) - 2k1,$$

$$dx(j+Nt) = sb2(j) - sb1(j) - 2k1,$$

$$dy(i) = \frac{1}{2z+1}\sum_{k=-z}^{z} Dtf(st2(i) - k1 + k) - \frac{1}{2z+1}\sum_{k=-z}^{z} Dtf(st1(i) + k1 + k),$$

$$dy(j+Nt) = \frac{1}{2z+1}\sum_{k=-z}^{z} Dbf(sb2(i) - k1 + k) - \frac{1}{2z+1}\sum_{k=-z}^{z} Dbf(sb1(i) + k1 + k),$$

$$\alpha(n) = arctg(dy(n)/dx(n)),\ N = Nb + Nt,\ n = 1..N,$$

where *N* is total number of straight line ranges for both arrays of distances from top and bottom of bounding box; *z* is number of points for averaging, *z* depends on scanning resolution, for 300 dpi *z* = 2.

We propose rule for reliable skew-angle φ estimation: φ is weighted average of angles which differ from φ by variance less than 1. For appropriate angles selection iterative procedure is used. Initial estimation is calculated as:

$$\varphi = \sum_{n=1}^{N}\alpha(n)\times dx(n) / \sum_{n=1}^{N} dx(n),\quad \delta = \frac{1}{N-1}\sqrt{\sum_{n=1}^{N}(\varphi - \alpha(n))^2}.$$

If $\delta > 25$ then special processing for positive and negative angles which are close to 45° and -45° is performed:

$$bp(x) = \begin{cases} 1, x > 38^\circ \\ 0 \end{cases},\quad bn(x) = \begin{cases} 0, x < -38^\circ \\ 0 \end{cases},$$

$$qap = \sum_{n=1}^{N} bp(\alpha(n)),\quad qan = \sum_{n=1}^{N} bn(\alpha(n)),$$

$$sap = \sum_{n=1}^{N}\alpha(n)\times bp(\alpha(n)),\quad san = \sum_{n=1}^{N}\alpha(n)\times bn(\alpha(n)),$$

if $qap > 0$ and $qan > 0$, then $\delta=0$, if $sap > san$, then $\varphi = sap/qap$, otherwise $\varphi = san/qan$, where *bp* and *bn* are functions for indication of big positive and negative angles, correspondingly; *qap* and *qan* are numbers of big positive and negative angles; *sap* and *san* are sum of big positive and negative angles.

Iterations are continued while variance $\delta > 1$ and array $\alpha(n)$ is not empty: if $(\alpha(n) < \varphi - \delta)$ or $(\alpha(n) > \varphi + \delta)$ then $\alpha(n)$ and $dx(n)$ are excluded from corresponding arrays; further we make new estimation of weighted average of angles φ and variance δ as above. In case when angles significantly differ from each other, all angles are excluded from $\alpha(n)$ on some iteration. It corresponds to a situation, when we cannot make decision about skew-angle.

4. ROTATION IN PLACE

Image has to be rotated by minus skew-angle. Usually for rotation by angle φ the following transformation matrix is used:

$$[x' \quad y'] = [x \quad y]\begin{bmatrix}\cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi\end{bmatrix}.$$

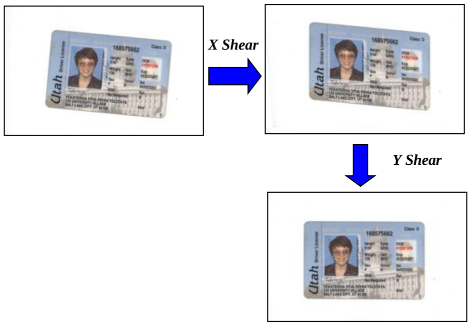

Figure 6: Example of rotation by two shears.

However in this case additional memory buffer for at least a part of rotated image is necessary. It can be impractical for copying devices where memory size is limited and/or operations with memory are relatively slow. It is preferable to use the same memory buffer where initial image is stored. Such algorithms are named 'in place' or *in situ*. Several papers describe decomposition of transformation matrix on two or three shears. Practical approach for rotation via two shears for rows and columns separately is proposed in [2] (see fig. 6 for example). The approach applies the following decomposition:

$$[x' \quad y'] = [x \quad y]\begin{bmatrix}\cos\varphi & 0\\ \sin\varphi & 1\end{bmatrix}\begin{bmatrix}1 & -\tan\varphi\\ 0 & \sec\varphi\end{bmatrix}.$$

Various interpolation algorithms can be applied during rotation. Bilinear interpolation is good trade-off between image quality and processing speed. Application of this technique allowed us to meet strong limitations of embedding systems.

5. RESULTS AND CONCLUSION

Proposed algorithm is extremely fast. Processing time on ARMv7 800 MHz for 1920x1080 RGB 24 bpp image is about 0.3 s without application of SIMD instructions. For 300 dpi scanning resolution our method requires about 150 Kb of additional memory only. So, it is applicable not only in software but also in firmware of embedded systems.

For conventional A4 and Letter-size document images paper [4] claims that a skew by as a little as 0.1° is noticeable for observer. Size of card images is smaller; and according to our investigation skew-angle of 0.1° is almost unnoticeable for naked eye. Survey among 14 observers has shown that skew-angle in 0.4° of card images is critical for visual perception.

For evaluation of deskew quality we randomly selected 10 cards from our test set and processed these cards by scanning software of the scanners HP ScanJet G4010, Epson Perfection V300 Photo and Canon CanoScan 8800F. Also we checked deskew function in well-known OCR applications ReadIRIS 11 and FineReader 10. We calculated percentage of deskewed images, average angle after deskew and percentage of cases, when angle after deskew is less than 0.4°. Table 1 contains results of our investigation. Our solution shares the first place with HP software and ReadIRIS.

We tested proposed algorithm for various types of document images different from card images. Proposed technique works well for all types of documents that we tested. Moreover, we applied this deskew technique for deskewing several rectangular objects placed on scanner platen and obtained good results. At the same time proposed algorithm is simple enough. Its implementation on C programming language contains about 400 lines only. So, it is an effective approach for plenty of scanning/copying applications and several other image correction tasks.

TABLE 1 DESKEW QUALITY

	Averaged angle after deskew, °	*Deskewed images, %*	*Angle after deskew is less than 0.4°, %*
HP G4010	**0.2**	**100**	**90**
Epson V300	3.5	40	30
Canon 8800	1.3	90	50
ReadIRIS	**0.2**	**100**	**90**
FineReader	2.2	60	60
Proposed	**0.2**	**100**	**90**

6. REFERENCES

[1] Beusekom, J., Shafait, F. & Breuel, T., *Resolution Independent Skew and Orientation Detection for Document Images*, Proc.of SPIE: Document Recognition and Retrieval XVI, 2009.

[2] Catmull, E., Smith, A., *3-D transformations of images in scanline order*, Proc. of SIGGRAPH'80, pp. 279–285, 1980.

[3] Chou, C.H., Chu, S.Y., Chang, F., Estimation of skewangles for scanned documents based on piecewise covering by parallelograms, Pattern Recognition 40, pp. 443 – 455, 2007.

[4] Hull, J., Document image skew detection: survey and annotated bibliography, Document Analysis Systems II, pp. 40-64, 1998.

[5] Konya, I., Eickeler, S., Seibert, C., *Fast Seamless Skew and Orientation Detection in Document Images*, Proc. of ICPR, pp. 1924-1928, 2010.

[6] Meng, G., Pan, C., Zheng, N., Sun, C,. *Skew Estimation of Document Images Using Bagging*, IEEE Transactions on image processing, vol. 19, No. 7, 2010.

[7] Vil'kin, A.M., Safonov, I.V., Egorova, M.A., *Bottom-up page segmentation based on texture features*, Proc. of PRIA, vol. 2, pp. 373-376, 2010.

[8] Yuan, B., Tan, C.L, *Fiducial line based skew estimation*, Pattern Recognition 38, pp. 2333 – 2350, 2005

About the author

Ilia V. Safonov received his MS degree in automatic and electronic engineering from Moscow Engineering Physics Institute/University (MEPhI), Russia in 1994 and his PhD degree in computer science from MEPhI in 1997. Since 2004, Dr. I. Safonov is with Samsung Research Center in Moscow, where he is engaged in image and video processing projects.

Ilya V. Kurilin received his MS degree in radio engineering from Novosibirsk State Technical University (NSTU), Russia in 1999 and his PhD degree in computer science from NSTU in 2006. At present time Dr. I. Kurilin is in Samsung Research Center, where he is engaged in photo and documents image enhancement projects.