RESTORATION OF IMAGES SCANNED FROM THICK BOUND DOCUMENTS

Zheng Zhang, Chew Lim Tan

School of Computing, National University of Singapore
3 Science Drive 2, Singapore 117543
Email: {zhangz, tancl}@comp.nus.edu.sg

ABSTRACT

Perspective distortion always occurs while scanning thick, bound documents. This distortion mainly causes two sources of degradation for the scanned grayscale image – i) shadow along the 'spine' of the book, and ii) warp of the words in the shadow. In this paper, we propose a restoration system to solve these two problems. Our system first produces a vertical projection profile to detect which side of the image the shadow lies on, and a run-length method is used to find the boundary between the shadow and the clean area. We then apply a modified Niblack's method to remove the shadow. We use a connected component analysis to adjust the location and orientation of the warped word in the shadow area, i.e. the words within the boundary detected earlier. The implementation results are presented in this paper. Our system will be used in a text retrieval project for National Archives of Singapore.

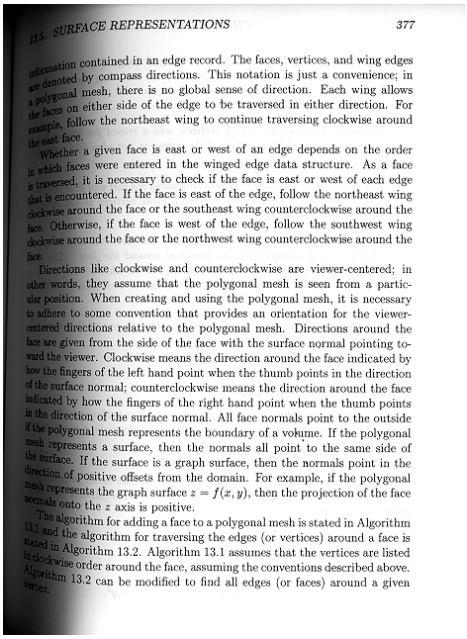
1. INTRODUCTION

While scanning pages from a thick, bound book, the page to be scanned is distorted and not flat on the document glass of the scanner. Such distortion mainly causes two sources of degradation for the scanned grayscale image – i) shadow along the 'spine' of the book, and ii) warp of the word in the shadow. There are many reasons for the need to restore such document image. First, in order to have a clear and easy view of the document image, it is necessary to remove the shadow, and restore the warped text line to a straight text line. Second, the restoration is necessary before using OCR to extract the text from the image.

In the current literature, there are some related techniques reported. Baird discusses a model of character degradation [1]. His model does not account for the nonlinear distortions produced from perspective distortion, which means it could not be used in our case. Baird improved his model to a 6-parameter model [2]. But the algorithm for estimating distributions on all of the model parameters to fit real image populations closely is still an open problem. Kanungo introduces a model for perspective distortion [3]. However, in his model there are some assumptions, and some parameters may not be known in reality. So in our system we do not adopt any degradation model. We use a modified Niblack's binarization method to remove the shadow, and a connected component analysis to adjust the warped words in the shadow.

2. SHADOW REMOVAL

A typical grayscale image for a page scanned from a thick, bound book is shown in Figure 1. As stated in the introduction, there is a shadow along the 'spine' of the book. The following subsections describe the procedures for shadow removal.


13.5. SURFACE REPRESENTATIONS 377

information contained in an edge record. The faces, vertices, and wing edges are denoted by compass directions. This notation is just a convenience; in a polygonal mesh, there is no global sense of direction. Each wing allows the faces on either side of the edge to be traversed in either direction. For example, follow the northeast wing to continue traversing clockwise around the east face.

Whether a given face is east or west of an edge depends on the order in which faces were entered in the winged edge data structure. As a face is traversed, it is necessary to check if the face is east or west of each edge that is encountered. If the face is east of the edge, follow the northeast wing clockwise around the face or the southeast wing counterclockwise around the face. Otherwise, if the face is west of the edge, follow the southwest wing clockwise around the face or the northwest wing counterclockwise around the face.

Directions like clockwise and counterclockwise are viewer-centered; in other words, they assume that the polygonal mesh is seen from a particular position. When creating and using the polygonal mesh, it is necessary to adhere to some convention that provides an orientation for the viewer-centered directions relative to the polygonal mesh. Directions around the face are given from the side of the face with the surface normal pointing toward the viewer. Clockwise means the direction around the face indicated by how the fingers of the left hand point when the thumb points in the direction of the surface normal; counterclockwise means the direction around the face indicated by how the fingers of the right hand point when the thumb points in the direction of the surface normal. All face normals point to the outside if the polygonal mesh represents the boundary of a volume. If the polygonal mesh represents a surface, then the normals all point to the same side of the surface. If the surface is a graph surface, then the normals point in the direction of positive offsets from the domain. For example, if the polygonal mesh represents the graph surface $z = f(x, y)$, then the projection of the face normals onto the z axis is positive.

The algorithm for adding a face to a polygonal mesh is stated in Algorithm 13.1 and the algorithm for traversing the edges (or vertices) around a face is stated in Algorithm 13.2. Algorithm 13.1 assumes that the vertices are listed in clockwise order around the face, assuming the conventions described above. Algorithm 13.2 can be modified to find all edges (or faces) around a given vertex.

Figure 1. A grayscale image scanned from a thick

2.1. Shadow boundary detection

Note that only the words in the shadow are warped. So according to the text line in the clean area, we can adjust the location and orientation of the warped words. This requires the knowledge of the boundary between the shadow and the clean area. This step is a preparation for warped words adjustment to be described in section 3, but obviously it must be done before the shadow removal.

Since the shadow lies on either the left side or the right side of the image, a vertical projection profile is produced first, and then the peak in the profile shows which side of the image the shadow lies on. We then perform a horizontal line-by-line scan of the image. Each line-scan starts from the side near the peak position. For each line, a break point, *b*, is found, and the boundary between shadow and clean area is defined as a set, *B*, of all the break points. Mathematically, *B* can be expressed as

$$B = \{(b,i) \mid 0 \le i < n, b = F(i,T)\} \qquad (1)$$

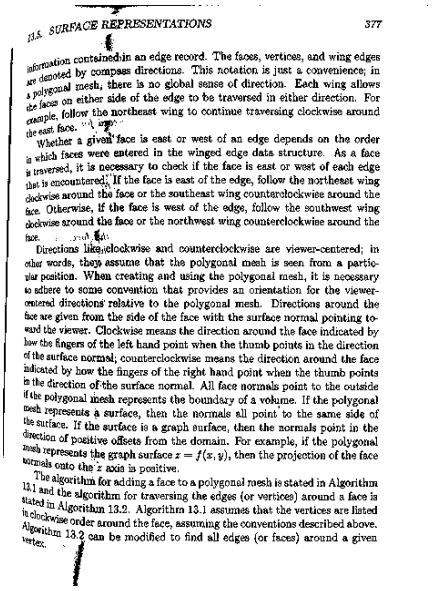
where *n* is the number of horizontal pixel lines in the image, *T* is a predefined threshold parameter, and *F* is a function returning the length of the first run of pixels in *ith* line whose intensity value is greater or equal to *T*. In our experiment, we use $T = V_{max} / 4$, where V_{max} is the maximum intensity value in the grayscale image, to obtain a good result for *B*.

2.2. Modified version of Niblack's binarization method

To remove the shadow, we use a modified version of Niblack's algorithm [4]. The idea of Niblack's method is to vary the threshold over the image, based on the local mean, *m*, and local standard deviation, *s*, computed in a small neighborhood (normally a window size 15*15 is used) of each pixel. A threshold for each pixel, *p(x,y)*, is computed from $T(x,y) = m(x,y) + k*s(x,y)$, where *m(x,y)* and *s(x,y)* are the local mean and local standard deviation calculated in a window centered at *(x, y)*, and *k* is a user defined parameter and is negative in value.

There are two reasons that we could not adopt Niblack's method directly. One is that the Niblack's method is sensitive to the *k* value. It is quite difficult to find a single *k* value that produces good results for most of our test images. The other is that the result always has much pepper noise in the shadow area, even if a proper *k* value is chosen.

In our modification, each standard deviation, *s(x,y)*, is normalized by dividing it by the dynamic range of standard deviation, *R*. Furthermore, the local mean, *m(x,y)*, is utilized to multiply, instead of adding, the standard deviation terms. These have the effect of amplifying the contribution of standard deviation, which produces results with much less pepper noise than the original one. These


13.5. SURFACE REPRESENTATIONS 377

information contained in an edge record. The faces, vertices, and wing edges are denoted by compass directions. This notation is just a convenience; in a polygonal mesh, there is no global sense of direction. Each wing allows the faces on either side of the edge to be traversed in either direction. For example, follow the northeast wing to continue traversing clockwise around the east face.

Whether a given face is east or west of an edge depends on the order in which faces were entered in the winged edge data structure. As a face is traversed, it is necessary to check if the face is east or west of each edge that is encountered. If the face is east of the edge, follow the northeast wing clockwise around the face or the southeast wing counterclockwise around the face. Otherwise, if the face is west of the edge, follow the southwest wing clockwise around the face or the northwest wing counterclockwise around the face.

Directions like clockwise and counterclockwise are viewer-centered; in other words, they assume that the polygonal mesh is seen from a particular position. When creating and using the polygonal mesh, it is necessary to adhere to some convention that provides an orientation for the viewer-centered directions relative to the polygonal mesh. Directions around the face are given from the side of the face with the surface normal pointing toward the viewer. Clockwise means the direction around the face indicated by how the fingers of the left hand point when the thumb points in the direction of the surface normal; counterclockwise means the direction around the face indicated by how the fingers of the right hand point when the thumb points in the direction of the surface normal. All face normals point to the outside if the polygonal mesh represents the boundary of a volume. If the polygonal mesh represents a surface, then the normals all point to the same side of the surface. If the surface is a graph surface, then the normals point in the direction of positive offsets from the domain. For example, if the polygonal mesh represents the graph surface $z = f(x, y)$, then the projection of the face normals onto the z axis is positive.

The algorithm for adding a face to a polygonal mesh is stated in Algorithm 13.1 and the algorithm for traversing the edges (or vertices) around a face is stated in Algorithm 13.2. Algorithm 13.1 assumes that the vertices are listed in clockwise order around the face, assuming the conventions described above. Algorithm 13.2 can be modified to find all edges (or faces) around a given vertex.

Figure 2. Binarization result for Figure1.

modifications also reduce the sensitivity of parameter *k*. Equation (2) presents the revised formula.

$$T(x, y) = m(x, y) \cdot [1 + k \cdot (1 - \frac{s(x, y)}{R})] \qquad (2)$$

where *m(x,y)* and *s(x,y)* are as in Niblack's formula. We use $R = 100$ and $k = 0.1$ for grayscale images. Figure 2 shows the binarization result for Figure 1.

2.3. Enhancement of binarization result

After the binarization is done, there always exists some large-size noise (see Figure 2). To improve the binarization result, the connected components are constructed (8-neighbors algorithm) and filters utilizing the property of connected components are applied to remove the noise. Another important reason for connected component construction is that we shall use it for the warped words adjustment to be described in section 3.

The filter consist of two parts, namely shape-filter and size-filter. The shape-filter rejects long and irregular-shape objects that are usually not text characters, and the size-filter simply rejects big objects whose sizes are greater than a threshold value. The connected component area that is most common then becomes the shape-filter. This is reasonable since the text characters are almost similarly sized. Connected components whose sizes lie outside the permissible range of the shape-filter are then removed. Mathematically, the shape-filter can be expressed as Equation (3).

$$F(c) = \begin{cases} true & if \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2} \le \sqrt{j \cdot size\_of(c)}, \forall (x_1, y_1) \in c \\ false & otherwise \end{cases} \qquad (3)$$

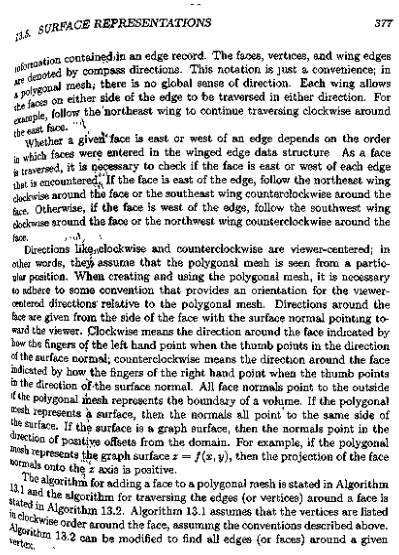


Figure 3. Enhanced binarization result for Figure1.

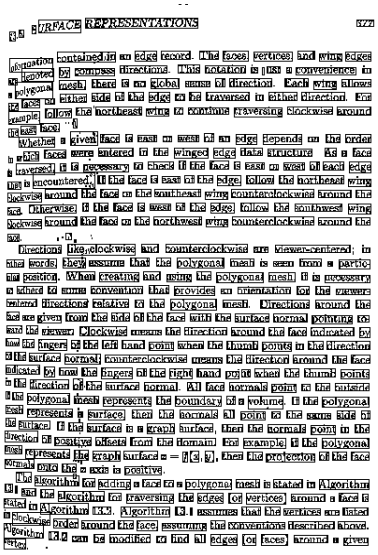


Figure 4. Words indicated by their bounding boxes.

where *c* denotes a connected component, *(x0, y0)* and *(x1, y1)* are coordinates for gravity center and an arbitrary point of *c* respectively, *size_of* is a function returning the number of black pixels in *c*, and *j* is a predefined parameter. We use *j* = 2.5 for our grayscale images. The result after applying filters is shown in Figure 3.

3. WARPED WORDS ADJUSTMENT

As we can see in Figure 1-3, the words in the shadow are warped because the page is not flat while scanning. In our system, we use a connected component analysis to adjust the warped words for both location and orientation. The following subsections describe the details.

3.1. Connected component basis clustering

Some of the connected components may consist of one or several characters, while the others are part of some characters, such as the top dot and the body of a character like 'i' or 'j'. If we adjust the warped words by moving and rotating connected components directly, the characters in the same word may not lie on the same straight line and may have different orientation, which means poor result will be produced. So the connected components are first clustered into words by applying a nearest neighbor algorithm. The algorithm initializes all the connected components to be unmarked and then perform the following iterative steps:

Step 1: Generate an empty word, i.e. the word contains zero connected components. Push the first unmarked connected component into the word, and make it marked.

Step 2: For each unmarked connected component, *C1*, and each component, *C2*, in the current word, if there exists a pixel *p1* in *C1* and a pixel *p2* in *C2* such that the distance between *p1* and *p2* is less than a threshold distance, *D*, *C1* is marked and pushed into current word. Then the algorithm goes back to Step 2. If no more connected component is added in, the algorithm will check whether all the connected components are marked, if yes, the algorithm terminates, else go back to Step 1 and start a new iteration.

The result of applying this algorithm is shown in Figure 4. Each word is indicated by its bounding box.

3.2. Word basis clustering

Merely clustering connected components into words is not enough, since for a given word we do not know which text line it belongs to, i.e. there is no way to move the word to the correct location. Our system groups the words into text lines by using a modified "box-hands" method [5]. The reason we could not adopt the original "box-hands" method is that it works when the text line is a straight horizontal line, which is obviously not the case in our test images.

After the construction of the bounding box of the word at the end of section 3.1., they are extended to give chains of connected boxes. As shown in Figure 5, the bounding box is extended by adding to their left and right sides two equal parallel-quadrilateral extensions, called the "box hands". The length and height of the hands are equal to the height and half of the height of the word bounding box respectly, and the orientation of the hands is decided by the orientation of the word. Our system detects

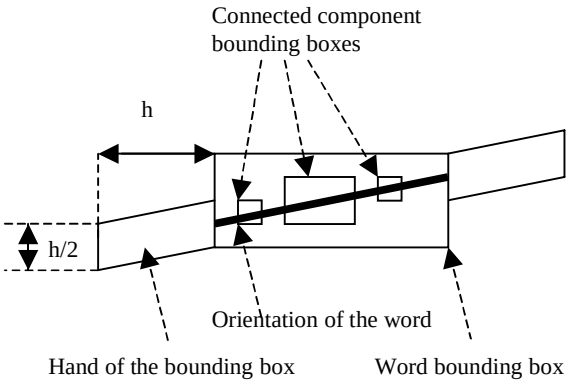
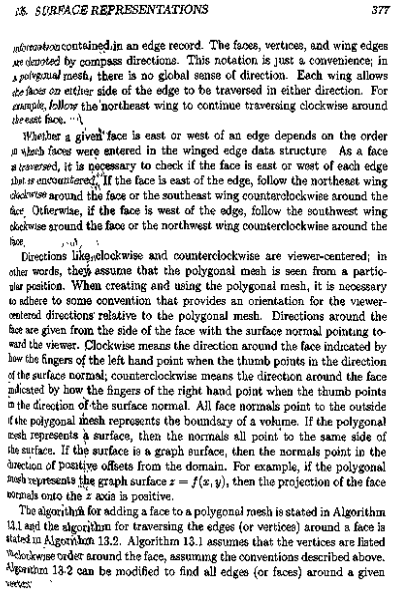


Figure 5. Extension of the word bounding box.


13.8. SURFACE REPRESENTATIONS 377

information contained in an edge record. The faces, vertices, and wing edges are denoted by compass directions. This notation is just a convenience; in a polygonal mesh, there is no global sense of direction. Each wing allows the faces on either side of the edge to be traversed in either direction. For example, follow the northeast wing to continue traversing clockwise around the east face.

Whether a given face is east or west of an edge depends on the order in which faces were entered in the winged edge data structure. As a face is traversed, it is necessary to check if the face is east or west of each edge that is encountered. If the face is east of the edge, follow the northeast wing clockwise around the face or the southeast wing counterclockwise around the face. Otherwise, if the face is west of the edge, follow the southwest wing clockwise around the face or the northwest wing counterclockwise around the face.

Directions like clockwise and counterclockwise are viewer-centered; in other words, they assume that the polygonal mesh is seen from a particular position. When creating and using the polygonal mesh, it is necessary to adhere to some convention that provides an orientation for the viewer-centered directions relative to the polygonal mesh. Directions around the face are given from the side of the face with the surface normal pointing toward the viewer. Clockwise means the direction around the face indicated by how the fingers of the left hand point when the thumb points in the direction of the surface normal; counterclockwise means the direction around the face indicated by how the fingers of the right hand point when the thumb points in the direction of the surface normal. All face normals point to the outside if the polygonal mesh represents the boundary of a volume. If the polygonal mesh represents a surface, then the normals all point to the same side of the surface. If the surface is a graph surface, then the normals point in the direction of positive offsets from the domain. For example, if the polygonal mesh represents the graph surface $z = f(x, y)$, then the projection of the face normals onto the z axis is positive.

The algorithm for adding a face to a polygonal mesh is stated in Algorithm 13.1 and the algorithm for traversing the edges (or vertices) around a face is stated in Algorithm 13.2. Algorithm 13.1 assumes that the vertices are listed in clockwise order around the face, assuming the conventions described above. Algorithm 13-2 can be modified to find all edges (or faces) around a given vertex.

Figure 6. Final result for Figure 1.

the word orientation by adopting a Hough transform [6]. It takes the centers of the bounding boxes of the connected components that belongs to a word as the points in the image space of Hough transform, and finds the line that most of centers lie on. A text line is found by clustering all the words whose hands of bounding boxes touch each other.

3.3. Warped words adjustment

The warped words adjustment consists of two steps – location adjustment and orientation adjustment. For a text line, *L*, found in section 3.2., we perform a Hough transform on the centers of all the connected components in *L*, and return a straight line, *S*, for *L*. Then for all the warped words in *L*, i.e. the words whose centers of bounding boxes lie within the shadow boundary (detected in section 2.1.), the location adjustment is done by moving the warped word such that the center of the word bounding box lies on its vertical projection point on *S*. For each warped word, the angle, *A*, between the word orientation (detected in section 3.2) and *S* can be calculated, since the functions of the two lines are known. The orientation adjustment is done by rotating the word around its center of bounding box by –*A* degrees.

The final result for Figure 1 is shown in Figure 6. Comparing with Figure 3, the warped words are moved to the correct location and rotate to the correct orientation. At the moment, no correction is made yet to the shape of the word. However, the general appearance and the readability of the original document image has been greatly improved and enhanced.

4. CONCLUSION AND FUTURE WORKS

In this paper, we describe the problems of distorted images while scanning thick, bound documents, and a restoration system that solves these problems for the grayscale images scanned. This system can successfully remove the shadow, and adjust the warped words. Note that although the location and orientation are adjusted, the shape of the word is not fully restored. One of our future works is to replace the connected components analysis with some "optical flow" techniques, so that the system can also correct the shape of the warped words.

REFERENCES

[1] H. Baird. "Document Image Defect Models", In Proc. of IAPR *Workshop on Syntactic and Structural Pattern Recognition*, Murray Hill, NJ, pp. 38-46, June 1990.

[2] H. Baird. "Document Image Quality: Making Fine Discriminations", In Proc. of IAPR Int'l Conf. on Document Analysis and Recognition, Bangalore, India, pp. 459-462, September 20-22, 1999.

[3] T. Kanungo, R.M. Haralick and I. Phillips. "Global and Local Document Degradation Models", In Proc. of IEEE *International Conference on Document Analysis and Recognition*, Tsukuba Science City, Japan, pp. 730-734, October 1993.

[4] W. Niblack. "An Introduction to Image Processing", Prentice-Hall, Englewood Cliff, NJ, pp. 115-116, 1986.

[5] C. Strouthopoulos, N.Papamarkos, and C.Chamzas. "Identification of Text-Only Areas in Mixed-Type Documents", *Engng Applic. Artif. Intell*., Elsevier Science Ltd, Great Britain, Vol. 10, No. 4, pp. 387-401, 1997.

[6] R. Jain, R.Kasturi, and B.G.Chamzas. "Machine Vision", McGRAW-HILL International editions, pp. 218-223, 1995.