ABSTRACT

Title of dissertation: PROCESSING CAMERA-CAPTURED DOCUMENT IMAGES: GEOMETRIC RECTIFICATION, MOSAICING, AND LAYOUT STRUCTURE RECOGNITION

Jian Liang, Doctor of Philosophy, 2006

Dissertation directed by: Dr. Daniel DeMenthon
Dr. David Doermann
Professor Rama Chellappa
Institute for Advanced Computer Studies

This dissertation explores three topics: 1) geometric rectification of camera-captured document images, 2) camera-captured document mosaicing, and 3) layout structure recognition. The first two topics pertain to camera-based document image analysis, a new trend within the OCR community. Compared to typical scanners, cameras offer convenient, flexible, portable, and non-contact image capture, which enables many new applications and breathes new life into existing ones. The third topic is related to the need for efficient metadata extraction methods, critical for managing digitized documents.

The kernel of our geometric rectification framework is a novel method for estimating document shape from a single camera-captured image. Our method uses texture flows detected in printed text areas and is insensitive to occlusion. Classification of planar versus curved documents is done automatically. For planar pages, we

obtain full metric rectification. For curved pages, we estimate a planar-strip approximation based on properties of developable surfaces. Our method can process any planar or smoothly curved document captured from an arbitrary position without requiring 3D data, metric data, or camera calibration.

For the second topic, we design a novel registration method for document images, which produces good results in difficult situations including large displacements, severe projective distortion, small overlapping areas, and lack of distinguishable feature points. We implement a selective image composition method that outperforms conventional image blending methods in overlapping areas. It eliminates double images caused by mis-registration and preserves the sharpness in overlapping areas.

We solve the third topic with a graph-based model matching framework. Layout structures are modeled by graphs, which integrate local and global features and are extensible to new features in the future. Our model can handle large variation within a class and subtle differences between classes. Through graph matching, the layout structure of a document is discovered. Our layout structure recognition technique accomplishes document classification and logical component labeling at the same time. Our model learning method enables a model to adapt to changes in classes over time.

PROCESSING CAMERA-CAPTURED DOCUMENT IMAGES: GEOMETRIC RECTIFICATION, MOSAICING, AND LAYOUT STRUCTURE RECOGNITION

by

Jian Liang

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2006

Advisory Commmittee:

Professor Rama Chellappa, Chair/Advisor
Dr. Daniel DeMenthon, Co-Advisor
Dr. David Doermann, Co-Advisor
Professor Larry Davis, Dean's representative
Professor Min Wu
Professor Ray Liu

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Motivation

In this dissertation we present our work on three main topics:

- Geometric rectification of camera-captured document images,
- Camera-captured document mosaicing, and
- Recognition of layout structure in document images.

The first two topics are motivated by the recent trend in the OCR community of augmenting the use of flat-bed scanners with digital cameras [38, 19, 35]. From a technical point of view, cameras offer convenient, flexible, portable, and non-contact image capture, which opens the door to many new applications and gives new life to existing ones. From a market point of view, the vast number of digital cameras owned by consumers provide a large potential market for document capture and OCR. Both drive the recent trend of camera-based document analysis.

This trend brings many opportunities as well as challenges to the OCR community. For example, handheld devices (such as PDAs and cell phones) equipped with cameras are ideal platforms for mobile OCR applications such as recognition of street signs in foreign languages, out-of-office digitization of documents, and text-to-voice input for the visually impaired. In industrial market, high-end cameras have

been used for digitizing thick books and fragile historic manuscripts unsuitable for scanning; in consumer market, camera-based document capture is in use in the desktop environment [65]. A challenge facing the OCR community is that, due to the differences between scanners and cameras, traditional scanner-oriented OCR techniques are not generally applicable to camera-captured documents. Although digital cameras have recently made advances in resolution, noise level, shutter speed, white balancing, exposure metering, etc., fundamental obstacles exist in using them for the purpose of OCR. Unlike a flat-bed scanner which fixes the image plane, carefully controls lighting, and obtains large-sized images with a moving optical component, a camera-captured image may suffer from problems such as non-planar page shape, uneven lighting, low resolution, perspective distortion, out-of-focus or motion blur, and under- or over- exposure. As a result, we either must modify traditional OCR techniques to make them compatible with new images or modify the images so they can use available OCR algorithms. Both have pros and cons. From an engineering perspective, the second approach has the advantage of keeping existing working modules intact, which is important in terms of reliability and code reuse. Therefore, our first two topics focus on processing camera-captured document images to make them OCR compatible.

Our third topic originates from the need for metadata extraction in document image analysis. Metadata plays an important role in applications such as indexing and retrieval, automatic routing, re-authorization, abstract generation, and device dependent formatting. For example, despite the phenomenal success of full text search engines on the web, their heavy dependence on the 'quality' of query keywords

(i.e., the more common the keywords, the less the discriminating power) and their inability to express constraints that are not content-based limit their performance. It is very difficult to formulate the query, to "find all correspondences sent to Mr. Smith," as a full text search query because the document class, *correspondence*, and the functional attribute, *sent to*, are not related to any keywords. The name, *Smith*, is so common that the turn-out would be too large to be useful. In another example, most PDF or HTML documents are formatted for large desktop or laptop displays and are inappropriate for the small screens on handheld devices such as PDAs and cell phones. This means that the content must be reformatted according to the terminal's requirement upon delivery, and such reformatting clearly depends on the metadata associated with the content.

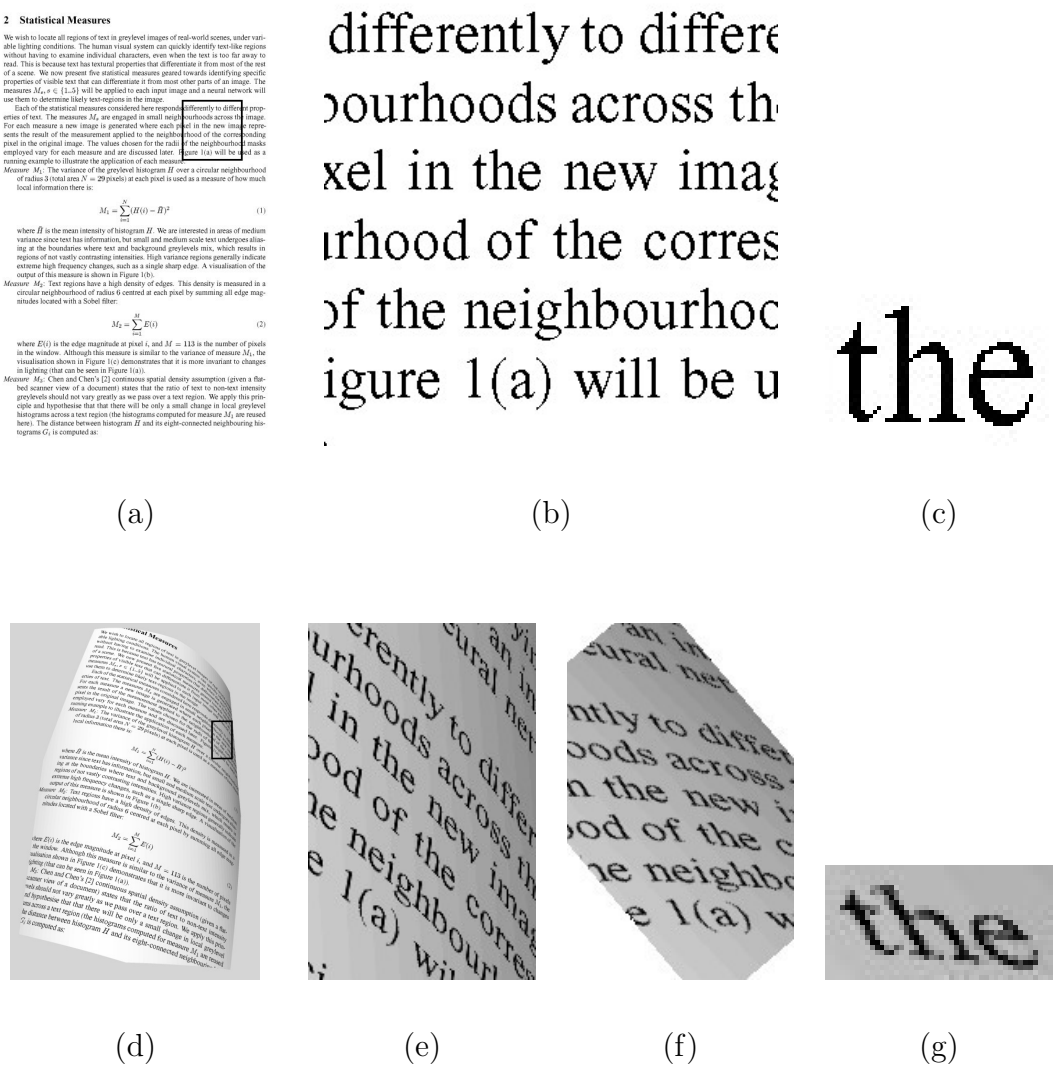
Most current OCR systems can recognize the text content, but few of them extract extensive metadata. Although metadata identification from text is possible by using Natural Language Processing (NLP) techniques, it is often unreliable. This is because NLP only has access to text content that cannot carry other useful visual information about the document. In particular, we find that layout styles convey important information about the function of the document as well as functions of different zones on the page. Many publishers restrict their authors and editors to carefully designed templates that generate a consistent style in their publications; many non-public domain documents such as personal correspondences also follow widely accepted layout conventions. The understanding of layout styles allows human readers to distinguish a letter from a report without even knowing the language of the document. Our third topic, therefore, questions how to simulate the human

ability to recognize the document layout and apply it to the problem of document classification and logical component labeling.

1.2 Geometric image rectification

Many factors contribute to the difficulties of using scanner-oriented OCR technique to process camera-captured document images [38], such as uneven lighting, out-of-focus blur, motion blur, complex background objects, and *non-planar pages with perspective distortion*. It is well-known that state-of-the-art OCR software packages do not properly handle a non-planar document under perspective projection. As seen in Figure 1.1, at the page level such distortions bend straight margins and text lines, thus defeating the assumptions of many well-known page segmentation algorithms [52, 49, 32]. At the word and character level, the distortion make character segmentation difficult because characters can no longer be split perpendicular to the text base line. Even if characters are successfully segmented, the distorted characters are unlikely to be represented in the training data set, causing low recognition rates. To a lesser degree, these challenges also apply to planar pages. The experimental results summarized in Tables 3.8 and 3.9 show disappointing OCR performance for synthetic camera-captured document images (both curved and planar). The synthetic images used for the tests are virtually free of noise and blur, and have sufficient resolution. This eliminates the opportunity of improving performance through pure 2D image enhancement.

In the literature, the geometric rectification of camera-captured document im-



(a) (b) (c)

(d) (e) (f) (g)

Figure 1.1: Comparison between flat and warped document images. (a) A clear scan of a document page. (b) A sub-image of (a) enlarged. (c) Word "the" enlarged from (b). (d) The same document page with curved shape under perspective projection. (e) A sub-image of (d) with similar content as (b). (f) Text line segmentation might be possible (locally) after rotating (e) so that text lines are roughly horizontal. (g) Word "the" with distorted characters that OCR cannot recognize.

ages largely depends on additional knowledge outside the image. In one branch of this area, 3D range data is required [5, 57, 68] so that shape is directly known. This approach only suits large projects such as digital library acquisition because it requires special hardware. Another area of research assumes a flat document [14, 56] to simplify the underlying page shape. Obviously, these methods for flat pages cannot handle an opened book with curved surfaces. To process curved pages, either the camera pose must be restricted [7] or additional metric information of the page [67, 24] is required. Overall, to the best of our knowledge, no current method other than ours exists for processing general document images captured by cameras.

Our approach does not require multiple views, extra 3D or metric data about the page, specific pose/shape knowledge, or camera calibration. We make three basic assumptions. First, the document page should contain sufficient printed text content. This requirement is valid considering that the user is interested in document analysis, not general image analysis. Second, the document is either flat or smoothly curved (i.e., not torn or creased). And third, the camera is a *standard* pin-hole camera in which the x-to-y sampling ratio is unity and the principal point (where the optical axis intersects the image plane) coincides with the image center. Most digital cameras satisfy the third assumption.

Under these three assumptions, we show that we can constrain the physical page by a developable surface model, obtain a planar-strip approximation of the surface using texture flow data extracted from the image, and use the 3D shape information to restore the frontal-flat document view. Our approach presents a unified solution for both planar and curved documents. Figure 1.2 illustrates our

approach. It takes one camera-captured document image as input, either planar or curved, and outputs the frontal-flat view of the page.

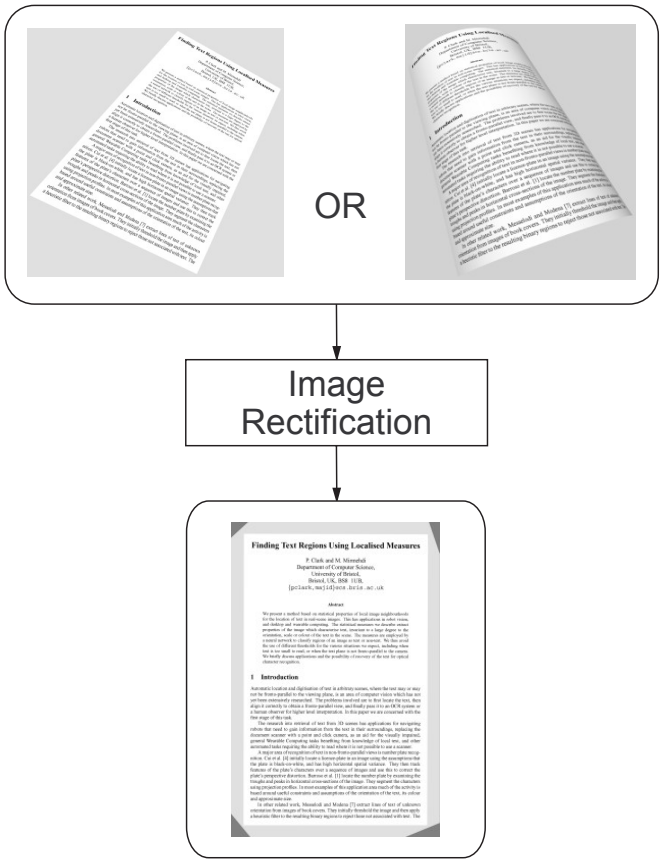


Figure 1.2: Example of geometric image rectification.

1.3 Document mosaicing

Another problem associated with processing camera-captured documents involves the conflict between limited resolution and field of view for most consumer-grade digital cameras. For low resolution cameras, such as PDA or cellphone cameras, if a whole letter-size document is included in the field of view, the characters might be

illegible. On the other hand, if the resolution is increased by either zooming in or moving closer to the page, then only a small portion of the page can be captured. There are two possible solutions: 1) keep the field of view large, take multiple low resolution images, then apply *super-resolution* techniques [8, 21, 54] to generate a high resolution image of the entire page; or 2) take multiple high resolution pictures of small portions of the document, then apply mosaicing techniques to generate a composite result. As [2] shows, super-resolution methods have some inherent limitations. More importantly, in practice it is difficult to keep handheld cameras at a fixed position. Varying poses make image registration very difficult, if not impossible, to meet the requirement of super-resolution. As for mosaicing, most methods proposed for document images are designed for scanners [61, 72], where overlapping images differ only by rotation and translation. A few methods are developed using cameras but they require additional hardware to restrict the camera pose [50, 76]. Video mosaicing methods allow changes in scale and perspective. However, frame-to-frame difference is usually minute [45, 64]. Furthermore, if the camera motion is not pure panning and zooming, i.e., if the optical center moves, most video mosaicing produces a panorama view that is not a projective view of the world [55]. In the context of document mosaicing, this means that the result image is still distorted.

Our mosaicing approach for camera-captured documents does not impose restrictions on camera pose/motion and suits images with large perspective distortion and small overlapping areas. In other words, our method allows the user to take pictures from arbitrary positions. We assume that the portion of the document

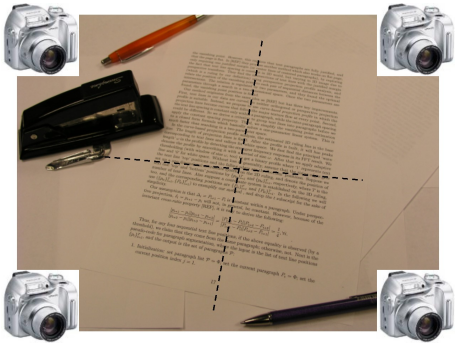
in each image is *almost* flat. We first remove perspective distortion using geometric rectification methods, then register the images pairwise. Finally, we seamlessly blend them to produce a high resolution composite image which contains the frontal flat view of the document.

Figure 1.3 illustrates the concept of camera-captured document mosaicing, where four pictures, each capturing a portions of a document, are mosaiced into a high resolution composition.

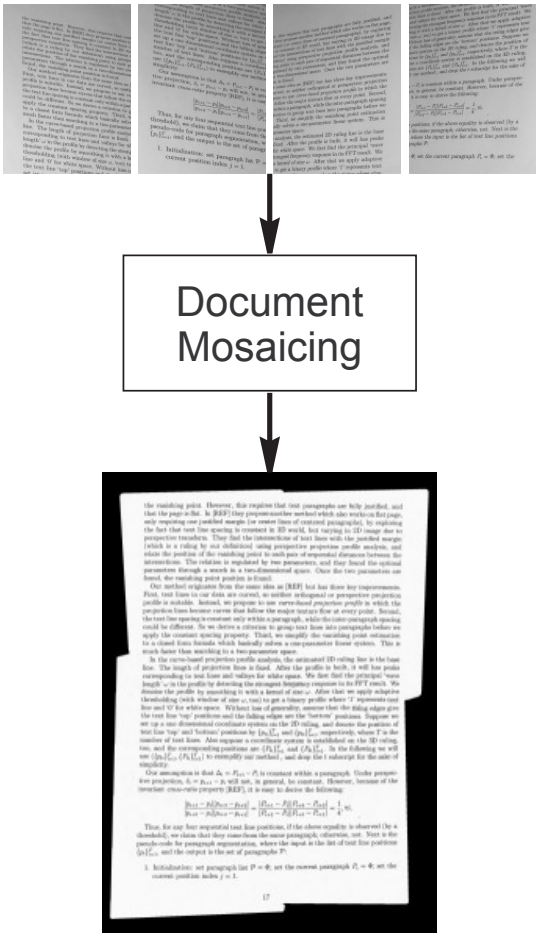
1.4 Layout structure recognition

The third topic of our work focuses on the analysis of layout structure in document images. The layout of a document is designed to facilitate its readers to understand the content and convey information absent in the text. Therefore, layout analysis is an important area in document image understanding.

In particular, we notice that documents in one class usually share a common layout style and documents from various classes can often be distinguished by their different layout styles. Hence, the analysis of layout structure can assist document classification. Within a document class, physical layout is closely linked to logical structure [25]. The logical structure of a document image involves logical roles of individual zones, reading order, and logical relationship among zones. Typically, a logical component in a document class is consistently represented by a specific zone in the page. As a result, analysis of layout structures provides an approach to discover logical structures.



(a)



(b)

Figure 1.3: Example of document mosaicing. (a) Four images cover a whole document. (b) Mosaicing produces a high resolution composition.

In the literature, classification of document images and extraction of logical structures from document images are studied as two separate subjects. Most previous work in document image classification use a global representation of the page layout that does not provide explicit local features. Documented work on logical structure analysis usually performs functional analysis at the component level without considering global layout styles. Furthermore, many approaches requires a fair amount of training and models must be retrained from scratch if classes change or new classes are added.

We present a unified approach to both document classification and logical structure analysis using a graph-based model. Our model can accommodate heterogeneous features, both global and local. We call the process that matches models to instances as *layout structure recognition*. Through this process, we simultaneously determine the class of the document instance and its logical structure (see Figure 1.4 for a conceptual illustration). With an adaptive learning method, a document model can be initialized with a relatively small number of samples and improved with new samples later on. This avoids the problem of expensive retraining when classes change or new classes emerge.

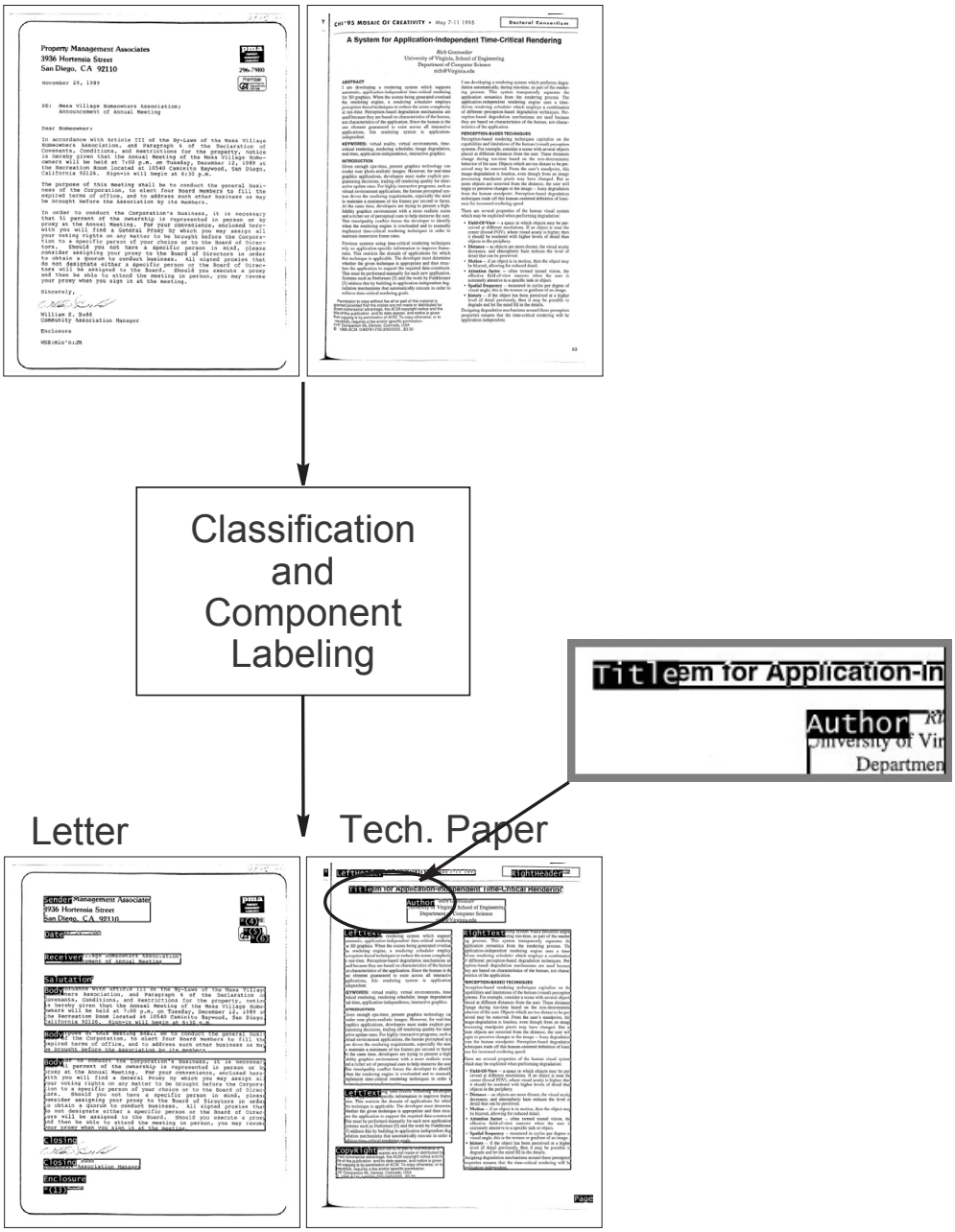


Figure 1.4: Example of document classification and logical component labeling.

Chapter 2

Analysis of Texture Flow in Document Images

2.1 Motivation

We are interested in the analysis of texture flow fields in document images because they provide a powerful tool for analyzing the underlying page shape, and the latter presents one of the key problems in rectification of camera-captured document images.

Informally, a flow field defines an orientation function at every point in the space. In the 2D plane $\mathcal{R}^2$, one way to describe the flow field is:

$$\theta(x, y) : \mathcal{R}^2 \rightarrow \mathcal{S}^1$$

which defines a dominant orientation θ at (x, y).

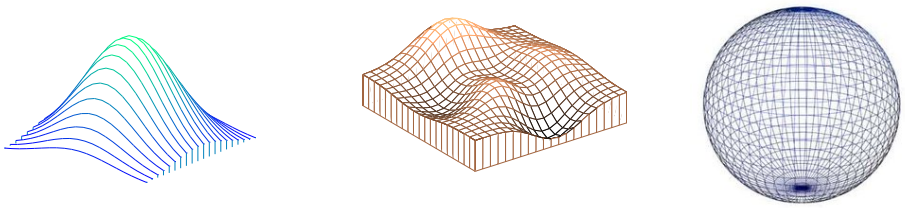
A flow field can be visualized by short line segments or continuous curves following the local orientations. Psychological observations suggest that abstract representations of continuous 2D flow fields should be such that locally the line segments or curves are parallel [3, 59]. Inversely, an image with a texture pattern that exhibits local parallelism gives a viewer the perception of a flow field. We call this a *texture flow* field. A typical example is the pattern of a zebra's stripes.

More than one texture flow field can co-exist in one space. For example, a piece of fabric usually exhibits two orthogonal texture flow fields. This also holds

true for a document page in which we can identify a *major texture flow* field and a *minor texture flow* field. Text lines and white line space form very prominent parallel structures that define the major field, while the vertical character strokes present a weaker (hence the name) minor field. For most languages and scripts, vertical strokes represent important elements in characters, and characters align along text base lines. Therefore, these two texture flow fields are common to most printed documents.

It is well known that texture on surfaces can assist the shape perception process in human visual system. Texture flows fields have the same effect [36]. During this interpretation process, we usually make unconscious assumptions, such as that the flow field follow the *lines of curvature* direction, or that it follow the *geodesic* direction, or that it represents contours cut by a group of parallel planes. These assumptions are extensively utilized in line drawing to convey shape information (see Figure 2.1).

In the context of document pages, our physical world knowledge ensures that 1) the two texture flow fields defined above are locally orthogonal to each other, 2) they are both geodesics of the page surface, 3) globally the page can be flattened, and 4) on the flat page each texture flow field points to a consistent direction. These assumptions help a viewer to quickly obtain a good idea of local surface orientation and global surface shape. Certain exceptions, such as a paragraph of italic text that defies the orthogonality between two fields or a large figure or picture absent with these two fields, can make this process harder. In these situations, shape information can be interpolated or extrapolated using surrounding normal



(a) (b) (c)

Figure 2.1: Shape perception from line arts. The curves in both (a) and (b) are cutting contours by (a) one group of parallel planes, and (b) two groups of parallel planes. The 'latitude' lines in (c) are cutting contours by a group of horizontal planes, and the 'longitude' lines are geodesics.

text areas. Furthermore, page boundaries, text margins, and picture borders, can all be clues for generating shape perception.

While these auxiliary clues are helpul, they are not always reliable because of possible occlusion. If a shape estimation method depends on these clues, it will have difficulty when another object occludes the document and creates a false text margin or page boundary. On the contrary, techniques based on texture flow fields in printed text are not affected by occlusion because occlusion only eliminates some portions of the fields and does not change the remaining part. The shape of the visible part of the page can always be found using the visible texture flow fields.

In the following sections, we first review related work, then describe our approach to estimating texture flow fields in document images.

2.2 Related work

As with many computer vision problems, the first step in dealing with complex real life images involves finding the region of interest. In our case, the region of interest is the document in the image; to be precise, we are interested in the printed text area (including tables) but not arbitrary figures or pictures. We focus on printed text (and tables) because it presents more consistent major and minor texture flows than figures or pictures. In the literature, the process of finding text areas in images is called *text identification*. Methods for text identification in complex images can be roughly grouped as gradient-based, color-based, and texture-based. For more details of these three groups of techniques, refer to [38]. In practice, these methods all begin by finding possible text pixels and follow with a grouping/verification procedure. The output can be either a binary mask or bounding boxes that enclose text areas.

Image binarization classifies each pixel as either foreground or background. For most documents, pixels belonging to printed markings should be labeled foreground and all other pixels background. The problem may become complicated if there are multi-layer foreground or background, such as overlapping and multi-color areas, or textured paper background. Binarization greatly reduces the image's complexity and almost all scanner-oriented OCR techniques are designed for binary images. Binarization is usually accomplished by thresholding a gray level image. It has been widely accepted that for documents with non-uniform background color and/or brightness or uneven illumination, adaptive thresholding provides a powerful

solution [66].

During scanning, a document may not be perfectly aligned with the scanner frame, resulting in text lines skewed with respect to the horizontal axis. Skew removal algorithms attempt to restore perfectly horizontal text lines. When multiple text blocks exist at different skew angles, the problem is not well-defined. Most skew removal algorithms assume one skew angle for the whole image and rely on printed text to estimate the skew. Skew detection is related to texture flow detection in document images because, locally, the major texture flow direction defines the skew angle. Therefore, all the following skew detection methods have the potential of detecting major texture flow, and, to some extent, they are also applicable to minor texture flow.

There are several major skew detection techniques, including Hough transform [37, 1, 33], auto-correlation [74, 9], projection profile analysis [10], and connected component clustering [53, 43]. Hough transform is widely used for line detection, therefore can detect text lines, which, in turn, assist in estimating the skew angle. Its main drawback involves high computational cost. Some variations [33] are less computationally expensive. Auto-correlation based methods compute the correlation of the image with itself at a certain offset (w, h). The result yields maxima, when w and h are such that text lines in the two images overlap. The offsets that result in maximum correlation are related to the skew angle. By varying (w, h), one finds the best skew angle that produces consistently maximum correlation. This method provides good results for dense text area with constant line spacing. The method of projection profile analysis, as its name suggests, computes the profile of projec-

tion of the binary image at a given angle. If the angle corresponds to the correct skew, the profile exhibits a clear peak-and-valley pattern due to the separation of text lines and white space, which maximizes the entropy of the profile[1]. Connected component clustering methods either find the nearest neighbor of each component or group components into text lines, then use the orientation of nearest-neighbor-pairs, or the direction of text lines, to estimate the skew. All these approaches can be applied to the entire image or to selected small regions for the sake of speed. In the latter case, results from different regions are usually combined by voting or averaging because we assume only one skew angle for the page.

2.3 Texture flow estimation

Our texture flow estimation module consists of three steps. The first step involves text identification and binarization, which filters out irrelevant objects in the image and converts gray level pixels to binary bits. The next step detects the major texture flow, in a multi-resolution fashion in terms of both spatial resolution and angle precision. In the second step, a rough estimate of the minor texture flow is also obtained. Based on this rough estimate, the third step refines the precision of minor texture flow.

[1]Under the assumption that the profile value follows a Gaussian distribution, its entropy can be computed by standard deviation.

2.3.1 Text identification

We adopt the gradient-based methodology and implement the following procedure to find the text area and binarize the image:

1. Compute the edge map of the image using the Laplacian of Gaussian method [27] and retain pixels with strong edge magnitude;

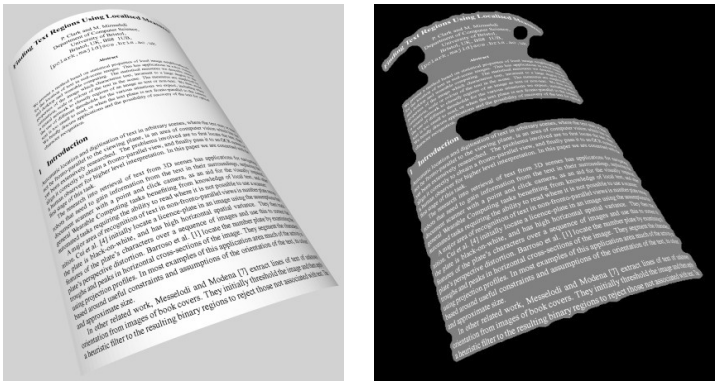
2. Perform a *close* (dilation followed by erosion) morphological operation to fill holes and gaps, and the result is the text area.

3. Within the text area, apply Niblack's adaptive thresholding method [66] to the original image to obtain a binary text image.

Our method works well with our experiment data. Figure 2.2 shows the results of one example. In real images, strong texture in non-text areas and soft text (because of out-of-focus or motion blur) could affect the results of our method. More sophisticated techniques would be needed to address the text identification problem in real images, which is a complicated topic deserving a thesis of its own. We do not, however, focus on it in this dissertation.

2.3.2 Major texture flow

Because the major texture flow is equivalent to the local text line direction, it can be found using various skew detection methods summarized in Secton 2.2. We choose the projection profile analysis method for its simplicity and robustness. Essentially, for a given point, we place a window of size $w \times h$ centered at the point and rotated



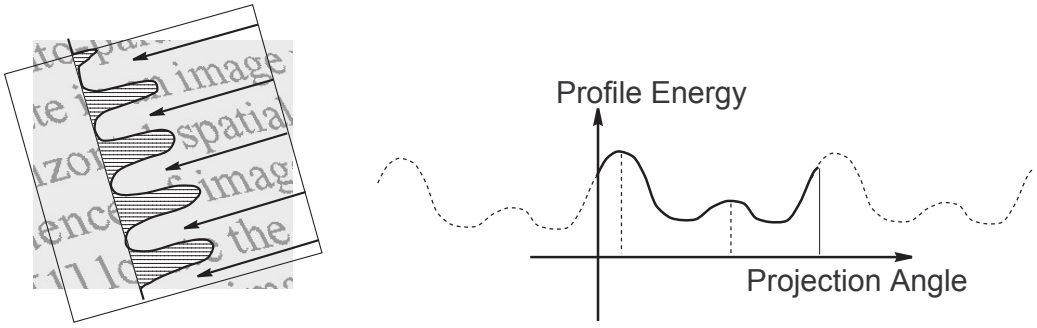
(a) (b)

Figure 2.2: Text identification and binarization. (a) Original document image. (b) Binary text in white overlaid on text area in gray.

by an angle α, then compute the projection profile of the image inside the window along the w edge. We take the standard deviation of the profile as its energy measure $\mathcal{E}$. By gradually changing α from 0 to π, we obtain a sequence of $\mathcal{E}$ measures. Ideally, the energy sequence should have maxima at the angles corresponding to the texture flow directions (see Figure 2.3) and typically, the largest peak indicates the angle of the major texture flow and the second one is for the minor texture flow. However, noise is inevitable in a small sampling window[2] because of the randomness of character shape. Especially, in areas where the text is sparse or non-text elements are not removed by text identification, the result may have multiple maxima in the $\mathcal{E}$-α curve. In most cases, the incorrect peaks are lower than the two correct ones, so they can be removed easily. However, there are exceptions. Because the

[2]The window size is mainly limited by the curvature in texture flows. It is also restricted by the computational cost.

angles corresponding to the false peaks are usually random, we can detect them by comparing the neighboring information.



(a) (b)

Figure 2.3: Projection profile analysis. (a) A projection profile built for a given angle, showing a clear peak-and-valley pattern as the angle coincides with the text line direction. (b) Ideal projection energy vs. angle graph where within $[0, \pi)$ two peaks correspond to text line and vertical character stroke directions, respectively.

Relaxation labeling is a well-known method [29] for utilizing contextual information. This method organizes data as a graph in which nodes have both correct and incorrect labels as candidates. It assumes that a correct label at a node is supported by correct labels at neighboring nodes, and incorrect labels are not or less supported. So, the collected support from neighboring nodes can be used to find the correct label among the candidates. For us, the nodes are the sampling points at which we compute the $\mathcal{E}$-α data, and the labels are the α angles corresponding to maxima in $\mathcal{E}$. At neighboring nodes, the correct angles are similar due to the continuity of texture flows, so they support each other.

More specifically, we divide the image into blocks (of size $X \times Y$) and use the block centers as sampling points, or the nodes. At each node, we sample α in $[0, \pi)$ with precision Φ, collect the $\mathcal{E}$-α data and select K highest peaks from the curve. For node i, the energy values of the candidates are denoted by $\mathcal{E}_{ik}$ and corresponding angles are represented as α_{ik}, $k = 1, \ldots, K$. Each candidate is associated with a confidence value within 0 and 1, which is initially set to

$$C_{ik} = \mathcal{E}_{ik} / \left(\max_{k=1}^{K} (\mathcal{E}_{ik}) \right).$$

The contextual information of neighboring nodes is quantified by a *compatibility function* $r_{ij}(k_1, k_2)$, which measures the influence of candidate k_2 of node j to candidate k_1 of node i. In conjunction with the confidence value C_{jk}, we use the following formula to compute the support a candidate receives from surrounding nodes:

$$s_{ik} = s(i, k; \{r\}, \{C\}) = \sum_{j \in U_i} \sum_{k_2=1}^{K} r_{ij}(k, k_2) C_{jk_2},$$

which sums the compatibilities between label k at node i and all other labels in the neighborhood U_i, weighted by the corresponding confidence values. In our implementation, we include eight neighbors of node i in U_i. The contextual support is incorporated into the confidence of label k at node i using the insights of Hummel and Zucker [30]. Here, we choose the following iterative implementation:

$$\begin{aligned} \tilde{C}_{ik}^{t} &\leftarrow C_{ik}^{t} + \delta s_{ik}^{t}, \\ C_{ik}^{t+1} &\leftarrow \frac{\tilde{C}_{ik}^{t}}{\sum_{k=1}^{K} \tilde{C}_{ik}^{t}}, \end{aligned}$$

where δ controls the iteration step size, t and $t+1$ represent the present and next iteration count.

As for the compatibility function r_{ij}, we define it as:

$$r_{ij}(k_1, k_2) = \cos(\Gamma(\alpha_{ik_1}, \alpha_{jk_2})),$$

where

$$\Gamma(a, b) = \min(\mathrm{mod}(a - b, \pi), \mathrm{mod}(b - a, \pi))$$

computes the *best aligned* difference between the two angles. The value of r reaches maximum 1 when the two angles differ by an even multiple of $\pi/2$ and minimum 0 when the difference is an odd multiple of $\pi/2$. In other words, r is maximized when the two angles are parallel and minimized if they are orthogonal.

We find that a few iterations (typically ten) are sufficient to increase the confidence of a correct label to be maximum among all competing candidates at the node, if we use a large δ to accelerate the iteration. After relaxation labeling, the angles corresponding to the major texture flow become top candidates. We store them in E_0^M and remove them from the candidates. Then, we re-run the relaxation labeling algorithm. This time, the top candidates correspond to the minor texture flow directions. They are stored in E_0^m.

Even with the relaxation process, E_0^M and E_0^m may include errors. It could be that text in a block is too sparse, or non-text elements are overwhelmingly present in the block, such that the correct angles are not represented by any peaks in the $\mathcal{E}$-α curve. Nevertheless, we implement a verification step to correct the errors. We

check the texture flow estimate at every node against the average of its neighbors. Any estimate with a difference above a certain threshold is replaced by the local average. Thus, we ensure that both E_0^M and E_0^m are coherent.

In our implementation, we use a multi-resolution approach to compute texture flow fields for improving the computational efficiency. We use relatively large spatial sampling size ($X \times Y$) and angle sampling precision (Φ) for E_0^M and E_0^m. Then we use a smaller block size ($x \times y$) to re-divide the image and interpolate or extrapolate E_0^M and E_0^m to obtain E_1^M and E_1^m, which are the texture flow estimates at the centers of small blocks. At each block center, we scan the angle range $[E_1^M - \Phi, E_1^M + \Phi]$ with a finer step $\phi (< \Phi)$ and apply the same projection profile analysis method to find the best major texture flow estimate corresponding to the maximum energy. The results are stored in E_2^M. Finally, we interpolate/extrapolate E_2^M to obtain a dense major texture flow field, E^M, that covers every pixel in the text area.

For the minor texture flow, the process from E_1^m to E^m (the counterpart of E^M) is described in the following section.

2.3.3 Minor texture flow

With varying width scripts, it is impossible to align vertical character strokes in different text lines, therefore the projection profile analysis method does not work as accurately for minor texture flow detection as for detecting major texture flow. It can provide a coarse estimate E_1^m, but we need another method to refine the result to obtain E^m.

We notice that in printed text, the horizontal and vertical strokes usually present the major linear structures. Therefore, we can extract these linear structures, remove those aligned with E^M, and use the remaining ones to estimate E^m. We use a directional filter to search for line segments of a certain length at a specific direction. Figure 2.4(b) shows one example filter (20-pixel long, 60 degree angle). When we convolve a directional filter with a binary text image, the result has large responses at centers of line segments whose directions coincide with the filter's direction. We remove short line segments by setting the response below a threshold to zero, then average the remaining response within a window to measure the 'strength' (S) of local linear structures at the specified direction (β).

Ideally, S should be maximized when β is aligned with the vertical strokes. To address possible errors caused by noise and the randomness of character structure, we fit a third order polynomial curve to S-β data and select the angle of the curve peak as the vertical stroke direction. As a result, we obtain minor texture flow estimate, E_2^m, at the centers of re-divided image blocks. We interpolate/extrapolate E_2^m to compute E^m, the minor texture flow field.

2.4 Synthetic data generation

We use synthetic images to evaluate our texture flow estimation module. There are three reasons for using synthetic images. First, synthetic images do not contain CCD noise, uneven lighting, motion blur or out-of-focus blur, background objects, and lens distortion, so we can concentrate on our core problem. Second, we can

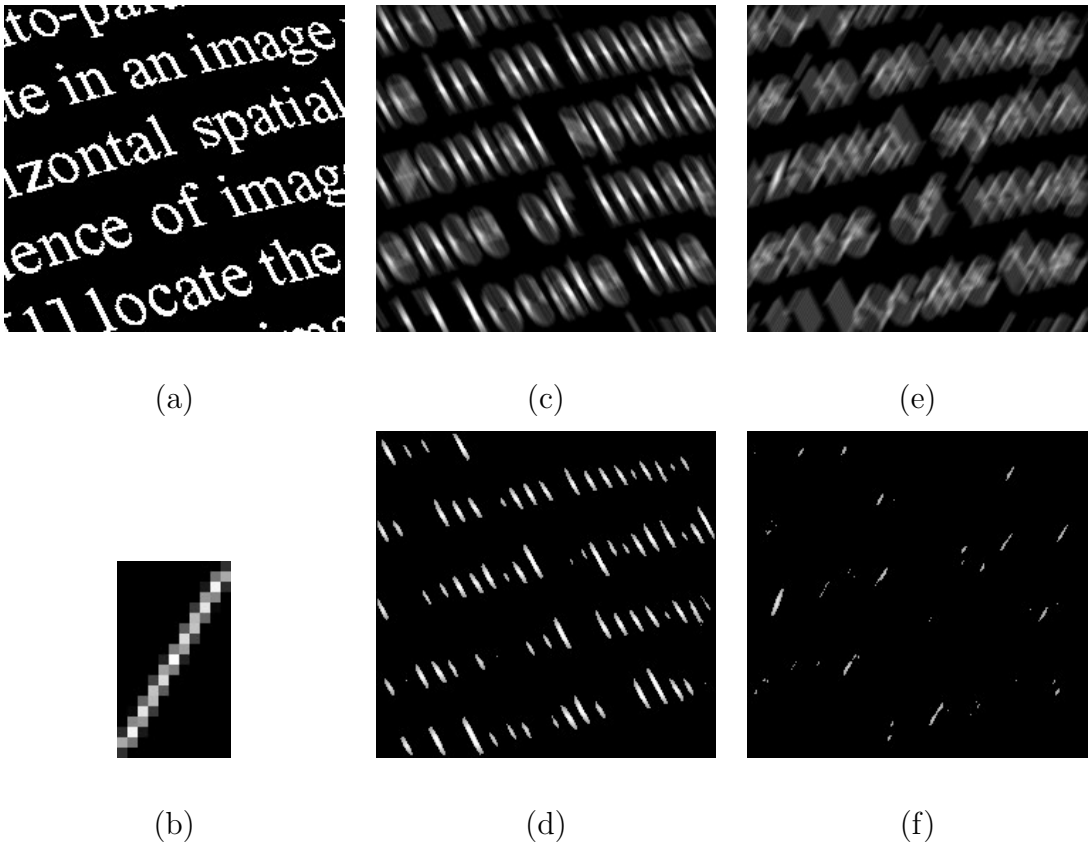


Figure 2.4: Vertical character stroke direction detection. (a) A sub-image of binarized text image from the document in Figure 2.2(a). (b) A directional filter tuned to 60° angle, enlarged to show details. (c) Output of applying a 120° filter to (a). (d) Thresholded result of (c). (e) Output of a 60° filter. (f) Thresholded result of (e).

generate a large amount of images with known poses, shapes, or focal lengths, so that we may understand the results better. The third, and most critical reason is that only with synthetic images can we inexpensively obtain ground truth data for the purpose of evaluation.

Synthetic images are generated using a module described in Appendix A. Along with the images, it also generates ground truth data of the major/minor texture flow fields, ruling lines and their vanishing points, and surface normals, so we can evaluate our intermediate results at each step [3].

We converted five pages in a PDF file to TIFF images at 300dpi, and we use them as the flat document images. Their sizes are all 1600×2500 pixels. Each page is skewed by three angles, $15°$, $0°$, and $-15°$. After skewing, the parts outside the original page frame are cropped (see Figure 2.5(c)(d)). We designed four sets of pose parameters, each defining the rotation and translation of the document page in the camera's coordinate system, plus the focal length. The combination of five pages, three skews, and four poses gives us 60 synthetic images of planar documents (see Figure 2.5(a)(b)(c)). For curved documents, we designed two cylindrical shapes (see Figure 2.5(d)), which, together with the previous combinations, provide a total of 120 images.

For images of both planar and curved documents, images with non-zero skew have some *false* text margins and page boundaries because of the cropping effect. Such images present difficulties for rectification algorithms that rely on text margins or page boundaries [75, 13, 14, 12, 24]. For curve documents, the skew also results

[3] Except for texture flows, the other intermediate results are discussed in Chapter 3.

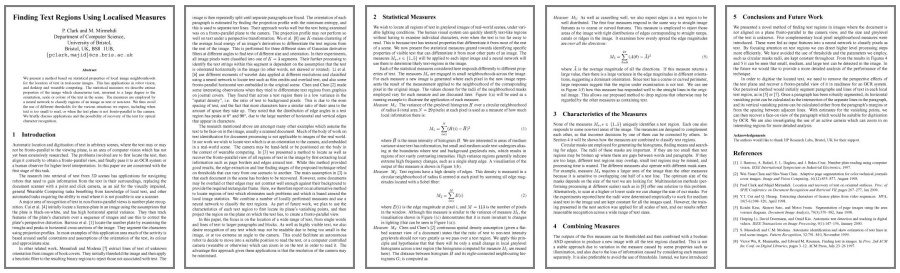
in an angle between the cylinder directrix and the text lines, which most current algorithms [78, 79, 73, 7, 69] cannot handle.

2.5 Evaluation

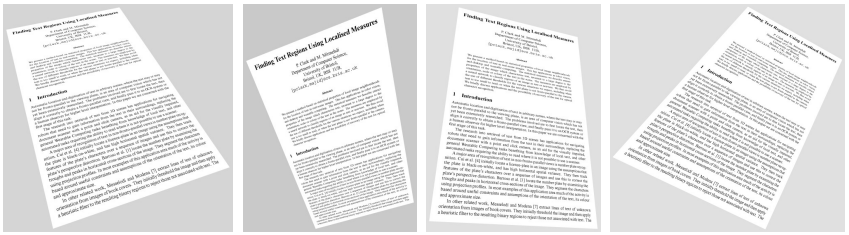
Once we obtain the estimated texture flow field and ground truth data, we take a group of sample points in the text area in the image and compute the average flow direction error compared to the ground truth data as the performance benchmark for the particular image. For every image, we produce one benchmark number for each texture flow field estimate, and, for a group of images, we further compute the mean and standard deviation of individual benchmarks as group-wise performance indices. For perfect estimation, both the mean and standard deviation should equal zero.

Table 2.1 summarizes the texture flow estimation benchmarks obtained from 60 planar pages. The first row shows the overall performance, while the other rows illustrate results grouped by page, pose, or skew. The numbers are in the form of '*mean/standard deviation*', all in degrees. Overall, we observe satisfactory precision, with the average error in E^M being 0.31 degrees and 0.90 degrees for E^m. We also notice that the error in E^m is consistently larger than that of E^M, which is understandable since the text line and white line space separation presents a significantly more prominent parallel pattern compared to vertical character strokes.

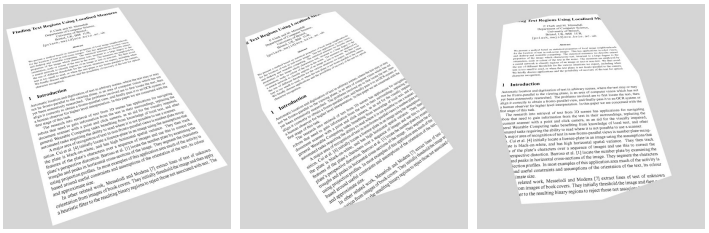
In terms of the effect of different poses, page contents and skew angles, we do not record any significant variation in texture flow estimation errors among different



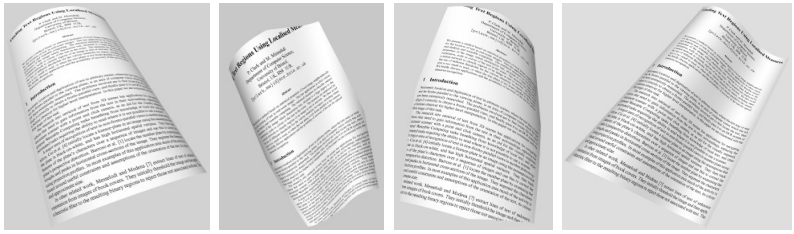
(a)



(b)



(c)



(d)

Figure 2.5: Synthetic document image samples. From left to right (a) flat page no. 1 through no. 5, (b) pose no.1 through no. 4, (c) 0°, 15°, and −15° skew, (d) shape no. 1, pose no. 1, 0° skew; shape no. 2, pose no. 2, 15° skew; shape no. 1, pose no. 3, −15° skew; shape no. 2, pose no. 4, 0° skew.

groups. It is possible that when the tilt angle of the plane increases beyond those in the test images, we could see larger errors. This is especially so with E^m, because its accuracy is linked to the stroke length, which will be greatly affected by severe perspective foreshortening caused by a large tilt. However, under severe perspective distortion, the character strokes will smear into each other and become inseparable even with perfect rectification. So, in practice, when excessive tilt increases the errors of texture flow estimation to an intolerable level, it also defies the need for rectification.

Table 2.2 shows the results obtained from 120 synthetic curved document images. The overall performance is satisfactory. The average error for E^m (1.12 degrees) is close to that observed in planar pages, with only 0.22 degrees. The average error for E^M (0.80 degrees) suffers an 0.49 degrees increase compared to that of planar pages, which is considerably larger in both absolute and relative sense. This occurs because in our test data the curvature of E^M is much larger than E^m, so the local variation of E^M is larger, which increases the difficulty of estimation.

Similar to Table 2.1, Table 2.2 does not show significant differences in texture flow estimation among different page, pose, and skew groups.

The block size ($X \times Y$ and $x \times y$) and angle sampling rate (Φ and ϕ) have an important effect on both accuracy and computational speed. Smaller values favor accuracy but slow computation. A good trade-off between accuracy and speed depends on the maximum texture flow curvature in the image. For planar pages, the curvature of texture flows is small, so we can use relatively large values. For

(mean/std)×1°	Major texture flow E^M	Minor texture flow E^m
All	0.31/0.04	0.90/0.34
Page 1	0.34/0.05	0.76/0.22
Page 2	0.30/0.03	0.66/0.21
Page 3	0.32/0.04	0.89/0.19
Page 4	0.31/0.02	0.89/0.28
Page 5	0.31/0.04	1.31/0.39
Pose no.1	0.31/0.04	1.04/0.21
Pose no.2	0.33/0.05	0.91/0.47
Pose no.3	0.33/0.03	0.96/0.15
Pose no.4	0.29/0.02	0.70/0.36
0° skew	0.30/0.02	0.91/0.34
15° skew	0.35/0.04	0.89/0.34
−15° skew	0.29/0.04	0.90/0.36

Table 2.1: Evaluation of texture flow estimation for synthetic images of planar documents.

curved pages, the optimal values should be computed using the estimated curvature of texture flows. However, this presents a dead lock. In practice, we simply use relatively small and fixed values.

(mean/std)$\times 1^\circ$	Major texture flow E^M	Minor texture flow E^m
All	0.80/0.16	1.12/0.44
Page 1	0.80/0.14	0.89/0.22
Page 2	0.76/0.17	0.90/0.27
Page 3	0.82/0.18	1.00/0.27
Page 4	0.82/0.17	1.14/0.41
Page 5	0.79/0.17	1.69/0.44
Pose no.1	0.83/0.14	1.20/0.34
Pose no.2	0.85/0.14	1.12/0.50
Pose no.3	0.75/0.13	1.30/0.46
Pose no.4	0.76/0.21	0.87/0.35
0° skew	0.72/0.13	1.03/0.45
15° skew	0.80/0.17	1.11/0.41
-15° skew	0.87/0.15	1.22/0.46

Table 2.2: Evaluation of texture flow estimation for synthetic images of curved documents.

2.6 Discussion

In this chapter, we presented an approach to detecting texture flow fields in document images. We extract two texture flow fields, which represent the local text line direction and vertical character stroke direction, respectively. Our method works with both planar and curved document images.

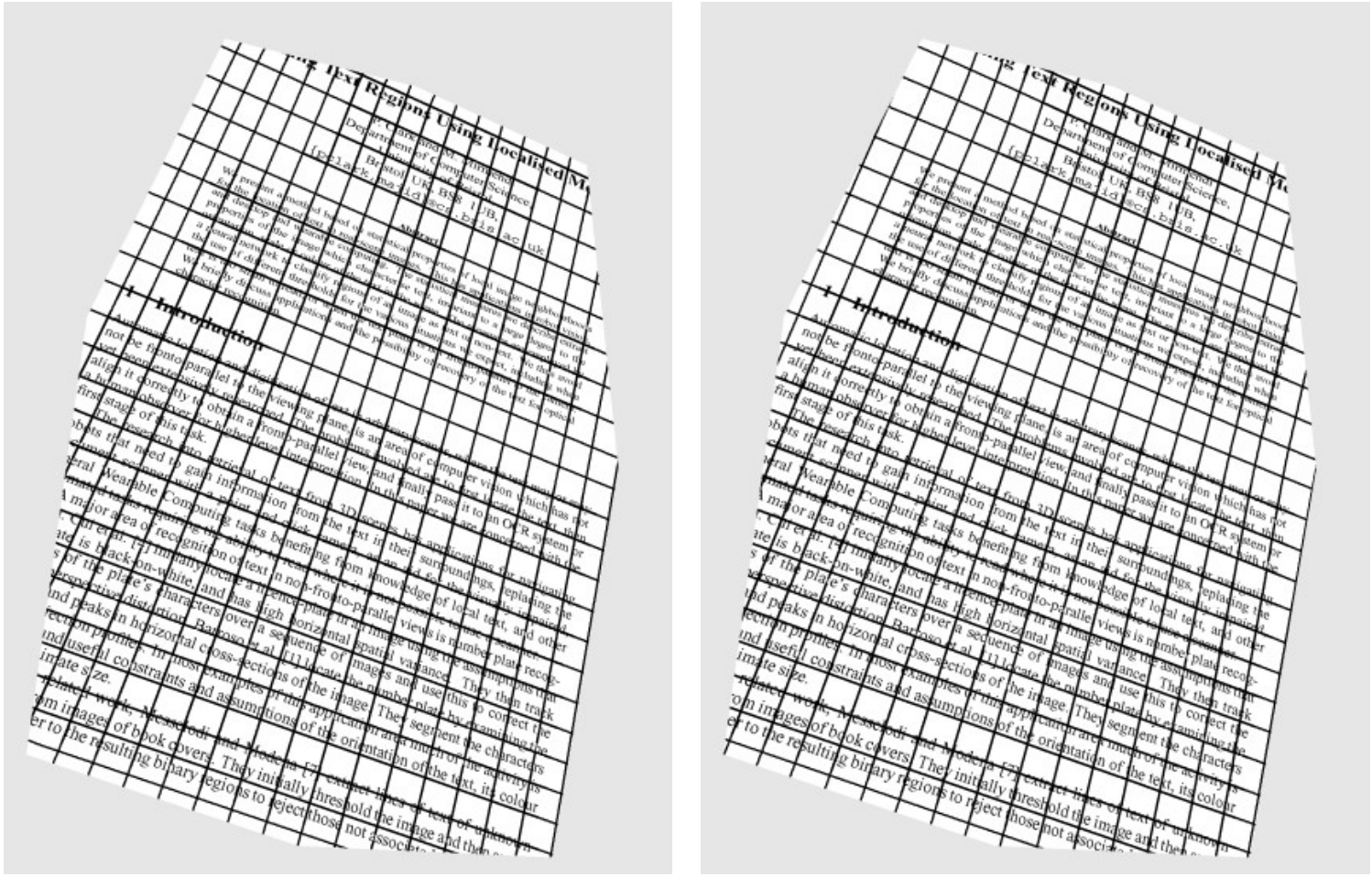
Figures 2.6, 2.7, and 2.8 show the ground truth and estimated texture flows from synthetic images of both planar and curved documents. Figure 2.9 shows estimated texture flows from two real images, in which one is a planar document and the other is a curved document. In all cases, both the major and the minor texture flows are quite accurate, with the major one slightly better. Most errors occur near text margins, mainly because less text is available to support the estimation.

As we have discussed, our text identification module is based on the high gradient property of text. In our synthetic images, it works well because the images contain no blurring. In real images, there are two possible problems. The first one is the loss of text in areas affected by out-of-focus or motion blur. This problem does not affect the accuracy of texture flow estimation in other areas. The second problem is false detection in strongly textured non-text areas such as pictures in the document or an object in the background. The second problem is more challenging because non-text elements cause errors in texture flow estimation and affect subsequent procedures including shape estimation and image rectification. To overcome it, text properties, other than high gradient, should be utilized. Because text identification in complex real images is a complicated problem deserving a thesis of its own and

there are many on-going research efforts addressing this problem (e.g., [11, 42]), we do not address this problem any more in this dissertation.

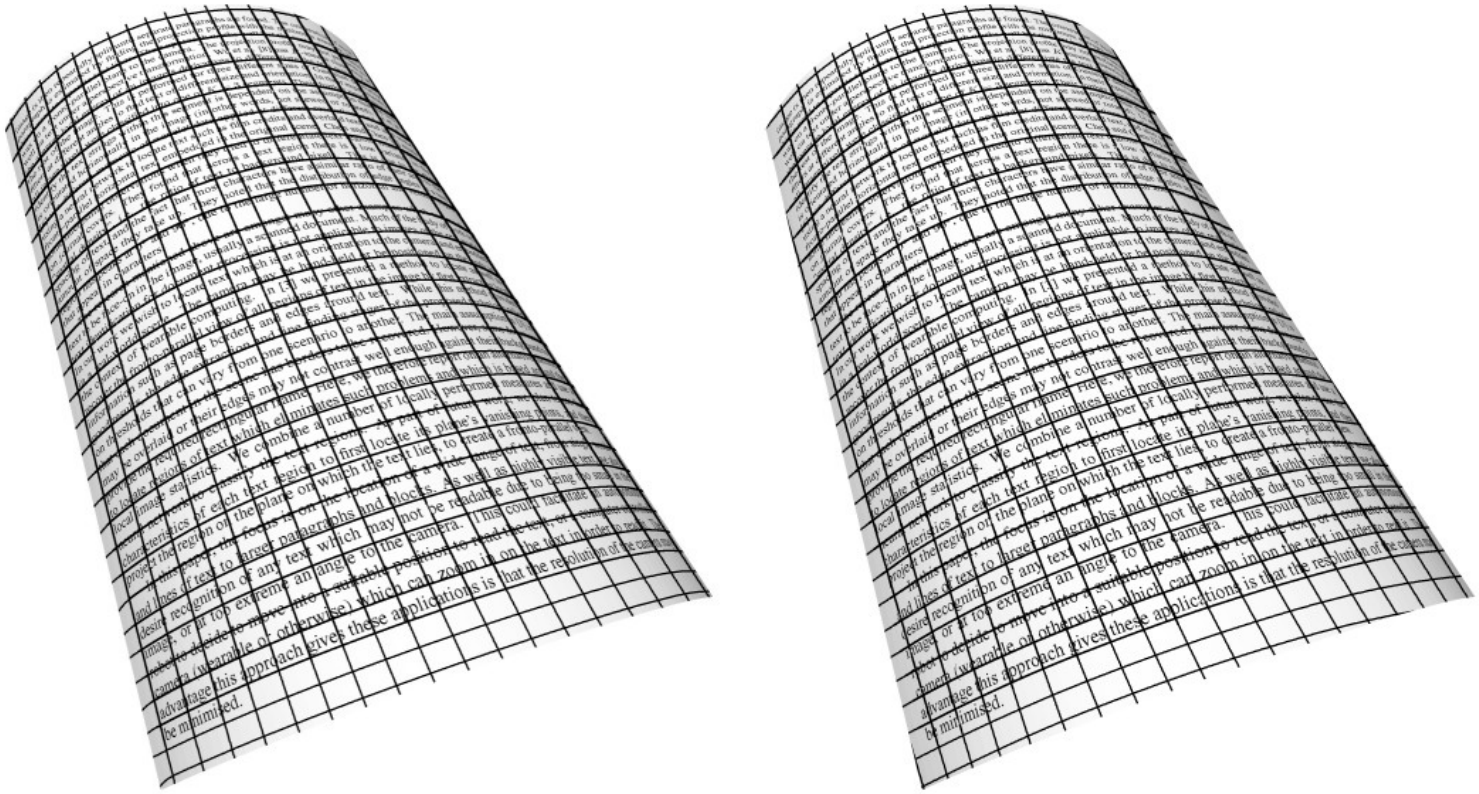
Problems with our texture flow detection algorithm arise mainly from errors in text identification. Other than that, accuracy drops at text boundaries compared to the interior of text areas because there is less text near boundaries to support the estimation. The limited length of vertical character strokes is the reason for relatively low accuracy of minor texture flow estimation. In our test images, this length is usually less than 25 pixels. Due to quantization, the end of the stroke may have an one-pixel displacement, which amounts to $2.3°(= tan^{-1}\frac{1}{25})$. As for the major texture flow, its accuracy is limited by the length of text line sections that are (almost) straight. In our data, it is much larger than 25 pixels. In general, we expect a higher accuracy for major texture flow estimation unless text lines are extremely curved. In such extreme cases, there is no need for rectification anyway.

With the mesh shown in Figures 2.6, 2.7, 2.8 and 2.9, it may seem inviting to finish rectification by 'morphing' the irregular mesh toward a rectangular mesh. While this method may be applied to a small portion of the page, it is inappropriate for the whole document because we do not know the aspect ratio of each cell in the mesh. We must recover the local surface orientation to obtain the aspect ratio of each cell, and further integrate local information in a global way to obtain the scale ratio of neighboring cells. In other words, we still need to estimate the page shape.



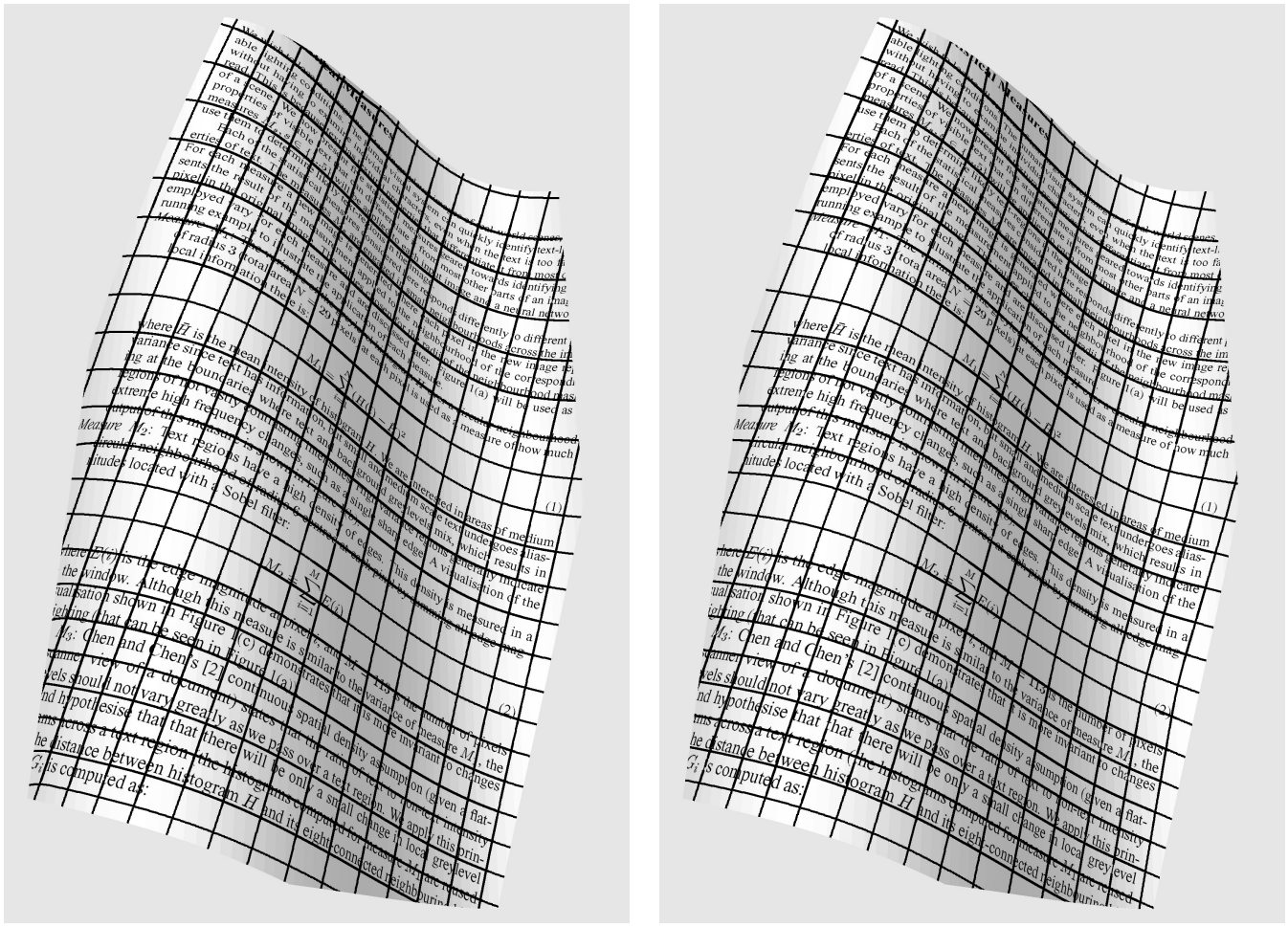
(a) (b)

Figure 2.6: Texture flow result of a synthetic planar page. (a) Ground truth (b) Estimation



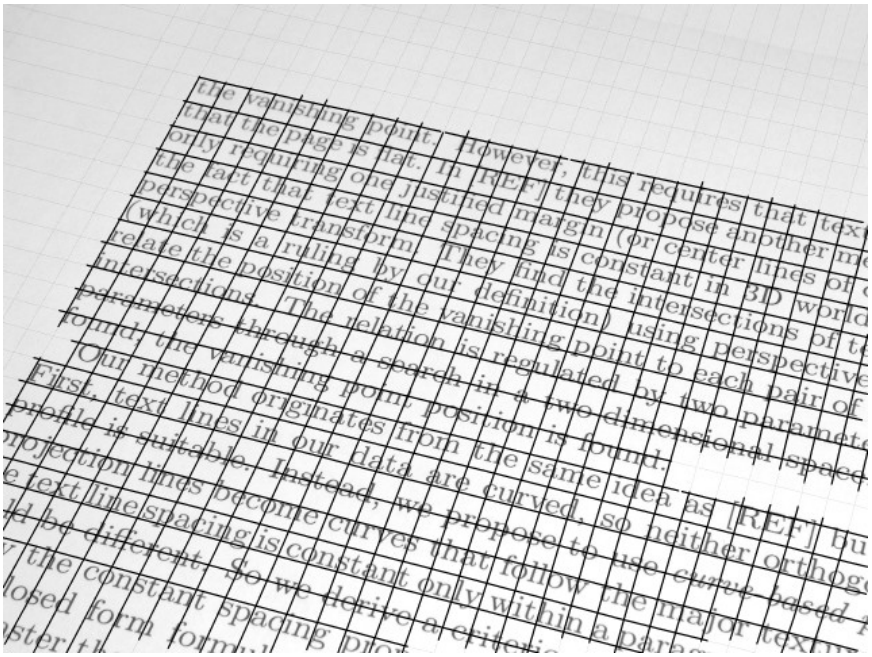
(a) (b)

Figure 2.7: Texture flow result of a synthetic curved page. (a) Ground truth (b) Estimation

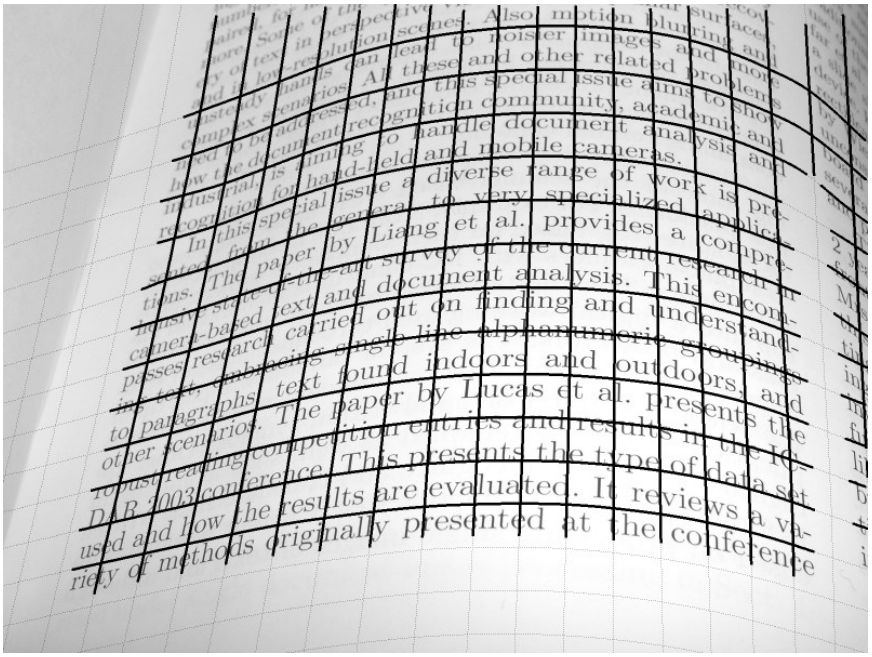


(a) (b)

Figure 2.8: Texture flow result of another synthetic curved page. (a) Ground truth (b) Estimation



(a)



(b)

Figure 2.9: Texture flow results on real images. (a) Planar page (b) Curved page

Chapter 3

Rectification of Camera-captured Document Images

3.1 System overview

The system work flow of our image rectification framework is illustrated in Figure 3.1. The text identification and texture flow estimation modules were described in Chapter 2.

Because a plane is a special case of a curved surface, our shape estimation method developed for curved pages can also handle planar pages. However, for planar pages we can use a simpler method that is faster. Therefore, we develop a hypothesis testing module to discriminate planar and curved documents, using the geometric property of surfaces and texture flows. For planar pages, we use the two texture flow fields to find the horizontal and vertical vanishing points and compute the homogeneous transformation matrix with which we remove the perspective distortion from the original image. For curved pages, we model the page shape by a developable surface, which can be approximated by a group of planar strips. We estimate each strip's position and flatten the page by rectifying the planar strips in a piecewise manner.

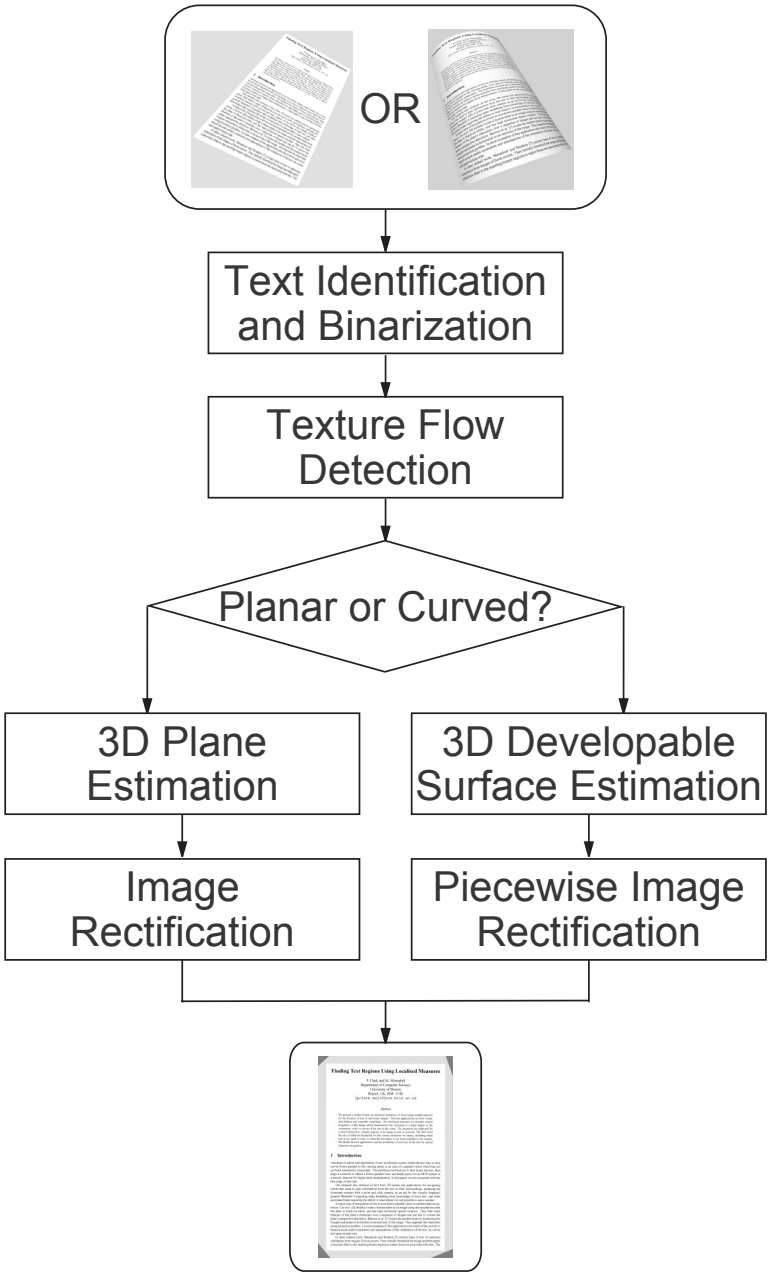


Figure 3.1: Work flow of geometric image rectification

3.2 Rectification of planar document images

3.2.1 Related work

An image of a planar document page captured by a camera is subject to perspective distortion unless the optical axis of the camera remains perpendicular to the page. As a result of the perspective distortion, the characters further away from the camera appear smaller, and text lines appear convergent at one point, called the *vanishing point*. In general, the projection transformation from a 3D world plane to the 2D image plane can be described by a 3×3 homogeneous matrix $\mathbf{H}$ with eight degrees of freedom (dof). If all eight dof's are known, the perspective distortion can be removed completely. From the OCR viewpoint, however, not all of the eight dof's are equally important.

It is shown in [41] that $\mathbf{H}$ can be uniquely decomposed into a concatenation of three matrices, $\mathbf{S}$, $\mathbf{A}$ and $\mathbf{P}$, which are similarity, affine, and 'pure projective' transformations, respectively:

$$\mathbf{H} = \mathbf{SAP} = \begin{pmatrix} s\mathbf{R}_{2\times 2} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{\beta} & -\frac{\alpha}{\beta} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ l_1 & l_2 & l_3 \end{pmatrix}$$

In matrix $\mathbf{S}$, s is an isotropic scaling factor that cannot be determined from the image alone and is not part of the eight dof's; $\mathbf{R}_{2\times 2}$ is a 2D rotation matrix involving one angle, and $\mathbf{t}$ is a translation vector defining x- and y- displacements. Matrix $\mathbf{S}$ contains three dof's in total. From a single image, $\mathbf{t}$ cannot be determined, neither is it important for our rectification purpose. So, we are reduced to only one

rotation angle.

In matrix $\mathbf{P}$, $\mathbf{l}_\infty = (l_1, l_2, l_3)^T$ represents the vanishing line of the world plane in the image. This line contains two dof's because the representation of $\mathbf{l}_\infty$ is homogeneous. In matrix $\mathbf{A}$ two more dof's exist, α and β, and their geometric interpretation specifies the images of the circular points [62]. Overall, the rotation angle, the vanishing line, and the circular points contain five dof's which determine a *planar metric rectification* [41].

In [41], it is shown that the following knowledge of the world plane can constrain these five dof's:

- A known angle between lines;
- Equality of two (unkown) angles;
- A known length ratio.

Any combination of sufficient (at least five) and independent constraints can be used to solve for a planar metric rectification. Dependent constraints can only improve the accuracy [41]. Zandifar et al. [75] apply this method to rectify presentations and posters captured by video cameras. They find line segments in the image using edge detection and mean shift clustering. Lines of similar directions in the image are assumed parallel in the world plane and grouped together, and lines in groups with significantly different dominant directions are assumed orthogonal in the world plane. Typically, these lines correspond to poster boundaries, presentation frames, and text lines in the poster/presentation. Their method requires five pairs of

lines orthogonal in the world plane, which provide five right angles, or, equivalently, the horizontal and vertical vanishing points[1] and two right angles, to solve for the five dof's.

In practice, however, it is difficult, if not impossible, to obtain five *independent* constraints. For example, if the edges of five right angles come from two groups of parallel lines in the world, then they provide only three independent angles [41]. This problem can be circumvented by reducing the number of required dof's [47]. The rotation in $\mathbf{R}$ is not necessary because, from the OCR point of view, it can be handled by deskewing. The two dof's in α and β can be interpreted as the ratio $\frac{\alpha}{\beta}$, which controls the shearing, and $\frac{1}{\beta}$, which controls the x-to-y aspect ratio. This aspect ratio is not critical from the viewpoint of OCR because most OCR engines automatically normalize each character image to a fixed size. This leaves us with only three dof's, which are immediately solvable using the horizontal and vertical line segments, or, equivalently, using the horizontal and vertical vanishing points [13, 14].

Pilu [56] uses a bottom up method to locate both horizontal and vertical linear clues. They use the convergent points of linear clues as the two vanishing points. Clark et al. [13, 14] use a *perspective* projection profile analysis method, similar to the *orthogonal* projection profile analysis typically used in deskewing, to locate the horizontal vanishing point. They propose two methods for vertical vanishing point detection. In [13], they locate the vanishing point by computing the convergent point of the two margins of a fully justified text block. In [14], they extract the left

[1]The two vanishing points are equivalent to three angle constraints.

margin of a left justified text block or the central line of centered text block, then use the equidistant property of text lines to compute the position of the vertical vanishing point along the margin or central line. The equidistant property of text lines is equivalent to a *known-length-ratio* constraint. When not sufficient text is present, page or frame boundaries can be used [12] if they are visible.

3.2.2 Discrimination of planar and curved documents

Perspective projection preserves linearity, so straight text lines on planar documents remain straight in the camera-captured image. Furthermore, these co-planar and parallel 3D lines share a common vanishing point in the image [26]. These two properties do not hold true for curved text lines on curved documents[2]. Therefore, we can determine whether an image contains a planar or curved document by testing the linearity and convergence of text lines, which, in our case, can be verified using the major texture flow field. For the same reason, the minor texture flow field is also useful.

Let $\{p_i\}_{i=1}^{N}$ be a set of points evenly sampled in the text area, and $\{\alpha_i\}$ be the flow directions at sample points. The homogeneous representation of the flow tangent line at each point (a line passing through the given point with the direction of the flow) is given by

[2]Under perspective projection, if a curve lies on a *plane of sight* (a plane passing through the optical center), then its projection is a straight line in the image. However, text lines on a curved document can simultaneously satisfy this requirement. Their projections cannot converge at a single point, either.

$$\mathbf{l}_i = \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix} \times \begin{pmatrix} \cos\alpha_i \\ \sin\alpha_i \\ 0 \end{pmatrix}$$

where $p_i = (x_i, y_i)$.

Under the planar page hypothesis, all these flow tangent lines converge at a vanishing point, say $\mathbf{v}$ (in homogeneous representation), which can be written as

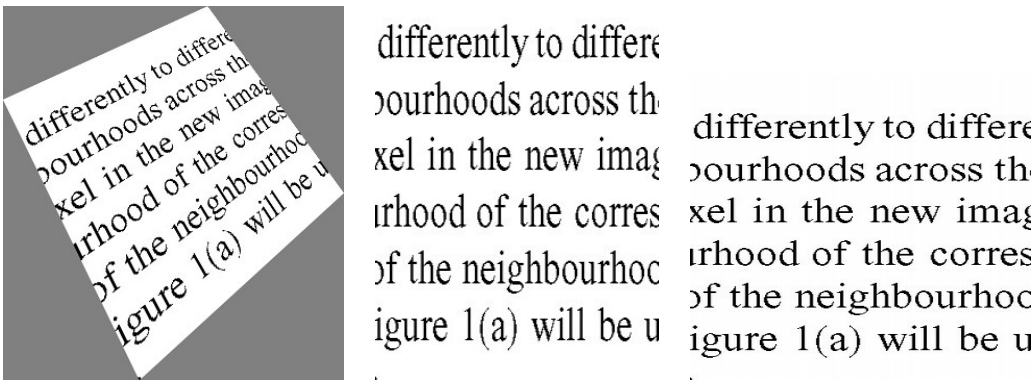
$$\mathbf{l}_i^\top \mathbf{v} = 0, \forall i.$$

This means that $\mathbf{v}$ lies in the null space of the sub-space spanned by $\{\mathbf{l}_i\}$; in other words, the rank of $\mathbf{L} = (\mathbf{l}_1, \ldots, \mathbf{l}_N)$ is less than three. On the contrary, under the curved document hypothesis, $\mathbf{v}$ does not exist, which means that the null space of $\mathbf{L}$ is $\emptyset$ and $\mathbf{L}$ has full rank.

We use SVD decomposition to test the hypotheses. Let S_1 and S_3 be the largest and least eigenvalues of $\mathbf{L}$, respectively. We use S_3/S_1 as the convergence quality measure. If it rests below a predefined threshold, we decide that $\mathbf{L}$ does not have full rank. In our implementation, we have a tighter threshold for the test based on major texture flow field, and a weaker one for minor field, because major texture flow field estimation is more accurate. If either test indicates that $\mathbf{L}$ has full rank, we decide that the document is curved; otherwise, it is planar.

3.2.3 Plane surface estimation

For planar document images, as a result of the previous hypothesis test, we obtain $\mathbf{v}_h$ and $\mathbf{v}_v$, the vanishing points of major and minor texture flow tangent lines. As Section 3.2.1 shows, a metric rectification has five dof's. The line connecting $\mathbf{v}_h$ and $\mathbf{v}_v$ is $\mathbf{l}_\infty$, the vanishing line of the world plane. The knowledge of $\mathbf{l}_\infty$ reduces the projective transformation to an affine transformation. The positions of the vanishing points in the world plane (the infinity points at North and East) allow us to remove the shearing and rotation from the affine transformation. However, we cannot recover the x-to-y ratio. In summary, we can obtain a homogeneous transformation that remove perspective distortion up to an unknown x-to-y aspect ratio. In practice, we set the x-to-y ratio one. For OCR, this is sufficient given the discussion in Section 3.2.1.



(a) (b) (c)

Figure 3.2: Non-unique image rectification results. (a) A perspective distorted image. (b) and (c) are two possible rectification results that have different x-to-y aspect ratios.

If we assume a zero offset for the principal point in the image plane, we can further compute the camera focal length and hence the surface normal. Suppose the two vanishing points are $\mathbf{v}_h = (x_h, y_h)^\top$ and $\mathbf{v}_v = (x_v, y_v)^\top$, then the 3D directions of the horizontal and vertical lines on the page in the camera coordinate system are given by

$$\begin{aligned} \mathbf{V}_h &= (\mathbf{v}_h^\top, f)^\top, \\ \mathbf{V}_v &= (\mathbf{v}_v^\top, f)^\top, \end{aligned} \tag{3.1}$$

where f is the focal length. Due to their orthogonality,

$$\mathbf{V}_h^\top \mathbf{V}_v = 0, \tag{3.2}$$

and it follows that

$$f = \sqrt{-\mathbf{v}_h^\top \mathbf{v}_v},$$

if $\mathbf{v}_h^\top \mathbf{v}_v < 0$.

Special care should be taken when either $\mathbf{v}_h$ or $\mathbf{v}_v$ lies at the infinity of the image plane. Whether at the infinity or not, let us define $\overline{\mathbf{v}}_h$ and $\overline{\mathbf{v}}_v$ as the unit 2D vectors in their directions respectively. In theory, only two cases are possible. In the first, both vanishing points are at the infinity. In this case, Equation 3.1 becomes

$$\begin{aligned} \mathbf{V}_h &= (\overline{\mathbf{v}}_h^\top, 0)^\top, \\ \mathbf{V}_v &= (\overline{\mathbf{v}}_v^\top, 0)^\top, \end{aligned}$$

and Equation 3.2 becomes

$$\overline{\mathbf{v}}_h^\top \overline{\mathbf{v}}_v = 0,$$

neglecting f at all. Therefore, we cannot solve for f. However, this case implies that the page is parallel to the image plane, so there is neither perspective distortion nor the need for rectification.

In the second case, only one vanishing point lies at the infinity. Without loss of generality, let $\mathbf{v}_h$ be at the infinity. This means that all text lines in the 3D world are parallel to the image plane, so their projections in the image plane are also parallel lines. Meanwhile, the 3D minor texture flow is not parallel to the image plane, causing foreshortening in this direction. In this case, Equation 3.1 becomes

$$\begin{aligned} \mathbf{V}_h &= (\overline{\mathbf{v}}_h^\top, 0)^\top, \\ \mathbf{V}_v &= (\mathbf{v}_v^\top, f)^\top, \end{aligned}$$

and Equation 3.2 becomes

$$\overline{\mathbf{v}}_h^\top \mathbf{v}_v = 0,$$

again not involving f. Therefore we still cannot solve for f. This time, however, we do have perspective distortion. Since we do not have f, we are back to the situation where we can remove the distortion up to an unknown aspect ratio.

In practice, due to noise, we may arrive at theoretically impossible vanishing point combinations. It could be that $\mathbf{v}_h^\top \mathbf{v}_v > 0$; or at least one vanishing point

lies at the infinity, but $\overline{\mathbf{v}}_h^\top \overline{\mathbf{v}}_h \neq 0$. Whatever the situation, we cannot solve for f and can only compute a homogeneous transformation to remove the shearing and foreshortening, leaving an unknown x-to-y ratio.

In the general case where f can be computed, we can calculate the plane orientation as

$$\mathbf{N} = \mathbf{V}_h \times \mathbf{V}_v,$$

where $\mathbf{N}$ is the plane normal vector, and $\times$ denotes the cross product operation. The full knowledge of the plane surface is then determined by f and N.

3.2.4 Metric rectification

Having determined the page plane, we can remove perspective distortion completely. The needed homogeneous transformation is computed in the following way:

Consider an arbitrary point, (x_0', y_0'), in the image plane. In the camera's 3D coordinate system, its position is $(x_0', y_0', f)^\top$, where f is the focal length. The corresponding 3D point, $\mathbf{W}$, in the document page must lie on the line of sight through the optical center and the point $(x_0', y_0', f)^\top$. So, $\mathbf{W} = d(x_0', y_0', f)^\top$, where $d(> 0)$ is an unknown depth factor. Let $\overline{\mathbf{V}}_h (= \mathbf{V}_h/|\mathbf{V}_h|)$ and $\overline{\mathbf{V}}_v (= \mathbf{V}_v/|\mathbf{V}_v|)$ be the 3D unit vectors representing the directions of 3D major and minor texture flows. Suppose that we set up a 2D coordinate system in the document plane so the x-axis is aligned with $\overline{\mathbf{V}}_h$ while the y-axis is (must be) aligned with $\overline{\mathbf{V}}_v$. Every point on the document plane, thus, has a 2D coordinate (x, y). Assume that $\mathbf{W}$ is at (x_0, y_0) within the 2D coordinate system, then the 3D position, $\mathbf{P}$, of any point (x, y) in the

document plane can be computed by

$$\mathbf{P} = (x - x_0)\overline{\mathbf{V}}_h + (y - y_0)\overline{\mathbf{V}}_v + \mathbf{W},$$

or, in matrix form,

$$\mathbf{P} = \begin{pmatrix} \overline{\mathbf{V}}_h & \overline{\mathbf{V}}_v & \mathbf{W} \end{pmatrix} \begin{pmatrix} 1 & 0 & -x_0 \\ 0 & 1 & -y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}.$$

The camera's internal parameters determine the transformation from 3D world coordinate system to camera image plane. The parameters of a general projective camera model can be defined by a 3×3 upper triangular matrix $\mathbf{K}$, which has five degrees of freedom [26]. Two dof's correspond to the offset of the principal point in image plane; one is the focal length; one is the x-to-y pixel aspect ratio; the last one is a shear parameter. Most digital cameras have unit x-to-y ratio and zero shear. Also, the principal point offset is typically zero. Therefore, the $\mathbf{K}$ matrix can be simplified to

$$\mathbf{K} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

where f is the focal length. A 3D point $\mathbf{P} = (X, Y, Z)^\top$ in the camera's coordinate system projects to a point (x', y') in the image by

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \mathbf{K} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}, \tag{3.3}$$

where

$$\begin{aligned} x' &= u/w, \\ y' &= v/w. \end{aligned}$$

Thus, the homogeneous transformation from document plane to image plane is the concatenation

$$\mathbf{H} = \mathbf{K} \begin{pmatrix} \overline{\mathbf{V}}_h & \overline{\mathbf{V}}_v & \mathbf{W} \end{pmatrix} \begin{pmatrix} 1 & 0 & -x_0 \\ 0 & 1 & -y_0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{3.4}$$

The inverse of $\mathbf{H}$ maps every point in the image plane back to the frontal-flat view of the document page and is called the *rectification* matrix. That is,

$$\begin{aligned} (x, y) &\overset{\mathbf{H}}{\rightarrow} (x', y'), \\ (x', y') &\overset{\mathbf{H}^{-1}}{\rightarrow} (x, y). \end{aligned}$$

In Equation 3.4, d and (x_0, y_0) can take any value. The value of (x_0, y_0) decides an irrelevant translation of the rectified image within the destination plane. The depth factor d determines the scale of the rectified image — the larger the depth, the larger the rectified image.

Suppose $\mathbf{W} = (0, 0, df)^\top$, and $(x_0, y_0) = (0, 0)$. Let $\mathbf{v}_h = (x_h, y_h)^\top$ and $\mathbf{v}_v = (x_v, y_v)^\top$ so $\mathbf{V}_h = (x_h, y_h, f)^\top$ and $\mathbf{V}_v = (x_v, y_v, f)^\top$. Then, Equation 3.4 becomes

$$\mathbf{H} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_h & x_v & 0 \\ y_h & y_v & 0 \\ f & f & df \end{pmatrix} = f \begin{pmatrix} x_h & x_v & 0 \\ y_h & y_v & 0 \\ 1 & 1 & d \end{pmatrix}. \tag{3.5}$$

and it follows that

$$\mathbf{H}^{-1} = \frac{1}{df(x_h y_v - x_v y_h)} \begin{pmatrix} y_v d & -x_v d & 0 \\ -y_h d & x_h d & 0 \\ y_h - y_v & x_v - x_h & x_h y_v - x_v y_h \end{pmatrix},$$

$$\begin{aligned} x' &= \tfrac{x x_h + y x_v}{x+y+d}, \\ y' &= \tfrac{x y_h + y y_v}{x+y+d}, \end{aligned}$$

and

$$\begin{aligned} x &= d\tfrac{y_v x' - x_v y'}{(y_h - y_v)x' + (x_v - x_h)y' + (x_h y_v - x_v y_h)}, \\ y &= d\tfrac{x_h y' - y_v x'}{(y_h - y_v)x' + (x_v - x_h)y' + (x_h y_v - x_v y_h)}, \end{aligned} \tag{3.6}$$

which maps a point, (x', y'), in the input image to the rectified image, (x, y). Equation 3.6 shows that the magnitudes of x and y are proportional to the magnitude of d. This confirms our claim that the depth of $\mathbf{W}$ determines the scale of the rectified image.

Because we cannot determine the positive directions of $\mathbf{V}_h$ and $\mathbf{V}_v$ (corresponding to the 'left' and 'up' in a flat document) from texture flow analysis alone, the rectified image may be reversed in x- or y- direction, or both. A simple solution is to pass the rectification result and several reversed versions to an OCR engine

and select the one with the best recognition confidence. More sophisticated methods also exist [70].

Some examples of camera-captured planar documents (both synthetic and real) and their rectification results are shown in Figures 3.3, 3.4, and 3.5. In Figure 3.5, the top two real images in the left column are captured with the camera pointing straight at the document, resulting in no perspective distortion and both the horizontal and vertical vanishing points at the infinity. In the bottom image, only the vertical vanishing point is at the infinity. In all three cases, full metric rectification is impossible. However, the rectified images are satisfactory.

3.3 Rectification of curved document images

3.3.1 Related work

Compared with flat page rectification, the de-warping of curved document images is significantly more difficult because of the infinite number of dof's associated with the surface geometry.

One way of dealing with the surface reconstruction problem involves completely removing the unknown dof's with direct 3D shape knowledge obtained from special equipment. Brown et al. [5] use a structured light system to gather range data of deformed manuscripts, and they combine this data with the 2D image captured by a digital camera to restore the flat frontal view. In their work, the 3D surface is represented by a triangular mass-spring particle mesh, which has an energy function associated with the lengths of the springs. They 'force' the mesh to

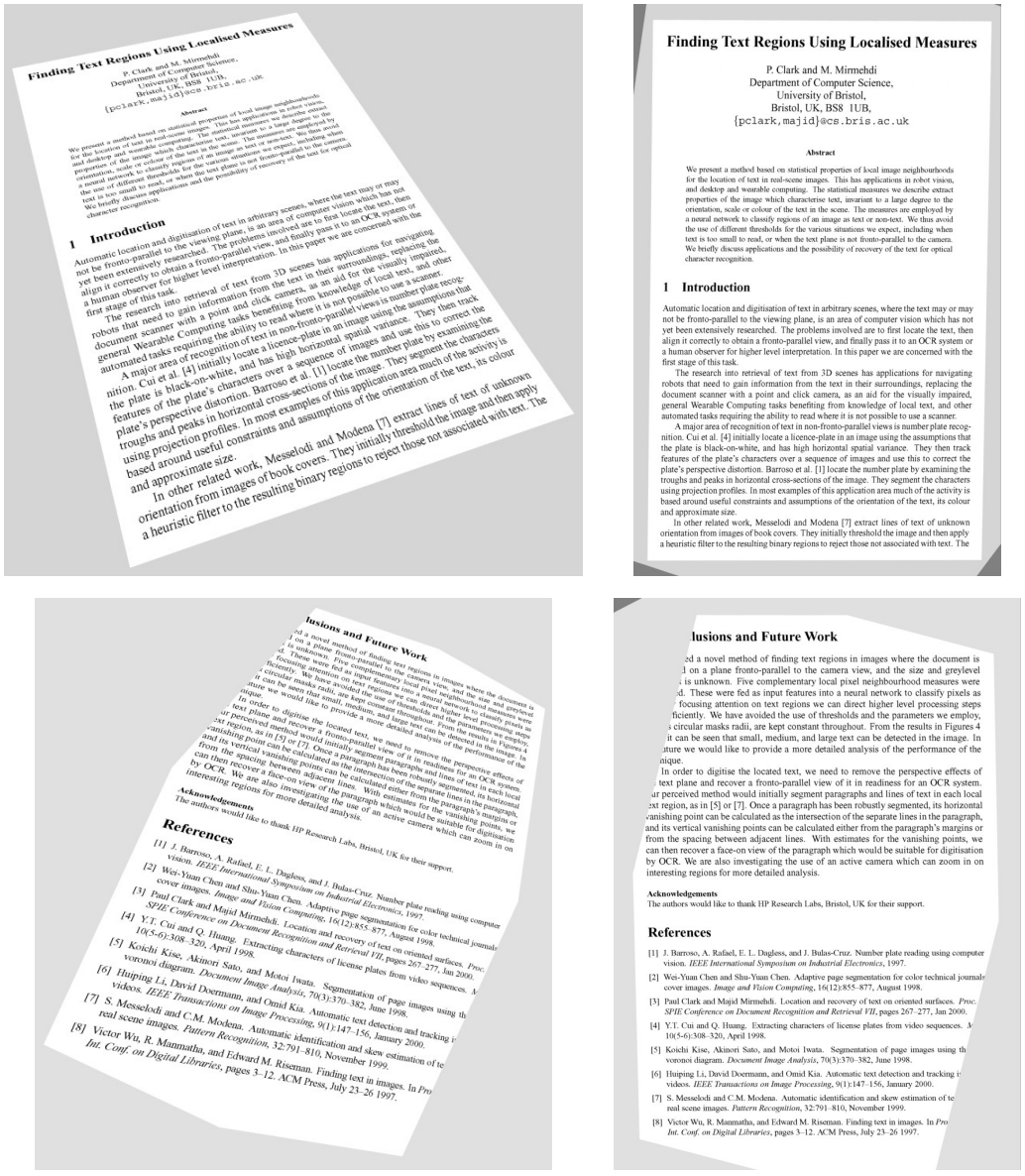


Figure 3.3: Comparison of synthetic images of planar documents and rectification results. The false text margin in the lower image has no effect on the result.

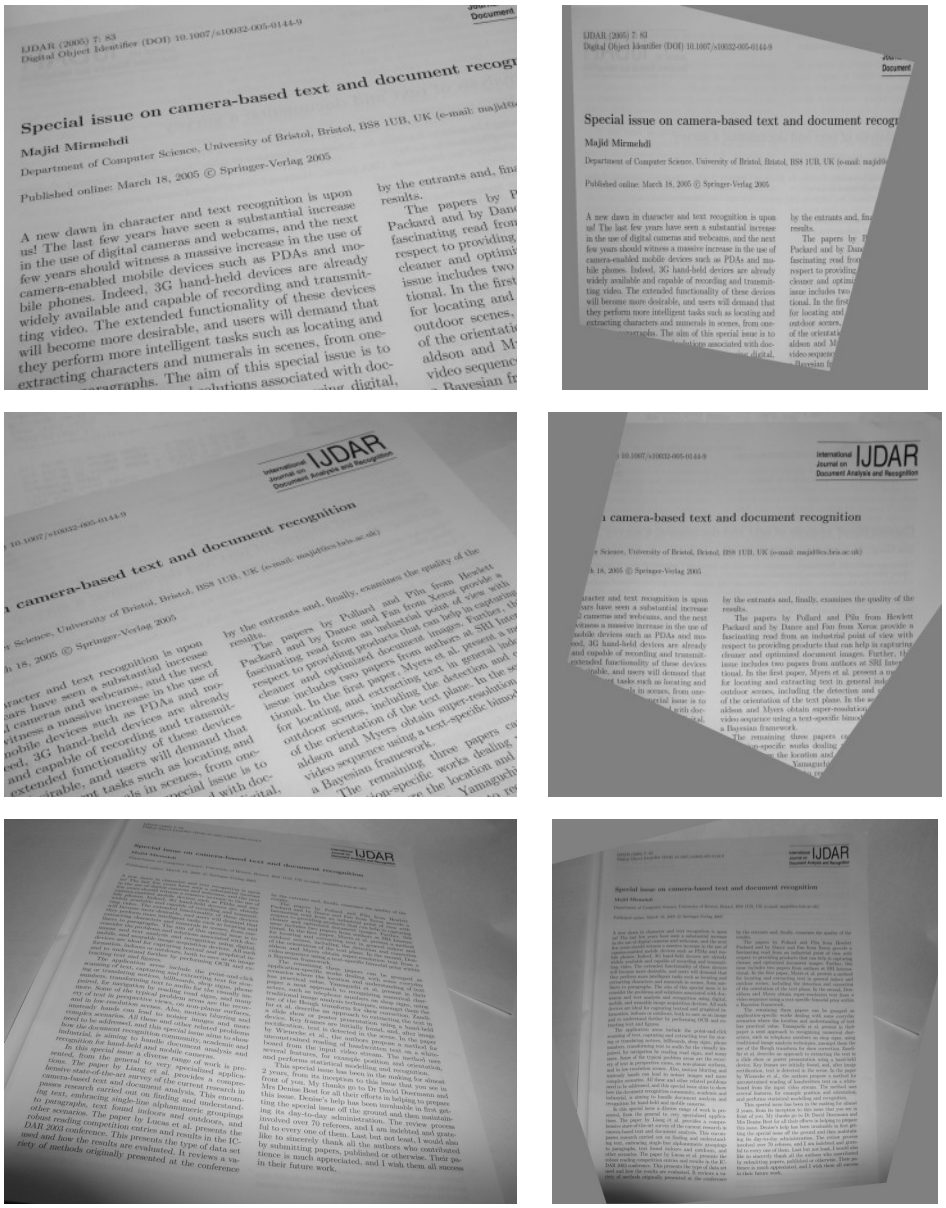


Figure 3.4: Comparison of images of real planar documents and rectification results. Both full page and partial page can be handled.

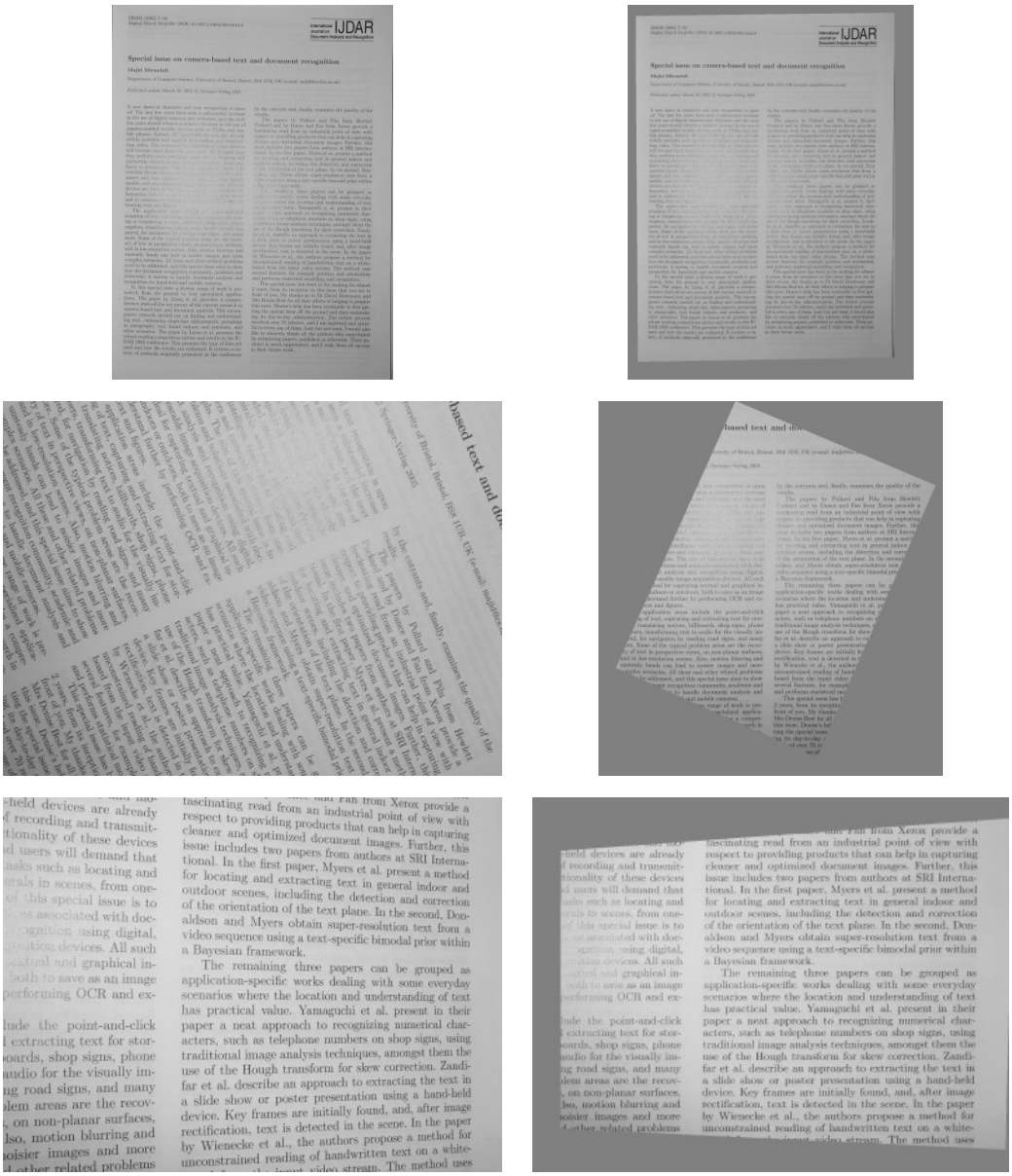


Figure 3.5: Comparison of images of real planar documents and rectification results.

Results are satisfactory despite full metric rectification is unavailable.

a flat plane under minimum energy constraint through an iterative process. This is similar to physically flattening the manuscript with minimal tearing and stretching. Their extended work [4] revises the energy as a function of the angles between springs, and this leads to a one-pass computation without iteration. Pollard et al. also use a structured light system to obtain range data of opened books, then fit a discrete developable surface to the 3D data through an iterative optimization [58]. The basic ideas of [5, 4] and [58] share many common elements.

Realizing that 3D data or calibration is not always available in many applications, researchers have tried to induce indirectly the shape from a single 2D image. However, 3D shape estimation from a single 2D image alone presents a difficult, if not impossible, problem for general cases. The theories in this area heavily rely on prior knowledge of lighting or surface geometry, and many treat only orthographic projection model equivalent to a pin-hole camera with infinite focal length. However, for many cameras (particularly the zoomless cameras attached to mobile devices), the focal length tends to be small. The lighting condition in camera-captured images is typically unconstrained and unknown a priori, which defies the use of shape-from-shading methods. The markings on document pages are usually neither isotropic (considering that the space between lines is much wider than the space between words or characters), nor do they form distinct and repetitive textel structures. Therefore, shape-from-texture methods do not suit this case well. For stereo vision or structure-from-motion techniques, they do not work with a single image. Therefore, general shape-from-X techniques can give some qualitative, but not accurate quantitative, 3D information needed for our task.

In some special cases, people use additional constraints to facilitate the shape estimation process. For example, for the scans of books in flatbed scanners, Zhang et al. [77] propose a shape-from-shading based method to estimate the curved shape of opened books. Their work pivots on two key points: 1) the page shape near the spine of an opened book is cylindrical, and 2) during scanning the flatbed scanner casts light at a fixed direction. Therefore, the shading on the white area of the paper is proportional to the cosine of the angle between the surface normal and the incident light. By comparing the shade to the white area (in the flat part of the page), they can recover the local surface orientation. In [7], Cao et al. address opened books captured by cameras. They also assume that the page shape is cylindrical, and they require a straight frontal view of the page so the cylinder's generatrix is orthogonal to the camera's optical axis. Under these conditions, they extract text lines on the page to locate the cylinder's directrices and compute the 3D directrix shape from two 2D curved text lines. In [69], Ulges et al. make the same cylindrical shape and straight frontal view assumptions, but take a different approach. They observe that line spacing in 3D world is uniform over the entire page, while change of depth leads to change of line spacing in the image. By investigating this changes in line spacing, they can estimate the depth difference up to a scaling factor. They locate the sheared bounding boxes of each character (which in 3D space are rectangles), use the depth information to infer the aspect ratios of each box, and restore their correct rectangular shape. In [24], a method is proposed to estimate the shape of a developable surface. It has the potential of handling document page under general poses. However, its requirement for metric knowledge of a closed contour in the

surface limits its applicability.

Given the difficulty involved in 3D structure analysis, another alternative focuses on 2D image processing techniques that could counter the distortion caused by curved surface and perspective projection. Such methods usually aim to straighten curved text lines to meet OCR requirements. Zhang and Tan [78, 79] restore the overall linearity of text lines in scans of thick books where curve distortion becomes obvious near the book spine. They locate the curved portion of a text line by clustering nearby connected components, and they move the components back to the baseline determined by other components in the straight portion. However, the distortion of each component (character) is not addressed. Wu and Agam [73] build a mesh using curved text lines and morph the mesh toward a rectangular mesh in order to reduce the distortion. Their method does not address the perspective foreshortening in the vertical direction.

Given multiple images, the shape can be estimated using stereo vision, or other structure-from-motion techniques. In [68], Ulges et al. report their work on recovering document shape using general stereo vision techniques. They place the document in a cubic frame with known physical size and take two images from slightly different positions. The cubic frame enables them to compute the epipolar geometry, which simplifies point matching in two images and calibrates the stereo camera. They obtain depth information using standard triangulation given the correspondence in two images and camera calibration. The rest of their work on flattening is similar to [5, 4, 58].

3.3.2 Page surface modeling

The shape of a curved document belongs to a family of 2D surfaces called *developable* surfaces, as long as the document is not torn, creased, or deformed by a soak-and-dry process. In mathematical terms, developable surfaces are 2D manifolds that can be isometrically mapped to an Euclidean plane. In other words, developable surfaces can unroll to a plane without tearing or stretching. This developing process preserves intrinsic surface properties, such as arc length and angle between lines on the surface. Because of their ability to map to a plane, developable surfaces are widely used to construct sophisticated volumes out of flat material with minimum stretching, ranging from ships to French fries containers.

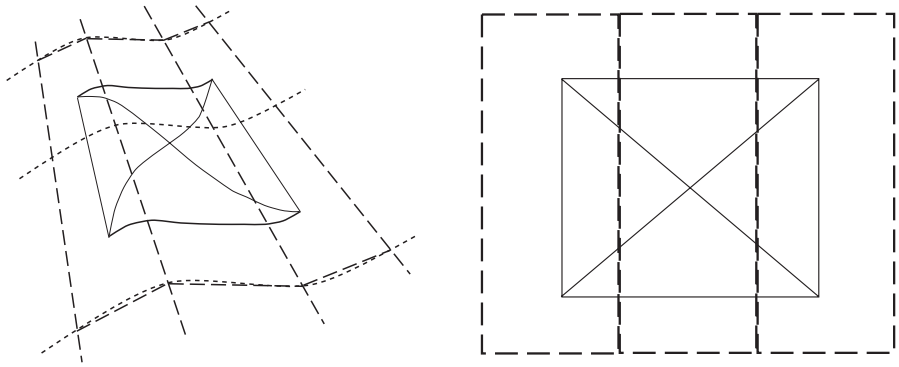
Developable surfaces represent particular cases of a more general class of surfaces called *ruled* surfaces. Ruled surfaces are envelopes of a one-parameter family of straight *lines* in 3D space. That is, a ruled surface can be thought to be swept in space by straight lines (*rulings* or *rectilinear generators*) which lie entirely on the surface. The degrees of freedom of a ruled surface are significantly fewer than for a general 2D surface because we can specify the surface by 1) giving a curve (the *directrix*) on the surface that crosses all rulings and 2) specifying the ruling direction at every crossing point. Developable surfaces are even more restricted; they are envelopes of a one-parameter family of *planes*. For developable surfaces, all tangent planes at points along one ruling coincide, which means the movement of the rectilinear generator cannot be arbitrary. Given this property, we can approximate a developable surface with a finite number of planar strips that come from

the family of tangent planes. Although this is only a first order approximation, it is sufficiently accurate for our application, while greatly cutting the number of unknowns for which we need to solve.

It is well known [22] that a developable surface can be either a flat plane, a cylinder, a cone, the envelope of tangents to a twisted 3D curve, or the combination of any of the above connected smoothly at common rulings. Let us consider how to represent each type of developable surface with planar strips. For planes, the approximation is precise and trivial. For cylinders, rulings are parallel; for cones, rulings are convergent at a vertex. Any two rulings of a cylinder or a cone, therefore, are co-planar. The strip on the common plane between two neighboring rulings can approximate the small piece of cylindrical or conic surface between the two rulings. Any two neighboring planar strips formed in this way join seamlessly at the common ruling border (see Figure 3.6). However, this statement does not hold true for a developable surface formed by tangents to a twisted 3D curve, where these tangents (rulings) are not necessarily co-planar. In general, two co-planar lines may not exist on such a surface. For a small portion of the surface, of course, we can find an optimal plane to approximate it. For example, if given two segments of two rulings, we can optimize the plane by minimizing the sum of squares of distances between segment end points and their projections on the plane. However, two neighboring strips constructed this way will not join seamlessly. This being said, it does not present a prohibitive problem. First, the seams decrease as we increase the number of strips so rulings become denser. Second, in practice, planes, cylinders and cones are sufficient to describe most curved documents' shape. Any deviation from these

basic types is small compared to other sources of noise, such as lens distortion.

Figure 3.6 illustrates our idea of approximating a curved document surface with planar strips. As the number of strips increases, the approximation becomes increasingly accurate. The de-warping can be accomplished by rectifying the strips piece by piece.



(a) (b)

Figure 3.6: Strip-based approximation to a developable surface. (a) Three planar strips approximate a developable surface. (b) The surface is de-warped piecewise.

3.3.3 Projected rulings

Rulings are straight 3D lines that lie entirely on the curved document page. We call their projections on the image *projected rulings*, or *2D rulings*. To find 3D rulings, we first need to locate their 2D projections. Similarly, we can distinguish 2D texture flows detected in the image and their 3D counterparts on the document surface. A 3D texture flow field defines a 3D orientation, or, equivalently, a unit 3D vector for every point on the document. This vector lies in the tangent plane at that point.

Recall that all points along a ruling on a developable surface share the same tangent plane. It follows that the 3D flow vectors at all points along a ruling lie in a common tangent plane. Furthermore, they are all parallel[3].

On the other hand, if the texture flow vectors at all points along a 3D curve are parallel, this curve must be a ruling. To prove this, consider any two points. If the 3D major and minor texture flow vectors at these points are parallel, respectively, then the two tangent planes at these points must also be parallel[4]. Therefore, all the tangent planes along the 3D curve is the same. On a developable surface, this can be true only if the curve is a ruling, or, the surface is a plane.

Therefore, we have the following properties:

The 3D major and minor texture flow vectors along any 3D ruling on a developable document surface are constant (i.e., the flow directions are parallel, respectively).

The 3D major and minor texture flow vectors along a non-ruling curve on a non-planar developable document surface cannot both be constant (i.e., they cannot both be parallel).

As a result, 2D texture flow vectors along a 2D ruling converge to a common vanishing point (see Figure 3.7). This vanishing point may be at the infinity if the 3D flow vectors are parallel to the image plane. If a group of 2D flow vectors do not

[3]To prove the parallelism of all 3D major (minor) texture flow vectors along a ruling, imagine that we unroll the document surface onto the tangent plane at the ruling. In the flat document, the texture flow vectors are parallel. So, on the tangent plane of the curved surface, they are also parallel.

[4]Because the surface normal *must* be the cross product of major and minor texture flow vectors.

converge at a single point, they are most likely not parallel[5] and, therefore, their base points do not lie on a 2D ruling. This property can help us detect 2D rulings.

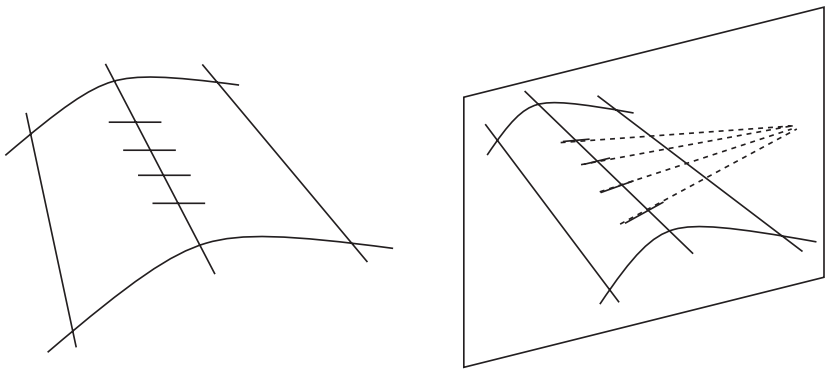


Figure 3.7: Parallel 3D texture flow vectors along a 3D ruling and convergent 2D texture flow vectors along the corresponding 2D ruling.

For any given point in the image, and any given line r through it, we take M sample points ($\{(x_i, y_i)\}_{i=1}^{M}$) along the part of the line contained in the text area. We denote the 2D major texture flow unit vector at sample points by $\{\mathbf{t}_i\}_{i=1}^{M}$, where $\mathbf{t}_i = (t_i^x, t_i^y)^\top$ and $|\mathbf{t}_i| = 1$. We do not use the minor flow at this time because it is less accurate. We use exactly the same convergence testing method based on singular value decomposition described in Section 3.2.2 to compute the convergence quality. If the convergence quality is good, we declare r as a 2D ruling.

This method succeeds in areas with a large curvature of the major texture

[5]It is possible for a group of non-parallel 3D vectors to project to a group of convergent 2D vectors. However, this requires all the *planes of sight* of these 3D vectors form a pencil of planes that share a common 3D line. At the same time, in our context, the base points of these 3D vectors *must* be on one plane of sight so the base points of their projections are co-linear. It is extremely unlikely to satisfy both simultaneously.

flow, because a small deviation from the correct 2D ruling direction causes a relatively large change in the values of $\{\mathbf{t}_i\}$, and hence affects the convergence quality. Fortunately, a small curvature of the major texture flow means a nearly flat surface, and in this case the accuracy of the 2D ruling becomes less important for the rectification result. In the extreme case where the curvature approaches zero so the surface becomes completely flat, there is no unique ruling for any given point, i.e., any line through it represents a correct ruling.

Through any point on a non-planar developable surface there is one and only one 3D ruling, so any two 3D rulings do not intersect. The only exception is the vertex of a cone. However, a conical vertex can not reside inside a document page, otherwise the paper must be creased at this point. Therefore, any two 3D rulings do not intersect within the document. Under perspective projection, the projections of two non-intersecting 3D rulings do not intersect in the image plane, either, unless one is occluded by the other. In the case of occlusion, the invisible part does not need rectification. So, we can always assume that any two 2D rulings do not intersect within the document. This property allows us to use the detection results in high curvature areas, which are more accurate, to help find rulings in other areas.

We first find a group of N *reference points*, $\{p_i\}_{i=1}^{N}$, in the image within the text area. These reference points should be such that the 2D rulings through them roughly cover the entire text area (see Appendix B). For each p_i, we compute the convergence quality measure c_{ij} for a range of possible ruling directions, denoted by $\{\phi_{ij}\}$ ($\phi_{ij} \in [0, \pi)$). We want to find a group of ruling directions with the best overall convergence quality where nearby rulings do not intersect each other inside

the text area, i.e.,

$$\{\hat{\phi}_i\}_{i=1}^{N} = \underset{\phi_i \in [0,\pi)}{\operatorname{argmin}} \left[\sum_{i=1}^{N} c_i(p_i, \varphi_i) + \sum_{i=1}^{N-1} \Psi(p_i, p_{i+1}; \phi_i, \phi_{i+1}) \right],$$

where

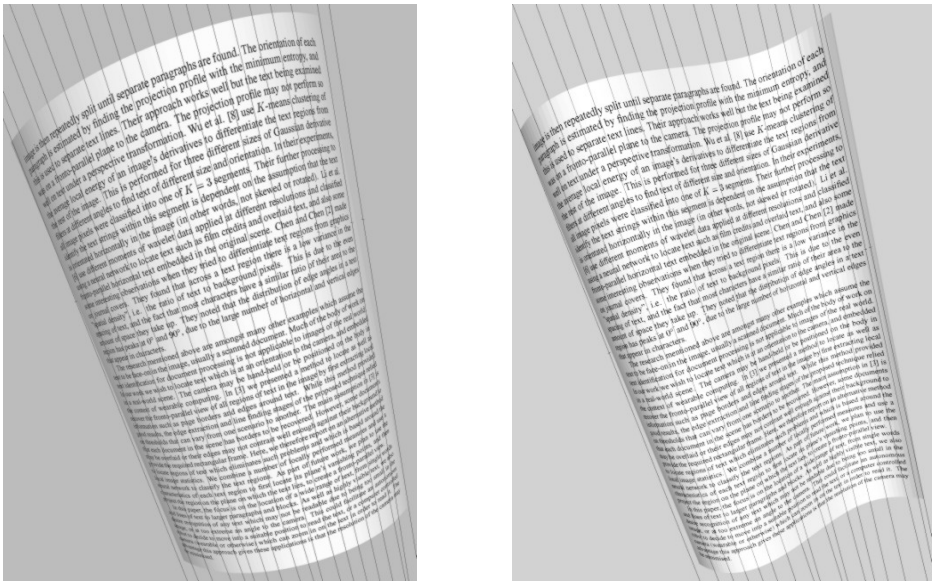
$$\Psi(p_i, p_{i+1}; \phi_i, \phi_{i+1}) = \begin{cases} \infty, & \text{if } r_i \text{ and } r_{i+1} \text{ intersect within the text area,} \\ 0, & \text{otherwise.} \end{cases}$$

prohibits two nearby rulings to intersect within the text area. The cost function decomposes into terms that depend only on each pair of $(\varphi_i, \varphi_{i+1})$, thus we can solve this minimization problem using a dynamic programming method. Figure 3.8 illustrates the result.

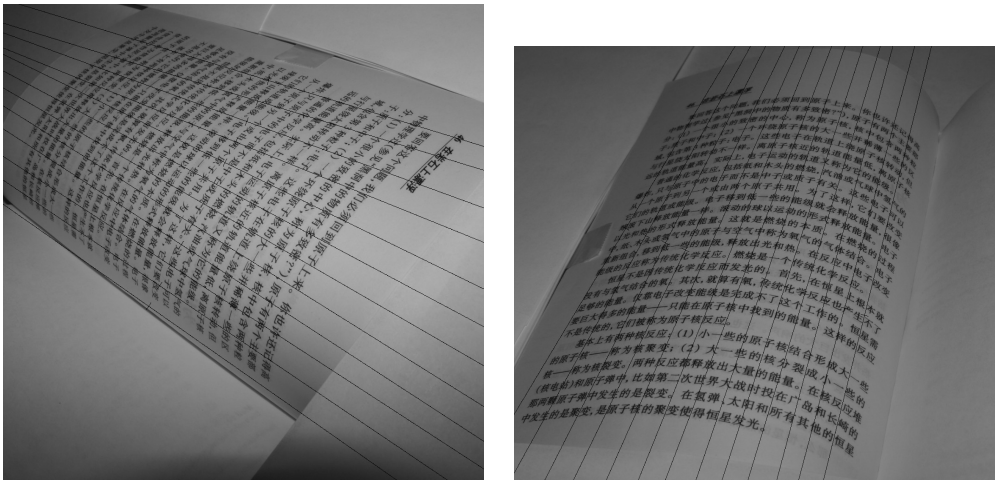
3.3.4 Vanishing points of rulings

Under perspective projection, an infinite 3D line projects to a 2D line terminating at its *vanishing point* [26]. More importantly, all parallel 3D lines have the same vanishing point in the image. So, a vanishing point is determined only by the direction of the 3D line, not its position. Inversely, the 3D line direction is determined solely by its vanishing point, because it is parallel to the ray through the camera's optical center and the vanishing point. Figure 3.9 shows three parallel 3D lines, L_1, L_2, and L_3, projecting onto the image plane I as 2D lines, l_1, l_2 and l_3, respectively. The 2D lines converge at the vanishing point V. Point O is the optical center, and $\overline{OV}$ gives the 3D direction of the 3D lines. Therefore, the vanishing points of 2D rulings are important to defining their 3D counterparts.

Our method for estimating vanishing points of rulings originates from the same



(a)



(b)

Figure 3.8: Projected ruling detection results in (a) synthetic images and (b) real images.

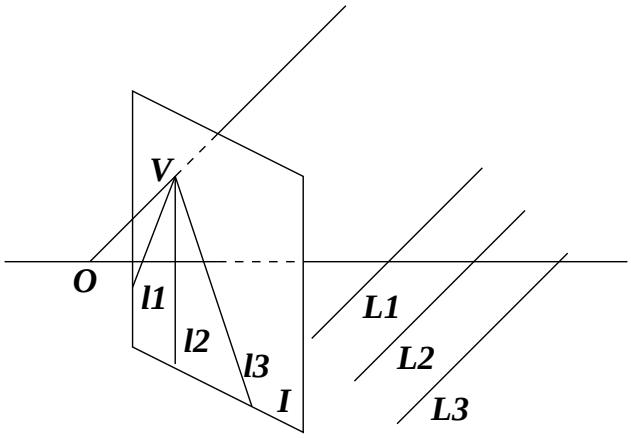


Figure 3.9: Projections of parallel 3D lines share a common vanishing point on the image plane.

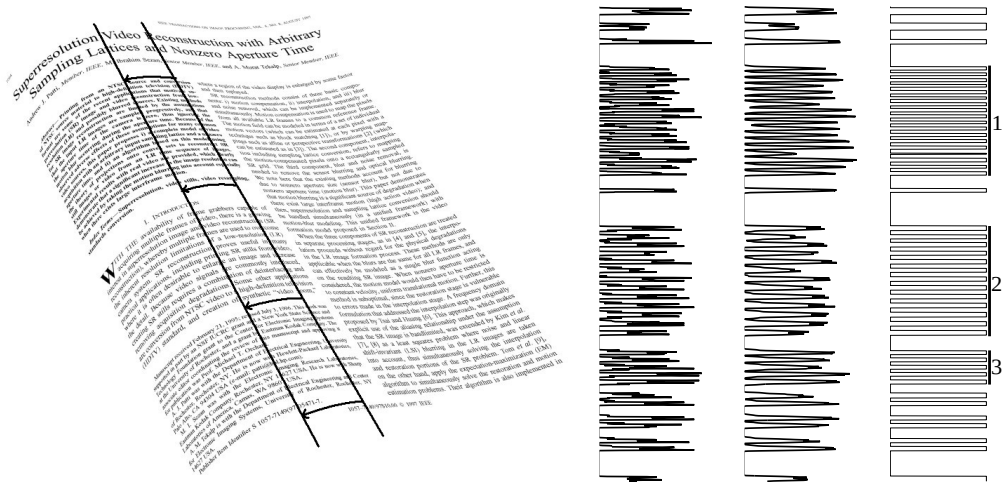
insights as [14], which are as follows: Text lines are equally spaced on the page, while in the image they are no longer equally spaced due to perspective. The page's tilt determines the degree of change in line spacing, and the tilt is related to the position of vanishing points of rulings. In [14], Clark et al. find the intersections of text lines with the justified text margin (or, equivalently, the central line of centered text blocks) using perspective projection profile analysis, then solve for the position of the vanishing point using the distances between the intersections. However, this method works only with planar pages, requires a justified text margin (or the central line), and is computationally expensive because of the search in a two-parameter space.

Our method offers three key improvements. First, text lines in our data are curved, so we propose to compute *curve-based projection profiles* in which the projec-

tion paths are curves that follow the major texture flow at every point. Second, the text line spacing is constant only within a paragraph, while the inter-paragraph spacing could be different. So, we derive a criterion to group text lines into paragraphs before we apply the constant spacing property. Third, we simplify the vanishing point estimation to a closed form solution which essentially solves a one-parameter linear system. This processes much faster than searching in a two-parameter space.

In the curve-based projection profile analysis, we select the estimated 2D ruling line as the base line (see Figure 3.10). The lengths of projection paths are fixed. The profile has peaks corresponding to text lines and valleys for white space. We first find the principal 'wave length' λ of the profile by detecting the strongest frequency response in its FFT result. We de-noise the profile by smoothing it with a kernel of size λ. After that, we apply adaptive thresholding (also with window of size λ) to obtain a binary profile where '1' represents text line and '0' white space. Without loss of generality, assume the rising edges give the 'top' positions of text lines and the falling edges give the 'bottom' positions. We create a one-dimensional coordinate system along the 2D ruling, and denote the 'top' and 'bottom' positions of text lines by $\{p_{ti}\}_{i=1}^{T}$ and $\{p_{bi}\}_{i=1}^{T}$, respectively, where T is the number of text lines. Also, a coordinate system is established on the 3D ruling, and the corresponding positions are $\{P_{ti}\}_{i=1}^{T}$ and $\{P_{bi}\}_{i=1}^{T}$ (see Figure 3.11). In the following, we use the top positions ($\{p_{ti}\}_{i=1}^{T}$, and $\{P_{ti}\}_{i=1}^{T}$) to describe our method and drop the t subscript for the sake of simplicity.

We know $\Delta = P_{i+1} - P_i$ is constant within a paragraph. Under perspective projection, $\delta_i = p_{i+1} - p_i$ is not, in general, a constant. Because of the invariant



(a) (b) (c) (d)

Figure 3.10: Curve-based projection profile (CBPP). (a) The two straight lines represent two base lines between which a curve-based projection profile is computed along the text line directions. (b) The CBPP profile. (c) Smoothed result of (b). (d) Binarized result of (c). Three paragraphs are identified.

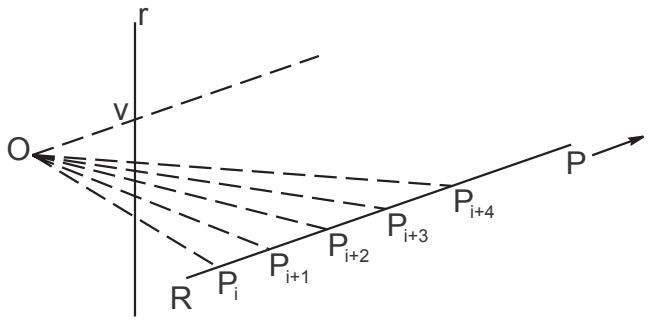


Figure 3.11: Vanishing point of a 2D ruling corresponds to the point at infinity on the 3D ruling.

cross-ratio property [26], the intervals in the 3D world and their counterparts in the image satisfy the following equality:

$$\frac{|p_{i+1}-p_i||p_{i+3}-p_{i+2}|}{|p_{i+2}-p_i||p_{i+3}-p_{i+1}|}=\frac{|P_{i+1}-P_i||P_{i+3}-P_{i+2}|}{|P_{i+2}-P_i||P_{i+3}-P_{i+1}|}=\frac{\Delta\cdot\Delta}{2\Delta\cdot 2\Delta}=\frac{1}{4},\forall i. \tag{3.7}$$

Thus, for any four consecutive text lines, if the above equality holds (within a threshold), we claim they come from the same paragraph; otherwise, they do not. The following is the pseudo-code for paragraph segmentation, where the input is the list of text line positions $\{p_i\}_{i=1}^{T}$, and the output is the set of paragraphs $\mathcal{P}$:

1. Set paragraph list $\mathcal{P}=\emptyset$; set current paragraph $P_c=\emptyset$; set the current position index $j=1$.
2. Take $\{p_i\}_{i=j}^{j+3}$ and verify it against Eq. 3.7.
3. If Eq. 3.7 holds (within a given error bound) $P_c :\leftarrow P_c\cup\{p_i\}_{i=j}^{j+3}$.
4. Otherwise, $\mathcal{P} :\leftarrow \mathcal{P}\cup\{P_c\}$, and reset $P_c=\emptyset$.
5. $j :\leftarrow j+1$.
6. Stop if $j=T-2$; otherwise go to step 2.

Careful readers may notice that this process implicitly requires at least four text lines in a paragraph. This usually works fine with most documents; for documents with short paragraphs, such as yellow pages, the verification error bound can be relaxed.

If we let P_{i+3} converge toward ∞, then p_{i+3} converge toward v, which is the coordinate of the vanishing point along r (see Figure 3.11). Eq. 3.7 becomes

$$\frac{|p_{i+1}-p_i||v-p_{i+2}|}{|p_{i+2}-p_i||v-p_{i+1}|} = \lim_{P_{i+3}\to\infty} \frac{|P_{i+1}-P_i||P_{i+3}-P_{i+2}|}{|P_{i+2}-P_i||P_{i+3}-P_{i+1}|} = \frac{1}{2}, \forall i,$$

which is a linear equation in v. Given the paragraphs $\mathcal{P}$, we can solve for the optimal position v in a Least Square sense. Because v is only a scalar variable, the linear equations are simply in the form of $\mathbf{X}v = \mathbf{Y}$, where $\mathbf{X}$ and $\mathbf{Y}$ are two column vectors, so $v = (\mathbf{X}^\top\mathbf{Y})(\mathbf{X}^\top\mathbf{X})^{-1}$.

Similarly, the discussion above applies to $(\{p_{bi}\}_{i=1}^T, \{P_{bi}\}_{i=1}^T)$, therefore the two sets of paragraphs found using the text line 'top' or 'bottom' positions can be used together in computing v.

3.3.5 Page shape estimation

The knowledge of projected texture flows, rulings and their vanishing points reveal some information about the local surface orientation. This local information is noisy. Furthermore, we do not know the camera parameters yet. In the following, we describe our approach for estimating the developable surface normals plus the camera focal length that optimally satisfy the texture flow and ruling estimates.

The orientations of the planar strips that approximate the document shape can be described by a group of unit 3D surface normals denoted as $\{\mathbf{N}_i\}_{i=1}^L$, where L is the number of strips. We have $L = N - 1$, where N is the number of detected rulings. In addition, we need the 3D position of one reference point on each strip to describe its plane fully. We backproject 2D ruling reference points onto the

document page as the 3D reference points, denoted as $\{\mathbf{P}_i\}_{i=1}^{L}$.

It is impossible to recover absolute depth from a single image unless we have a priori metric knowledge of the page surface. We can only recover $\{\mathbf{P}_i\}$ up to a scale factor. Moreover, $\{\mathbf{P}_i\}$ is not independent of $\{\mathbf{N}_i\}$. If a set of surface normals for any 3D surface is known, then, in theory, we can use integration to find the 3D position of any point on the surface given an initial point. So, if $\{\mathbf{N}_i\}$ is known, we can give $\mathbf{P}_1$ an arbitrary depth (which corresponds to the varying scale factor) and use discrete integration to get the other $\{\mathbf{P}_i\}$. It follows that the real unknowns are f and $\{\mathbf{N}_i\}$.

Before we describe our method to estimate f and $\{\mathbf{N}_i\}$, let us first define the variables (see Figure 3.12):

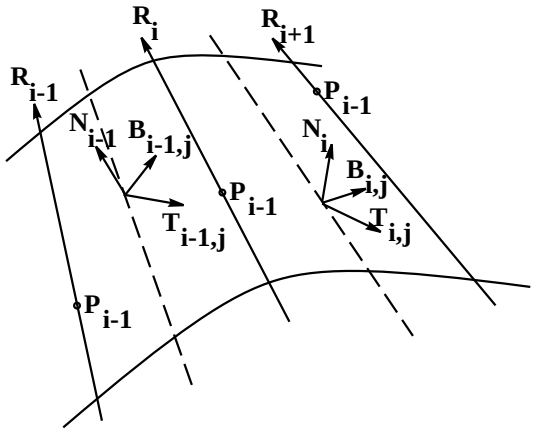


Figure 3.12: Definitions of variables used in page shape estimation.

- Wanted unknowns:
  - 3D normals: $\{\mathbf{N}_i\}_{i=1}^{L}$

- 3D reference points: $\{P_i\}_{i=1}^{L+1}$
- Focal length: f

- Available data:
  - Projected rulings in the image: $\{\mathbf{r}_i\}_{i=1}^{L+1}$
  - Projected reference points in the image: $\{p_i\}_{i=1}^{L+1}$
  - Projected texture flow in the image: $\mathbf{t}$ and $\mathbf{b}$ at every point

- Other related variables:
  - 3D rulings: $\{\mathbf{R}_i\}_{i=1}^{L+1}$
  - 3D texture flow: For the i-th strip, we select a group of J_i sample points inside the strip, and define $\mathbf{T}_{ij}$ as the 3D major texture flow vector at the j-th point, and $\mathbf{B}_{ij}$ as the minor texture flow vector.
  - 3D line-of-sight vector: For the j-th sample point in the i-th strip, we define $\mathbf{V}_{ij}$ as its line-of-sight vector which originates from optical center $\mathbf{O}$ toward the sample point.

All the 2D and 3D vectors are of unit length.

The 3D vectors are orthogonal to the surface normal at the sample points. At the same time, they are coplanar with the 3D line-of-sight vectors and their 2D counterparts. Therefore

$$\begin{aligned} \mathbf{R}_i &= \eta((\mathbf{r}_i \times \mathbf{V}_i) \times (\mathbf{N}_i + \mathbf{N}_{i-1})), \\ \mathbf{T}_{ij} &= \eta((\mathbf{t}_{ij} \times \mathbf{V}_{ij}) \times \mathbf{N}_i), \\ \mathbf{B}_{ij} &= \eta((\mathbf{b}_{ij} \times \mathbf{V}_{ij}) \times \mathbf{N}_i). \end{aligned} \tag{3.8}$$

where $\eta(\cdot)$ represents the normalization operator, i.e., $\eta(\mathbf{v}) = \mathbf{v}/|\mathbf{v}|$. Note that we use $\mathbf{N}_i + \mathbf{N}_{i+1}$ to approximate the surface normal at any point along $\mathbf{R}_i$.

There are four constraints that we can derive from the developable property of the page and the property of printed text:

- Orthogonality between surface normals and rulings: Ideally, we would want $\mathbf{N}_{i-1}^\top \mathbf{R}_i = \mathbf{N}_i^\top \mathbf{R}_i = 0$. Since we have fixed $\mathbf{R}_i$ to be orthogonal to $\mathbf{N}_{i-1} + \mathbf{N}_i$, we only need to check $\mathbf{R}_i^\top (\mathbf{N}_i - \mathbf{N}_{i-1})$. We define $\mu_1 = \sum_{i=1}^{L-1} (\Delta \mathbf{N}_i^\top \mathbf{R}_i)^2$ where $\Delta \mathbf{N}_i = \mathbf{N}_i - \mathbf{N}_{i-1}$, and ideally $\mu_1 = 0$.

- Parallelism of text lines inside each strip: Text line directions are represented by $\mathbf{T}_{ij}$. We use $\mu_2 = \sum_i \sum_j |\mathbf{T}_{ij} - \overline{\mathbf{T}}_i|^2$ to measure their parallelism, where $\overline{\mathbf{T}}_i$ is the average of all $\mathbf{T}_{ij}$ within the i-th strip. Ideally $\mu_2 = 0$.

- Geodesic property of text lines crossing two neighboring strips: The text lines on two neighboring strips form two different angles with the 3D ruling that separates the strips. After unwarping, the angles do not change. If the text line is straight in the unwarped image, the sum of the two angles must be π. We use $\mu_3 = \sum_i ((\overline{\mathbf{T}}_{i+1} - \overline{\mathbf{T}}_i)^\top \mathbf{R}_i)^2$ to measure this straightness, and ideally $\mu_3 = 0$.

- Orthogonality between text line direction and vertical stroke direction: The

orthogonality can be measured by $\mu_4 = \sum_i \sum_j |T_{ij}^\top B_{ij}|^2$, which in the idea case should be zero.

In our experiments, we embedded two additional constraints:

- Smoothness: We use $\mu_5 = \sum_i |\Delta \mathbf{N}_i|^2$ to measure the surface smoothness. A large value indicates abrupt changes in normals of neighboring strips and, therefore, should be penalized.

- Unit length: Each normal should be of unit length. We measure this by $\mu_6 = \sum_i (1 - |\mathbf{N}_i|)^2$.

The overall optimization objective function is the weighted sum of all constraint measurements,

$$F(\mathbf{X}) = \sum_{i=1}^{6} \alpha_i \mu_i$$

where $\mathbf{X}$ represents all normals and the focal length f, and α_i are weights. Notice that f affects the line-of-sight vectors, which contributes to the objective function through Eq. 3.8.

Overall, given $\{\mathbf{r}_i\}$, $\{\mathbf{t}_{ij}\}$ and $\{\mathbf{b}_{ij}\}$, the objective function is fully determined by the unknown $\{\mathbf{N}_i\}$ and f. The optimal set of $\{\mathbf{N}_i^*\}$ and f^* should minimize F.

A good initial value of $\mathbf{X}$ is essential for optimizing this highly non-linear objective function. Such initial values can be obtained using the estimated vanishing points of rulings. These vanishing points, when given the focal length, determine the direction of 3D rulings. Since surface normals are orthogonal to 3D rulings, this eliminates one degree of freedom from the unknown normals. The remaining

degree of freedom allows a normal to rotate inside the plane orthogonal to the ruling. So, a set of rotation angles determines the objective function. Furthermore, the computation of the objective function involves either each individual normal (in μ_2, μ_4, μ_6), or two neighboring normals (in μ_1, μ_3, μ_5). Therefore, we can use a dynamic programming search to find the set of rotation angles that gives the minimum objective function output.

The focal length is not covered by the dynamic programming search, however, as it is independent of the surface normals. We have to perform an exhaustive search for the initial focal length. More specifically, we select a set of possible focal lengths constrained by the physical lens specification and, for each value, we find the 'best' surface normals and compute the objective function. We fit a third-order polynomial curve to the objective function values vs. the focal lengths and find the best initial focal length f^0 at the minimum of the curve. Then, we compute the best initial normals $\{\mathbf{N}^0\}$, using f^0.

Our non-linear optimization module derives from a subspace trust region method based on the interior-reflective Newton method described in [16, 15]. Each iteration involves the approximate solution of a large linear system using the method of preconditioned conjugate gradients.

After we have estimated the surface normals and focal length, we can select an arbitrary depth for one of the 3D reference point (which determines the depths of all other reference points) and fully determine the depths of all planar strips. In the rectification process, these planar strips are rectified piece by piece using the method we develop for planar document pages. We can arbitrarily scale the final image by

modifying the strips' rectification matrices, so the depth value is not critical.

3.3.6 Frontal-flat view restoration

We use Eq. 3.4 in Section 3.2.4 to remove the perspective distortion of each planar strip that approximates the curved document page. Let us repeat Eq. 3.4 here:

$$\mathbf{H} = \mathbf{K} \begin{pmatrix} \overline{\mathbf{V}}_h & \overline{\mathbf{V}}_v & \mathbf{W} \end{pmatrix} \begin{pmatrix} 1 & 0 & -x_0 \\ 0 & 1 & -y_0 \\ 0 & 0 & 1 \end{pmatrix}.$$

The matrix $\mathbf{K}$ is determined by focal length f. For the i-th strip, we substitute $\overline{\mathbf{T}}_i$ for $\overline{\mathbf{V}}_h$, and $\overline{\mathbf{B}}_i$ for $\overline{\mathbf{V}}_v$. We need to compute $\mathbf{W}$, the 3D reference point position on the strip, and (x_0, y_0), the position of the reference point in the final image. We denote them as $\mathbf{P}_i$ and (x_i, y_i).

Suppose $\mathbf{H}_{i-1}^{-1}$ is the rectification matrix for $(i-1)$-th strip. We map the 2D reference point $p_i = (x_i', y_i')$ to the destination image using $\mathbf{H}_{i-1}^{-1}$, and that position is (x_i, y_i).

We know $\mathbf{P}_i$ and $\mathbf{P}_{i-1}$ are both on the $(i-1)$-th strip, so $(\mathbf{P}_i - \mathbf{P}_{i-1})$ is perpendicular to the normal of $(i-1)$-th strip, i.e., $\mathbf{N}_{i-1}^\top(\mathbf{P}_i - \mathbf{P}_{i-1}) = 0$. Let $\mathbf{P}_i = (X, Y, Z)^\top$, then Eq. 3.3 produces

$$\begin{cases} fX - x_i'Z &= 0 \\ fY - y_i'Z &= 0 \end{cases},$$

where (x_i', y_i') is the image of $\mathbf{P}_i$ in the image plane. After some manipulation we obtain

$$\begin{pmatrix} & \mathbf{N}_{i-1}^{\top} & \\ f & 0 & -x_i' \\ 0 & f & -y_i' \end{pmatrix} \mathbf{P}_i = \begin{pmatrix} \mathbf{N}_{i-1}^{\top}\mathbf{P}_{i-1} \\ 0 \\ 0 \end{pmatrix}.$$

which computes $\mathbf{P}_i$ using the information of $(i-1)$-th strip.

In summary, we start by setting $\mathbf{P}_1 = (x_1', y_1', f)^{\top}$ and $(x_1, y_1) = (0, 0)$, then compute $\mathbf{H}_1$. We use $\mathbf{P}_1$ and $\mathbf{H}_1$ to compute $\mathbf{P}_2$ and $\mathbf{H}_2$, so on and so forth. Eventually, all planar strips are rectified.

However, the perspective-free planar strips will not fit perfectly to form the flat document we want. First, if the surface happens to be formed by a family of tangents to a twisted 3D curve, planar strip approximation is not seamless (see Section 3.3.2). Second, and more likely, the noise and error in the data cause conflicts among the constraints that govern the shape optimization process, so the planar strips are not seamless. Whatever reason, the seams between neighboring strips cause discontinuity along their borders in the rectified image. We need to merge those strips to produce a seamless document.

Let us first define a few more variables (see Figure 3.13):

- $\{\mathbf{r}_i\}_{i=1}^{L+1}$: 2D rulings.
- $\{s_i\}_{i=1}^{L}$: strips in the original image separated by 2D rulings.
- $\{\mathbf{H}_i^{-1}\}_{i=1}^{L}$: homography rectification matrices for planar strips.
- $\{s_i'\}_{i=1}^{L}$: rectified strips in the destination image.

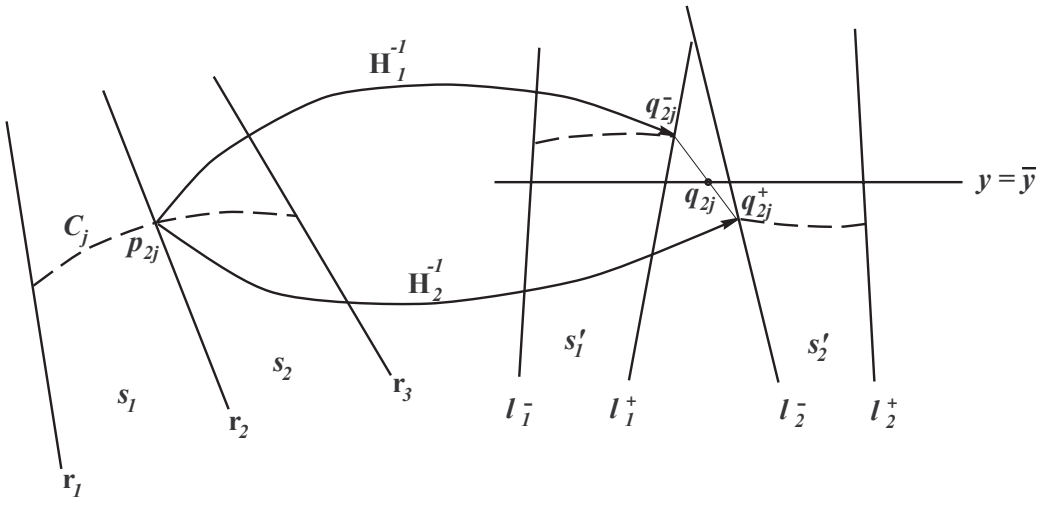


Figure 3.13: Computing the target grid points for seamless morphing.

- $\{\mathcal{C}_j\}_{j=1}^{J}$: a group of curves following major texture flow in the original image.
- $\{\mathcal{C}'_j\}_{j=1}^{J}$: $\mathcal{C}'_j = \cup_{i=1}^{L}\mathcal{C}'_{ij}$, rectification of $\{\mathcal{C}_j\}_{j=1}^{J}$ in the destination image, where $\mathcal{C}'_{ij} = \mathbf{H}_i(\mathcal{C}_j \cap s_i)$ is the section in s'_i.
- p_{ij}: intersection points of $\mathcal{C}_j$ and $\mathbf{r}_i$.
- q_{ij}^{+} and q_{ij}^{-}: $p_{ij} \xrightarrow{\mathbf{H}_{i-1}^{-1}} q_{ij}^{-}$, $p_{ij} \xrightarrow{\mathbf{H}_{i}^{-1}} q_{ij}^{+}$.
- y_{ij}^{-} and y_{ij}^{+}: y-coordinates of q_{ij}^{-} and q_{ij}^{+}.
- $\bar{y}_j = (\sum_i (y_{ij}^{-} + y_{ij}^{+}))/(2L)$: mean value of $\{q_{ij}^{-}\}_{i=1}^{L} \cup \{q_{ij}^{+}\}_{i=1}^{L}$.

Ideally, $\mathcal{C}_j$ should be mapped to a straight horizontal line in the destination image, which means q_{ij}^{-} and q_{ij}^{+} coincide, and $y_{ij}^{\pm}$ is constant, $\forall i, j$. However, due to imperfections in shape estimation, $\mathcal{C}'_j$ usually is not straight, nor horizontal, and is broken. We correct this problem by dividing each strip into small pieces, and for

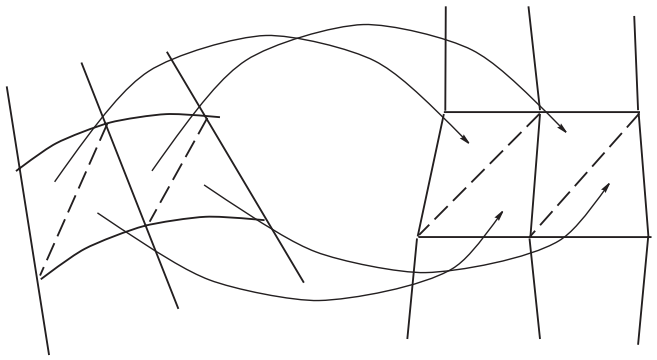
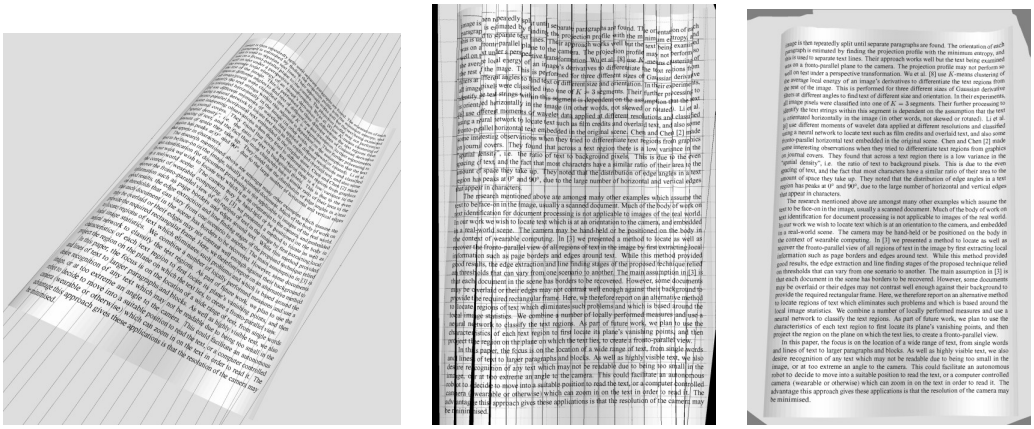


Figure 3.14: Computing the affine transform for triangular patches.

each piece we compute a rectification transformation. This step amounts to local morphing on top of the planar strip rectification. The control points in the original image are q_{ij}, which lie on $\mathcal{C}_j$. Their corresponding points ($\{q_{ij}\}$) in the destination image should satisfy the continuity, straightness and horizontal properties of $\mathcal{C}'_j$. Therefore, we set q_{ij} to the intersections of line $y = \bar{y}_j$ and line $\overline{q^-_{ij}q^+_{ij}}$.

In the original image, we construct two triangles based on four neighboring points, i.e., $\triangle(q_{ij}q_{i+1j}q_{ij+1})$ and $\triangle(q_{i+1j+1}q_{i+1j}q_{ij+1})$. The four points have $\bar{q}_{ij}$, $\bar{q}_{i+1j}$, $\bar{q}_{ij+1}$, $\bar{q}_{i+1j+1}$ as their target positions. For each triangle, we compute an affine transformation to map its vertices to the targets (see Figure 3.14). After the triangle based adjustment, we obtain a seamless document (see Figure 3.15).

Figures 3.16 and 3.17 compare camera-captured images of curved documents (both synthetic and real) and their rectified counterparts. Overall, the rectified images are close to the frontal-flat view of the documents, despite some imperfection near text boundaries.



(a) (b) (c)

Figure 3.15: Post-processing flattened strips to obtain seamless document image. (a) 2D rulings found for a document. (b) Piecewise rectification result. Note the gap and discontinuous text lines. (c) After post-processing, the document image is seamless.

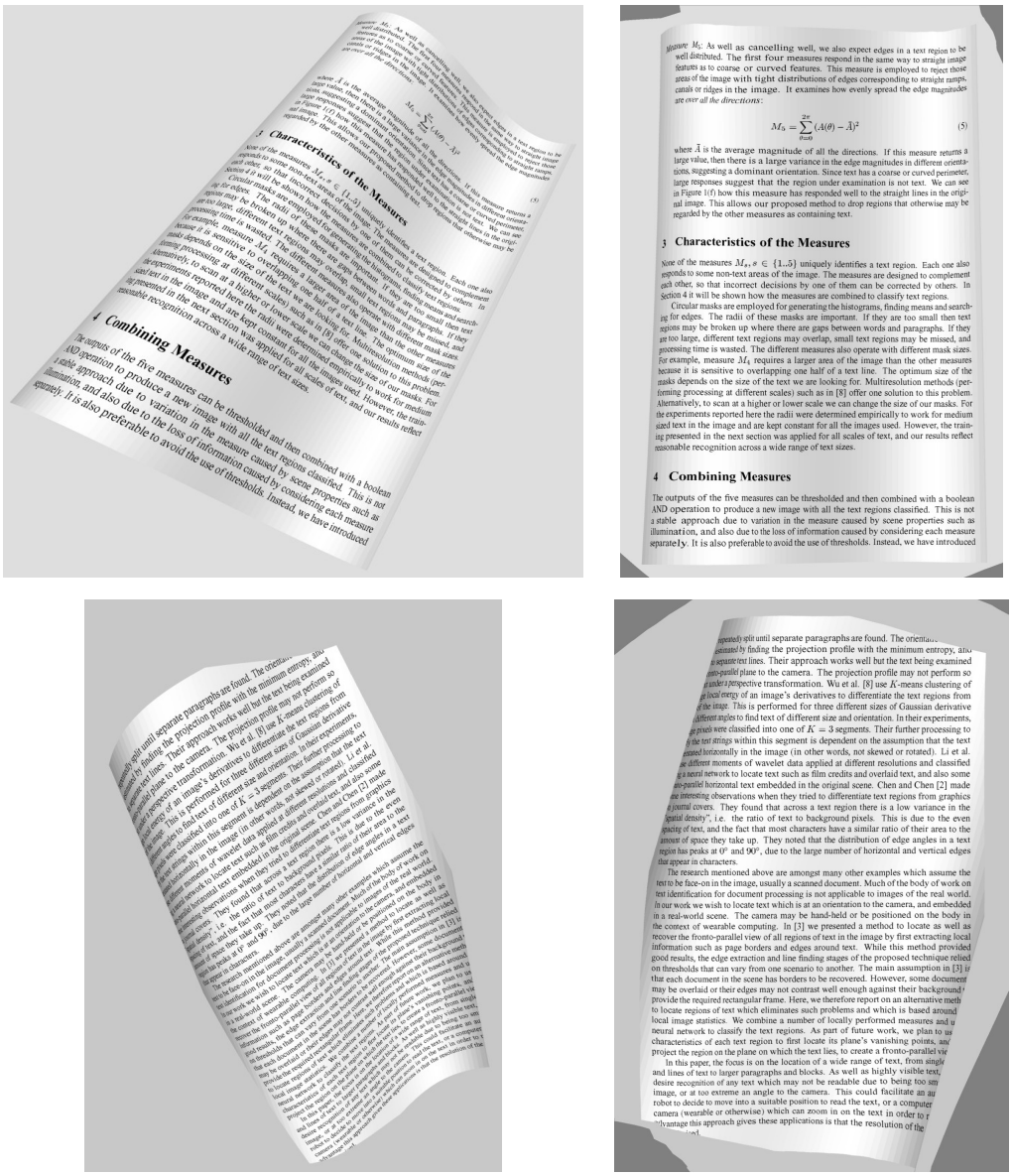


Figure 3.16: Comparison of synthetic images of curved documents and rectification results.

Figure 3.17: Comparison of images of real curved documents and rectification results.

3.4 Evaluation

3.4.1 Evaluation methodology

We evaluate the performance of image rectification using synthetic images. We benchmark the quality of the rectified image by OCR rates, i.e., we compare the OCR rates obtained from the original image with the rectified one. Ultimately, our algorithms are designed for document analysis, thus OCR rates can be viewed as the most important performance index. We compute both *character recognition score* (CRS) and *word recognition rate* (WRR) using the OCR ground-truthing tool described in [80]. Both CRS and WRR are percentage numbers, where 100% means perfect recognition, and 0% means complete failure. CRS is computed using the *shortest editor's distance* measure, which essentially finds the best approximate match between two text strings. WRR provides the actual percentage of words with all characters correctly recognized.

With synthetic images, we can also compare the estimated 2D and 3D rulings, focal length, and surface normals to the ground truth data used or generated by the synthesis module (see Appendix A). Section 2.4 describes our synthetic data.

Given 2D rulings and surface normals as directional values, we measure their precisions by the average direction error which is an angle. Such measurements are independent of image scales. However, the coordinates of vanishing points in the image plane and focal length are both metric values dependent on the image scale. If we compare the estimates with ground truth directly, the difference will be scale dependent, too. A good benchmark should be independent of the image scale.

First, we notice that the rays from the optical center to the vanishing points of rulings are parallel to the 3D rulings, so the precision of vanishing points can be equivalently measured by the corresponding 3D ruling direction. To position the optical center with respect to the image plane, we assume perfect knowledge of the focal length. The 3D ruling direction error does not solely originate from the CBPP-based estimation of vanishing points along 2D rulings; it also derives partially from the error in 2D ruling estimation.

Second, we benchmark the focal length estimation in a similar way. We take a reference point in the image and compare two rays from this point to the optical centers given by the correct focal length and the estimated value, respectively, which provides an error angle. This measurement is scale independent unless the reference point is at the principal point (under our assumption, the image center), in which case the error angle is always zero. In our test, we choose one image corner — all corner produces equivalent result if the principal point coincides the image center — so the angle between the ray and the optical axis has the physical interpretation of being half of the *field of view*. By this interpretation, the error in field of view measures the focal length accuracy.

Having argued for scale independent benchmarks, we also acknowledge image size and resolution have an important impact on the performance of our algorithms. In particular, high resolution benefits the minor texture flow estimation, which, in turn, has an impact on shape estimation, which a completely scale independent benchmark should take into account. One solution is to control the precision of the texture flow — by adding noise to ground truth, for example.

In our tests, the sizes and resolutions of synthetic images do not vary excessively. We did not observe large variances in texture flow precision, either. Therefore, we settle with the semi scale-independent benchmarks, as described above.

3.4.2 Performance of shape estimation

All 60 synthetic images of planar documents are correctly classified as 'planar' by our system. Table 3.1 summarizes the evaluation results in the same format as Table 2.1, i.e., the first row shows the overall averaged performance and standard variations, while the other rows illustrate results of controlled groups. We have three major groups exploring the effects of document content, page pose, and skew angles, each having several sub-groups. All numbers are in degrees. In average, the field of view error is 3.30 degrees, while the plane-normal error is 2.40 degrees. Considering we do not require any camera calibration as input, such shape estimation results exceeds expectation.

In general, we do not notice significant variation in the precisions of field of view and plane normal among different sub-groups for each major group, except for the sub-group `Pose no. 1`, `Pose no. 2` and `-15° skew`. Examination of the test images reveals the combination of these poses and skew results in images where text lines are (*almost*) parallel (for example, see the right-most image in Figure 2.5(c)), one of the configurations where the focal length cannot be (accurately) estimated.

All the 120 synthetic images of curved documents are correctly classified as 'curved' by our system. Table 3.2 and Table 3.3 summarize the performance of

(mean/std)×1°	Field-of-view	Plane normal
All	3.30/3.63	2.40/3.01
Page 1	2.93/2.75	2.28/1.98
Page 2	2.00/1.25	1.34/0.93
Page 3	2.88/2.43	1.80/1.54
Page 4	4.81/5.84	3.66/5.58
Page 5	3.86/4.07	2.92/2.58
Pose no.1	5.06/3.07	3.69/2.30
Pose no.2	4.39/5.70	3.14/5.00
Pose no.3	2.01/0.88	1.39/0.68
Pose no.4	1.73/1.97	1.38/1.70
0° skew	2.22/1.71	1.61/1.36
15° skew	2.16/1.55	1.57/0.86
−15° skew	5.51/5.28	4.01/4.63

Table 3.1: Evaluation of shape estimation for synthetic planar document images.

shape estimation, including 2D and 3D rulings, field of view and surface normals.

The data in Table 3.2 show good performance in 2D and 3D ruling direction estimation and no significant difference within the major groups, except among three skew angles. As the absolute skew angle increases, we could expect a drop in 2D ruling estimation precision. Consider the extreme case where skew angle is 90° so the major texture flow becomes parallel to the rulings (i.e., cylinder generatrix, in our context). In this case, any 2D major texture flow vectors in the document converge, so we cannot identify correct 2D rulings using the major texture flow. Instead, we should use the minor texture flow.

Ideally, the skew angle should be first estimated — locally we can define the skew angle as the angle between 3D major texture flow and 3D rulings — then, if it is small, we should use the major texture flow to estimate 2D rulings. Otherwise, we use the minor texture flow. However, this creates a dead lock since we could not find 3D rulings unless we have 2D rulings. In practice, usually the skew angle is not excessive, and because the major texture flow shows higher precision in estimation (shown in Table 2.1 and 2.2), we choose to use the major texture flow in all cases.

We notice 3D ruling errors are consistently greater than 2D ruling errors. Because 2D angles are the projection of 3D angles onto the image plane, thus they are always smaller than their 3D counterparts.

In Table 3.3, we note good overall performance in terms of field of view and surface normals, which are 3.08 degrees and 2.44 degrees, respectively, in average after the optimization process. The surface normals receive a larger improvement compared to the initial estimation than field of view estimations, because the opti-

(*mean/std*)×1°	2D ruling	3D ruling
All	1.82/1.26	2.91/1.88
Page 1	2.35/1.46	3.22/1.82
Page 2	1.58/1.06	2.17/1.31
Page 3	1.44/0.76	2.80/1.15
Page 4	1.92/1.48	3.33/3.03
Page 5	1.76/1.27	2.99/1.28
Pose no.1	2.11/1.36	2.95/1.71
Pose no.2	0.91/0.60	1.72/1.03
Pose no.3	2.20/1.27	3.71/2.48
Pose no.4	2.09/1.27	3.22/1.41
0° skew	0.90/0.47	1.80/0.87
15° skew	2.14/1.07	3.49/2.22
−15° skew	2.42/1.50	3.41/1.76

Table 3.2: Evaluation of 2D and 3D ruling estimation for synthetic images of curved document.

mization constraints can effectively decrease the excessive errors (if any) in planar strip normals using the good neighboring strip normals. The focal length is optimized in a global sense, so the optimization effect is not as significant.

(mean/std)×1°	Initial estimation		Optimized estimation	
	Field-of-view	Surface normal	Field-of-view	Surface normal
All	3.21/3.42	3.90/2.68	3.08/3.41	2.44/2.72
Page 1	3.18/3.45	3.97/1.93	3.02/3.52	2.21/1.04
Page 2	3.73/3.69	3.22/1.52	3.40/3.68	2.06/0.85
Page 3	2.52/2.82	3.47/1.20	2.57/2.80	2.18/0.72
Page 4	3.18/3.37	4.02/2.48	3.33/3.35	2.50/1.19
Page 5	3.43/3.88	4.82/4.82	3.09/3.89	3.29/5.98
Pose no.1	5.72/4.32	3.73/1.43	5.65/4.42	2.45/1.17
Pose no.2	3.08/2.43	3.68/1.91	2.93/2.48	2.10/0.73
Pose no.3	2.39/2.50	4.39/2.41	2.02/2.07	1.83/0.78
Pose no.4	2.22/3.45	3.74/4.03	2.32/3.54	3.36/4.97
0° skew	2.17/2.15	2.63/1.76	2.18/2.14	1.81/0.54
15° skew	3.29/4.33	5.09/3.44	3.15/4.35	2.91/4.40
−15° skew	4.20/3.09	3.87/1.78	3.96/3.13	2.56/0.94

Table 3.3: Evaluation of shape estimation for synthetic images of curved documents.

To investigate the relationship between shape properties and the estimation accuracy, we design two experiments. In the first, we fix the pose, and use seven

shapes with similar appearance but different curvature (see Figure 3.18). Each shape is combined with the five document pages, resulting in 35 total images. Table 3.4 summarizes the accuracies of texture flow and ruling estimation, while Table 3.5 shows shape estimation performance. In the second experiment, we fix the shape and use seven different poses (see Figure 3.19). The results are summarized in Tables 3.6 and 3.7.

Not surprisingly, both the increases in surface curvature and tilt angle increase the difficulty of estimating texture flow, ruling, field of view, and surface normal. Especially with the last two, most curved, and most tilted documents, almost all benchmarks drop sharply. In practice, however, it is rare to have images containing documents as curved or tilted as these. The results of our algorithms are reasonable for practical purposes. We also observe that the precision of surface normal is more closely related, than the precision of field of view, to the error in texture flow and rulings. Field of view is a global parameter, thus its dependency on local estimates of texture flow and rulings are not as direct as surface normals.

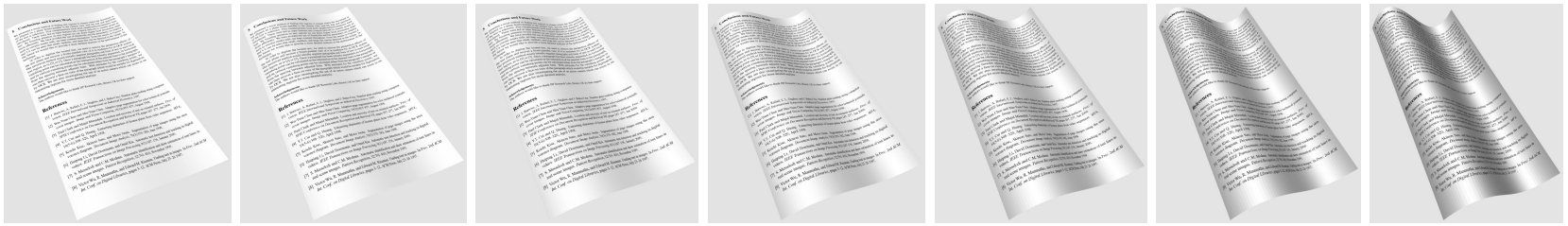


Figure 3.18: Seven shapes with increasing curvature.

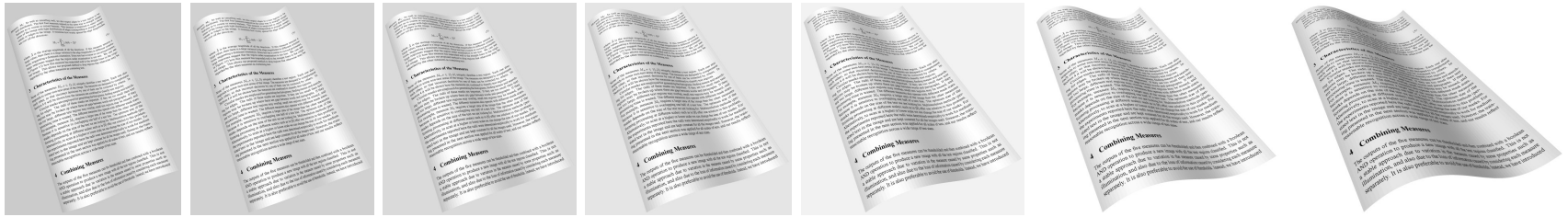


Figure 3.19: Seven poses with increasing tilt angle.

(mean/std)×1°	Major texture flow	Minor texture flow	2D ruling	3D ruling
Shape no.1	0.41/0.02	1.09/0.20	2.06/0.80	2.28/0.72
Shape no.2	0.52/0.03	1.10/0.16	2.15/1.10	2.48/0.79
Shape no.3	0.62/0.04	1.02/0.12	2.09/0.70	2.46/0.60
Shape no.4	0.72/0.03	1.02/0.13	1.83/0.93	2.06/0.72
Shape no.5	0.89/0.12	1.27/0.78	1.79/0.34	2.22/0.67
Shape no.6	1.47/0.74	1.15/0.38	3.59/1.33	4.10/1.20
Shape no.7	1.72/0.28	1.26/0.51	5.04/1.87	10.98/4.82

Table 3.4: Effects of varying surface curvature on texture flow and ruling estimation.

3.4.3 OCR performance on rectified images

Qualitatively, we observe visually satisfactory results in rectified images in Figures 3.3, 3.4, 3.5, 3.16, and 3.17. Quantitatively, we measure image quality by the OCR results. That is, we compare the OCR rates on original images to those obtained from rectified images. We extract the text from the PDF files as ground truth.

Table 3.8 summarizes the OCR evaluation results on synthetic planar document images and rectification output. Overall, they demonstrate significant improvements in both character recognition score (CRS) and word recognition rate (WRR). The average CRS and WRR (97.08% and 95.91%) of rectified images come close to the state-of-the-art OCR rate on high quality scans (which is usually above

(*mean/std*)×1°	Initial estimation		Optimized estimation	
	Field-of-view	Surface normal	Field-of-view	Surface normal
Shape no.1	1.09/1.18	1.49/0.21	0.98/1.25	1.06/0.14
Shape no.2	1.53/0.64	1.76/0.37	1.40/0.65	1.34/0.31
Shape no.3	1.71/0.76	1.98/0.58	1.33/0.92	1.57/0.22
Shape no.4	1.47/0.64	2.25/0.56	1.23/0.76	1.43/0.20
Shape no.5	1.02/0.40	3.33/0.97	0.61/0.20	1.86/0.62
Shape no.6	1.40/0.60	4.84/2.90	1.43/0.55	2.52/1.33
Shape no.7	1.39/1.66	6.45/4.49	1.10/1.17	4.65/2.61

Table 3.5: Effects of varying surface curvature on shape estimation.

(*mean/std*)×1°	Major texture flow	Minor texture flow	2D ruling	3D ruling
Pose no.1	0.62/0.04	0.61/0.07	1.48/0.59	3.74/4.39
Pose no.2	0.69/0.04	1.01/0.69	1.64/0.77	2.20/0.78
Pose no.3	0.70/0.03	0.90/0.13	1.91/0.54	2.26/0.58
Pose no.4	0.68/0.03	1.05/0.07	2.63/1.33	2.74/0.99
Pose no.5	0.75/0.05	1.16/0.09	2.73/1.24	2.66/0.89
Pose no.6	1.19/0.86	2.17/1.97	2.77/1.24	3.28/2.11
Pose no.7	4.05/3.25	2.00/0.80	4.79/2.03	4.32/2.06

Table 3.6: Effects of varying tilt angles on texture flow and ruling estimation.

(mean/std)×1°	Initial estimation		Optimized estimation	
	Field-of-view	Surface normal	Field-of-view	Surface normal
Pose no.1	0.74/0.82	2.28/0.45	0.67/0.79	1.55/0.18
Pose no.2	1.48/0.72	3.28/1.47	1.62/1.22	2.00/0.65
Pose no.3	1.13/0.75	2.37/0.77	0.87/0.77	1.66/0.26
Pose no.4	1.14/0.92	2.09/0.56	0.92/0.68	1.54/0.10
Pose no.5	1.70/0.49	2.71/1.16	1.73/0.57	1.56/0.38
Pose no.6	4.62/4.88	4.47/2.26	4.15/5.00	3.09/2.85
Pose no.7	7.92/3.91	6.09/2.44	7.66/4.07	3.78/2.53

Table 3.7: Effects of varying tilt angles on shape estimation.

99%). The small gap from 99% can be explained partially by the fact two of the five pages contain mathematical formulas, and all of them possess mathematical symbols in the text, which poses difficulties for OCR engines.

Table 3.9 shows the same level of improvement for curved pages. Compared to Table 3.8, the additional curvature in shape costs the average CRS and WRR scores approximately 10%. We observe most OCR errors occur near text area boundaries where the texture flow estimation is prone to errors due to the lower percentage of text in the neighborhood. For planar pages, this poses less a problem because the shape computation has only one global surface normal to estimate, thus some local errors are acceptable. For curved documents, such errors carry into the estimated local orientation, thus appearing in the rectified image, and affecting OCR.

(*mean/std*)×1%	Original image		Rectified image	
	CRS	WRR	CRS	WRR
All	26.14/36.14	22.92/34.24	97.08/3.55	95.91/3.80
Page 1	29.14/35.32	23.99/34.21	96.28/6.14	96.70/3.07
Page 2	24.32/39.06	22.41/37.31	98.63/1.22	98.24/1.81
Page 3	24.56/38.85	23.25/37.11	96.31/3.09	94.50/3.09
Page 4	28.57/36.34	24.18/34.74	96.77/2.41	95.19/3.19
Page 5	24.11/37.08	20.79/33.67	97.43/3.00	94.92/5.87
Pose no.1	33.11/45.12	29.90/42.75	95.86/5.79	96.27/2.25
Pose no.2	31.99/46.83	30.74/45.13	97.81/1.33	96.28/2.31
Pose no.3	34.02/23.49	28.17/21.41	98.26/1.15	97.08/1.94
Pose no.4	5.44/10.43	2.89/6.39	96.41/3.53	94.01/6.40
0° skew	10.68/17.55	8.39/14.66	99.28/0.65	98.57/1.29
15° skew	17.73/25.22	14.51/22.07	95.76/4.86	95.69/2.03
−15° skew	50.02/46.75	45.86/45.70	96.21/2.73	93.47/5.01

Table 3.8: OCR evaluation of *planar* pages using character recognition scores (CRS) and word recognition rates (WRR).

(*mean/std*)×1%	Original image		Rectified image	
	CRS	WRR	CRS	WRR
All	23.05/19.52	14.29/16.52	87.64/25.38	83.83/24.75
Page 1	24.18/19.94	15.09/17.06	95.92/2.95	92.59/4.61
Page 2	23.90/20.67	15.24/17.34	88.21/27.40	86.04/27.00
Page 3	21.76/19.18	13.75/16.44	86.60/26.80	81.94/25.67
Page 4	24.75/20.01	15.48/16.74	85.86/26.70	82.34/25.91
Page 5	20.66/19.14	11.87/16.13	81.60/32.04	76.22/30.61
Pose no.1	22.02/14.66	11.82/9.98	82.09/30.39	77.41/29.08
Pose no.2	29.88/27.58	21.83/25.45	92.26/17.66	88.55/17.40
Pose no.3	29.17/17.00	18.52/12.87	95.94/2.76	92.67/4.38
Pose no.4	11.71/12.45	6.00/7.26	91.76/18.08	87.70/18.44
0° skew	22.05/13.69	11.20/11.59	89.84/26.07	87.78/25.68
15° skew	19.10/16.44	10.27/10.06	93.12/4.35	87.75/6.54
−15° skew	28.00/25.75	21.40/22.78	79.95/34.21	75.95/32.74

Table 3.9: OCR evaluation of *curved* pages using character recognition scores (CRS) and word recognition rates (WRR)

3.5 Discussion

Our algorithms for geometric rectification of camera-captured document images work with both planar and curved documents. The evaluation results in the last section show satisfactory accuracy in shape estimation. The OCR performance tests indicate the rectified images are significantly more OCR compatible than the original images. Our method has potential usage in all camera-oriented OCR applications, such as text-to-voice input for the visually impaired, outdoor document archiving, digitizing fragile manuscripts for digital libraries, etc.

Our rectification method does not specify how to determine weighting coefficients used in shape optimization. Experiments with different values show no significant difference in the results. We find the relative order of magnitudes important, not the actual coefficient values. Given sufficient training data, it is possible to optimize these coefficients. However, this dissertation does not cover this subject.

Chapter 4

Mosaicing of Camera-captured Document Images

4.1 Motivation and related work

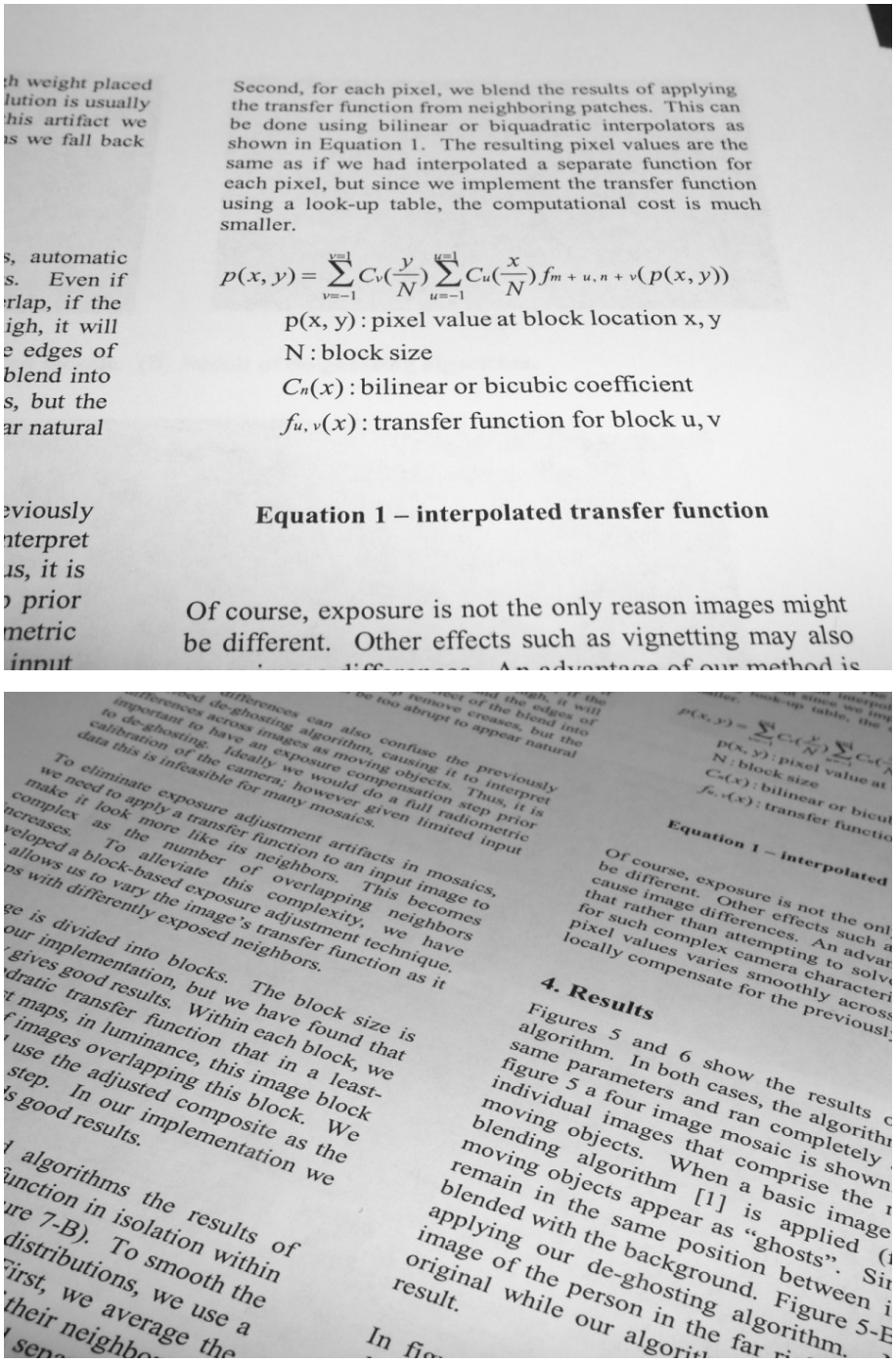
Digital image mosaicing has been studied for several decades, starting from the mosaicing of aerial and satellite pictures, now expanding into the consumer market for panoramic picture generation. Its success depends on two key components: image registration and image blending. The first aims at finding the geometric relationship between the to-be-mosaiced images, while the latter concerns the creation of a seamless composition.

Many researchers have developed techniques for the special case of document image mosaicing [31, 46, 50, 61, 72, 76]. Their basic idea involves creating a full view of a document page, often too large to capture during a single scan or in a single frame by stitching together many small patches. If the small images are obtained through flatbed scanners [31, 61], image registration is somewhat easier because the overlapping part of two images differ only by a 2D Euclidean transformation. However, if the images are captured by cameras, the overlapping images differ by a projective transformation. Virtually all reported work of which we are aware on document mosaicing using cameras imposes some restrictions on the camera position to avoid perspective distortion. Some simply require the user to point the camera straight at the document page [46, 72]. Others require hardware support. Nakao

et al [50] attach a video camera to a mouse, facing down at the document page. While a user drags the mouse across the page, a sequence of pictures are taken and registered pairwise with the help of mouse movement. In [76] a overhead camera is fixed facing down while the document moves on the desktop. Hardware support reduces projective transformations to Euclidean transformations. However, it also counters the convenience, flexibility, and portability of cameras.

Our goal is to remove the constraints on camera position and motion such that users can take pictures of a document from any position without requirement of special hardware. Figure 4.1 shows two image patches of a document captured by a camera. Due to unconstrained camera zooming and positions, these two images differ greatly in perspective, resolution, brightness, contrast, and sharpness. Although many methods have been proposed for image registration ([60, 34], to name a few), the images in Figure 4.1 still present great challenges because of large displacement, small overlapping area ($\sim 10\%$), significant perspective distortion, and periodicity of printed text which presents indistinguishable texture patterns everywhere. The Fourier-Mellin registration method [60] does not succeed. We also tried robust estimators (RANSAC and graduated assignment [23]) with a feature points detector (PCA-SIFT [34]), which failed because the periodicity of text leads to a large number of outliers (up to 90%) in feature point matches (see Figure 4.2).

With respect to image blending, Figure 4.3 reveals three possible problems that have not been well addressed. First, the lighting is inconsistent between two images, a common result of consumer grade cameras with inaccurate auto-exposure metering and on-camera flash. Conventional blending computes the weighted average in an



Second, for each pixel, we blend the results of applying the transfer function from neighboring patches. This can be done using bilinear or biquadratic interpolators as shown in Equation 1. The resulting pixel values are the same as if we had interpolated a separate function for each pixel, but since we implement the transfer function using a look-up table, the computational cost is much smaller.

$$p(x,y) = \sum_{v=-1}^{v=1} C_v\left(\frac{y}{N}\right) \sum_{u=-1}^{u=1} C_u\left(\frac{x}{N}\right) f_{m+u,n+v}(p(x,y))$$

p(x, y) : pixel value at block location x, y

N : block size

$C_n(x)$: bilinear or bicubic coefficient

$f_{u,v}(x)$: transfer function for block u, v

Equation 1 – interpolated transfer function

Of course, exposure is not the only reason images might be different. Other effects such as vignetting may also

Figure 4.1: Two examples of image patches for mosaicing.

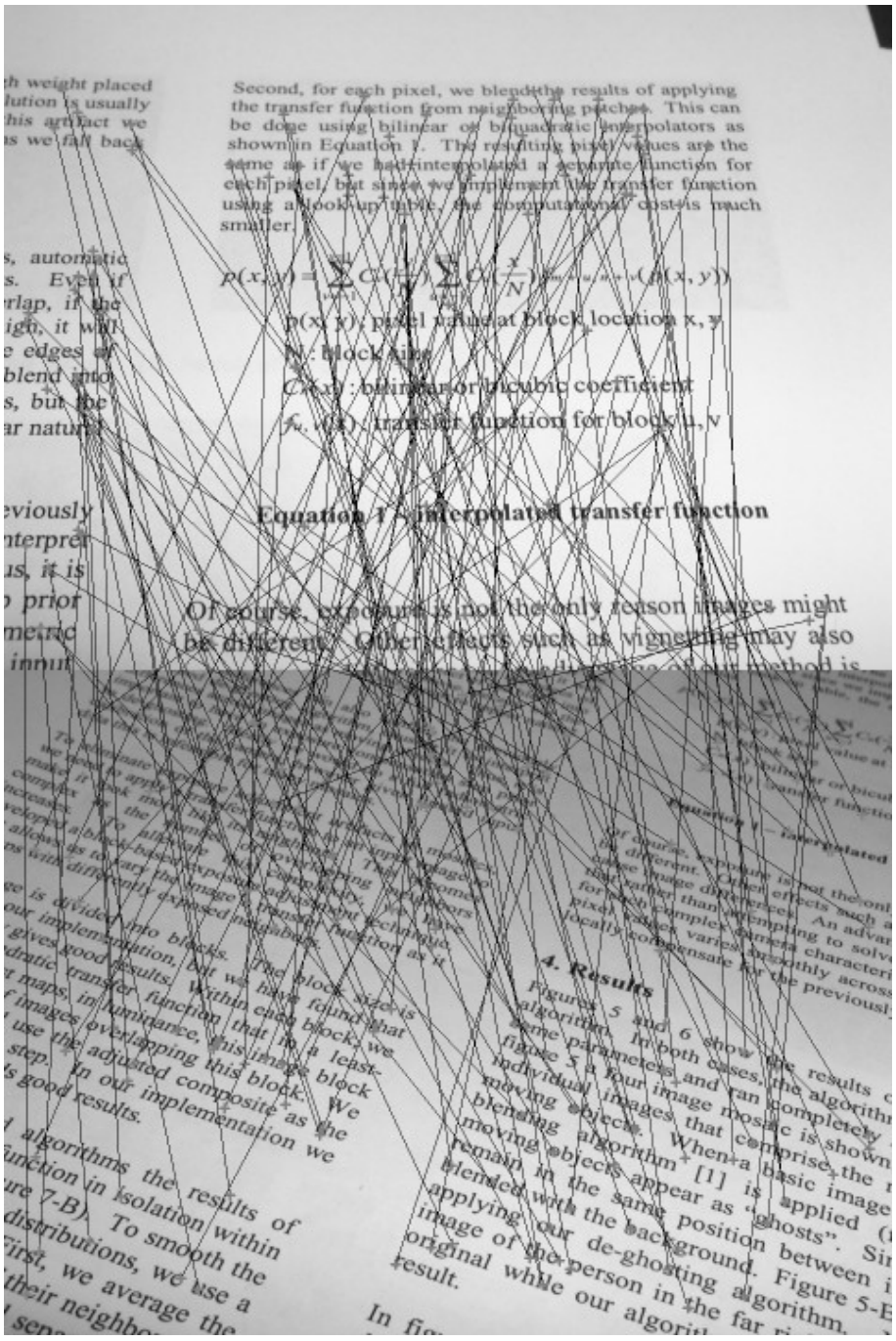


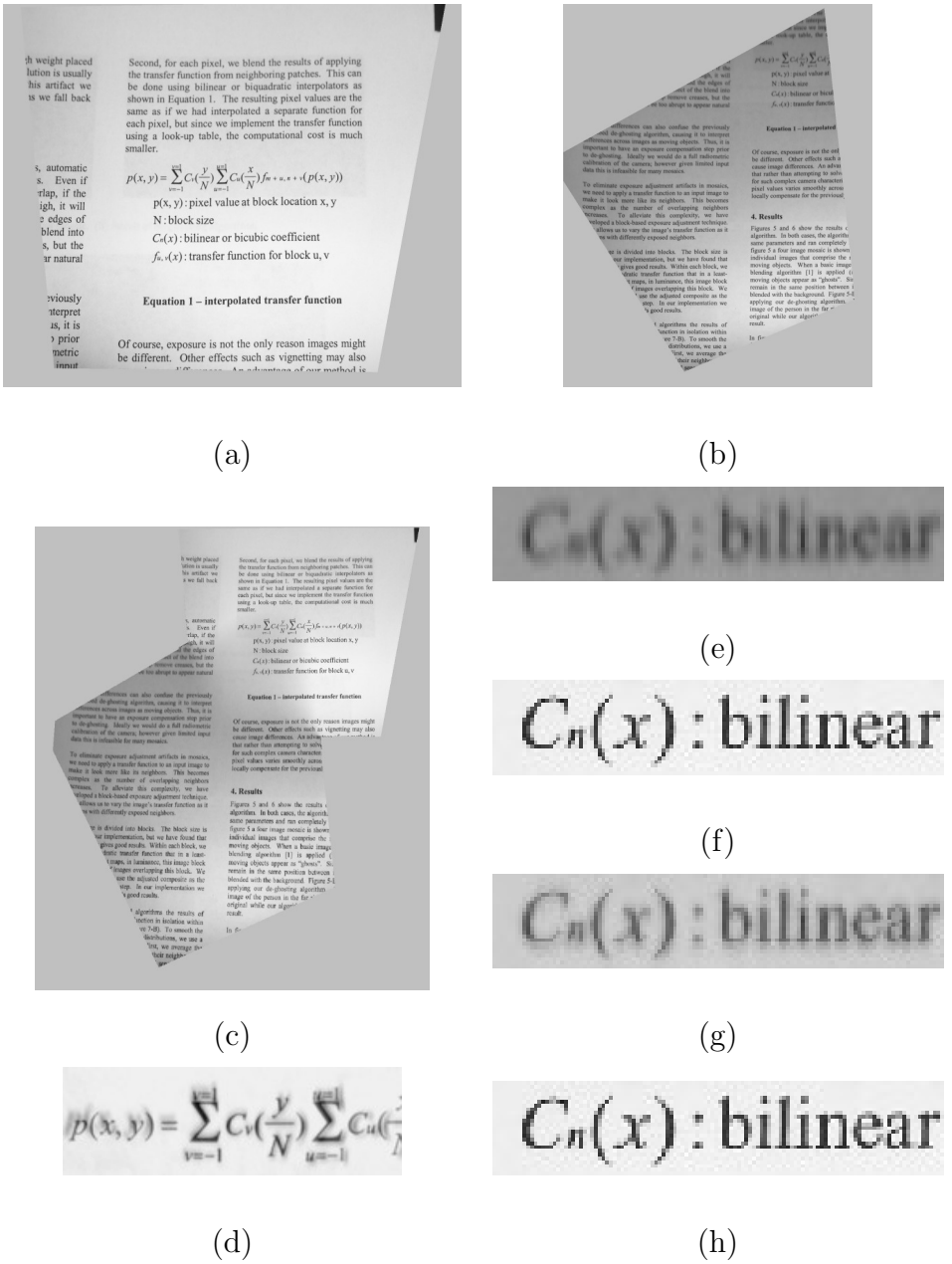
Figure 4.2: Matching points found by PCA-SIFT between two image patches.

overlapped area, i.e., $f = a_1 f_1 + a_2 f_2$, where f_1 and f_2 are pixel values from two images, a_1 and a_2 are two weights that sum to 1. By varying the weights, one achieves a gradual transition from one image to another across the overlapping area. Other more sophisticated methods exist, which are essentially variations of weighted averaging [6]. Though averaging may work for general images, they are not optimized for document images.

First, averaging methods treat only the overlapping area. They do not address the overall uneven lighting across images. Second, registration may have errors. In mis-registered areas, weighted averaging would result in double or so-called 'ghost' images. Third, two images may have different sharpness because of different resolution, noise level, zooming, out-of-focus blur, motion blur, or lighting change. Weighted averaging essentially reduces the sharpness of the sharper image by blending a blurred image into it. Figure 4.3(c) through (h) show the shortcomings of averaging method. For general scenery or portrait images, a certain amount of lighting variation and blurring is acceptable and 'ghost' can be softened by blurring. However, for document images, viewers and OCR packages expect sharp contrast between text and background and a minimum lighting variation. Therefore, blending does not present the best way of creating document mosaics.

4.2 System overview

Our proposed registration method for two overlapping views consists of two steps. First, we remove perspective distortion and rotation of individual views using text



(a) (b)

(e)

(f)

(c) (g)

(d) (h)

Figure 4.3: Challenges for blending of camera-captured document images. (a,b) Rectified images. (c) Mosaicing using weighted averaging. (d) 'Ghost' image due to mis-registration. (e) Small portion of (a). (f) Small portion of (b). (g) Weighted averaging result of (e) and (f) extracted from (c). (h) Result of our selective image blending method.

lines and vertical character strokes detected in document images. This step removes perspective foreshortening and rotation and leaves only a translation and a scaling between the two views. Then, we find feature point matches between views using PCA-SIFT. Although outliers still dominate, we can filter them out efficiently using a voting mechanism similar to Hough transform. After refining the transformation with cross-correlation block matching results we obtain an accurate registration result.

We treat the inconsistency of lighting by localized histogram normalization, which balances the brightness and contrast across two images as well as within each. Then, in the overlapped area, we perform a component level selective image composition, which preserves the sharpness of the printed markings, and ensures a smooth transition near the overlapping area border.

Overall, Figure 4.4 illustrates the system work flow.

4.3 Image registration

After rectifying the image patches, they ideally should be free of perspective distortion, and the overlapping portion of a pair of images should differ only by translation and scale. However, the problems of small overlapping area, large displacement, and periodicity of texture still prevent common registration methods from succeeding. For example, the Fourier-Mellin method still fails and PCA-SIFT still generates many false matches (see Figure 4.5) that defeat graduated assignment and make RANSAC impractical. Our solution filters out the outliers in matches using a vot-

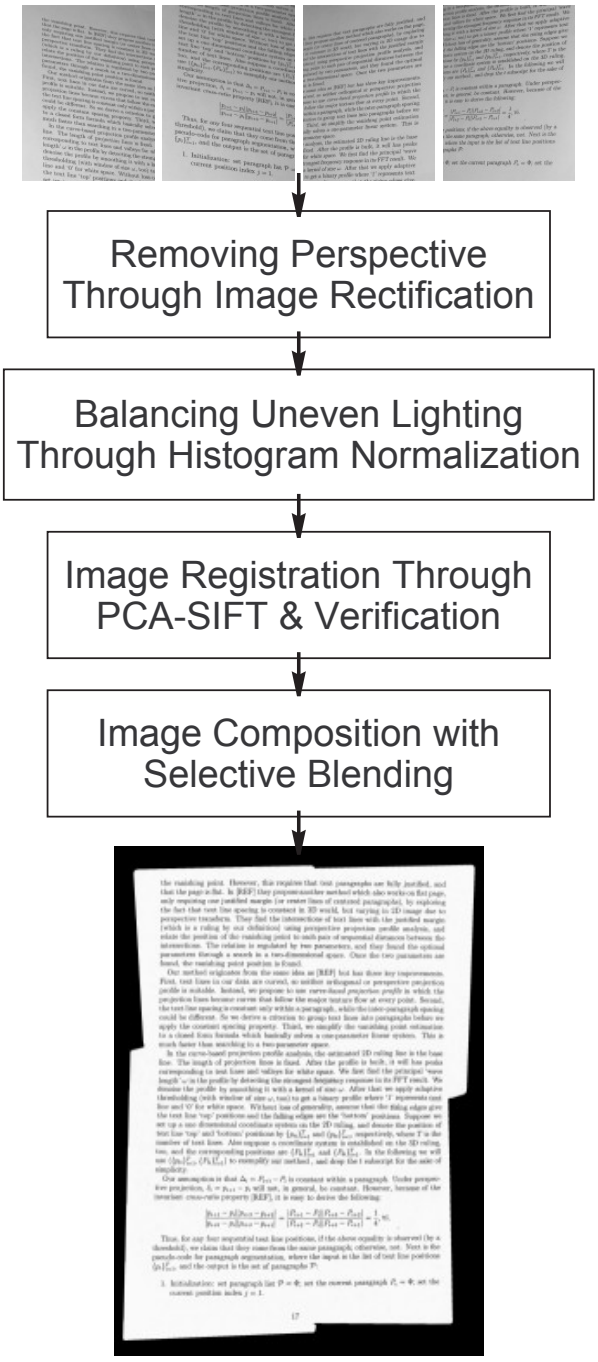


Figure 4.4: Work flow of document mosaicing

ing technique in the spirit of Hough transform, and uses the good matches to register the two images.

First, let us assume the scale is known. Suppose two images (called A and B) are placed within the same coordinate system after proper scaling, and the true translation of image B with respect to image A is (x_0, y_0). Let $\{p_i\}_{i=1}^N$ be the feature points in image A, and $\{q_i\}_{i=1}^N$ be the matched points in image B. If p_i and q_i are a correct match, we have $q_i - p_i = (x_0, y_0)$, and inequality otherwise. We compute all the displacements between matched points, i.e., let $q_i - p_i = (x_i, y_i)$. We have $(x_j, y_j) = (x_k, y_k)$ (we say that they are *compatible*), where j and k denote any two correct matches. In the meantime, the probability of $(x_s, y_s) = (x_t, y_t)$, where either s or t denotes an incorrect match, is extremely low assuming incorrect matches are randomly distributed across the image. We group the matches with equal displacement (within a certain quantization bound) into compatible groups. Ideally, all correct matches are assigned to one group, while each incorrect match constitutes a group of its own. Hence, the correct matches are the matches in the largest group, and their displacements represent the correct translation. In practice, due to the quantization in compatibility test, some incorrect matches that are similar may be placed in one group. Even so, the sizes of such groups are highly unlikely to surpass the size of the group of correct matches.

If the scale is imperfect, the compatibility among correct matches will degrade. A small scale error can be contained by the quantization in compatibility test. As the error increases, the group of correct matches will eventually split. Given a completely incorrect scale, the displacement distribution of correct matches will

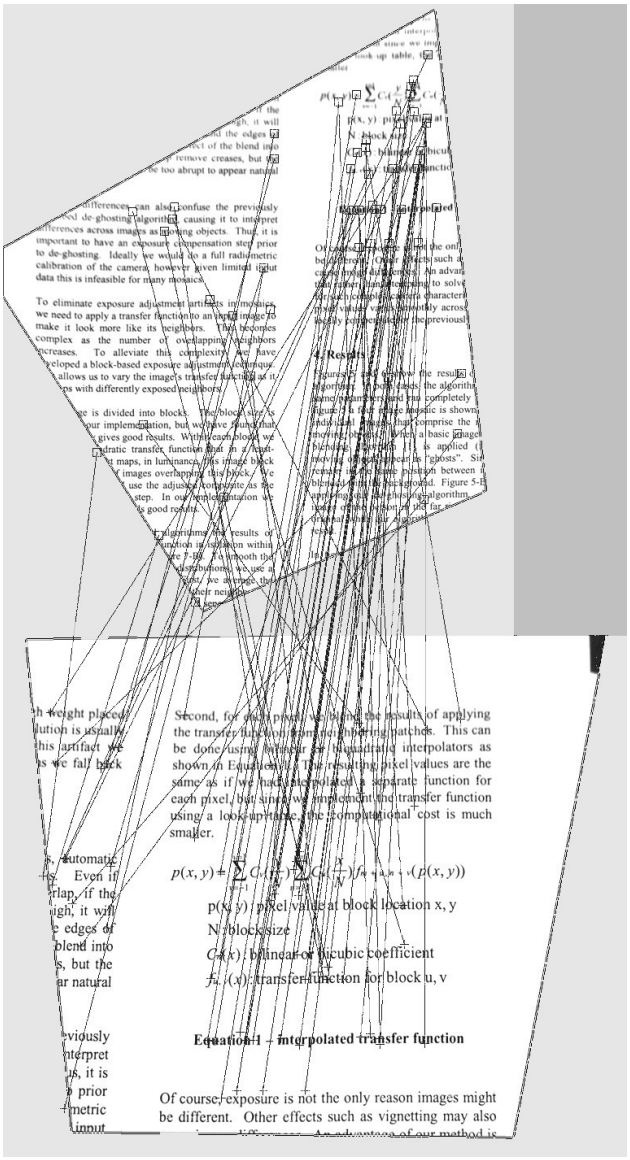


Figure 4.5: Matches found by PCA-SIFT on the rectified images are dominated by outliers.

be as random as incorrect matches, so the largest compatible group will split into single-match groups. In summary, the largest compatible group appears when the scale is correct.

Based on the above analysis, searching for the largest compatible group of matches can simultaneously solve the problems of finding 1) the correct matches, 2) the correct scale, and 3) the correct translation between two images. The specific procedure is as follows:

1. For every scale s in a range, construct the compatible groups and let $g(s)$ be the largest.

2. Select s^* which maximize $|g(s)|$ and s^* is the correct scale.

3. Find all matches in $g(s^*)$, compute the mean of their displacements, which is the correct translation.

For a given scale, we use a 2D histogram of the match displacements to find the compatible groups. We divide the 2D displacement space into bins, and the displacement of each match falls in one bin. To address quantization error at bin boundaries, we smooth the 2D histogram by a 3×3 averaging kernel. Then, the bin with the most votes is the largest compatible group. The optimal bin size should be proportional to the average position error of the correctly matched feature points. In practice, we find it not critical. We use 1/20 of the image diagonal length as the bin size.

Figure 4.6 shows the sizes of the first and second largest compatible groups found in 2D histograms for different scales. We use PCA-SIFT to find the matches

between the two images in Figure 4.3(a)(b). The highest peak in the solid curve identifies the correct scale. At the correct scale, the second largest group (only three votes) is much smaller than the largest group (12 votes). This shows good aggregation of correct matches. After examination, we found the second largest group resides in a neighboring bin of the largest group, and the three matches are approximately correct. These two groups would merge if the bin size is increased. With different bin sizes we obtain curves slightly different from those in Figure 4.6. The correct scale is always found.

The figure also shows that when the scale rests slightly larger than the best value, the solid curve drops while the dotted line climbs. This means some matches in the largest group shift to the second largest group in the neighboring bin. This confirms the largest group splits when the scale is not perfect. When the scale differs significantly from the best value, either to the left of right, the solid curve drops to two and the dotted curve stays at one. The largest group keeps two matches because PCA-SIFT repeated a pair of matched points in its output.

Given the best scale, we use the corresponding 2D histogram to find the matches aggregated in the largest group at this scale. Figure 4.7 shows the correct and incorrect matches.

Using the correct matches, we compute an initial projective transformation between the two images and map one into the other, as shown in Figure 4.8(a). Because good matches tend to reside near the overlapped region's center, the registration is inaccurate near the border. We further refine the registration using cross-correlation block matching. This results in a dense and accurate matched

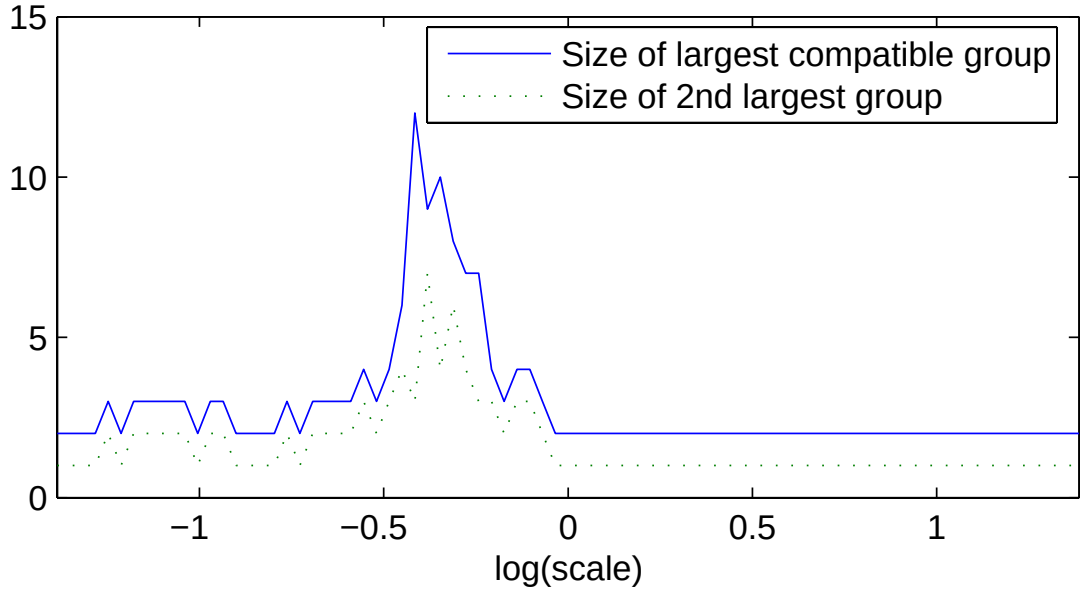


Figure 4.6: 2D histogram peak values vs. scales

point set covering the whole overlapped area, which generates a refined projective transformation (see Figure 4.8(b)).

4.4 Seamless composition

As stated in the introduction, three difficulties arise in creating a seamless document mosaic. The first occurs because of inconsistent lighting across two images. Documents are fundamentally binary with black print on white paper, and viewers' eyes are sensitive to varying shades in documents. Typically, the histogram of a document image is bimodal. Different lighting conditions cause the two modes to shift. One way of balancing the lighting across two document images involves binarizing both images. However, binarization introduces artifacts. Instead, we choose

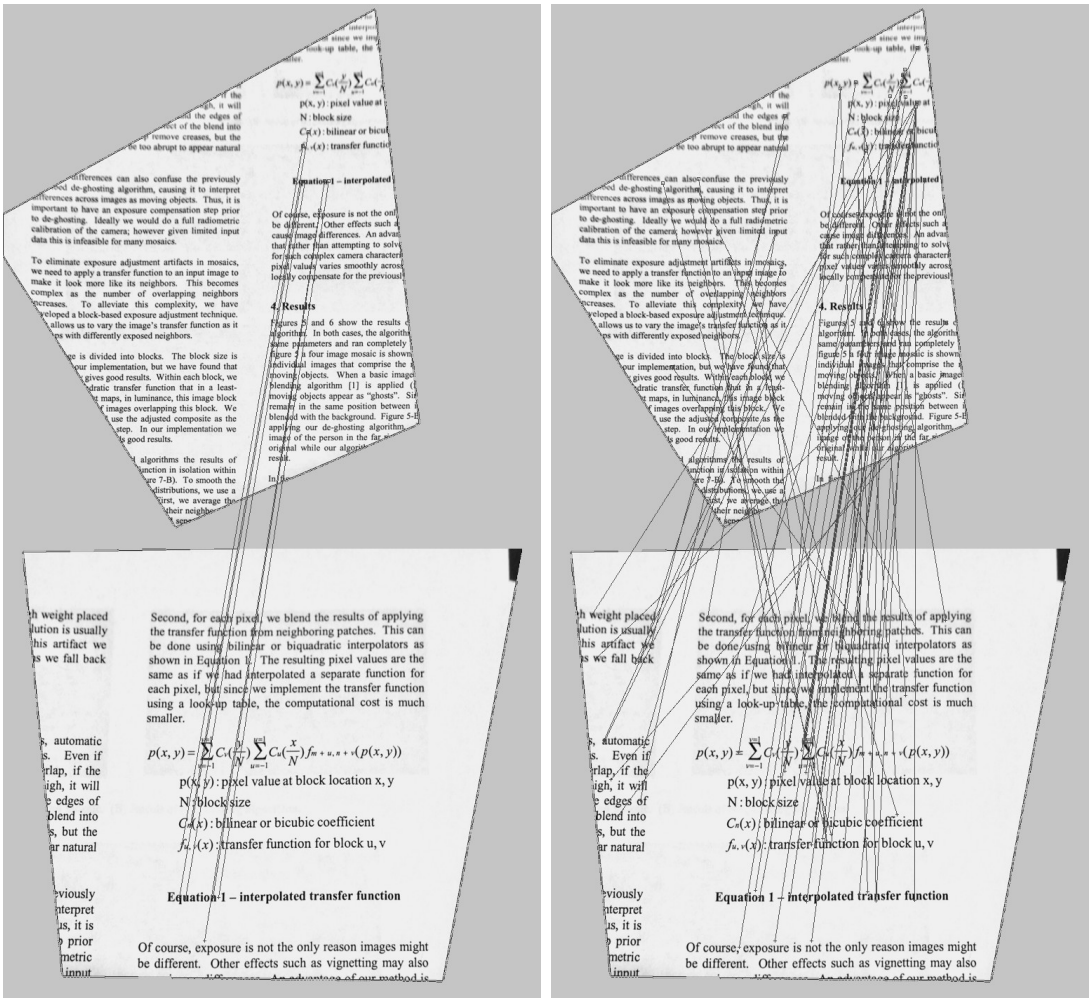
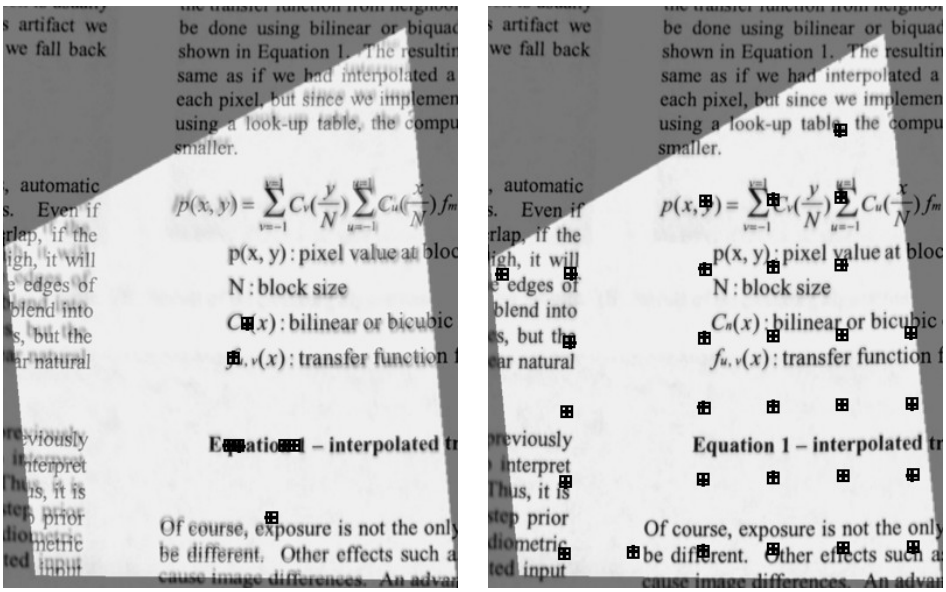


Figure 4.7: Correct (left) and incorrect (right) matches found by registration in the PCA-SIFT result.



(a) (b)

Figure 4.8: Image registration results. (a) Registration by correct PCA-SIFT matches shows misalignment around border area. Squares and crosses indicate the matched points. (b) Registration by block matching results is very accurate.

localized histogram normalization. Essentially, we normalize the histogram of a small neighborhood so the two modes shift to black and white (or deep dark and light gray), respectively. Histogram normalization preserves the transition between background and foreground, resulting in more pleasant text images for viewers (see Figure 4.9).

The second problem is registration error, and the third is different sharpness of patch images. We solve both with *selective image composition*, i.e., each pixel in the result is chosen from the image with the best sharpness. We measure sharpness in an image by the local average of gradient magnitudes. The index of the chosen image of a pixel is called this pixel's decision.

The pixel-level decisions can be represented by a map in which the same decisions are grouped into regions. The boundaries of decision regions may intersect characters and words. So, if we apply pixel-level decisions directly, some characters or words may consist of pieces with different sharpness chosen from different images, which is not desirable. Furthermore, mis-registration tends to break decision regions into small pieces, resulting in 'ghost' images.

Therefore we aggregate the pixel-level decisions at the word level. More specifically, we compose an averaging image for the overlapped area, binarize it, dilate the foreground, and find connected components. The dilation has two effects. First, areas that may contain 'ghost' images are merged into the nearest component. Second, dilation kernel's width exceeds its height, so components in a word more likely merge than components from upper or lower text lines. As a result, most connected components contain a word. All the pixels inside a connected component vote with

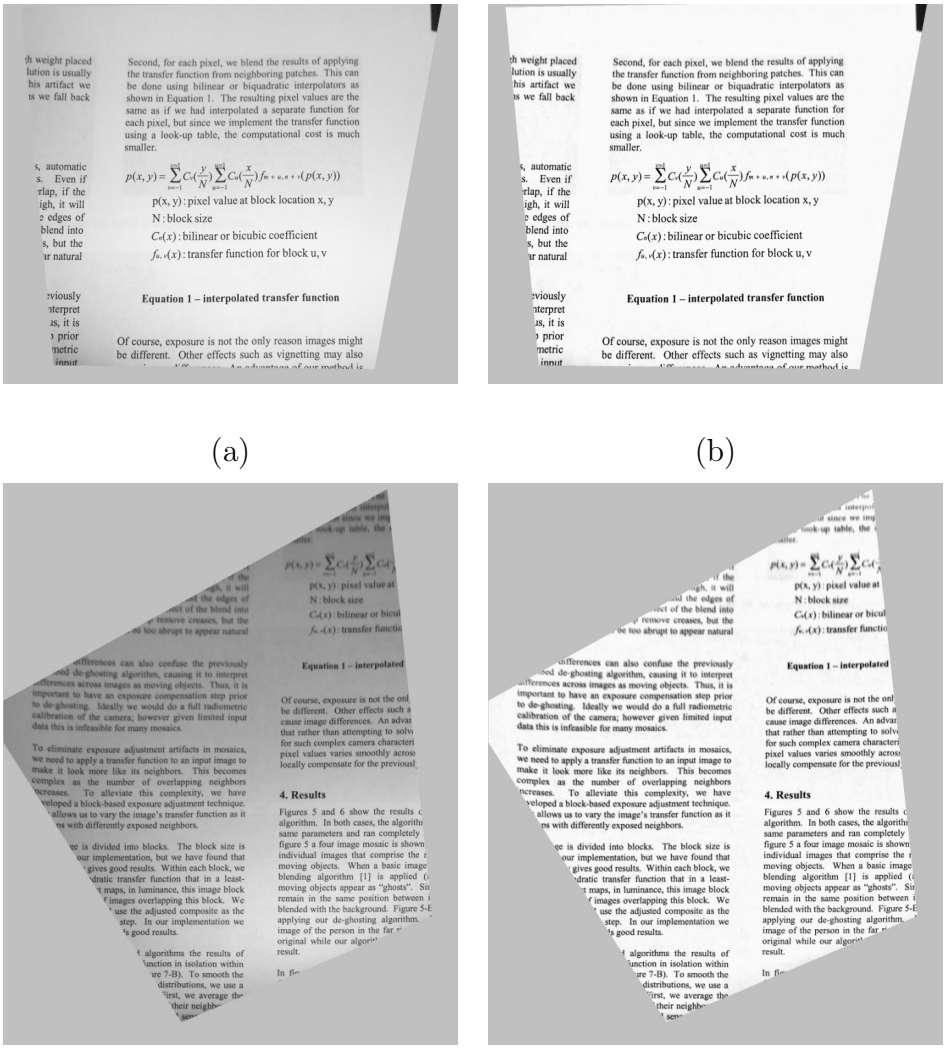


Figure 4.9: Results of localized histogram normalization. (a,c) Before. (b,d) After.

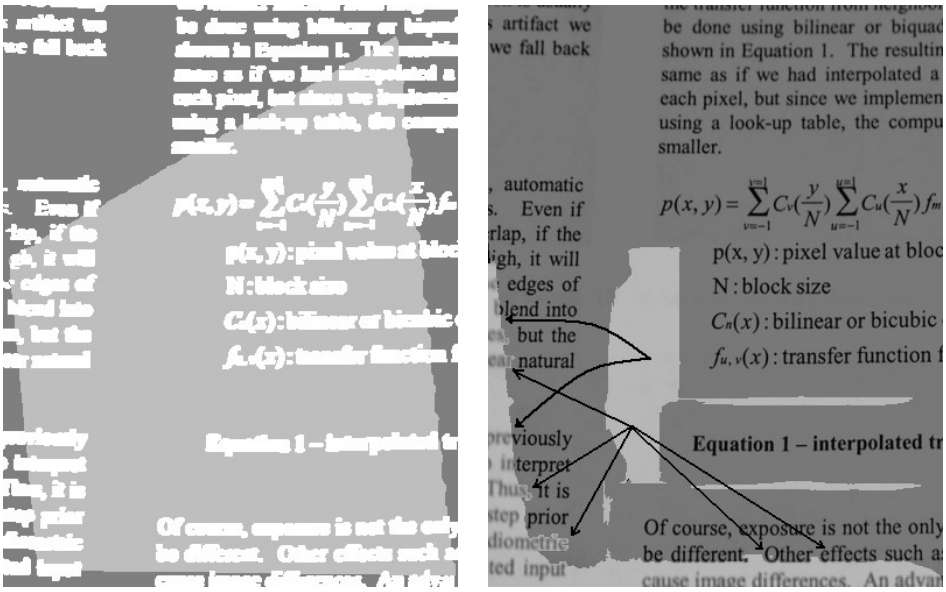
their pixel-level decision, and the majority vote comprises the component decision. The entire component is chosen from the corresponding image. This process ensures 'ghost' images are eliminated and words do not consist various sharpness. For background areas, the variation of sharpness is not visible, so we use the pixel-level decisions directly.

Figure 4.10 illustrates the process of selective image composition and the results. Figure 4.10(a) shows most components consist of a single word. In Figure 4.10(b), light gray and dark gray represent two component-level decision values. The straight arrows indicate words that initially cross boundaries of pixel-level decision regions. They are not broken by component-level decision regions.

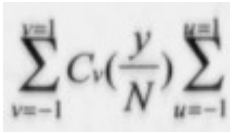
Words may still be broken, not by the decision regions, but by the boundaries of the overlapping area. The curve arrows point to two words in this case. It can be treated by overriding the component-level decision if a connected component is fully contained in only one image.

In the background area, the pixel-level decisions result in a large light gray region embedded in dark gray area. This does not create visible difference in the final image because the variation of sharpness in background is small.

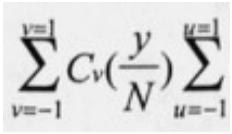
In Figure 4.10, the comparison between (c) and (d) shows our approach preserves the sharpness. In Figure 4.10(e), the overlapping area boundary is visible. It is eliminated in Figure 4.10(f).



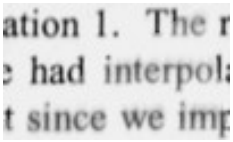
(a) (b)



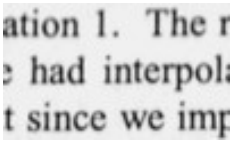
(c)



(d)



(e)



(f)

Figure 4.10: Selective image composition. (a) Connected component are represented by white. The overlapping area is represented by light gray. (b) The binary selection decision map distinguished by dark and light gray. (c,e) Weighted averaging result. (d,f) Selective image composition result.

4.5 Experimental results

In the first experiment, we took four pictures of a document. Each picture covers roughly 1/4 of the page. After geometric rectification, we select one as the base image, and one by one mosaic the other three onto the base image. Figure 4.11 shows the four original images and the mosaicing result. In the second test, we produced a full page mosaic from eight images (see Figure 4.12). In both tests, we experimented with different base images and different orders of adding the remaining images. Because the geometric rectification is not error free, the rectified images differ slightly by projective transformations. Therefore, the mosaics using different base images differ slightly from each other, too. The order for adding images has no visible effect. Both Figure 4.11 and Figure 4.12 show accurate registration and seamless composition.

4.6 Discussion

A novel method for mosaicing camera-captured document images is presented in this chapter. Our method makes use of a geometric rectification algorithm, which simplifies the image registration problem to the search of an unknown scale and translation. We find candidate correspondent points in image pairs and apply a grouping technique in the spirit of Hough transform to remove the outliers, which typically are over 90%. Our selective image composition approach eliminates double image defects and blurring in the composite image. Handheld devices are the main application platform of our algorithms. With our technique, a camera can simulate

the function of a scanner or photocopier.

Our experiments show we can obtain accurate mosaicing of full page planar documents. Compared to planar documents, curved documents are more difficult to rectify, so we find that the rectified images can not be registered by pure projective transformations. Local warping or morphing would be required to register them at pixel accuracy. Nevertheless, our method can serve as a pre-registration step, after which local projective transformations can be computed for small image blocks. This step, however, is out of the scope of this dissertation.

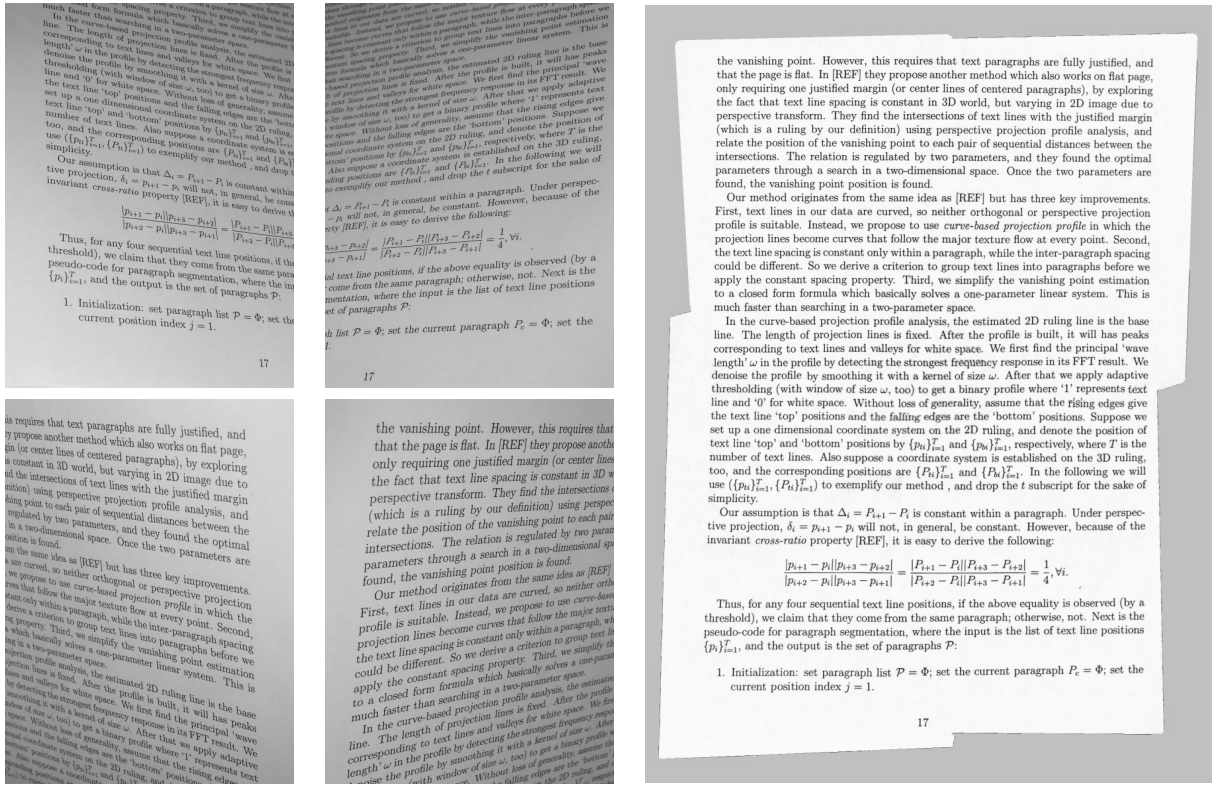


Figure 4.11: Four image patches and the full page mosaic.

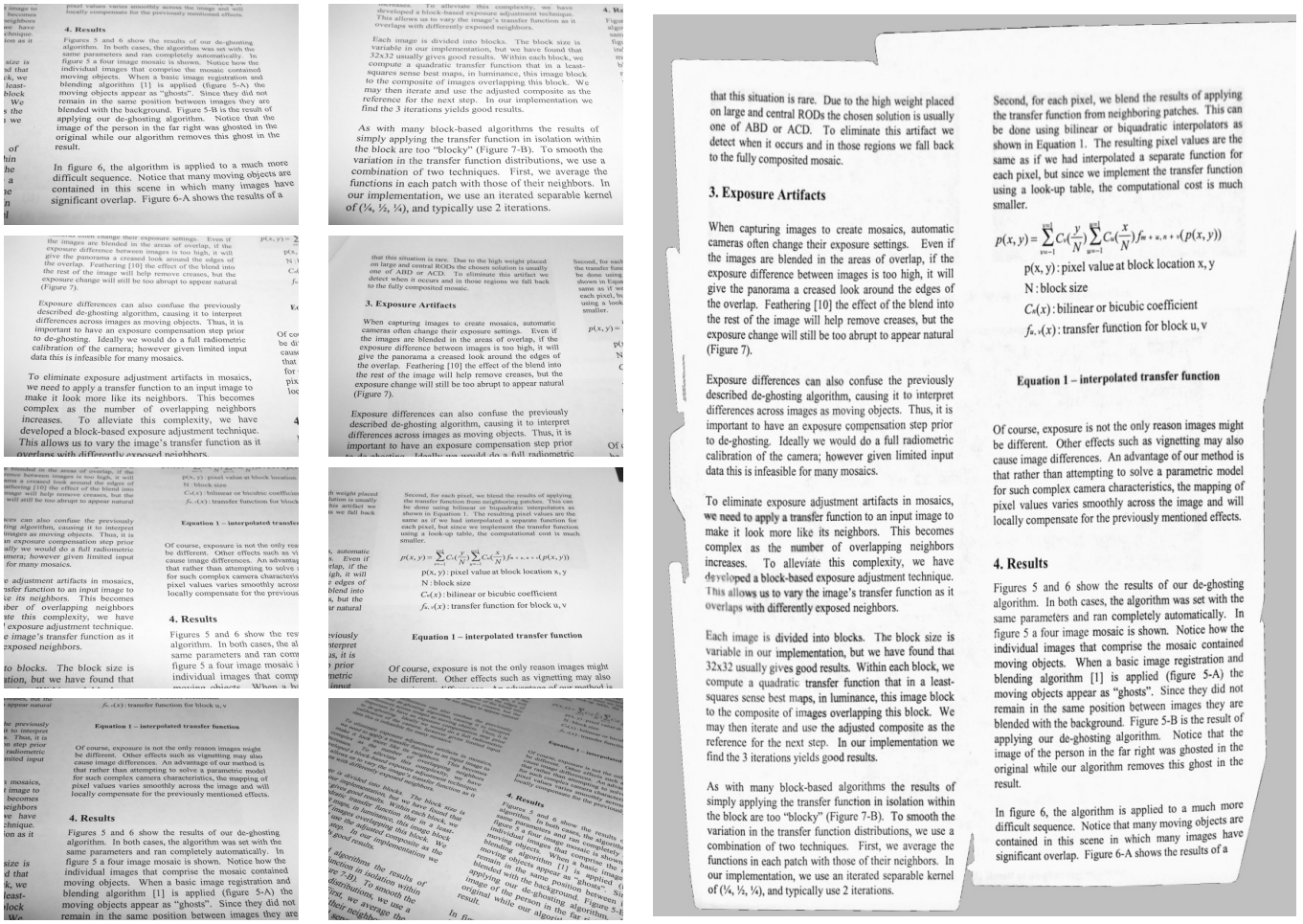


Figure 4.12: Eight image patches and the full page mosaic.

Chapter 5

Recognition of Layout Structure in Document Images

5.1 Motivation and related work

As the number of digitized documents grows, efficient and automatic techniques for metadata extraction become more important. Unlike electronic documents, which are compounded with metadata during generation, digitized documents are mostly stored in image and text format with a few tags about generation time, image property, etc. These primitive metadata cannot serve advanced needs. For example, for technical papers, a very important type of documents used by millions of researchers every day, an incomplete list of metadata worth extracting includes: genre, book title or journal title, subject, author(s) and contact information, title, keywords, abstract, publisher, editor(s), page number, citations, copyright notice, and so on. Other examples include correspondences such as memorandums, faxes, and letters. Most email users rely on the convenient search function of their software to find messages to/from a certain contact, within a date range, about a specific subject, etc. Such functions would be also highly valuable for digitized correspondences.

While manual tagging presents a possible solution, and in fact is in use in some commercial projects, its prohibitively high cost makes it unattractive. With the volume of digitized documents growing increasingly faster, the speed and cost concerns make automatic metadata extraction a necessity.

There are two major approaches to the problem of automatic metadata extraction. The first one relies on Natural Language Processing (NLP) techniques to analyze text content recognized by OCR. NLP can identify semantic elements and syntactic structures in text strings, such as names, locations, dates, special nouns, important key words, etc. The frequency of certain types of words, for example, can help determine the genre of the document. Guided by rules, NLP can extract sender's and receiver's names in the text of a letter. However, except for text, all other visual information of the document (e.g., font attributes, position of text blocks, number of columns, page orientation) is lost in the conversion from image to text. Such non-text features are important in determining the functional and logical roles of zones in the page. For example, a name appearing in a small and centered text block in the top half of the page is more likely to be the author of the document than the same name appearing in a long paragraph at the lower right corner. Thus, text content alone does not provide all the information needed by metadata extraction.

The second method approaches metadata extraction from the perspective of image analysis. In this approach, the physical structure of a document image is analyzed to identify the functional and logical role of different zones (logical labeling) as well as the entire document (document classification or similarity comparison).

In the literature, the problem of logical labeling is studied separately from the problem of document classification (or similarity comparison). Algorithms for the labeling problem mainly analyze the attributes of zones or their neighbors [20, 44, 17, 48, 51, 71] and often lack the power to grasp the overall page struc-

ture. On the other hand, algorithms for the classification problem typically have a global representation for the physical layout of a document image [63, 28, 18] and use pattern recognition techniques to obtain a distance or likelihood score. However, such global representations usually do not explicitly express local or regional features.

Because of the large difference between document classes, a feature set that works for one class may not suit another. Therefore, the ability to expand a system to include new features is important.

Another important practical issue is that many systems require a fair amount of training samples to build their models, especially statistical ones. For many tasks, number of classes may be large and over the time, classes may vary and new classes may appear. In these cases, it could be impractical if models need to be retrained from scratch.

Based on the above considerations, we choose 1) a graph-based representation of the page layout and 2) an adaptive learning approach for model training. Our graph-based model has the flexibility to incorporate both global structure and low level features of a document image. Also, it can be easily adapted to new features in the future. Adaptive learning methods allow the models to be initialized with a relatively small number of samples and to be adjusted with new samples later on. We use a graph matching method to perform *layout structure recognition*, which achieves document classification and logical labeling simultaneously.

5.2 System overview

Our system for layout structure recognition mainly uses the results of page segmentation. If zone classification and OCR are available, their results can also be included. Page segmentation [52, 49, 32] aims to separate a document image containing *heterogeneous* content into *homogeneous* zones. The definition of heterogeneity or homogeneity is, of course, application dependent. Typically, a document image is decomposed into text zones and non-text zones, and within the text zones columns and paragraphs are segmented. Zone classification identifies the type of segmented zones, such as e.g., figures, tables, handwriting, and noise.

There are three key components in the system. The first extracts features from the segmented zones, such as font attributes, line spacing, zone position and size, distances between nearby zones, zone type (if zone classification is performed), the existence of key words (if OCR is available), and construct a graph-based representation of the page layout. The second learns a model for each class of documents from graph instances and adaptively improves the model given new samples. The third is in charge of finding the best match from a model to an incoming instance. When there is only one model in the model base, the system only performs logical labeling; otherwise, both logical labeling and document classification are conducted.

Figure 5.1 illustrates the system overview, where the learning and recognition modules are depicted but the graph construction step is not explicitly shown. In the figure, the `verification` step, which corrects the errors in the result of automatic layout structure recognition, is the only one that requires human interaction. It is

not always in use, though. At the very beginning, when a model is not learned yet and there is no results from the recognition module, verification provides ground-truthed samples to train a new model. After a model is set up, if the error rates in recognition is high, the verified results help the adaptive learning module to improve the model. When a satisfactory model is obtained, the verification is stopped. If, over the time, the class changes such that new instances do not conform to the model and performance degradation is observed, verification and adaptive learning would be required again.

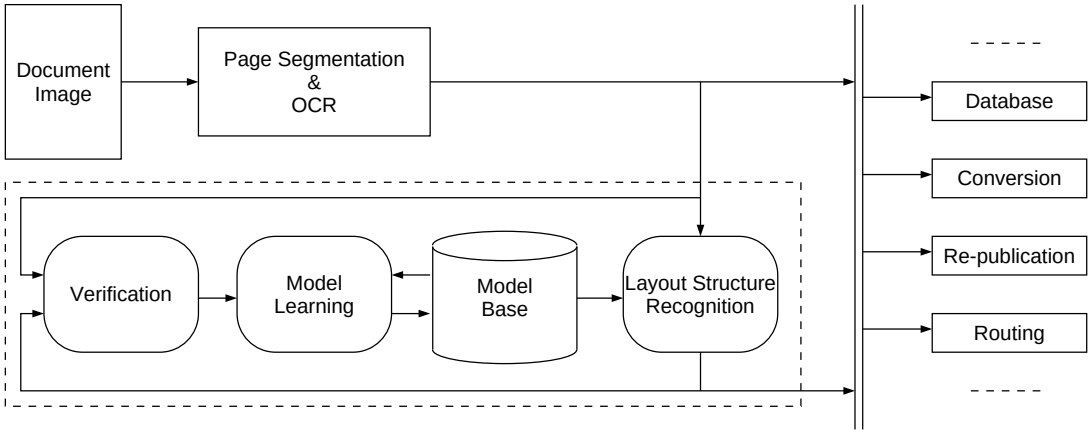


Figure 5.1: Framework of layout structure recognition system.

The labeling and classification results, together with results from page segmentation, zone classification, and OCR, enable many downstream applications that require metadata. Figure 5.2, for example, shows a digitized document and a HTML file generated using both metadata and text.

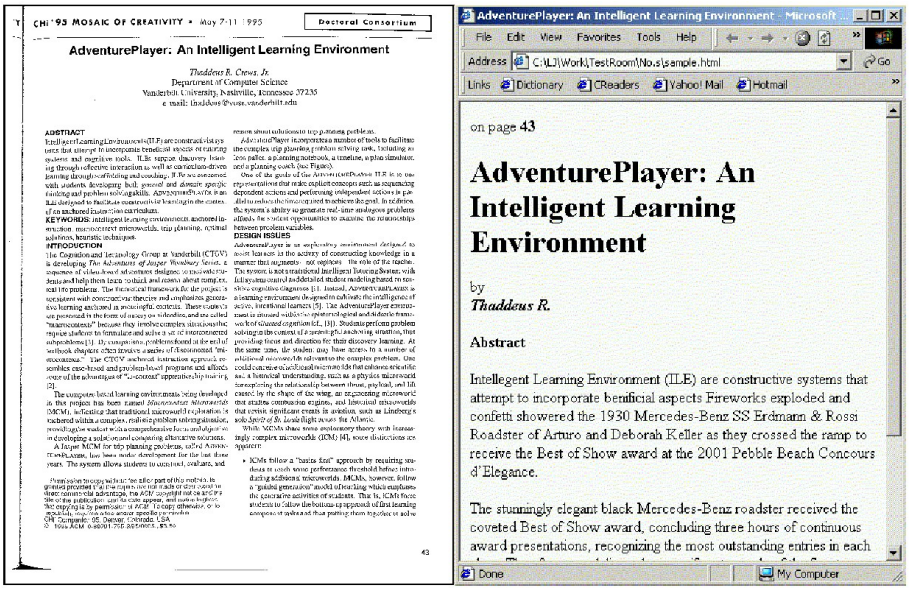


Figure 5.2: A digitized document is converted into HTML using meta data obtained from layout structure recognition and text from OCR.

5.3 Layout structure modeling

We model the physical layout of a structured document image by a *layout graph* [40]. A layout graph is a fully connected attributed graph in which each *node* corresponds to a segmented zone on a page. A node's attributes describe the properties of the zone, which could be geometric features, texture features, content features, or any other features of interest. The attributes associated with an *edge* between a pair of nodes reflects the relationship between the corresponding zones in the page, including spatial relation, distance, font size ratio, common words, etc. A layout graph summarizes the result of page segmentation and, if available, zone identification and OCR. Layout graphs provide a flexible representation of layout structures. With different node and edge attributes, they can describe both local

and global aspects of the layout structure of a document.

A layout graph instance summarizes the layout structure of a document instance, and a *layout graph model* summarizes the common layout structure of a document class. The node and edge attributes in a layout graph model describe the common features present in the document class. In addition, a layout graph model associate each attribute with a weight describing the stability of this attribute within the class. The weights for unstable attributes are low.

For a given document class, a layout graph instance usually has more nodes than the model. For example, page segmentation may over-segment, resulting in many small zones. Or, the page contains zones of handwritten text, stamps, or noise, not common to the document class. A match between a layout graph instance and a model assigns each node in the instance to a node in the model. A correct match between a layout graph instance and a model is, in general, a many-to-one mapping. More than one node in the instance map to one node in the model. In addition, an instance node may map to none of the model nodes, in which case we say it maps to *null* or $\emptyset$. Similarly, a model node may map to none of the instance nodes.

Figure 5.3 shows a layout graph model and an instance. Let $G_M(A, B, C)$ represent the model and $G_I(a, b, c, d)$ be the instance. Each of a, b, c, and d can be mapped to either A, B, C, or $\emptyset$. There are $4^4 = 256$ possible mappings. Two examples are as follows:

$$(A-a, B-b, C-c, A-d), (A-\emptyset, B-d, C-a, \emptyset-d, \emptyset-c)$$

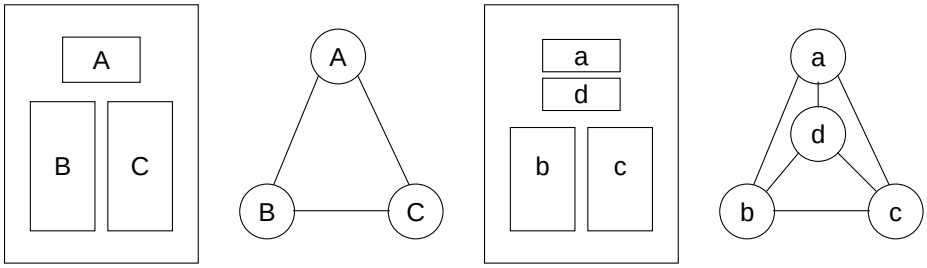


Figure 5.3: Concept of layout graph model A,B,C and instance a,b,c,d.

We transform the many-to-one mapping into a one-to-one mapping before measuring the match quality. If several nodes in a graph instance map to one node in the model, the corresponding zones in the page are grouped as a new zone, and the nodes are replaced by a new node. The node and edge attributes are recomputed accordingly. For example, under the mapping $(A-a, B-b, C-c, A-d)$, $G_I(a, b, c, d)$ transforms into $G'_I(a', b, c)$, where a' comes from a and d, and the mapping becomes $(A - a', B - b, C - c)$ which is one-to-one.

We denote a one-to-one mapping from a model to an instance by

$$h : (G_M \cup \emptyset) \rightarrow (G'_I \cup \emptyset).$$

We define the quality of a match between a layout graph instance and a model as the quality of the induced one-to-one mapping h, which is measured by the following cost function:

$$C(h) = \sum_{x \in (G_M \cup \phi)} C^n(x, h(x)) + \sum_{x,y \in (G_M \cup \phi)} C^e(\overline{xy}, \overline{h(x)h(y)}), \tag{5.1}$$

where $C^n(x, h(x))$ is the cost representing the differences between node x and $h(x)$, and $C^e(\overline{xy}, \overline{h(x)h(y)})$ is the cost measuring differences between edge $\overline{xy}$ and

$\overline{h(x)h(y)}$. We define $C^n(x, h(x))$ as the weighted sum of attribute-level costs:

$$C^n(x, h(x)) = \sum_{i=1}^{N} c_i((x)_i, (h(x))_i),$$

where $(\cdot)_i$ denotes the i-th attribute of a node, and N is the number of attributes of a node. $C^e(\cdot, \cdot)$ is defined similarly. Special costs are defined for the case where either x, y, $h(x)$ or $h(y)$ is $\emptyset$. The best match h^*, between an instance and a model is simply the one with the lest cost:

$$h^* = \underset{\forall h:(G_M \cup \phi) \to (G'_I \cup \phi)}{\operatorname{argmin}} C(h) \tag{5.2}$$

The specific form of an attribute-level cost function depends on the type of the attribute. In our implementation, we use SSD (sum of squared difference) for numerical attributes, and look up tables for qualitative attributes.

5.4 Model matching

Graph matching, in general, is NP-hard [23]. Practical solutions either employ branch-and-bound search with the help of heuristics, or nonlinear optimization techniques. It is more difficult to do many-to-one graph matching.

Our method of finding the match between a layout graph instance and a model consists of two steps (Figure 5.4). In theory, our method cannot guarantee the best match with the least cost, but, in practice, it usually finds a near-optimal solution. In the first step, we find the best one-to-one mapping from the model G_M to the instance G_I. The second step groups unmatched nodes in G_I to the matched ones in such a way that the cost of the final match is least. The success of our approach

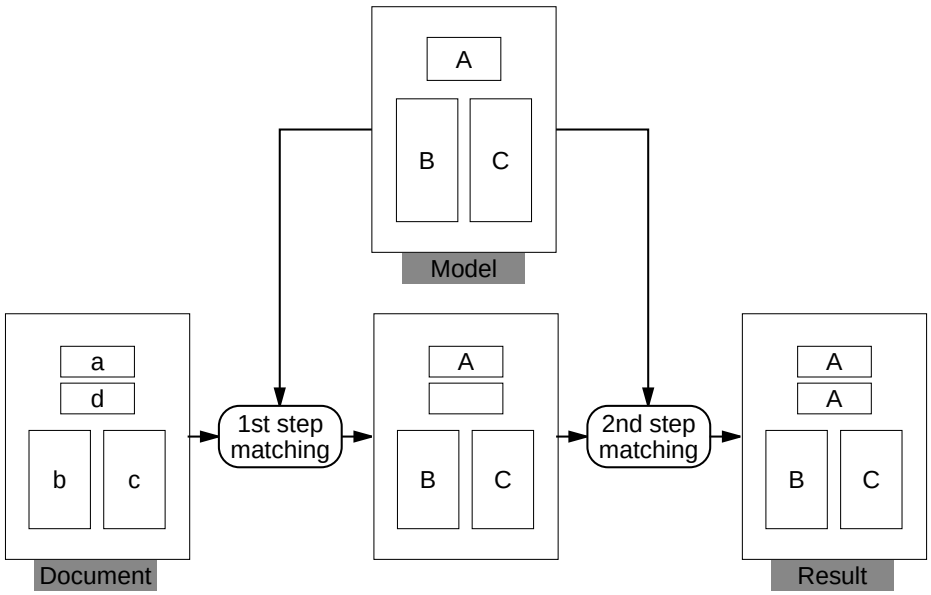


Figure 5.4: Two-step matching process

lies in our observation that, in a correct match, usually many nodes in the instance has an one-to-one mapping with the nodes in the model. The sub-graph consisting of such nodes can be interpreted as the 'back bone' of the graph. Our first step finds the sub-graph and the best sub-graph match, and the second step groups remaining nodes to the sub-graph.

The overall cost function is essentially a summation that can be computed in a incremental way, so we can use an efficient branch-and-bound search strategy to address the computational expense involved in the first step. In practice, the search usually finishes in less than one second. For documents with Manhattan style, we constrain two grouped zones must not overlap in the second step, which greatly reduces the number of possible grouping schemes. For non-Manhattan style documents, we drop this requirement. Figure 5.4 illustrates a sample matching

process. In the first step, three of the four zones are labeled. In the second step, the unlabeled zone d has four choices: be grouped with a, b, c, or to remain unlabeled. After comparing the costs of grouping d with a, b, c and $\emptyset$, the final decision groups d with a, and label both of them A.

5.5 Model learning

Given the definition of the layout graph model and cost functions, the values of attributes and their weights are learned from a set of training samples in a process called *model initialization* [40]. Model initialization finds the typical attribute values in the training set, and set the weights of stable attributes high.

Later on, if user feedback is available, a model is adaptively improved by a process called *adaptive model learning* [39]. For example, in a technical paper model, if a block above "title" is mistakenly labeled as "author", we increase the weight associated with the spatial attribute of the edge between node *title* and *author* — the value of this attribute states that *title is above author* — to increase the cost of such an incorrect labeling. Figure 5.5 shows an example where d is incorrectly matched to $\emptyset$ instead of A. In the improved model, the position attribute of node A is modified accordingly. Inspired by error back-propagation methods used in neural network training, we design the following method for improving model parameters.

Suppose $C(h)$ is the cost of the incorrect mapping h, $C(h^*)$ is the cost of the correct mapping h^*, both under the original model. Since h is the matching result, it follows $C(h) < C(h^*)$. Let $\mathcal{N}$ represent all incorrectly matched nodes in the model.

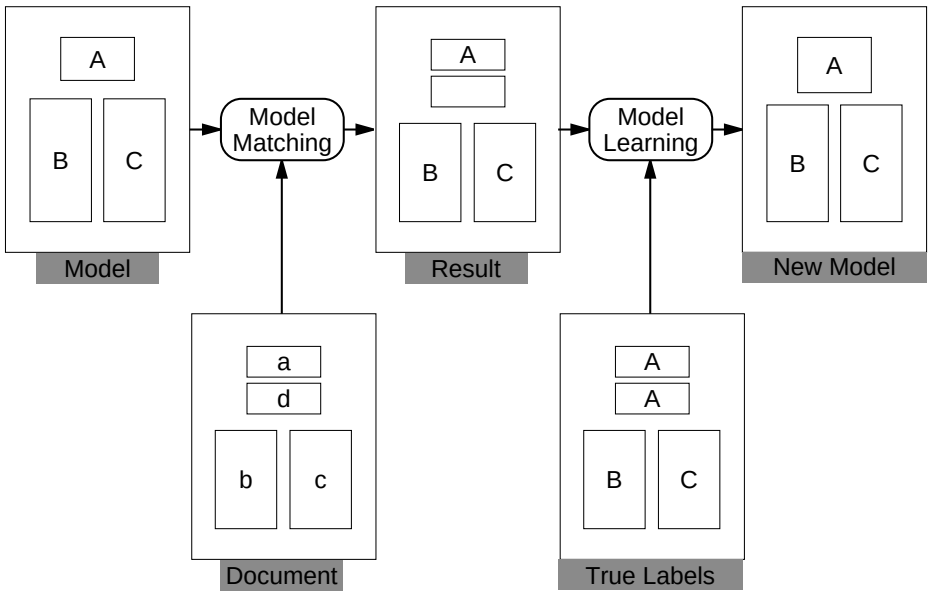


Figure 5.5: Adaptive model learning.

The attributes and weights associated with nodes in $\mathcal{N}$ are modified in such a way that $C'(h^*) - C'(h) < C(h^*) - C(h)$, where $C'(\cdot)$ is the cost using new attributes and weights. If the problem is persistent and repeats itself, eventually the modification will lead to $C(h) > C(h^*)$ such that the correct match results in the least cost.

More specifically, suppose v_m is a numeric attribute of a node $x \in \mathcal{N}$, v_i^* is the corresponding attribute of $h^*(x)$, and v_i is the attribute of $h(x)$. If $c(v_m, v_i) < c(v_m, v_i^*)$, where $c(\cdot, \cdot)$ is the attribute-level cost function, then, this means v_i is closer to v_m than v_i^*; in other words, v_m is unreliable. We shift v_m toward v_i^* and decrease the weight associated with v_m. Otherwise, if $c(v_m, v_i) > c(v_m, v_i^*)$, we increase the weight for v_m. In both cases, we achieve $C'(h^*) - C'(h) < C(h^*) - C(h)$.

5.6 Experiments

5.6.1 Title pages of technical papers

In the first set of experiments, we collected a total of 85 title pages from four technical journals or proceedings (Figure 5.6). These documents share a common style of two-column technical paper title pages, and have some slight difference in the font sizes, positions of headers, footers, page numbers, and so on. Because of the relatively rigid layout structure of these sample documents, we use mainly geometrical features as attributes in the layout graph model. In our implementation, the node attributes include the position and size of the bounding box of the corresponding region in the image, and the average font size of text in the region (font sizes are classified as *small*, *normal*, and *large*). The edge attributes express the spatial relation between two regions. For example, the left boundary of a region can be *to-the-left-of*, *aligned with*, or *to-the-right-of* the left boundary of another region. Similarly, we quantize the spatial relationship between two horizontal boundaries as *above*, *aligned*, or *below*. For each region, we consider its four boundaries plus a central line that divides it into two equal left and right halves. Between two regions, we use a set of nine qualitative descriptors (four for the two pairs of horizontal boundaries, four for the two pairs of vertical boundaries, and one for the two central lines) to describe their spatial relationship.

For attributes with numerical values (e.g., bounding box sizes) we use the absolute differences as the cost function, and for qualitative descriptors, we simply let the cost be 0 if there is no difference and 1 otherwise. In the model initialization

(a) CHI'95 (b) CHI'96 (c) PAMI (d) UIST'95

Figure 5.6: Sample pages from four publications

step, the numerical attribute values are set to the averages in the training samples, and their weights are set to the inverse of their variances. For an attribute with a qualitative descriptor, its weight is set to the inverse of the number of descriptors that are different from the dominant descriptor in the training set.

We use a leave-one-out strategy to test the *title page* model learned from these samples. More precisely, we take out one sample page as the test sample, and use the remaining pages to initialize the model. We call the initialization *round 0*. After that, in each round, we use all the training samples one by one to adaptively modify the model. The order of applying training samples in each round is not changed. For each round, the incorrectly labeled regions are counted, and the average error rate is defined as

$$e_i = \frac{\sum_{j=1}^{J} n_i^{(j)}}{\sum_{j=1}^{J} N^{(j)}}$$

where $N^{(j)}$ is the number of regions in the jth test sample, $n_i^{(j)}$ is the number of incorrectly labeled regions, i is the iteration round number, and J is the number of title pages. Here $J = 85$.

Figure 5.7 shows the average error rate vs. the training cycles. There are a total of 1347 zones in all sample pages. The declining trend of the error rate with the increase of training is clear. After every training sample has been used three times (after round 3), the error rate has dropped about 30%. In Figure 5.8 two title pages and their labeling results are shown.

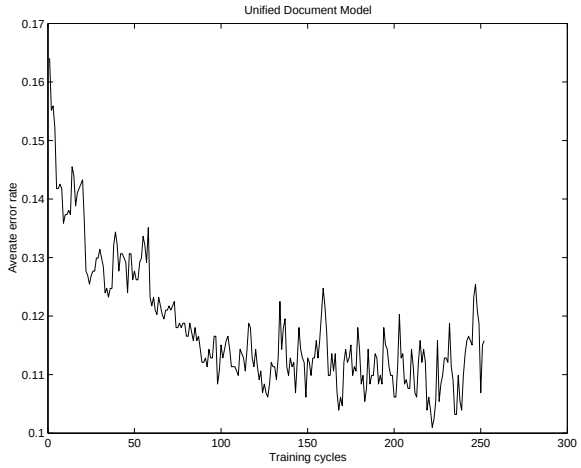
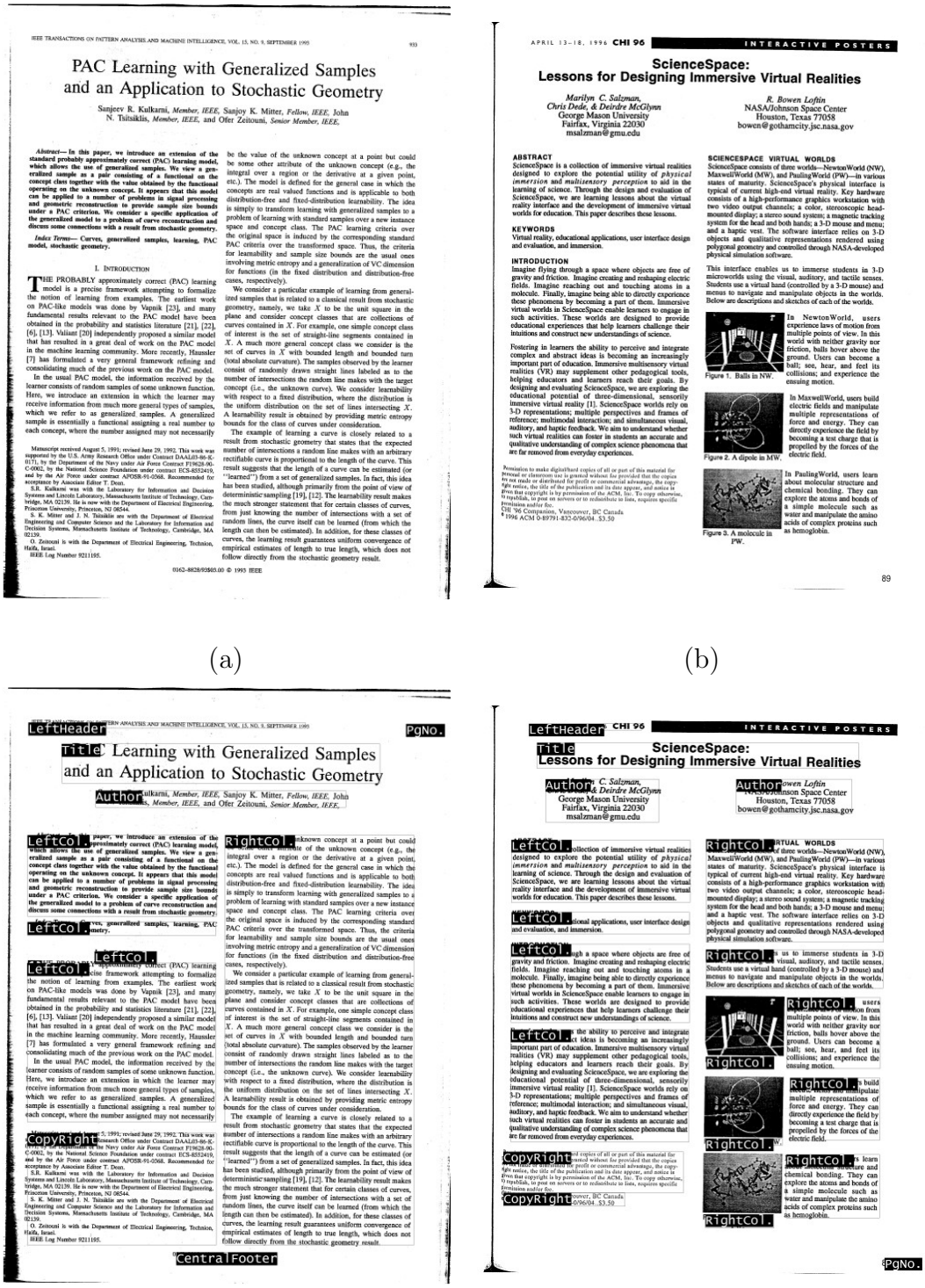


Figure 5.7: Average error rate vs. training cycles for unified model

We also grouped sample pages by the specific publication, build a model for each of them, and test the model using the group of samples. The results are similar to the unified model case, but we can have more power in distinguishing fine structures that are only defined within each publication. For example, PAMI has very strict layout rules which requires an abstract followed by keywords, and the first section must be an introduction with a centered section title. Therefore only for the PAMI model we define "Abstract", "Keywords", and "IntroTitle" labels. In Figure 5.9 the result of labeling using the PAMI model is shown, in which "Abstract", "Keywords", and "IntroTitle" are correctly found. Figure 5.10 shows the average



(a) (b) (c) (d)

Figure 5.8: Examples of labeling results of title pages. (a,b) Documents (c,d) Labeling results

error rate for each model vs. the iteration count. Two sets of labeling results are shown in Figure 5.11 and 5.12.

Using the four models, we carried out a document classification test. Each document is mapped to all four models, and its class is determined by the best model (i.e., with the least cost). For all 85 pages, we achieved 100% accuracy in classification.

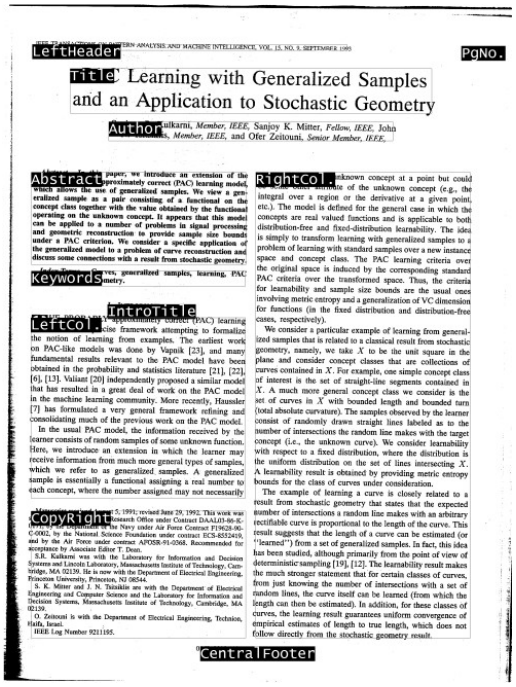
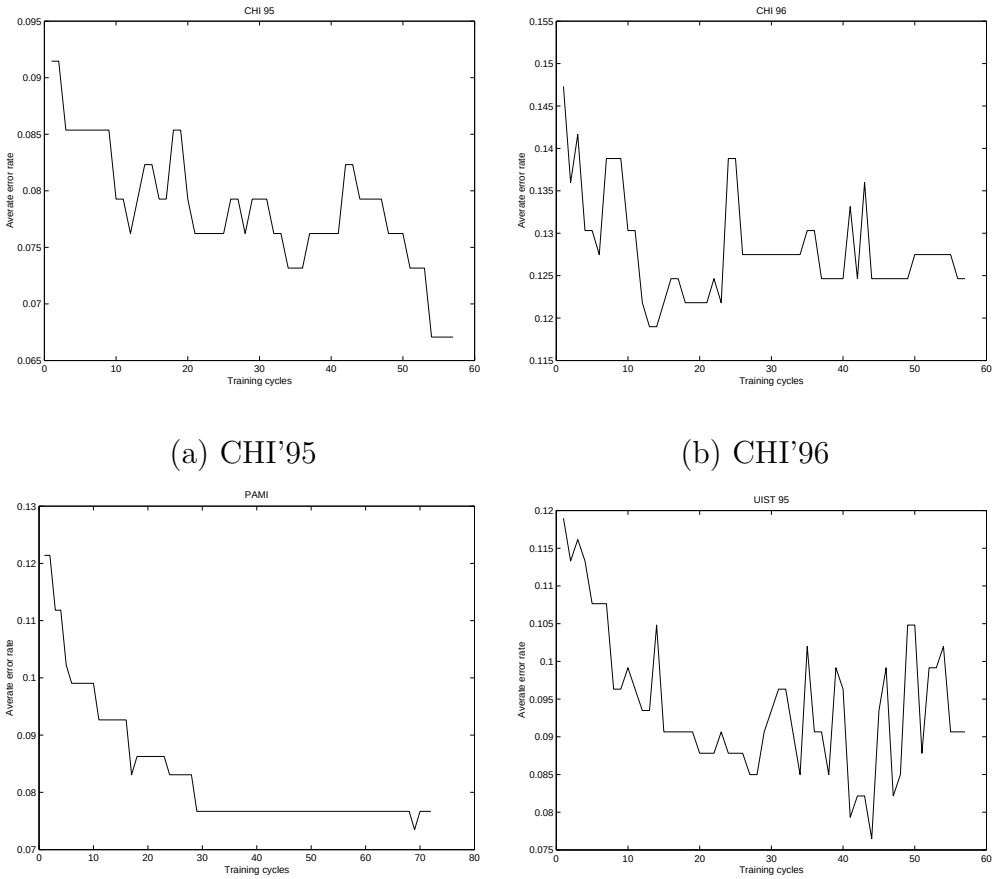


Figure 5.9: Labeling result for PAMI model

5.6.2 Business letters

In the second set of experiments, we applied our method on business letters. Compared to title pages, business letters have less consistent layout (see Figure 5.13). Therefore geometric features are not enough to describe the business letter model.



(a) CHI'95

(b) CHI'96

(c) PAMI

(d) UIST'95

Figure 5.10: Average error rates vs. training cycles for four models

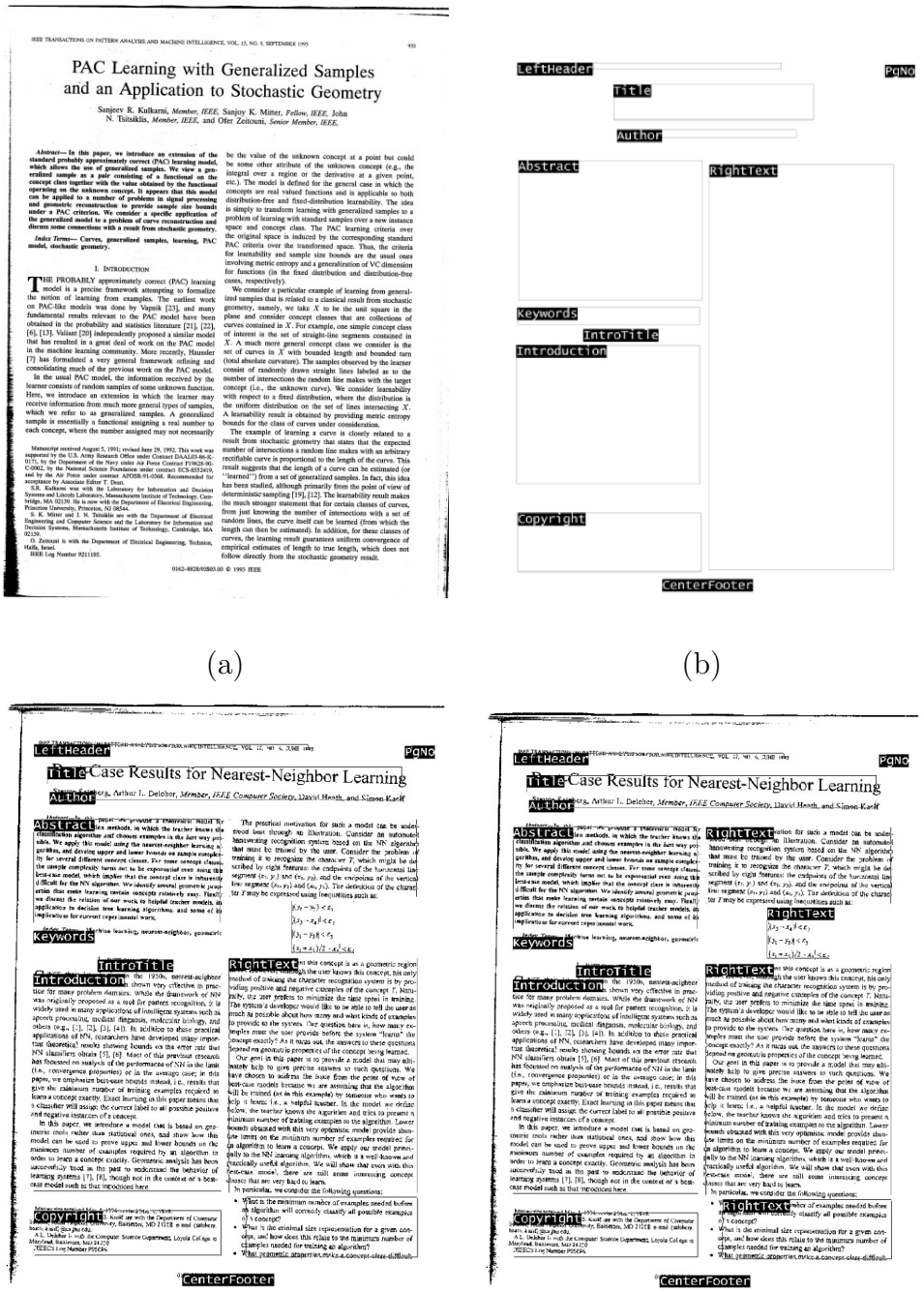
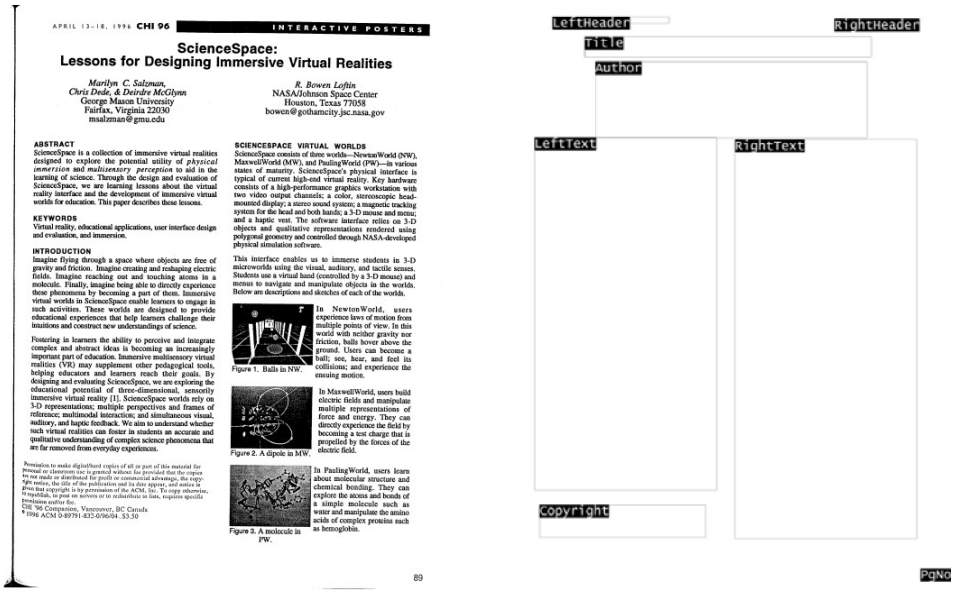
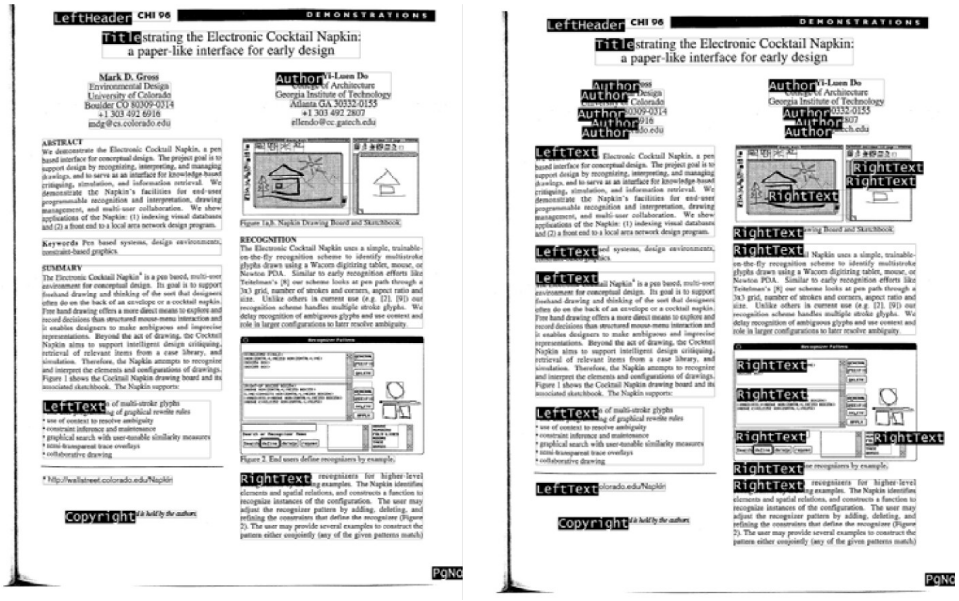


Figure 5.11: Examples of labeling results for title pages (part one). (a) A PAMI document. (b) Visualization of PAMI document model. (c) Labeling result of the first step matching. (d) Final labeling result.



(a) (b)



(c) (d)

Figure 5.12: Examples of labeling results for title pages (part two). (a) A CHI'95 document. (b) Visualization of CHI'95 document model. (c) Labeling result of the first step matching. (d) Final labeling result.

The lack of consistent geometric rules is compensated, on the other hand, by the consistent language features in business letters. For example, a *salutation* usually begins with "Dear Sir," "Dear Dr." or "To Whom It May Concern," and a *closing* usually is "Sincerely,", "Yours,", or "Truly yours". In the domain of English business letters we identify 10 logical components and 11 common content features that typically appear or do not appear in these components (see Table 5.2). The examples of the content features are shown in Table 5.1. As Table 5.2 suggests, the presence or absence of certain content features is strongly related to the logical component function.

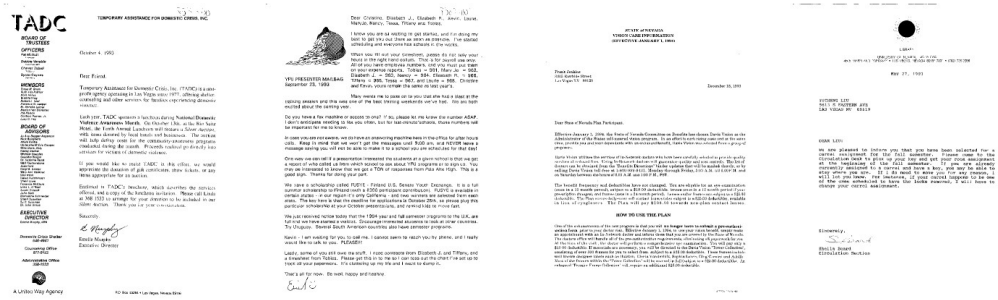


Figure 5.13: Samples of business letters

During the construction of a layout graph instance, each zone in the document is examined to determine the presence or absence of content features. A value $p_z^c \in [0,1]$ is defined to indicate the confidence of the presence of content feature c in zone z. In the model, for each node n we define an expectation value $e_n^c \in [0,1]$ for every content feature c, indicating to which degree we expect this logical component n to contain c. For example, $e_{SALUTATION}^{name}$ should be high, while $e_{SALUTATION}^{date}$ should be very low. We define the cost of labeling zone z as n based on the content feature c to be:

Content Feature	**Example**
name	Dr., Ms, Mr., David, Joe, Bill
location	Street, Ave, PO Box, MD, VA, Apt
organization	United Nation, Medical Center
date	September, 2002
greeting	Dear Sir or Madam, To Whom It May Concern
closing	Sincerely, Sincerely yours, Yours sincerely, Truly, Truly yours, Yours truly, Regards, Best regards,
zipcode	MD 20742, 00555-9642
subject	RE:, Subject:
ps	PS, PS., P.S.
enclosure	Enclosure, Encl., Enc.
cc	cc:, CC:

Table 5.1: Common indicative content features in English business letters

Logical Component	component frequency	Number of occurrences in logical component										
		name	loc.	org.	date	grt.	cls.	zip.	sub.	ps	encl.	cc
RECEIVER	66	54	49	42	3	0	0	52	0	0	0	0
SUBJECT	10	5	0	6	0	0	0	0	7	0	0	0
DATE	85	1	1	0	82	0	0	0	0	0	0	0
SALUTATION	116	75	1	21	2	99	0	0	2	0	0	0
BODY	116	5	3	44	103	55	2	2	2	1	0	0
SIGNATURE	116	109	8	63	1	0	90	0	0	0	0	0
ENCLOSURE	27	3	0	2	2	0	0	0	0	0	24	0
TYPISTINIT	27	0	0	8	0	0	0	0	0	0	0	0
PS	21	2	1	12	4	0	0	0	0	19	0	0
CC	6	6	1	1	0	0	0	0	0	0	0	6
Total	—	310	122	270	140	101	92	60	10	19	24	6

Table 5.2: Frequencies of content features in business letter samples

$$C(z, n, c) = (1 - e_n^c)p_z^c + e_n^c(1 - p_z^c). \tag{5.3}$$

The cost C as a function of e and p is shown in Figure 5.14. When $e < 0.5$, C decreases as p decreases; if $e > 0.5$, C decreases as p increases; when $e = 0.5$, C is independent of p.

Initially, the value of e_n^c is computed from the frequency of c appearing in component n in the training set. During the adaptive learning process, suppose label n should be assigned to zone z but is assigned to zone z' incorrectly, then the expectation value regarding content feature c is adjusted as follows:

- Increase e_n^c if $p_z^c > p_{z'}^c$;
- Decrease e_n^c if $p_z^c < p_{z'}^c$;

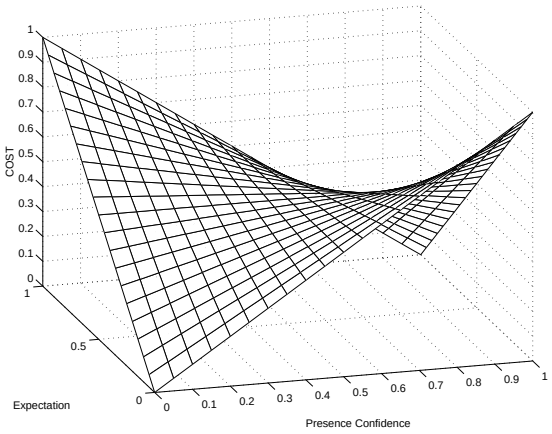


Figure 5.14: Cost as a function of expectation and presence confidence

As a result, $C(z,n,c) - C(z',n,c) < C'(z,n,c) - C'(z',n,c)$, where C' is the cost computed using the modified e_n^c value.

We collected 116 business letters, and tested our model using the same strategy as in the previous experiment. Figure 5.15 shows average error numbers vs. training cycles, and Figure 5.16 gives an example of labeling result. The total number of zones in 116 pages is 1389. The number of errors dropped ($\sim 60\%$) from about 500 to below 200 after 100 cycles of training, which shows the effectiveness of adaptive learning. Table 5.3 breaks down precision and recall for each logical component, where the components that are under-represented in the sample set (such as "CC," "SUBJECT") have worse numbers. Overall, the average precision and recall are both 87%, which is encouraging for a document class with very flexible structure.

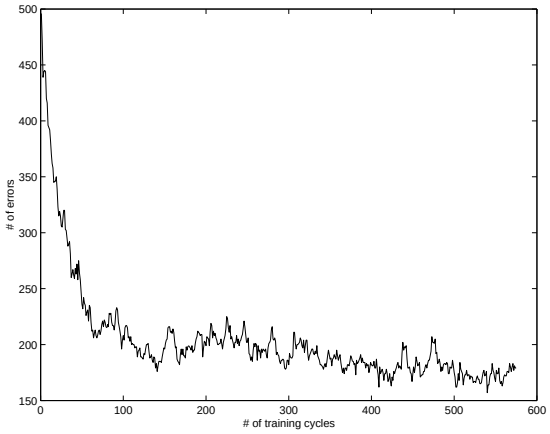


Figure 5.15: Average error rates vs. training cycles for business letter class

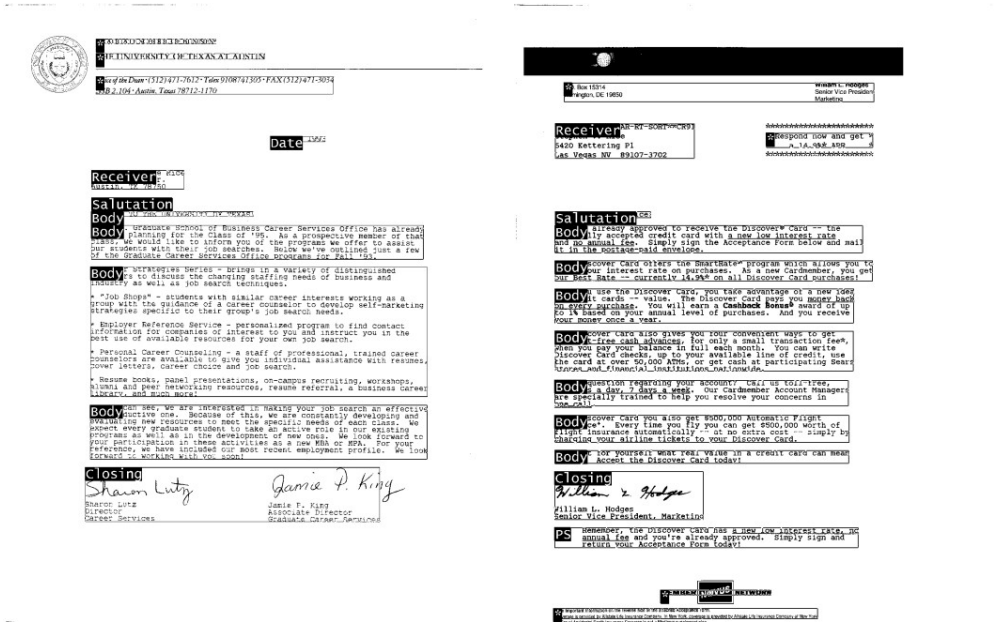


Figure 5.16: Examples of labeling results for business letters

Logical Component	Number of zones in 116 pages			Recall	Precision
	True	Found	Correct		
RECEIVER	68	66	60	88%	91%
SUBJECT	10	11	6	55%	60%
DATE	85	87	77	89%	91%
SALUTATION	116	116	108	93%	93%
BODY	484	457	438	96%	90%
SIGNATURE	197	198	162	82%	82%
ENCLOSURE	27	22	13	59%	48%
TYPISTINIT	28	27	15	56%	54%
PS	21	17	9	53%	43%
CC	6	5	0	0%	0%
(UNLABELED)	347	383	322	84%	93%
Overall	1389	1389	1210	87%	87%

Table 5.3: Precision and recall of different logical components

5.7 Discussion

At the beginning of this chapter, we analyzed several issues that a successful layout analysis system should consider:

1. The integration of local features and global structures.

2. The extensibility to new features.

3. The temporal change of a class.

4. The addition of a new class.

Our layout-structure-recognition approach satisfies all the above conditions. The graph-based model can integrate both local features (in nodes) and global structure (in edges) and include new features, if needed, in the future. The weighting factors associated with attributes give the model flexibility to handle large variation within a class. The recognition of layout structures achieves document classification and logical labeling simultaneously.

Furthermore, we provide an automatic model initialization and adaptive learning method. Our approach requires very few training samples to begin, and can improve the power of the model with user feedback afterward. Experiments show satisfactory results on both title pages of technical papers and business letters. In the literature, these two types of documents are usually processed separately, using different methods. Our approach provides a unified framework that works on both types, which demonstrates the power of our graph-based model.

Document classes can be organized in a hierarchical structure. Similarly, we could organize a tree-structured model base. For example, suppose we restrict ourselves to business letters and technical papers. At the top level of the model base, we would have two models, one for each type. Each top level model is the entry to a group of sub-models that describe specific classes in the genre. In our experiments on title pages, these sub-models would be the individual publication models. With such a model base, we could perform layout structure recognition in a coarse-to-fine, hierarchical way, which might improve both accuracy and speed.

Chapter 6

Summary and Conclusions

6.1 Summary of contributions

The following summarizes the contributions of this dissertation:

1. We developed a novel framework for geometric rectification of camera-captured document images.
    - We presented a planar-strip approximation model for curved document pages, based on developable surface properties.
    - We implemented a method for estimating page shape from a single document image, without requiring 3D data, camera calibration, or special camera pose.
    - We developed a unified solution for images of both planar and curved pages.
2. We designed a novel method for mosaicing camera-captured document images.
    - We developed an accurate image registration method for document images with large displacements, small overlapping areas, and severe projective distortions, without camera calibration or pose requirement.

- We implemented a selective image composition method that preserves text sharpness and eliminates 'ghost' images.

3. We developed a model matching approach for document layout analysis.

   - We designed a graph-based representation for modeling different layout styles, which incorporates heterogeneous features, both local and global.
   - We presented an model learning method that requires minimum initial training for a new document class and adapts a model to the changes of a class over the time.
   - We applied our layout structure recognition technique to both document image classification and logical component labeling.

6.2 Limitations and future work

In the future, we would like to address the following limitations as well as promising extensions of our work:

- Our text identification module needs improvement. Much research is going on in this area and we can adopt the successful techniques. In practice, a convenient GUI may be necessary to let users guide the text identification process.
- Texture flow estimation can benefit directly from the increase in image resolution, especially for minor texture flow. Although the increase in image size will have a negative effect on processing speed, we believe that the impact

can be minimized with rapid advances in hardware. For the same reason, we believe sufficient resources will be available on mobile platforms to implement our algorithms that currently run on desktop computers.

- Our document shape estimation method does not require 3D range data, metric data, or camera calibration. In some cases, such data may be available (at low accuracy, perhaps). For example, cameras equipped with IR sensors could generate a low resolution, low accuracy depth map along with the image; today's digital cameras can write the focal length of their auto-focus lens in the image files. We would like to incorporate such information, when available, into our framework.

- One of the three basic assumptions we make requires the principal point be at the image center. For applications running in a camera or in devices equipped with cameras, this holds generally true. This may not hold true, however, for processed images, such as those cropped from other images. In such cases, we need to estimate the principal point offset.

- The precision of geometric rectification for curved documents is relatively low compared to planar pages. Shape-from-multi-view provides a possible way to improve the accuracy. The correspondence problem is the key issue for shape estimation from multiple views. Enlightened by our work on document mosaicing, we propose to solve the correspondence problem in the following way: 1) estimate the shape in each view independently, 2) rectify the images, 3) register them using our image registration method, 4) fine-tune matches

locally, 5) map the matches back to the original images, 6) solve for the 3D shape using correspondences in multiple views.

- Three interesting questions exist in regard to our model-based layout structure recognition method. First, we would like to organize the models in a hierarchical tree structure and perform recognition in a hierarchical way. Second, the automatic maintenance of a hierarchical model base presents an interesting issue. This would involve detection of new document classes and insertion, deletion, modification, merging or splitting of nodes in the model tree. Third, our current framework aims at layout structures of single pages. The next step should expand to the processing of multiple pages.

6.3 Conclusion

For many decades, character recognition (OCR) has been the main issue for document image analysis. And scanners — especially flat-bed types — have been the main, if not only, imaging devices. With OCR considered a solved problem for clean printed text in major languages, new directions of document analysis appear at the horizon. Our work in this dissertation addresses two issues: 1) extending the imaging devices from scanners to digital cameras, and 2) extending the analysis result from text to metadata.

With the introduction of inexpensive and high-quality digital cameras, we predict a new trend of augmenting scanners with cameras. A great variety of new applications (especially on mobile devices) have been created, and many applications

once dominated by scanners possess a new life.

Two topics of our work cover the gap between current scanner-oriented OCR techniques and camera-captured document images. Our geometric rectification framework removes the distortion caused by non-planar document shape and camera perspective projection. Our document mosaicing technique provides a solution to uneven lighting, blurring, low resolution, and small field of view. Together, they provide a method to transform cameras into scanners, in the sense that the output images can be handled by current scanner-oriented OCR techniques. We expect them to be useful in all kinds of camera-based OCR applications, especially mobile devices.

Another topic addressed in this dissertation is related to the layout structure of a document page, which is lost in OCR text, to the function of the document and the functions of components in the page. Through the recognition of layout structures, we accomplish document classification and logical component labeling simultaneously. This enables us to extract semantic metadata without OCR text — if available, text can reinforce this capability. Metadata can greatly enhance the management of digitized documents.

Appendix A

Synthetic image generation for curved documents

Our synthetic curved document image generator takes a flat document image and a set of parameters as input and produces an image of a curved document, as well as ground truth data about the texture flows, surface rulings, and surface normals. Although there are computer graphics packages that can generate such images, we find it necessary to build our own generator because of our need for ground truth data.

In the first step, the input image is placed in the 2D plane with its center at the origin. Then, the image is rotated around its center by a *skew* angle ϑ within the 2D plane. After skewing, the parts that are outside the original rectangular frame are cropped (see Figure A.1).

The second step involves warping the 2D plane to a 3D surface. We use a cylinder surface model since it is the most usual case for opened books. The y-axis of the 2D plane is mapped to a generatrix of the cylinder (which is parallel to the Y-axis), while the x-axis is warped to be a directrix (see Figure A.2). The directrix is defined by a *polynomial curve* $S = (s_0, s_1, \ldots, s_n)$ on the Z-X plane, which is $Z(X) = \sum_{i=0}^{n} s_i X^i$. If $S = 0$, the directrix is a straight line, and the cylinder degrades to a plane. A point (x, y) in the 2D plane and its counterpart point (X, Y, Z) in the 3D world are related by

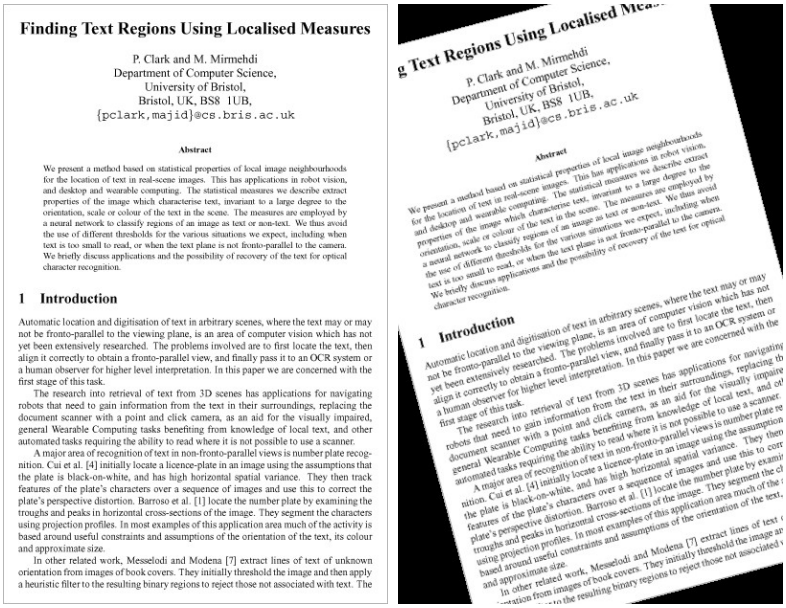


Figure A.1: An original image and the skewed image.

$$\begin{cases} y &= Y, \\ x &= |\int_{r=0}^{X} \sqrt{1 + (\frac{dZ}{dX}\Big|_{X=r})^2} dr| \\ &= |\int_{r=0}^{X} \sqrt{1 + (\sum_{i=1}^{n} i s_i r^{i-1})^2} dr|, \\ Z &= \sum_{i=0}^{n} s_i X^i. \end{cases}$$

Given (x, y), Y is fixed. We solve for $X(x)$ by interpolating a reverse lookup table computed from $x(X)$. Then, Z is easy to obtain.

In our model, the tangent vector $\mathbf{U}$ of the directrix at a surface point (X, Y, Z) is given by

$$\mathbf{U} = (1, 0, \frac{dZ}{dX})^\top = (1, 0, \sum_{i=1}^{n} i s_i X^{i-1})^\top.$$

The generatrix vector $\mathbf{V}$, which is also the 3D ruling vector $\mathbf{R}$, is equal to the

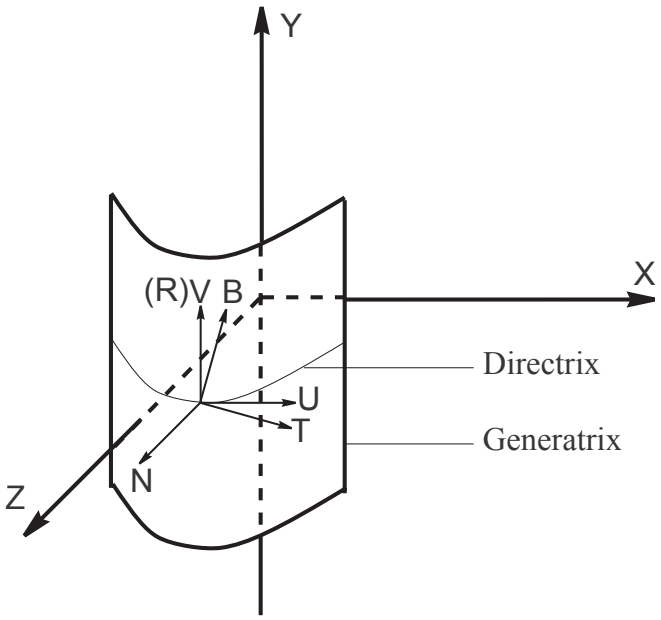


Figure A.2: Cylinder model for curved documents.

Y-axis vector, i.e.,

$$\mathbf{V} = \mathbf{R} = (0, 1, 0)^{\top}.$$

And the surface normal $\mathbf{N}$ at this point is simply the cross product of $\mathbf{U}$ and $\mathbf{V}$, i.e.,

$$\mathbf{N} = \mathbf{U} \times \mathbf{V}.$$

The 3D major texture flow vector $\mathbf{T}$ at this point can be computed by rotating $\mathbf{U}$ around $\mathbf{N}$ by the skew angle ϑ, i.e,

$$\mathbf{T} = \mathbf{U} \cos \vartheta + \mathbf{V} \sin \vartheta.$$

The 3D minor texture flow vector $\mathbf{B}$ is similarly calculated with a rotation angle $\vartheta + \pi/2$, i.e.,

$$\mathbf{B} = -\mathbf{U}\sin\vartheta + \mathbf{V}\cos\vartheta.$$

Equations A and A show that the skew ϑ controls the angle between the major texture flow and cylinder directrix as well the angle between the minor texture flow vector and cylinder generatrix (or ruling). For typical opened books, $\vartheta = 0$.

A set of three angles define the *rotation matrix* $\mathbf{R}$, and a 3D vector defines the *translation* $\mathbf{C}$. After rotation and translation, the 3D point $(X, Y, Z)^\top$ becomes $(X', Y', Z')^\top$ in the camera coordinate system. Accordingly, the surface normals, 3D texture flows, and 3D ruling directions are also rotated. Finally, the 3D point is projected onto the image plane through the pin-hole camera described by the *focal length* f. That is

$$\begin{cases} p &= \frac{fX'}{Z'}, \\ q &= \frac{fY'}{Z'}. \end{cases}$$

where (p, q) is the position on image plane (see Figure A.3).

To compute the 2D major texture flow $\mathbf{t}$ at point (p, q), we notice that $\mathbf{t}$ and its 3D counterpart $\mathbf{T}$ should be coplanar. This common plane is the cross product of $\mathbf{T}$ and the line-of-sight through the optical center and $(X', Y', Z')^\top$ or $(p, q, f)^\top$. Also, $\mathbf{t}$ lies on the image plane whose normal is $(0, 0, 1)^\top$. Thus

$$\mathbf{t} = (0, 0, 1)^\top \times (\mathbf{T} \times (p, q, f)^\top). \tag{A.1}$$

The 2D minor texture flow $\mathbf{b}$ and the 2D ruling $\mathbf{r}$ can be calculated in the same way.

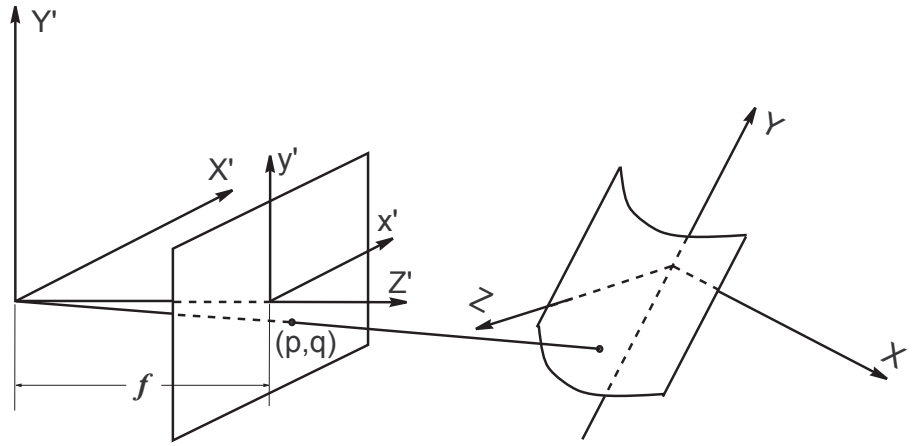


Figure A.3: A point on the curved page is projected onto image plane after rotation and translation.

The last set of parameters describe the light cast on the page. There could be any number of light sources, each characterized by a radiance R and a unit 3D vector $\mathbf{D}$ indicating the light direction. Under the Lambertian assumption, the brightness b' of a point in the final image is

$$b' = b \sum_{k=1}^{K} R_k \mathbf{D}_k^\top \mathbf{N},$$

where K is the number of light sources, b is the BRDF (bidirectional reflectance distribution function) value at the point in original flat document equal to its pixel value, and $\mathbf{N}$ is the unit surface normal at the point in 3D surface.

Above describes the generative process that maps a point in the original document image to the final synthetic image. However, in order not to leave any pixel in the synthetic image unfilled, we need a reverse process that maps every point in the synthetic image back to the original image. A precise reverse mapping is more

difficult to compute than a forward process. In computer graphics, this is usually solved by triangulating the surface and mapping each triangle as a whole piece. In our context, because the surface is developable, we can rectangulate the surface by rulings, i.e., approximate the surface by a group of planar strips (Figure A.4), and map each strip as a whole piece into the synthetic image. The specific procedure is as follows:

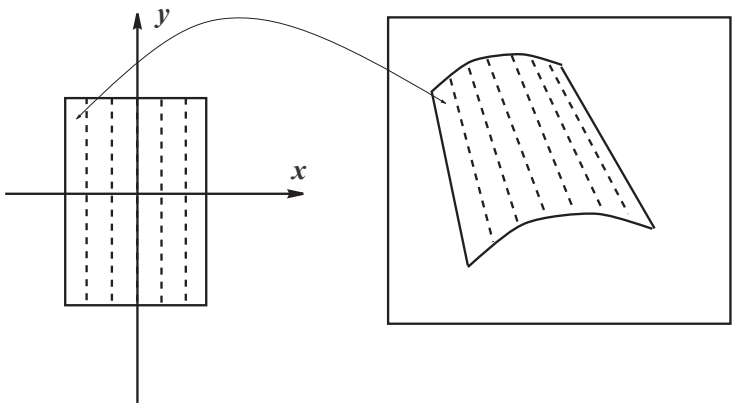


Figure A.4: Planar-strip approximation in synthetic image generation.

We divide the skewed image into narrow vertical slices. For each slice, we use the generative process to compute the 3D position $\mathbf{W}$ of its top-left corner after warping, rotation, and translation. We compute the surface normal $\mathbf{N}$ and two 3D texture flows ($\mathbf{T}$ and $\mathbf{B}$) at the 3D point. We also compute the position of its projection in the image as (x_0, y_0). Substituting $\overline{\mathbf{V}}_h$ by $\mathbf{T}$ and $\overline{\mathbf{V}}_v$ by $\mathbf{B}$ in Equation 3.4, and plugging in $\mathbf{W}$, (x_0, y_0), and f (for $\mathbf{K}$), we can compute a homogeneous transformation $\mathbf{H}$ that maps every point in the slice to the final image, and $\mathbf{H}^{-1}$ for the reverse mapping. Furthermore, to take the skew process into

account, the complete reverse mapping can be written as $\widetilde{\mathbf{H}} = \mathbf{S}^{-1}\mathbf{H}^{-1}$, where

$$\mathbf{S} = \begin{pmatrix} \cos\vartheta & \sin\vartheta & 0 \\ -\sin\vartheta & \cos\vartheta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

is the skew matrix. Let $\mathbf{H}$ map the four corners of the slice to the four vertices of a quadrilateral in the synthetic image, then, for every pixel inside this quadrilateral, we can use $\widetilde{\mathbf{H}}$ to map it back to the original image. Its color is computed using bilinear interpolation of four surrounding pixels. If it is mapped to the outside of the image frame, we set it to a pre-defined background color.

Under the planar-strip approximation, the 3D surface normal for any point inside the quadrilateral is simply $\mathbf{N}$. The 2D texture flows at point (p, q) can be computed from $\mathbf{T}$ and $\mathbf{B}$, using Equation A.1, since the 3D texture flows are constant within the quadrilateral.

As a summary, for any pixel in the synthetic image, we can compute its color, the 3D surface normal, 3D ruling direction, 3D major and minor texture flows, 2D ruling direction, and 2D major and minor texture flows at this point.

Appendix B

Selection of reference points for rulings

We first compute the center of mass for the text area. Let it be C. We estimate the optimal 2D ruling through C as r using major texture flow field. This ruling may not be very accurate, which is fine as long as the direction remains roughly correct. Let s be the line through C and perpendicular to r. We select reference points along s and within the text area. We also select two more reference points on s *just* outside the text area, as Figure B.1 shows. In this way, the 2D rulings through the reference points can cover the entire text area. Ideally, we would like to have denser 2D rulings in areas where the surface curvature is high and fewer rulings in low curvature areas. However, in our implementation, we select reference point at equal distances for the sake of simplicity. We find that this simple scheme works sufficiently well. The distance controls the number of 2D rulings we need to estimate, which, in turn, controls the computation speed and shape approximation accuracy.

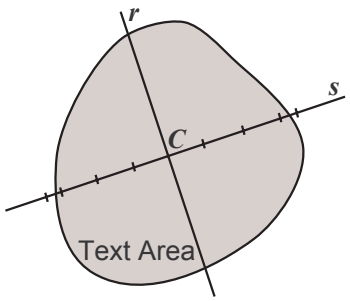


Figure B.1: Selection of reference points for rulings.

BIBLIOGRAPHY

[1] A. Amin and F. S. A document skew detection method using the hough transform. *Pattern Analysis and Applications*, 3(3):243–253, 2000.

[2] S. Baker and T. Kanade. Limits on super-resolution and how to break them. *IEEE Transactions on Pattern Analysis and Machine Intellegence*, 24(9):1167–1183, 2002.

[3] O. Ben-Shahar and S. W. Zucker. The perceptual organization of texture flow: A contextual inference approach. *IEEE Transactions on Pattern Analysis and Machine Intellegence*, 25(4):401–417, April 2003.

[4] M. S. Brown and C. J. Pisula. Conformal deskewing of non-planar documents. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 998–1004, June 2005.

[5] M. S. Brown and W. B. Seales. Image restoration of arbitrarily warped documents. *IEEE Transactions on Pattern Analysis and Machine Intellegence*, 26(10):1295–1306, October 2004.

[6] P. J. Burt and E. H. Adelson. A multiresolution spline with application to image mosaics. *ACM Transactions on Graphics*, 2(4):217–236, 1983.

[7] H. Cao, X. Ding, and C. Liu. Rectifying the bound document image captured by the camera: A model based approach. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 71–75, 2003.

[8] D. Capel and A. Zisserman. Super-resolution enhancement of text image sequences. In *Proceedings of the International Conference on Pattern Recognition*, volume 1, pages 600–605, 2000.

[9] A. Chaudhuri and S. Chaudhuri. Robust detection of skew in document images. *IEEE Transactions on Image Processing*, 6(2):344–349, 1997.

[10] G. Ciadiella, G. Scafuro, M. T. Degrandi, M. R. Spada, and M. P. Roccotelli. An experimental system for office document handling and text recognition. In *Proceedings of the International Conference on Pattern Recognition*, pages 739–743, 1988.

[11] P. Clark and M. Mirmehdi. Finding text regions using localised measures. In *Proceedings of the British Machine Vision Conference*, pages 675–684, 2000.

[12] P. Clark and M. Mirmehdi. Location and recovery of text on oriented surfaces. In *Proceedings of SPIE Document Recognition and Retrieval VII*, pages 267–277, 2000.

[13] P. Clark and M. Mirmehdi. Estimating the orientation and recovery of text planes in a single image. In *Proceedings of the British Machine Vision Conference*, pages 421–430, 2001.

[14] P. Clark and M. Mirmehdi. On the recovery of oriented documents from single images. In *Proceedings of Advanced Concepts for Intelligent Vision Systems*, pages 190–197, 2002.

[15] T. Coleman and Y. Li. On the convergence of reflective Newton methods for large-scale nonlinear minimization subject to bounds. *Mathematical Programming*, 67(2):189–224, 1994.

[16] T. Coleman and Y. Li. An interior, trust region approach for nonlinear minimization subject to bounds. 1996.

[17] A. Dengel, R. Bleisinger, R. hoch, F. Fein, and F. Hones. From paper to office document standard representation. *Computer*, 25(7):63–67, 1992.

[18] M. Diligenti, P. Frasconi, and M. Gori. Hidden tree Markov models for document image classification. *IEEE Transactions on pattern analysis and machine intelligence*, 25(4):519–523, 2003.

[19] D. S. Doermann, J. Liang, and H. Li. Progress in camera-based document image analysis. In *Proceedings of the Internatioanl Conference on Document Analysis and Recognition*, volume 1, pages 606–616, Aug 2003.

[20] V. Eglin and A. Gagneux. Visual exploration and functional document labeling. In *Proceedings Of The Sixth International Conference On Document Analysis And Recognition*, pages 816–820, 2001.

[21] M. Elad and A. Feuer. Restoration of a single superresolution image from several blurred, noisy, and undersampled measured images. *IEEE Transactions on Image Processing*, 6(12):1646–1658, 1997.

[22] A. Goetz. *Introduction to Differential Geometry.* Addison Wesley Publishing Company, 1970.

[23] S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *IEEE Transactions on Pattern Analysis and Machine Intellegence*, 18(4):377–388, April 1996.

[24] N. Gumerov, A. Zandifar, R. Duraiswarni, and L. S. Davis. Structure of applicable surfaces from single views. In *Proceedings of European Conference on Computer Vision*, pages 482–496, 2004.

[25] R. M. Haralik. Document image understanding: Geometric and logical layout. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 385–390, 1994.

[26] R. Hartley and A. Zisserman. *Multiple view Geometry in Computer Vision.* Cambridge University Press, 2000.

[27] B. K. P. Horn. *Robot Vision.* The MIT Press, 1986.

[28] J. Hu, R. Kashi, and G. Wilfong. Document image layout comparison and classification. In *Proceedings of the Internatioanl Conference on Document Analysis and Recognition*, pages 285–289, 1999.

[29] R. A. Hummel and S. W. Zucker. On the foundations of the relaxation labeling proceeses. *IEEE Transactions on Pattern Analysis and Machine Intellegence*, 5:267–287, 1983.

[30] R. A. Hummel and S. W. Zucker. On the foundations of the relaxation labeling processes. *IEEE Transactions on Pattern Analysis and Machine Intellegence*, 5:267–287, 1983.

[31] F. Isgrò and M. Pilu. A fast and robust image registration method based on an early consensus paradigm. *Pattern Recognition Letters*, 25(8):943–954, 2004.

[32] A. K. Jain and B. Yu. Document representation and its application to page decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):294–308, 1998.

[33] H.-F. Jiang, C.-C. Han, and K.-C. Fan. A fast approach to the detection and correction of skew documents. *Pattern Recognition Letters*, 18:675–686, 1997.

[34] Y. Ke and R. Sukthankar. PCA-SIFT: a more distinctive representation for local image descriptors. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, volume 2, pages 506–513, 2004.

[35] K. Kise and D. Doermann, editors. *First International Workshop on Camera-Based Document Analysis and Recognition*, Aug. 2005.

[36] D. C. Knill. Contour into texture: Information content of surface contours and texture flow. *Journal of the Optical Society of America Association*, 18(1):12–35, Jan 2001.

[37] D. X. Le, G. Thoma, and H. Weschler. Automated page orientation and skew angle detection for binary document images. *Pattern Recognition*, 27(10):1325–1344, 1994.

[38] J. Liang, D. Doermann, and H. Li. Camera-based analysis of text and documents: a survey. *International Journal on Document Analysis and Recognition*, 7(2+3):84–104, July 2005.

[39] J. Liang and D. S. Doermann. Logical labeling of document images using layout graph matching with adaptive learning. In *International Workshop on Document Analysis Systems*, pages 224–235, 2002.

[40] J. Liang, D. S. Doermann, M. Ma, and J. K. Guo. Page classification through logical labelling. In *Proceedings of the International Conference on Pattern Recognition*, pages 477–480, 2002.

[41] D. Liebowitz and A. Zisserman. Metric rectification for perspective images of planes. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 482–488, 1998.

[42] R. Lienhart and A. Wernicle. Localizing and segmenting text in images and videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(4):256–268, 2002.

[43] N. Liolios, N. Fakotakis, and G. Kokkinakis. Improved document skew detection based on text line connected-component clustering. In *Proceedings of IEEE International Conference on Image Processing*, volume 1, pages 1098–1101, 2001.

[44] M. Lipshutz and S. L. Taylor. Functional decomposition of business letters. In *The Fourth Symposium On Document Analysis And Information Retrieval*, pages 435–447, 1995.

[45] S. Mann and R. W. Picard. Video orbits of the projective group: A simple approach to featureless estimation of parameters. *IEEE Transactions on Image Processing*, 6(9):1281–1295, 1997.

[46] M. Mirmehdi, P. Clark, and J. Lam. Extracting low resolution text with an active camera for OCR. In *Proceedings of the IX Spanish Symposium on Pattern Recognition and Im age Processing*, pages 43–48, May 2001.

[47] G. K. Myers, R. C. Bolles, Q.-T. Luong, J. A. Herson, and H. B. Aradhye. Rectification and recognition of text in 3-D scenes. *International Journal on Document Analysis and Recognition*, 7(2+3):147–158, July 2005.

[48] D. Nagy. A prototype document image analysis system for technical journals. *Computer*, 25(7):10–22, 1992.

[49] G. Nagy, S. Seth, and M. Viswanathan. A prototype document image analysis system for technical journals. *Computer*, 25(7):10–22, 1992.

[50] T. Nakao, A. Kashitani, and A. Kaneyoshi. Scanning a document with a small camera attached to a mouse. In *Proc. WACV'98*, pages 63–68, 1998.

[51] D. Niyogi and S. N. Srihari. An integrated approach to document decomposition and structural analysis. *International Journal Of Imaging Systems And Technology*, 7:330–342, 1996.

[52] L. O'Gorman. The document spectrum for page layout analysis. *IEEE Transactions on Pattern Analysis and Machine Intellegence*, 15(11):1162–1173, Nov. 1993.

[53] J. H. Park, I. H. Jang, and N. C. Kim. Skew correction of business card images acquired in pda. *IEE Proceedings: Vision, Image and Signal Processing*, 152(6):668–676, 2005.

[54] A. J. Patti, I. Sezan, and A. M. Tekalp. Superresolution video reconstruction with arbitrary sampling lattices and nonzero aperture time. *IEEE Transactions on Image Processing*, 6(8):1064–1076, 1997.

[55] S. Peleg and J. Herman. Panoramic mosaics by manifold projection. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 338–343, 1997.

[56] M. Pilu. Extraction of illusory linear clues in perspectively skewed documents. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, volume 1, pages 363–368, 2001.

[57] M. Pilu. Undoing paper curl distortion using applicable surfaces. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, volume 1, pages 67–72, 2001.

[58] S. Pollard and M. Pilu. Building cameras for capturing documents. *International Journal on Document Analysis and Recognition*, 7(2+3):123–137, July 2005.

[59] A. R. Rao and R. C. Jain. Computerized flow field analysis: Oriented texture fields. *IEEE Transactions on Pattern Analysis and Machine Intellegence*, 14(7):693–709, July 2003.

[60] B. S. Reddy and B. N. Chatterji. An FFT-based technique for translation, rotation, and scale-invariant image registration. *IEEE Transactions on Image Processing*, 5(8):1266–1271, 1996.

[61] K. Schutte and A. M. Vossepoel. Accurate mosaicking of scanned maps, or how to generate a virtual a0 scanner. In *Proc. ASCI'95*, pages 353–359, 1995.

[62] J. G. Semple and G. T. Kneebone. *Algebraic Projective Geometry*. Oxford University Press, 1979.

[63] C. Shin, D. Doermann, and A. Rosenfeld. Classification of document pages using structure-based features. *International Journal on Document Analysis and Recognition*, 3(4):232–247, 2001.

[64] R. Szeliski. Video mosaics for virtual environment. *IEEE Computer Graphics and Applications*, 16(2):22–30, 1996.

[65] M. J. Taylor, A. Zappala, W. M. Newman, and C. R. Dance. Documents through cameras. *Image and Vision Computing*, 17(11):831–844, 1999.

[66] Ø. D. Trier and T. Taxt. Evaluation of binarization methods for document images. *IEEE Transactions on Pattern Analysis and Machine Intellegence*, 17(3):312–315, 1995.

[67] Y.-C. Tsoi and M. S. Brown. Geometric and shading correction for images of printed materials a unified approach using boundary. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 240–246, 2004.

[68] A. Ulges, C. H. Lampert, and T. Breuel. Document capture using stereo vision. In *Proceedings of the 2004 ACM Symposium on Document Engineering*, pages 198–200, 2004.

[69] A. Ulges, C. H. Lampert, and T. M. Breuel. Document image dewarping using robust estimation of curled text lines. In *Proceedings of the Internatioanl Conference on Document Analysis and Recognition*, pages 1001–1005, 2005.

[70] A. Vailaya, H. Zhang, C. Yang, F.-I. Liu, and A. Jain. Automatic image orientation detection. *IEEE Transactions on Image Processing*, 11(7):746–755, 2002.

[71] H. Walischewske. Learning regions of interest in postal automation. In *Proceedings Of The Fifth International Conference On Document Analysis And Recognition*, pages 317–340, 1999.

[72] A. P. Whichello and H. Yan. Document image mosaicing. In *Proceedings of the International Conference on Pattern Recognition*, pages 1081–1083, 1998.

[73] C. Wu and G. Agam. Document image de-warping for text/graphics recognition. In *SPR 2002, Int. Workshopon Statistical and Structural Pattern Recognition*, volume 2396 of *Lecture Notes in Computer Science*, pages 348–357, 2002.

[74] H. Yan. Skew correction of document images using interline cross-correlation. *Computer Vision: Graphical Models and Image Processing*, 55(6):538–543, 1993.

[75] A. Zandifar, R. Duraiswami, and L. S. David. A video-based framework for the analysis of presentations/posters. *International Journal on Document Analysis and Recognition*, 7(2+3):178–187, July 2005.

[76] A. Zappala, A. Gee, and M. J. Taylor. Document mosaicing. *Image and Vision Computing*, 17(8):585–595, 1999.

[77] L. Zhang, Z. Zhang, C. L. Tan, and T. Xia. 3D geometric and optical modeling of warped document images from scanners. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 337–342, 2005.

[78] Z. Zhang and C. L. Tan. Restoration of images scanned from thick bound documents. In *Proceedings of IEEE International Conference on Image Processing*, pages 1074–1077, 2001.

[79] Z. Zhang and C. L. Tan. Correcting document image warping based on regression of curved text lines. In *Proceedings of the International Conference on Document Analysis and Recognition*, volume 1, pages 589–593, 2003.

[80] G. Zi. Groundtruth generation and document image degradation. Technical Report LAMP-TR-121/CAR-TR-1008/CS-TR-4699/UMIACS-TR-2005-08, 2005.