A Morphological Image Preprocessing Suite for OCR on Natural Scene Images

Megan Elmore
Georgia Institute of Technology
melmore@cc.gatech.edu

Margaret Martonosi
Princeton University
mrm@princeton.edu

ABSTRACT

As demand grows for mobile applications, research in optical character recognition (OCR), a technology well-developed for document imaging, is shifting focus to the recognition of text embedded in digital photographs or video. Segmenting text and background in natural scenes is a difficult classification problem, and the accuracy of this segmentation is of utmost importance when the output of an OCR system will be transformed as in translation or speech synthesis. Our work proposes an image preprocessing suite that, through text detection, auto-rotation, and noise reduction, improves the accuracy of OCR analysis in a camera-based translation system. Our novel approaches for foreground/background detection and skew estimation using morphological edge analysis show immediate improvement in OCR accuracy, and our experimental classification of text regions using unsupervised feature-based clustering provides a good background for future research in applying signal processing and machine learning techniques to the problem of text detection.

1. INTRODUCTION

Optical character recognition, or OCR, is a powerful tool for bringing information from our analog lives into our increasingly digital world. This technology has long seen use in document imaging; by scanning or photographing documents we may convert them into soft copies that can be edited, searched, reproduced and transported with ease. With digitally reproduced information at our fingertips, the world's business, research, and governmental spheres grow more productive.

As portable computing is becoming more accessible to the public, new and exciting applications of character and image recognition have started to emerge. Some modern mobile devices can use pictures of words or barcodes to search, download and purchase; currently-deployed applications for saving contact information from business cards may soon allow businesspeople to carry only one personalized card with no physical copies to share. Screen readers already voice digital text to aid the blind, and future OCR-enhanced applications embedded in mobile devices may allow blind persons to "see" the wider world untethered.

A particularly interesting application of OCR is to combine character recognition with natural language translation. With these two technologies intertwined and deployed for mobile devices, the language barrier faced by tourists and immigrants lowers; visitors can take pictures of public signage and have the same access to information as locals. The major technical hurdle of such an application is ensuring that the OCR output is accurate enough to allow a useful translation — recognizing text from natural scenes and mobile camera images is much more difficult than analyzing clear document text. Environmental and camera conditions may reduce image clarity, and text must be segmented from a very complex rather than uniform background. As evidenced by the heavy focus on image processing concerns shown by prototype image translation systems ([8], [12], [28]) the challenge of adapting OCR to natural scene images must be accomplished before any accurate synthesis with translation or other transformative technologies is possible.

The goal of this research is to address with a preprocessing software suite the complications natural scenes imaged with a mobile device introduce to the OCR process. We first build on existing approaches for text detection, using a morphological edge-detection strategy to isolate connected components in an image that may be classified as text. To perform this classification, we cluster the extracted candidate regions based on their inherent size and color features, with the goal of partitioning the image into text and background elements. After identifying text, we detect and correct its angle of rotation using the skew of the edges detected in the text regions of the image. Finally we use a simple morphological approach to reduce noise from the rotation and conversion of text into black and white. It is our goal to develop a preprocessing system robust enough to allow us to build and deploy a portable, OCR-based translation system.

2. OUTLINE

In section 3, we discuss related work in image preprocessing for enhancing OCR output and in systems that combine OCR with natural language translation. Next, section 4 describes our prototype mobile application for translating image text. Section 5 contains implementation details for our image preprocessing suite. Section 6 outlines our experimental methods, and section 7 is a discussion of the results of our early experiments. Finally, section 8 summarizes the contributions of this paper and gives an overview of our future research goals.

3. RELATED WORK

OCR systems have been under development in research and industry since the 1950s (see [20] for a detailed history). Such systems use knowledge-based and statistical pattern recognition techniques to transform scanned or photographed images of text into machine-editable text files. There are several commercially-available OCR systems on the market today, such as ABBYY FineReader, OmniPage, and Microsoft Office Document Imaging. The research and open-source communities offer comparable systems like GOCR [1], Ocrad [4], and Tesseract [27] under free software licenses.

In these and similar systems, the OCR process has 5 main steps: preprocessing to improve the quality of the input image, binarization (to black and white), text segmentation, character recognition, and post-processing based on knowledge of the target language of the text. These steps are most effective when applied to document text, which when collected by a scanner is generally aligned and has clear contrast between text and its uniform background. However, using scene text, or text that appears in a complex background like a sign in a natural scene, as the input to an OCR system makes the preprocessing steps necessarily more complex. The imaging device, a portable camera, may cause scene text to be rotated, out of focus, or at a non-optimal perspective for legibility. At a more semantic level, scene text may have uneven lighting that makes contrast with the background less sharp and uniform, or there may be parts of the background that have similar properties to text after binarization. Liang, Doermann and Li [16] provide a detailed treatment of these and other challenges to camera-based document analysis; for the rest of this section we discuss existing work related to the preprocessing, binarization, and text segmentation challenges that are addressed by our proposed image preprocessing suite.

3.1 Binarization

Traditionally, OCR systems have a binarization step that classifies image pixels as text (black) or background (white). This rigid partitioning allows the text segmentation and character recognition steps to only analyze the regions of the image that most likely contain text. When preprocessing scene text, the binarization step is used both to isolate these candidate text regions for analysis and to provide the sharp contrast with anything considered as background needed by the later processing steps.

Binarization is commonly performed using thresholding, a technique to partition the colors in the image into two sets based on a histogram or other statistical measure. There are two major classes of thresholding algorithms: global, which decides on one partition for the entire image based the properties of the entire image, and local (or adaptive), which considers the properties within smaller regions to produce possibly differing partitions for each region. Otsu's method for grayscale images [22], a global thresholding technique prevalent in document recognition literature, uses a histogram to search for the partition that minimizes the variance in gray-levels within each set in the partition. Where local thresholding is more appropriate, it is common to use a technique similar to Niblack's method for grayscale images [21], which sets each region's threshold to be a fixed fraction of one standard deviation above or below the average gray level within the region. Experimentally, we found that binarizing the text selection of our image using Otsu's method more accurately preserved legibility.

3.2 Text Detection

When performing OCR on a natural scene image, the complexity of the background often precludes the separation of text from background based on thresholding information alone. Rather, binarization as a part of and or/after a process to detect regions that most likely contain text based on more complex properties. The survey of text information extraction by Jung, Kim and Jain [13] covers a wealth of approaches to text detection and localization; here, we discuss text detection techniques that rely upon the properties of color, edges and texture.

It is difficult to assess the relative merits of published approaches as there is not a common corpus of test images nor another established standard by which they each were tested. The survey paper by Jung et al. discusses some relative and qualitative merits of the approaches it covers, and Messelodi and Modena's paper on text detection [19] collects the results of the related work they cite in addition to providing an extensive discussion of properties used to distinguish text candidate regions in their and other connected component (sometimes written CC)-based approaches.

3.2.1 Color-based Text Detection and Heuristic Connected Component Filtering

Many text detection approaches first divide an image into contiguous regions, or connected components, of similar color and then analyze these regions for other properties characteristic of printed or handwritten text. [11] and [14] use color quantization to define connected components, while [19] extract connected components at the gray-level by creating three slices of the grayscale image using a head and a tail threshold from the histogram. Central to all of these works is a need for aggressive heuristic filtering to determine which connected components contain text. Some common classes of heuristics used across these works are color distribution within the component's region, character- or word-like size and dimensions, and location near other like components; our work explores the statistical importance of some common heuristics in a text candidate clustering scheme.

3.2.2 Edge-based Text Detection and Mathematical Morphology

A second class of text detection algorithms is edge-based algorithms, or those that use the positions at which their is a high level of contrast between neighboring pixels to determine the likely location of text. Many such algorithms isolate edges with the Canny edge filter [7], which can estimate horizontal, vertical, and diagonal edges by detecting local maxima across the image's intensity gradient in these directions.

Other edge detection approaches use the operators of mathematical morphology [25]. A morphological operation commonly used in edge detection is the morphological gradient, which is the difference between the dilation and erosion of an image by a connectivity-preserving structuring element. This operation results in a pixel-wise description of the contrast between the value of each pixel and its neighborhood corresponding to the structuring element.

One morphological approach that is prominent in text detection literature is that of Hasan and Karam [10]. They apply the morphological gradient to grayscale images blurred by morphological openings and closings, and strong edges are determined by preserving all pixels in the gradient image above a global gray-level threshold. Connected edges are then grouped by another closing and labeled by a fast forward-backward propagation algorithm, and the minimum bounding rectangles around pixels with the same label are analyzed as text candidate region. This text detection approach is robust to text orientation and skew but requires further exploration to determine its merit, as the authors provide only three qualitative examples of the algorithm's output. Our work uses this morphological edge-extraction approach, and we attempt to enhance the accuracy of extracting text by exploring more color, size, and spatial heuristics in combination with the extracted edge data.

3.2.3 Texture-based Text Detection

A final class of text detection algorithms have a texture-based approach. These algorithms use mathematical transforms and/or machine learning strategies to detect regions of an image whose pattern of pixels match what is considered a text-like texture, as opposed to flat regions of one color or other textures of the background. Some examples of texture-based detection schemes are Liu, Wang, and Dai's color texture approach using the Wavelet transform [18] and Shin et al.'s non-feature-based support vector machine (SVM) texture learning approach [26]. While these approaches show promise in distinguishing the nuanced difference between text and background in natural scene images, they are hampered by computational complexity and the difficulty in defining and standardizing the text-like pattern that these methods detect.

3.3 Additional Preprocessing

3.3.1 Auto-Rotation

When OCR input is taken from a hand-held camera or other imaging device whose perspective is not fixed like a scanner, text lines may skewed from their original orientation. OCR systems often use an auto-rotation algorithm to correct this skew before processing text further, but in natural scene images it is often necessary for the auto-rotation step to come after text detection and binarization so the original line orientation is estimated without considering any lines present in the complex background. Commercial and open-source OCR software also cannot often correct for large amounts of rotation, so extra auto-rotation processing may be helpful for camera-images black and white documents as well.

In traditional document analysis, text often falls into a rigid layout with set margins. An auto-rotation algorithm based on finding margins, such as that described in [6], falls easily out of this assumption of regularity. Their algorithm determines the angle of rotation from the sample set of horizontally-first text pixels encountered at regular vertical intervals. In their experiments, this algorithm shows a substantial improvement over the auto-rotation capabilities of the FineReader OCR software. However, this software is ill-suited for text images like those of road signs that lack margins.

A more robust approach to auto-rotation for use on natural scene images involves detecting the slope between connected components that make up a text line ([5], [19]). Such an approach does not depend on the location of lines relative to each other, but it does involve significant computational effort as neighboring components are merged until all lines are completely formed. We attempt to reduce this overhead by estimating slopes from morphological edge groupings, which inherently encompass the characters making up a word or line.

3.3.2 Noise Reduction

Noise is often a problem in camera-based text imaging; the input often has much lower resolution that that offered by scanners, and if the text is skewed, auto-rotation using the interpolation of the rotation transform alone introduces jagged edges. Though binarization with thresholding removes some static due to low resolution, it too can leave text edges jagged or of improper thickness because the contrast between text and background has not been captured sharply. To combat these problems, other noise reduction techniques, like blurring and the hit-miss transform, are often part of the preprocessing for natural scene OCR systems.

Blurring transforms like the Gaussian blur and morphological opening and closing are frequently used to reduce over-segmentation along edges. These transforms bring the color values of pixels closer to those of their neighbors, so they can help assure an averaged view of the contrast along regions of different color. They can also help relieve noise in the form of isolated pixels of different color from their surroundings, but this does result in an averaging of colors in the affected area into a blob of incorrect color rather than coloring the isolated pixels the same as their background.

The hit-miss transform is a morphological operation (see [25]) that can isolate specific patterns for preservation or removal. Using the hit-miss transform, our noise-reduction approach finds pixels whose neighborhoods match a known pattern for noise - perhaps isolated pixels of the wrong foreground/background type, or edges with sawteeth - and filters them from the target image.

3.4 Related Applications

One commercial OCR/translation application is Linguatec's Shoot and Translate. It can translate between six language pairs but can only be run on a supported high-end device. Statistics for recognition and translation accuracy are not available, though the product website shows correct translations of several signs with skew, perspective distortion, and improper lighting.

[8], [12], and [28] describe research efforts to produce OCR and translation applications. Like our current work, these papers show a strong early focus on accurate text detec-

tion. Using advanced machine-learning techniques for isolating and recognizing text, these papers report OCR accuracy from 80% to near 100%. Of particular interest for the future is the discussion in [28] of the difficulties in specializing such systems for translation of public signage; the abbreviations and grammar of such text seems to preclude effective translation by systems trained on formal documents.

4. APPLICATION OVERVIEW

The context of our image preprocessing suite is an application for mobile devices that allows users to translate text from camera images. This application is targeted to the pedestrian tourist who wants to understand informational or directional road signs in his or her native language. We describe here a prototype system which has been developed to test our image preprocessing techniques; it is a future goal to make this prototype deployable.

We have developed a prototype client program for the Android platform [2]. This client is responsible for allowing the user to take and review a photo with their device's built-in camera, to send the photo to a remote server for processing, and to receive and display the translation results. If users desires more narrowed results from the translation service - for example, they may wish to translate only one of several street signs captured in the image - or more accuracy given a noisy image, they may opt to select a smaller region of the captured image to send to the remote server.

In our prototype system, all image and text processing is currently done on a remote server. The first processing step is our image preprocessing suite, which aims to maximize the accuracy of text recognition so that the translation is conducted on the correct original text. Because an OCR system performs optimally when given clear, properly-aligned text as input, the preprocessing suite conducts text detection, binarization, auto-rotation, and noise reduction. A more detailed description of the preprocessing implementation is provided in Section 5.

After preprocessing, the server sends the image to an OCR engine for recognition. The OCR engine used in this prototype system is Tesseract [27], a platform developed by HP Labs between 1985 and 1995 and absorbed by Google in 2006. In 1995 it ranked in the top three OCR systems participating in the UNLV Test of OCR Accuracy [23], and when the document text experiments from the UNLV tests are run on the Tesseract system today it averages over 95% recognition accuracy. This engine is offered as free and open-source software under the Apache License 2.0.

Finally, the text extracted by Tesseract is supplied to a statistical machine translation system. For this prototype system we intend to use the Moses translation system [3] trained on the Europarl French and English corpora [15]. A future goal is to examine this system, in particular its ability to translate language from road signs when trained on this or other corpora, more fully.

5. IMAGE PREPROCESSING IMPLEMENTATION

The current focus of our research is the image preprocessing suite. All parts of this system were implemented in Matlab using the Image Processing Toolbox for fast morphological and other image manipulation operations. We provide in this section an overview of our system and closer analysis of its major contributions to text detection and auto-rotation.

5.1 System Overview

Our image preprocessing system is composed of three major parts: text detection and binarization; auto-rotation; and noise reduction. Figure 1 shows an overview of these processes.

The text detection and binarization step uses an edge-based morphological approach, drawing from the work described in [10], to identify candidate text regions. Applying the morphological gradient operator to a blurred grayscale version of the input image extracts strong edges, which are then grouped and labeled as distinct regions of interest.

Next, these candidate regions are analyzed individually. Text-like connected components (CCs) are extracted from the region by preserving the pixels with gray values in the range, either above or below the global gray-level threshold, judged to be the region's foreground. Simple size-based filtering eliminates from consideration regions that themselves are too small or large and regions that contain no connected components of text-like size and dimensions. The remaining regions are then clustered based on a vector of size and color features, resulting in a partition of the candidate region set into a text-like set and a non-text-like set. The text-like set, recognized as the set to which most median-sized (character-like) components fall, is preserved.

In the last stage of text detection, we select the minimum bounding rectangle with a small five pixel buffer on all sides around the text-like regions. Finally, the morphological edge groupings for the preserved regions are written to a "final region image," and the selected area of the original image is binarized based on the average preserved foreground range, weighted by the number of text-like connected components in each of these regions.

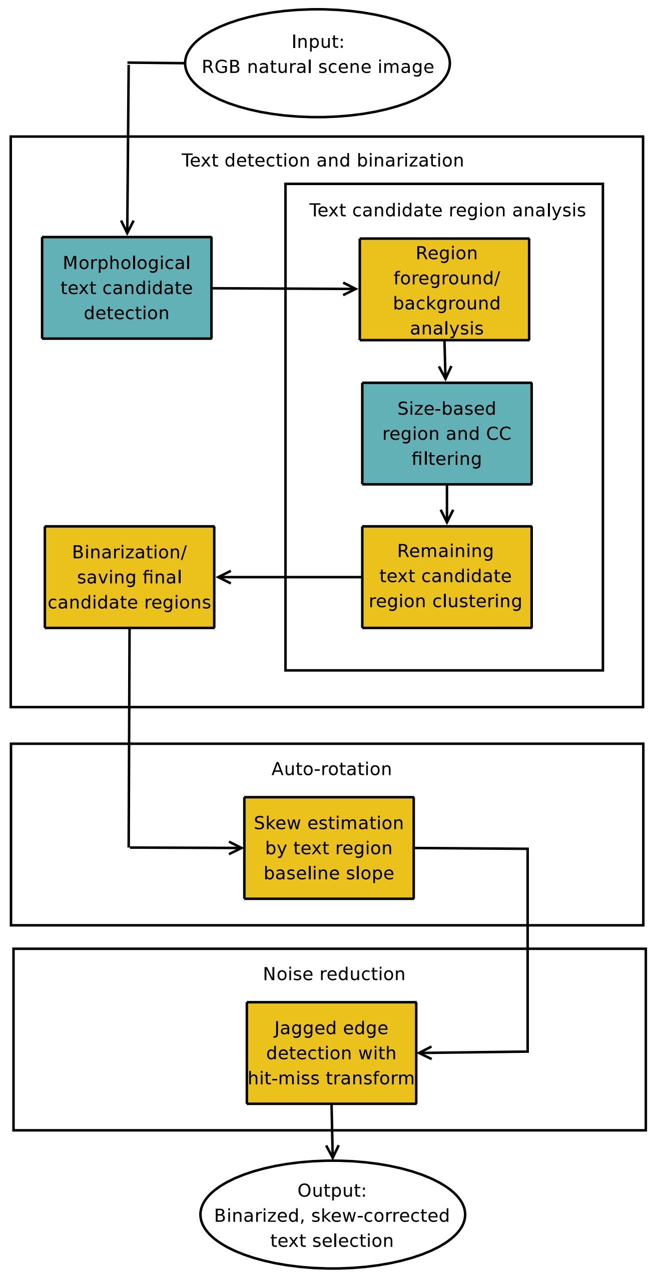
The binarized text selection and its region image then proceed to the auto-rotation stage. Here, the skew of text lines is estimated using the slopes of the baselines of the preserved regions, and the binarized text image is corrected by a rotation transformation.

Finally, the noise on text edges that was introduced by the rotation transformation is reduced using the morphological hit-miss transform. This transform, a relaxation of the standard morphological operations that use foreground patterns, allows us to search for patterns of foreground and background pixels surrounding single extra or missing pixels along edges. By detecting these pixels that make up jagged edges, we aim to smooth lines and curves to their pre-rotated state.

5.2 Major Contributions

5.2.1 Region Foreground/Background Analysis

Absent from the discussion in [10] is analysis of which connected components are important in each thresholded candi-



Input:
RGB natural scene image
Text detection and binarization
Text candidate region analysis
Morphological text candidate detection
Region foreground/ background analysis
Size-based region and CC filtering
Remaining text candidate region clustering
Binarization/ saving final candidate regions
Auto-rotation
Skew estimation by text region baseline slope
Noise reduction
Jagged edge detection with hit-miss transform
Output:
Binarized, skew-corrected text selection

Figure 1: An overview of the preprocessing suite. Regions shaded in blue are adapted from [10], while regions shaded in gold are new contributions from this research.

date text region. Ideally, pixels belonging to foreground text are on one side of the gray-level threshold and background pixels are on the other side, so we must judge which region is the foreground before continuing a connected component analysis.

In our text detection strategy, we determine the foreground color of a region by tracing its morphologically-grouped outer edges and determining whether the color just inside the edges is lighter (higher gray values) or darker (lower gray values) than the color outside the edges. Based on the assessment of this color, we pick out the elements higher or lower than the local threshold as foreground pixels. Foreground pixels are then separated into connected components by Hasan and Karam's labeling algorithm. Qualitatively, we have picked out connected components coming from letters, punctuation, or simple scene elements like single stones or rivets in signs.

5.2.2 *Text Candidate Region Clustering*

In order to be more adaptive to the wide range of features of scene text from public signage, we removed some restrictions from the text detection approach of [10]. First, we are applying the morphological edge analysis outside of the lower bound Hasan and Karam place on image resolution. They assume input resolution will not be below 9 pixels per millimeter, or around 230 ppi, but the input from a cell phone camera, webcam or similar portable imaging device regularly has an image resolution of 72 ppi. Second, we expect text in our corpora to be shortened or spaced widely for quick legibility by drivers and pedestrians, so we do not require a minimum of six character-like connected components to define a text-like region; rather, we keep regions with at least two character-like components in consideration.

Because we remove these restrictions, many of the regions remaining in consideration may actually contain background elements instead of actual text. As observed during testing and discussed further in Section 7, we have not found one set value of any one feature that classifies a region as text and not background across our test corpus. For this reason we use Weka, a data mining platform developed at the University of Waikato [9], to cluster the regions still under consideration based on a vector of fourteen features: the number of rows in the region; the number of columns in the region; the region's area in pixels; the percentage of total image area this region accounts for; the number of connected components in the region; the percentage of connected components in this region considered valid; the average gray value in the region; the gray-level threshold as determined by Otsu's method [22]; whether the foreground is the lighter or darker side of the threshold; the average gray value in the foreground range; the standard deviation of gray levels in the foreground range; the average gray value in the background range; the standard deviation of gray levels in the background range; and the ratio of foreground pixels to total pixels in the region. In our prototype system we use the expectation-maximization or EM algorithm with two desired clusters; tests using Weka's other implemented clustering algorithm, the simple K-means algorithm, performed similarly.

The result of this clustering is a two-set partition which, as observed during testing and described further in Section 7, maps closely to a partition between foreground and background elements. We must then determine which of the clusters is the one containing foreground elements. Our current approach is to see into which of the region clusters the majority of the valid connected components whose areas (in pixels) vary from the mean by up to one-quarter of a standard deviation. This choice assumes that the most text-like of all the connected components in the image are of near-mean size and that connected components with sizes far outside the mean are not text-like.

Once the text-like cluster is decided, components from the non-text-like cluster are filtered out of the image. We also filter out any components whose foreground color range is not the range shared by the majority of the components; this filter aims to remove some non-text-like components that were statistically like the rest of the text-like components.

From the size-based filtering step, we have kept a record of removed regions judged to have one text-like connected component. Due to the aforementioned wide letter spacing in some public signage and low resolution from our input device, some letters may be isolated or blurred together in the morphological grouping, but we still want to preserve their text information. At this point we examine the neighborhood of each component in the text-like cluster and "pull in" close discarded components, effectively returning to consideration characters isolated from their original word or important articles between words.

5.2.3 Skew Estimation

As discussed in Section 3, the approaches to skew estimation described in [5] and [19] estimate line slope using the centers of character components. We choose instead to estimate skew using the baseline of the morphological edge grouping for a region. The points along the baseline of text, when morphologically dilated so holes begin to close and edges draw closer together, are less prone to variations than character midpoints. Characters with tails hanging below a line occur infrequently in English and Romance languages (see [17] for English letter frequencies) so baseline variation will be minimal, but midpoints can come from letters of multiple sizes - tall letters like k, short letters like m, and letters with hanging tails like j - and thus can have a fair amount of variance around the actual skew direction.

For each region, we pick samples points along the baseline of the edge grouping and eliminate all points whose heights deviate from that of their left neighbor by more than one standard deviation from the mean difference between neighboring samples. We then use Matlab's Curve Fitting Toolbox to fit the remaining sample points to a linear model and determine the angle of rotation using the slope of the fit line.

Once the rotation angle has been determined for each region, we find the mean and standard deviation of the rotation angles and eliminate all those that deviate from the mean by more than three-quarters of a standard deviation; this helps eliminate outliers from irregular edge groupings derived from non-text. We then find the mean of the remaining rotation angles and output a black and white text image rotated by that amount.

6. EXPERIMENTAL METHODS

6.1 Test Corpora

The proposed image preprocessing suite, as implemented in Matlab, was applied to two test image corpora: a control corpus of 27 black and white document images, and an experimental corpus of 62 color scene images containing text from public signage in the Princeton, New Jersey and Johns Creek, Georgia areas. For the control corpus, the text candidate clustering step (see section 5.2.2) of the text detection algorithm was omitted, as there was minimal risk of non-text regions being marked as candidates because the control images had near uniform background.

All images in the test corpus may be found at our website for this project, http://www.princeton.edu/˜melmore/. We show two example images, along with the output produced from them during preprocessing, as Figures 5 and 6 in the Appendix of this paper.

All test images were taken with the Cingular 8525's embedded camera. The medium, or 640x320 pixel, image size setting was used; image resolution is 72x72 ppi.

6.2 OCR Accuracy Measurement

The text in the original test images and the images resulting from each stage of the preprocessing suite was recognized with Tesseract OCR [27].

To determine the accuracy of the recognition, we used the metric proposed by the Information Science Research Institute at UNLV for the Fourth Annual Test of OCR Accuracy [23]. If n is the ground truth number of text characters in the image, and m is the number of errors, or edit operations (insert, delete, and substitute) needed to transform the result from the OCR engine into the actual text in the image, the accuracy of the recognition is defined as

$$(n - m)/n \tag{1}$$

The upper bound on this metric is 100, a perfect recognition; there is no lower bound, as negative scores are possible when there are more errors than characters in the ground truth text of the image.

To calculate the accuracy for each test image, we used the accuracy program provided in the OCR Frontiers Toolkit, the companion software to the ISRI's 1999 book on OCR [24]. This toolkit is provided with Tesseract OCR to be used in evaluating one's build of the Tesseract system.

6.3 Rotation Measurement

To compare with the experimental angle of rotation, the ground truth angle of rotation for each test image was estimated to within 0.5° using GIMP 2.4.5.

7. EXPERIMENTAL RESULTS AND DISCUSSION

7.1 OCR Accuracy

OCR accuracy at each step in the preprocessing suite is a measure of the overall performance of this system. Figure 2 displays the average ISRI accuracy for the control document

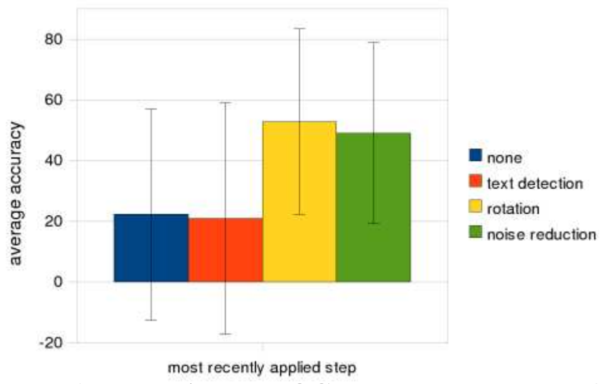


Figure 2: Average OCR accuracy on control corpus.

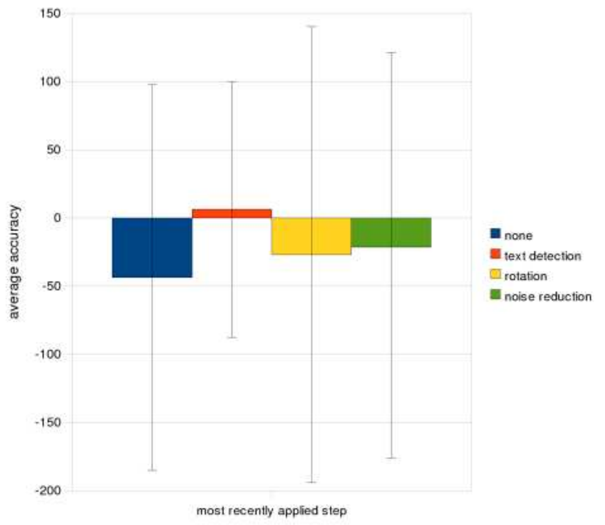


Figure 3: Average OCR accuracy on experimental corpus.

text corpus, and Figure 3 displays the average accuracy for the experimental scene text corpus.

For the control corpus, which consists of color images of black and white documents, OCR applied after the text detection step shows little change in accuracy than when OCR is applied to the unprocessed input image. The slight dip in accuracy, from a score of 22.27 to 20.97, is most likely due to binarization errors reducing the quality of the text in the image. The highest average accuracy - a score of 52.88, meaning roughly half the characters in the image are recognized correctly - occurs after the rotation stage. After noise reduction, the average accuracy score is 49.04, indicating that noise reduction alters some of the characters that were recognized correctly in the rotated image.

For the experimental corpus, accuracy scores frequently fall into the negative range because non-text elements in the image are falsely recognized as characters, creating noise in the text output for more errors than there are characters in the image. Without any preprocessing, the images in our corpus average an accuracy score of -43.57. All subsequent steps show an increase in accuracy over no processing, with the text detection step showing a positive average accuracy score of 6.37. Because of under- or over-rotation, the auto-rotation step and the noise reduction step that follows it dip back into the negative accuracy range with average scores of -26.82 and -21.34.

There is substantial deviation from the means shown in each corpus. In the control corpus, we see deviation between 30 and 40 points, which corresponds to small changes in quality of the input and of the intermediate preprocessing steps across the corpus. In the experimental corpus, however, we see much wider ranges of deviation. The deviation of the unprocessed input, which is 141.3 accuracy points above and below the mean, sheds light on the vast deviation in input image quality in the experimental corpus. A similarly wide deviation in accuracy scores is propagated through the preprocessing steps, indicating that the preprocessing suite performed either rather well or rather poorly on an individual test image. We hope to explain in subsequent sections the reasons this deviation in accuracy persists. We focus on text detection and rotation; discrepancies in noise reduction accuracy, while more difficult to quantify, are discussed qualitatively.

7.2 Significant Features in Text Detection

Our text detection scheme relies heavily upon a clustering phase, which selects the statistically significant features of the text candidate regions and partitions the set regions based upon the value of these features. Figure 4 shows the five features most frequently found to be statistically significant within a test image. Four of these five features involve the gray levels of pixels in the image, and the ratio of foreground-colored pixels to all pixels in a region is significant to all 62 images in the experimental corpus. This is a strong indication that the contrast between foreground and background and the variance within these two classes of colors is a significant feature distinguishing text regions from non-text regions.

On the other hand, it may be beneficial to seek other no-

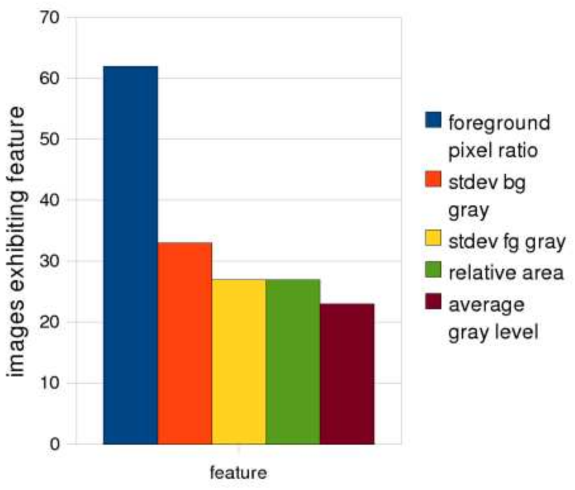


Figure 4: The top five statistically significant features detected in test images.

tions of what is text for such a text detection scheme to be effective. It would be ideal to strike upon a small set of significant features that accurately classify text, but in our experiments all fourteen features considered for candidate regions are selected as statistically significant for at least four images in the corpus. This leads to two conclusions: either scene images are too complex to have a small set of text-distinguishing features, or there are some features that we have not examined that do a better job of partitioning the region set into text and non-text in all cases. Given our initial results, a viable next step would be to apply machine learning algorithms to a larger corpus of images to see what is learned and how difficult current learning techniques are on such a complex corpus.

A final consideration is the choice of which of the two clusters determined by our feature analysis contains text. A wrong choice could lead to a final region image containing mostly or only non-text regions; while this obviously lowers the accuracy of text detection, it also bears upon the estimated angle of rotation in the auto-rotation stage. We observed a significant percentage of "wrong choices" over our corpus, and many of these choices do result in the introduction of background noise in the OCR output as well as wild deviation of the estimated angle of rotation from the actual angle of rotation. A future discovery of threshold values for the features that classify text versus non-text, or an improved scheme for determining into which cluster the most "text-like" of candidate regions' components fall, would likely reduce the wide deviation from the mean of OCR accuracy and improve the performance of each preprocessing step.

7.3 Text Detection Accuracy

The accuracy of the text detection step is crucial to system performance; if background noise is improperly classified as text, it remains in the image for the rotation and preprocessing steps. It is ideal, then, to partition the text candidates in the image clearly into text and non-text by the end of this procedure.

There are two components to accuracy in text detection: how much actual text is contained in the final bounding box, and how much non-text is contained in the final bounding box. Our findings are summarized in Table 1.

Ideally, the percentage of characters outside the bounding box should be 0. The low mean, 7.87%, for this metric within our experimental corpus indicates that our text detection scheme to be slightly strict at times. Though the scheme assigned a bounding box containing all characters to most images in the corpus, it was "too strict" when the text-like cluster was poorly selected. As discussed above, better selection of which candidate region cluster contains text may alleviate this problem.

As with every classification problem, we are concerned with false positive and false negative classifications. In this study, false positives are candidate regions that do not actually contain text but are believed by our code to contain text, and false negatives are candidate regions that contain text but are believed by our code to not contain text. We consider two metrics based on text region classifications: how many regions out of those believed to contain text are false positives, and how many regions out of those that actually contain text are false negatives.

Our experimental results show similar values for each metric: on average, about 30% of the final region image is composed of non-text regions, and about 30% of text regions get excluded from the final region image. As evidenced by the wide standard deviation, however, this is another case of some test images performing very well and some performing very poorly. Though the feature-based clustering of candidate regions needs improvement in itself, it is more often the incorrect selection of the text-like cluster that leads to high false positive and false negative rates.

7.4 Rotation Accuracy

The accuracy of the rotation phase is highly dependent on the accuracy of the text detection phase: the angle of rotation is estimated based on the slope of regions in the final region image, and if non-text regions are included in the final region image, the slopes captured may not match the slopes of actual text components.

Our control results show the rotation scheme to be accurate when the text detection process makes correct text classifications. Across 27 control images whose actual degrees of rotation range from 41.5° counterclockwise to 46° clockwise, the average magnitude of experimental error is 1.23°. Judging by a standard deviation from this mean of 2.60°, our rotation scheme is not greatly affected by differences in quality of the regions kept or the binarized image when there is no noisy background hindering text classification.

As discussed in sections 7.2 and 7.3, there are still many problems confounding the text detection stage, so rotations on improperly detected text are expectedly poor. Over the 62 test images whose actual degrees of rotation range from 10.5° counterclockwise to 22° clockwise, there is a 7.1° mean error with a standard deviation of 9.86°. If text detection

Table 1: Accuracy metrics for text detection

Metric	Mean	Standard Deviation
(text characters outside bounding box) / (total characters)	0.0787	0.1513
(false positive text region classifications) / (total positives)	0.3047	0.3080
(false negative text region classifications) / (total text regions)	0.3202	0.3317

on our experimental corpus also selected only and all regions containing text and binarized text and non-text pixels properly, we expect results would be similar to those of the control corpus.

7.5 Noise Reduction Analysis

The accuracy of the noise reduction step is a bit more difficult to gauge; on the control corpus, it results in a reduction of OCR accuracy from that of the rotation stage, and on the experimental corpus it results in a slight increase in accuracy. Qualitatively, we observe noise reduction helping the character recognition process in some regions of the image and hindering in others. Part (j) of Figure 6 shows one example of this phenomenon; the larger letters in the image are negatively affected, while the smaller letters become more accurately recognized. This sort of isolated improvement lends itself to the future integration of a translation system; our preprocessing suite could communicate with the translation dictionary to see when words in the original language are recognized at each step, ignoring any deterioration of already-recognized words. More research is necessary to judge the effect our noise reduction scheme, consisting of removals and additions of isolated pixels on edges, has on the character recognition process used by Tesseract and by other available OCR engines.

8. CONCLUSIONS AND FUTURE WORK

This paper has presented an image preprocessing system for use in machine recognition of text in natural scene images. Of the three steps of this preprocessing suite - text detection, auto-rotation, and noise reduction - text detection remains the most difficult problem. The original criteria, size ratios and presence of edges, used in Hasan and Karam's morphological strategy [10] cannot discern all types of background objects in natural scene images of public signage, and we have not found a more discerning criterion among the features used in the text candidate clustering of our image corpus. The proposed auto-rotation scheme has proved to be accurate when the angle of rotation is estimated using only text regions, and the noise reduction scheme often reveals better recognition of small characters after larger ones have been recognized in earlier steps, so it is up to improvements in the text detection phase to result in improvements of the entire system's text recognition accuracy.

As discussed in the presentation of results, an apparent next step is to attempt more sophisticated means of measuring what qualities define text in scene images. It could be of benefit to examine further signal processing techniques for edge detection and pattern recognition, and exploring machine learning techniques for extracting text-like qualities for use in classification also seems promising. A final venue for improving our text detection accuracy would be to examine the techniques employed by the open-source OCR engine Tesseract to see how our approach could cater to its strengths.

The proposed preprocessing suite is by no means complete. One future challenge is the integration of other preprocessing steps such as perspective correction, light correction, and better enhancement of letter shapes for increased legibility. While exploring more ways to increase the quality of images provided to the OCR engine, it is another future goal to examine which preprocessing steps may be done on a user's mobile device. It would be ideal if dependence on a remote mathematical processing server were eliminated and a user could translate scene text regardless of network availability.

Finally, we hope to make our prototype mobile application for scene text translation deployable. This requires further research into the capabilities of the Android platform or other development environments and integration of a machine translator to the pre- and post-processing systems. Our initial results show that intermediate translation of OCR output could help our system learn when words are correctly recognized or help correct a poorly-recognized word knowing its context. This and all future steps will help in our goal of designing a translation system that is accurate, adaptive, and above all general.

9. ACKNOWLEDGMENTS

We would like to thank Princeton University and the Computing Research Association's Distributed Mentor Project (DMP) for funding and technical support for this work.

10. REFERENCES

[1] GOCR. http://jocr.sourceforge.net/index.html.

[2] Google Android. http://code.google.com/android/.

[3] Moses. http://www.statmt.org/moses/.

[4] OCRAD. http://www.gnu.org/software/ocrad/ocrad.html.

[5] B. T. Ávila and R. D. Lins. A fast orientation and skew detection algorithm for monochromatic document images. In *Proc. 2005 ACM Symposium Document Engineering*, pages 118–126, Nov. 2005.

[6] W. Bieniecki, S. Grabowski, and W. Rozenberg. Image preprocessing for improving OCR accuracy. In *Proc. Int. Conf. Perspective Technologies in MEMS Design 2007*, pages 75–80, May 2007.

[7] J. F. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, pages 679–698, 1986.

[8] S. Ferreira, V. Garin, and B. Gosselin. A text detection technique applied in the framework of a mobile camera-based application. In *Proc. First Int. Workshop Camera-Based Document Analysis and Recognition*, pages 133–139, Aug. 2005.

[9] E. Frank and I. H. Witten. *Data Mining: Practical Machine Learning Tools and Techniques (Second*

Edition). Morgan Kaufman, June 2005.

[10] Y. M. Y. Hasan and L. J. Karam. Morphological text extraction from images. *IEEE Trans. Image Processing*, 9:1978–1983, Nov. 2000.

[11] H. Hase, T. Shinokawa, M. Yoneda, M. Sakai, and H. Maruyama. Character string extraction by multi-stage relaxation. In *Proc. Fourth Int. Conf, Document Analysis and Recognition*, pages 298–302, Aug. 1997.

[12] L. Jagannathan and C. V. Jawahar. Crosslingual access of textual information using camera phones. In *Proc. Int. Conf. Cognition and Recognition*, pages 655–660, Dec. 2005.

[13] K. Jung, K. I. Kim, and A. K. Jain. Text information extraction in images and video: A survey. *Pattern Recognition*, 37:977–997, May 2004.

[14] P. Kim. Automatic text location in complex color images using local color quantization. In *Proc. IEEE Region 10 Conference*, volume 1, pages 629–632, 1999.

[15] P. Koehn. Europarl: A parallel corpus for statistical machine translation. In *Proc. MT Summit X*, 2005.

[16] J. Liang, D. Doermann, and H. Li. Camera-based analysis of text and documents: a survey. *Int. Journal Document Analysis and Recognition*, 7:88–104, Jul. 2005.

[17] T. Linton. English letter frequencies. http://pages.central.edu/emp/lintont/classes/spring01/cryptography/letterfreq.html.

[18] C. Liu, C. Wang, and L. Dai. *Lecture Notes in Computer Science*, chapter Text detection in images based on color texture features, pages 40–48. Springer Berlin/Heidelberg, 2005.

[19] S. Messelodi and C. M. Modena. Automatic identification and skew estimation of text lines in real scene images. *Pattern Recognition*, 32:791–810, May 1999.

[20] S. Mori, C. Y. Suen, and K. Yamamoto. Historical review of OCR research and development. *Proc. IEEE*, 80:1029–1058, Jul. 1992.

[21] W. Niblack. *An Introduction to Digital Image Processing.* Prentice/Hall International, Englewood Cliffs, NJ, 1986.

[22] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Trans. Systems, Man, and Cybernetics*, 9:62–66, Jan. 1979.

[23] S. Rice, F. Jenkins, and T. Nartker. The fourth annual test of OCR accuracy. Technical Report 95-03, Information Science Research Institute, University of Nevada, Las Vegas, Jul. 1995.

[24] S. V. Rice, G. Nagy, and T. A. Nartker. *Optical Character Recognition: An Illustrated Guide to the Frontier.* Kluwer Academic Publishers, Apr. 1999.

[25] J. Serra. *Image Analysis and Mathematical Morphology.* Academic Press, New York, 1982.

[26] C. S. Shin, K. I. Kim, M. H. Park, and H. J. Kim. Support vector machine-based text detection in digital video. In *Proc. IEEE Signal Processing Society Workshop*, volume 2, pages 634–641, Dec. 2000.

[27] R. Smith. An overview of the Tesseract OCR engine. In *Proc. Ninth Annual Conference on Document Analysis and Recognition*, volume 2, pages 23–26, Sep. 2007.

[28] J. Yang, J. Gao, Y. Zhang, X. Chen, and A. Waibel. An automatic sign recognition and translation system. In *Proc. Workshop Perceptive User Interfaces*, pages 1–8, Nov. 2001.

APPENDIX

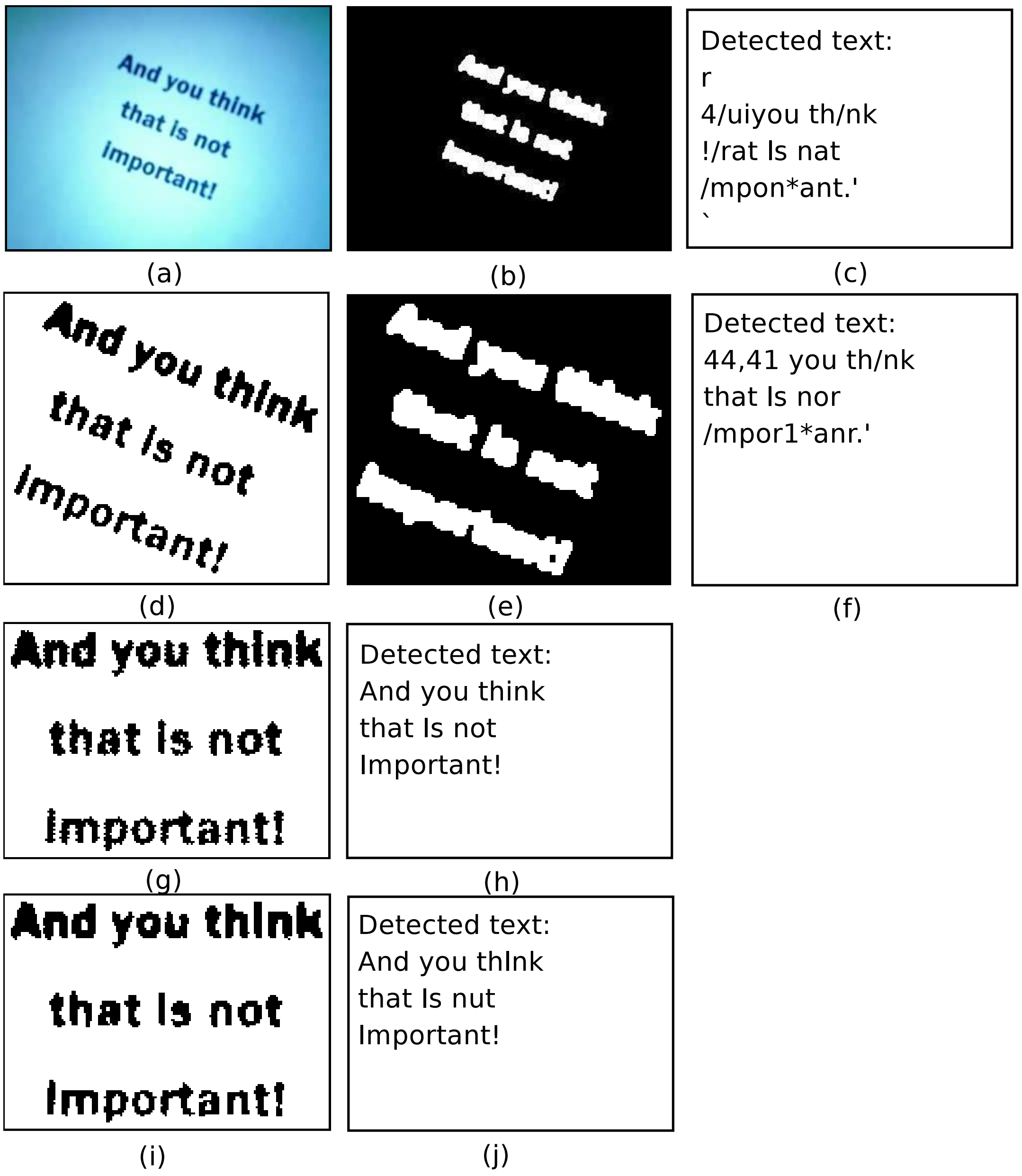
A. SAMPLE PREPROCESSING RESULTS

A.1 Control Corpus

Figure 5 shows the image and OCR output from the different stages of the image preprocessing suite when run on an input image from the control corpus.

A.2 Experimental Corpus

Figure 6 shows the image and OCR output from the different stages of the image preprocessing suite when run on an input image from the experimental corpus.



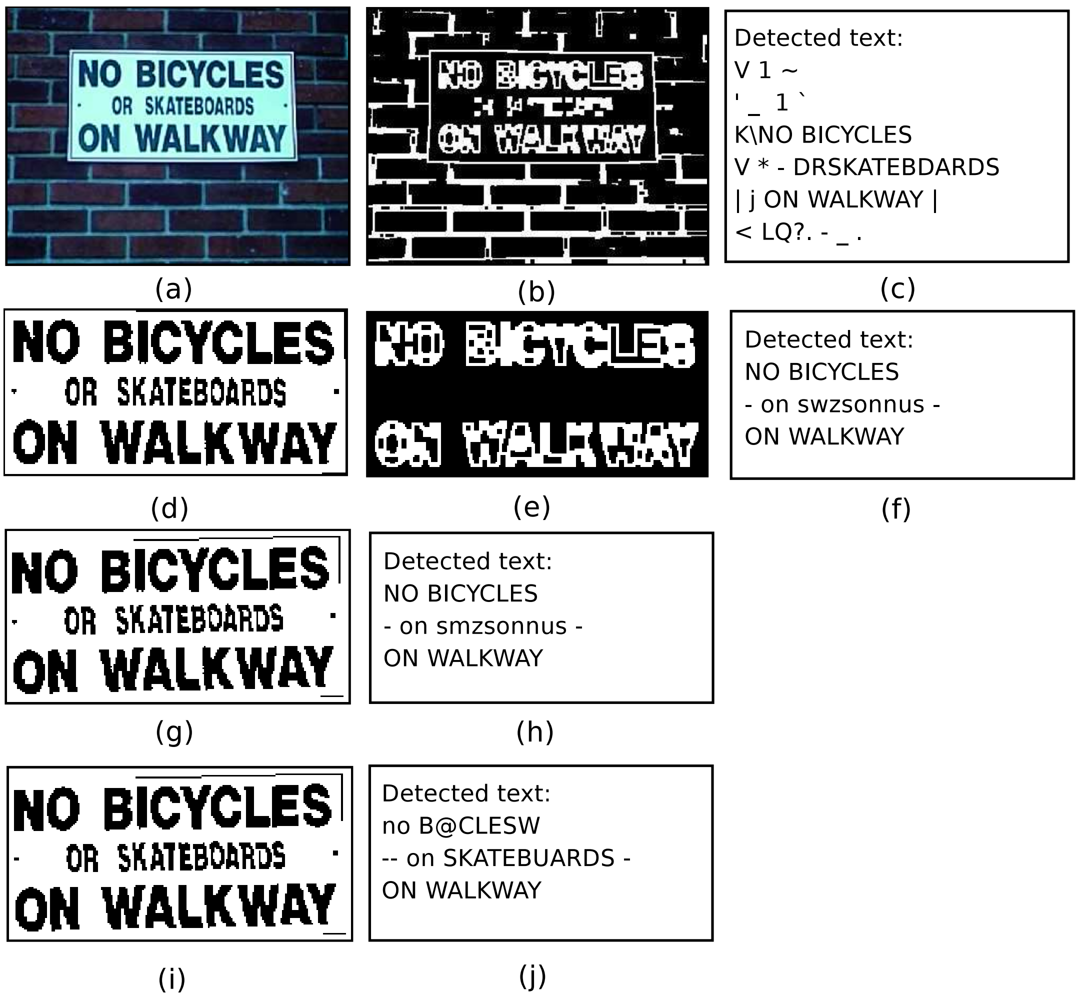
(c)
```
Detected text:
r
4/uiyou th/nk
!/rat Is nat
/mpon*ant.'
`
```

(f)
```
Detected text:
44,41 you th/nk
that Is nor
/mpor1*anr.'
```

(h)
```
Detected text:
And you think
that Is not
Important!
```

(j)
```
Detected text:
And you thlnk
that Is nut
Important!
```

Figure 5: Preprocessing output for one image in the control document text corpus. (a) original image, (b) edge grouping, (c) OCR output for (a), (d) binarized text candidate image, (e) final region image for text candidates, (f) OCR output for (d), (g) autorotated text image, (h) OCR output for (g), (i) noise reduced image, (j) OCR output for (i)



(c)

```
Detected text:
V 1 ~
' _ 1 `
K\NO BICYCLES
V * - DRSKATEBDARDS
| j ON WALKWAY |
< LQ?. - _ .
```

(f)

```
Detected text:
NO BICYCLES
- on swzsonnus -
ON WALKWAY
```

(h)

```
Detected text:
NO BICYCLES
- on smzsonnus -
ON WALKWAY
```

(j)

```
Detected text:
no B@CLESW
-- on SKATEBUARDS -
ON WALKWAY
```

Figure 6: Preprocessing output for one image in the experimental scene text corpus. (a) original image, (b) edge grouping, (c) OCR output for (a), (d) binarized text candidate image, (e) final region image for text candidates, (f) OCR output for (d), (g) autorotated text image, (h) OCR output for (g), (i) noise reduced image, (j) OCR output for (i)