ECE 415 Project First Draft

Optical Character Recognition

Group Partners:

- Yunjuan Wang <661 180 568>
- Usama Muneeb <656 392 492>

Section 1: Problem Statement

The goal is to develop an OCR (Optical Character Recognition) application that can take a scanned document image as an input and converts it to digital text with a high degree of accuracy. Traditionally, documents scanned into a computer are saved as PDFs and can be read only on a computer. OCR not only allows a user to scan documents and save them to a computer, but also to be able to edit the documents. Vividly, the application is such like a combination of CamScanner and Adobe Acrobat Pro. It can be widely used since it often occurs when someone tries to scan some paper work into editable files.

The final application we develop is web based (written in NodeJS and Python at backend).

Section 2: Implementation Details

We are using Python along with the following libraries:

- OpenCV for regular image processing procedures
- TensorFlow for implementing LeNet-5 neural network to recognize letters
- PDFRW for PDF write operations to generate a PDF file.

Also, we are using NodeJS for displaying a web-based user interface.

Section 3: Block Diagram of the System

The following block diagram summarizes our steps for achieving the OCR application.

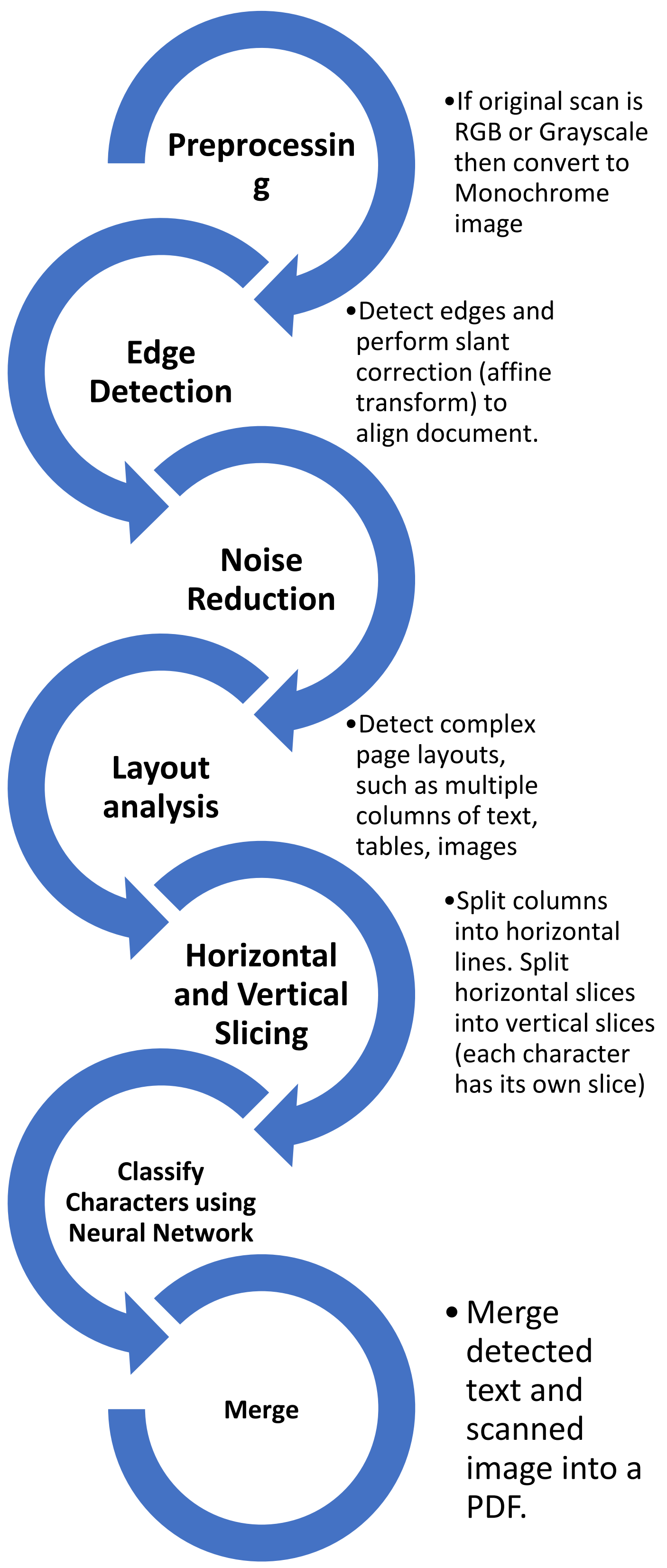

Preprocessin
g
•If original scan is RGB or Grayscale then convert to Monochrome image
Edge Detection
•Detect edges and perform slant correction (affine transform) to align document.
Noise Reduction
Layout analysis
•Detect complex page layouts, such as multiple columns of text, tables, images
Horizontal and Vertical Slicing
•Split columns into horizontal lines. Split horizontal slices into vertical slices (each character has its own slice)
Classify Characters using Neural Network
Merge
• Merge detected text and scanned image into a PDF.

Figure 1 Block Diagram

Section 4: Inputs and Outputs

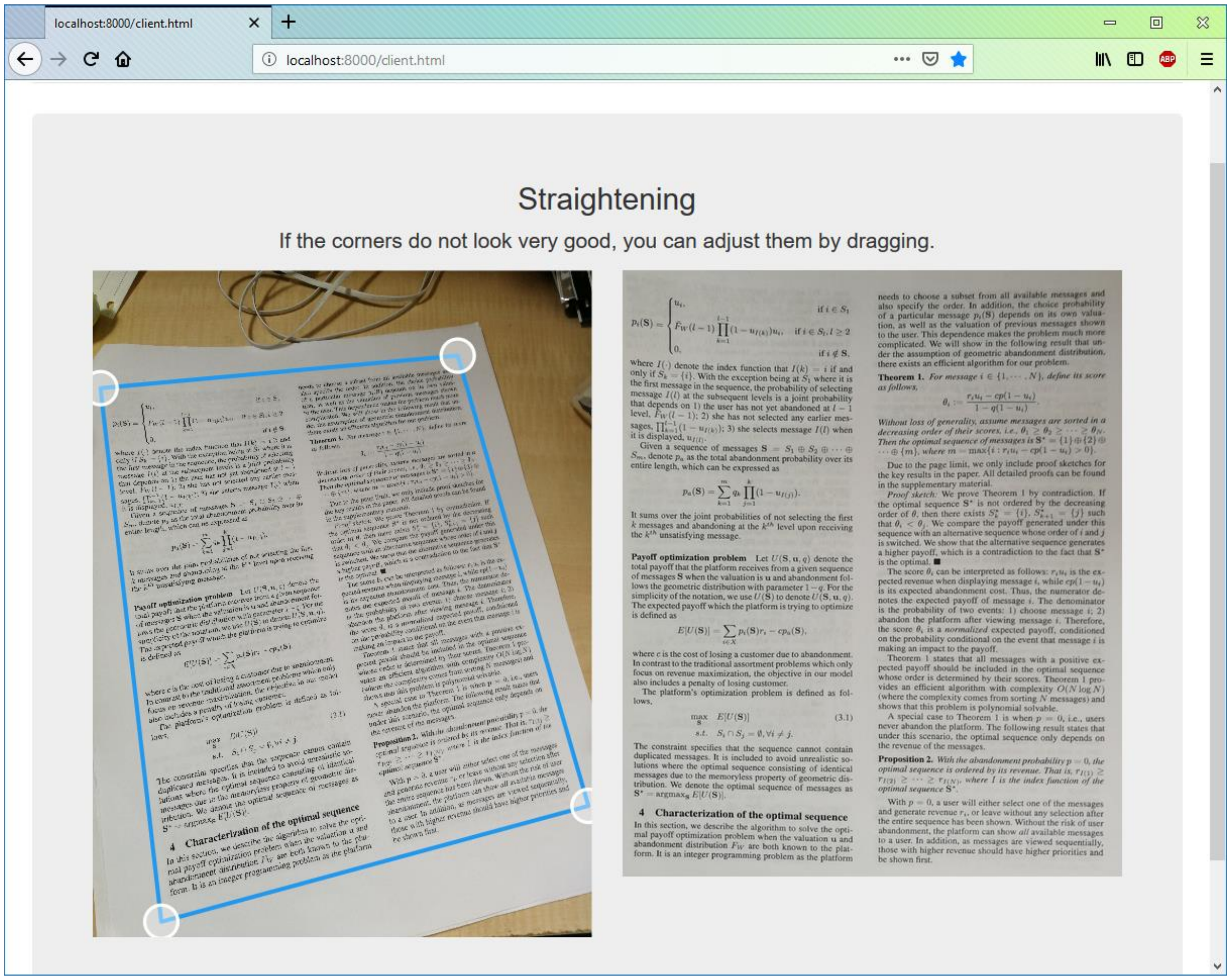
Our input is a JPG, JPEG, PNG or TIF image of a document scanned in at least 300 DPI.

The output is a PDF document which has the **original image in the background**, along with **hidden OCR text overlaid which can be selected and copied**.

Section 5: Walkthrough of the process

The whole process of OCR can be broken down into six main steps. Each step with its sub steps is summarized below.

1. **Edge Detection** – find the largest object in the image (which should be the paper)
    a. We convert the RGB image into **grayscale**.
    b. We choose a standard size for the image to perform the projective transform. We **resize** the image to height 800 pixels and the width is chosen to retain the aspect ratio.
    c. We perform **bilateral filtering** to reduce noise and smoothen the image, while preserving edges.
    d. We perform **adaptive thresholding** to create a monochrome image. Adaptive thresholding calculates the optimal threshold before applying it.
    e. We perform **median filtering** to get rid of small details.
    f. We **find the contours** of **ALL objects** inside the image. Each object is represented by a sequence of points which represent the contour. We choose four points for each object using OpenCV's polygon approximation feature.
    g. We find the **object with the largest area**. The area is computed using the four coordinates for every (approximated) polygon.
    h. We choose the object with the largest area. This object is considered to be the paper which we need to perform OCR on.
    i. We use these four **source coordinates** and map them to the four corners of the target image, which are (0,0), (0,WIDTH), (HEIGHT,0) and (HEIGHT,WIDTH). These are the **target coordinates**.
    j. This is done by computing a projective transform that maps these **source coordinates** to **target coordinates**.
    k. This results in a good enough approximation of the paper, as shown in the screenshot below.
    l. In case the approximation is not good, the user can adjust the corners and obtain better straightening.



Straightening

If the corners do not look very good, you can adjust them by dragging.

$$p_i(\mathbf{S}) = \begin{cases} u_i, & \text{if } i \in S_1 \\ \bar{F}_W(l-1)\prod_{k=1}^{l-1}(1-u_{I(k)})u_i, & \text{if } i \in S_l, l \geq 2 \\ 0, & \text{if } i \notin \mathbf{S}. \end{cases}$$

where $I(\cdot)$ denote the index function that $I(k) = i$ if and only if $S_k = \{i\}$. With the exception being at S_1 where it is the first message in the sequence, the probability of selecting message $I(l)$ at the subsequent levels is a joint probability that depends on 1) the user has not yet abandoned at $l-1$ level, $\bar{F}_W(l-1)$; 2) she has not selected any earlier messages, $\prod_{k=1}^{l-1}(1-u_{I(k)})$; 3) she selects message $I(l)$ when it is displayed, $u_{I(l)}$.

Given a sequence of messages $\mathbf{S} = S_1 \oplus S_2 \oplus \cdots \oplus S_m$, denote p_a as the total abandonment probability over its entire length, which can be expressed as

$$p_a(\mathbf{S}) = \sum_{k=1}^{m} q_k \prod_{j=1}^{k}(1-u_{I(j)}).$$

It sums over the joint probabilities of not selecting the first k messages and abandoning at the k^{th} level upon receiving the k^{th} unsatisfying message.

Payoff optimization problem Let $U(\mathbf{S}, \mathbf{u}, q)$ denote the total payoff that the platform receives from a given sequence of messages $\mathbf{S}$ when the valuation is $\mathbf{u}$ and abandonment follows the geometric distribution with parameter $1-q$. For the simplicity of the notation, we use $U(\mathbf{S})$ to denote $U(\mathbf{S}, \mathbf{u}, q)$. The expected payoff which the platform is trying to optimize is defined as

$$E[U(\mathbf{S})] = \sum_{i \in X} p_i(\mathbf{S})r_i - cp_a(\mathbf{S}),$$

where c is the cost of losing a customer due to abandonment. In contrast to the traditional assortment problems which only focus on revenue maximization, the objective in our model also includes a penalty of losing customer.

The platform's optimization problem is defined as follows,

$$\max_{\mathbf{S}} \quad E[U(\mathbf{S})] \qquad (3.1)$$
$$s.t. \quad S_i \cap S_j = \emptyset, \forall i \neq j.$$

The constraint specifies that the sequence cannot contain duplicated messages. It is included to avoid unrealistic solutions where the optimal sequence consisting of identical messages due to the memoryless property of geometric distribution. We denote the optimal sequence of messages as $\mathbf{S}^* = \text{argmax}_{\mathbf{S}} E[U(\mathbf{S})]$.

4 Characterization of the optimal sequence

In this section, we describe the algorithm to solve the optimal payoff optimization problem when the valuation $\mathbf{u}$ and abandonment distribution F_W are both known to the platform. It is an integer programming problem as the platform needs to choose a subset from all available messages and also specify the order. In addition, the choice probability of a particular message $p_i(\mathbf{S})$ depends on its own valuation, as well as the valuation of previous messages shown to the user. This dependence makes the problem much more complicated. We will show in the following result that under the assumption of geometric abandonment distribution, there exists an efficient algorithm for our problem.

Theorem 1. *For message $i \in \{1, \cdots, N\}$, define its score as follows,*

$$\theta_i := \frac{r_i u_i - cp(1-u_i)}{1-q(1-u_i)}.$$

Without loss of generality, assume messages are sorted in a decreasing order of their scores, i.e., $\theta_1 \geq \theta_2 \geq \cdots \geq \theta_N$. Then the optimal sequence of messages is $\mathbf{S}^* = \{1\} \oplus \{2\} \oplus \cdots \oplus \{m\}$, where $m = \max\{i : r_i u_i - cp(1-u_i) > 0\}$.

Due to the page limit, we only include proof sketches for the key results in the paper. All detailed proofs can be found in the supplementary material.

Proof sketch: We prove Theorem 1 by contradiction. If the optimal sequence $\mathbf{S}^*$ is not ordered by the decreasing order of θ, then there exists $S_k^* = \{i\}, S_{k+1}^* = \{j\}$ such that $\theta_i < \theta_j$. We compare the payoff generated under this sequence with an alternative sequence whose order of i and j is switched. We show that the alternative sequence generates a higher payoff, which is a contradiction to the fact that $\mathbf{S}^*$ is the optimal. ■

The score θ_i can be interpreted as follows: $r_i u_i$ is the expected revenue when displaying message i, while $cp(1-u_i)$ is its expected abandonment cost. Thus, the numerator denotes the expected payoff of message i. The denominator is the probability of two events: 1) choose message i; 2) abandon the platform after viewing message i. Therefore, the score θ_i is a *normalized* expected payoff, conditioned on the probability conditional on the event that message i is making an impact to the payoff.

Theorem 1 states that all messages with a positive expected payoff should be included in the optimal sequence whose order is determined by their scores. Theorem 1 provides an efficient algorithm with complexity $O(N \log N)$ (where the complexity comes from sorting N messages) and shows that this problem is polynomial solvable.

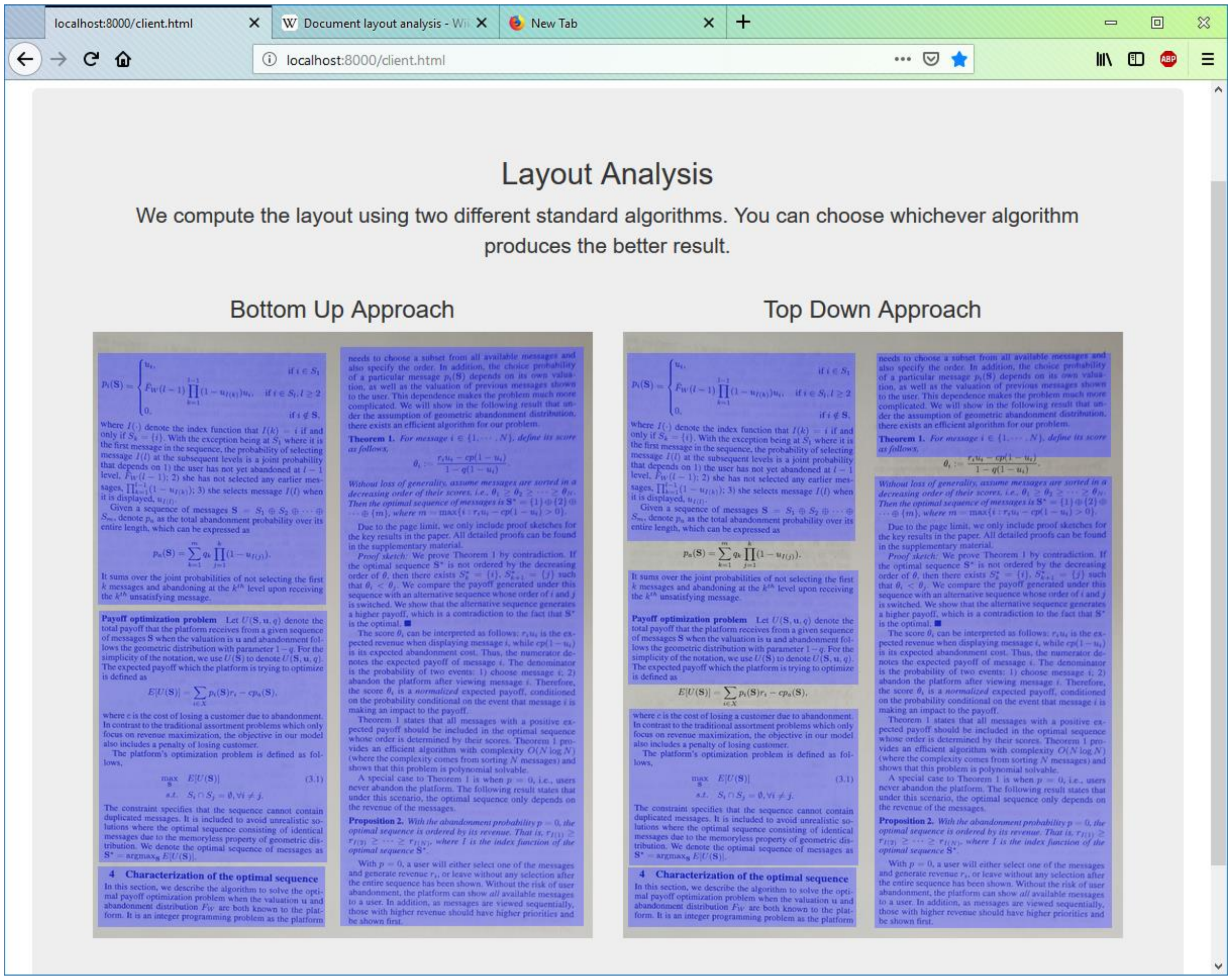
A special case to Theorem 1 is when $p = 0$, i.e., users never abandon the platform. The following result states that under this scenario, the optimal sequence only depends on the revenue of the messages.

Proposition 2. *With the abandonment probability $p = 0$, the optimal sequence is ordered by its revenue. That is, $r_{I(1)} \geq r_{I(2)} \geq \cdots \geq r_{I(N)}$, where I is the index function of the optimal sequence $\mathbf{S}^*$.*

With $p = 0$, a user will either select one of the messages and generate revenue r_i, or leave without any selection after the entire sequence has been shown. Without the risk of user abandonment, the platform can show *all* available messages to a user. In addition, as messages are viewed sequentially, those with higher revenue should have higher priorities and be shown first.

Figure 2 Screenshot - Straightening

2. <u>**Layout Analysis** – breaking the page down into individual objects.</u>
    a. There are three algorithms for doing layout analysis. We integrate two of them and let the user choose whichever layout analysis he thinks performs better. The layout analysis algorithms are described in Section 6. We also give the user a choice to add undetected regions and remove unnecessary regions from the layout analysis.


localhost:8000/client.html
Document layout analysis - Wi
New Tab
localhost:8000/client.html

Layout Analysis

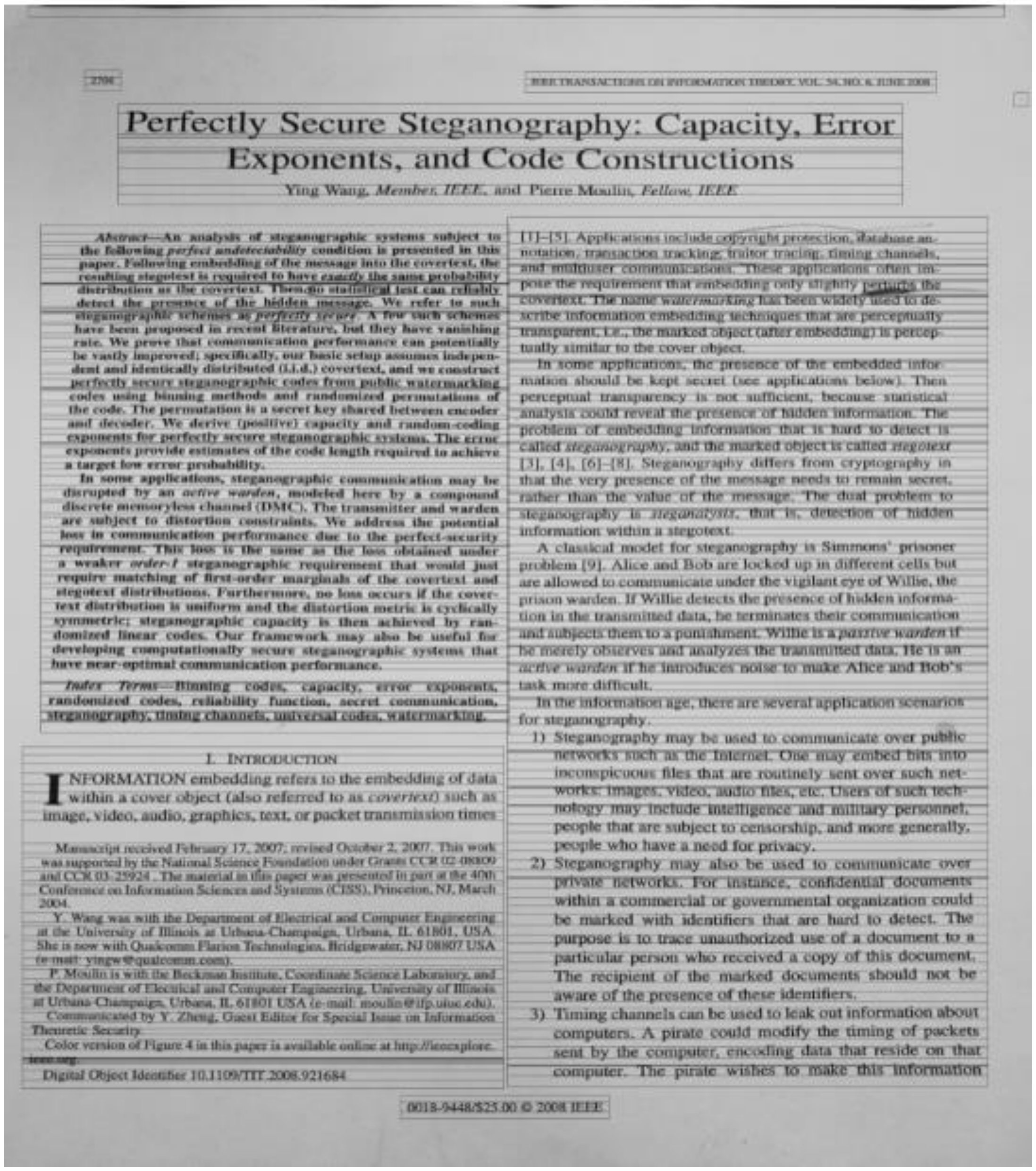
We compute the layout using two different standard algorithms. You can choose whichever algorithm produces the better result.

Bottom Up Approach

Top Down Approach

Figure 3 Screenshot - Layout Analysis

3. <u>**Line Splitting** – divide blocks of text into individual lines.</u>
    a. Iterate over each block of text given by the Layout Analysis algorithm.
    b. For each block, scan from the top most row of pixels to the bottom most row of pixels.
    c. Calculate the average pixel value of each row.
    d. Apply a threshold on the average pixel value.
    e. Whenever the average value goes below the threshold, create a horizontal line



2706 IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. 54, NO. 6, JUNE 2008

Perfectly Secure Steganography: Capacity, Error Exponents, and Code Constructions

Ying Wang, *Member, IEEE*, and Pierre Moulin, *Fellow, IEEE*

***Abstract*—An analysis of steganographic systems subject to the following *perfect undetectability* condition is presented in this paper. Following embedding of the message into the covertext, the resulting stegotext is required to have *exactly* the same probability distribution as the covertext. Then no statistical test can reliably detect the presence of the hidden message. We refer to such steganographic schemes as *perfectly secure*. A few such schemes have been proposed in recent literature, but they have vanishing rate. We prove that communication performance can potentially be vastly improved; specifically, our basic setup assumes independent and identically distributed (i.i.d.) covertext, and we construct perfectly secure steganographic codes from public watermarking codes using binning methods and randomized permutations of the code. The permutation is a secret key shared between encoder and decoder. We derive (positive) capacity and random-coding exponents for perfectly secure steganographic systems. The error exponents provide estimates of the code length required to achieve a target low error probability.**

In some applications, steganographic communication may be disrupted by an *active warden*, modeled here by a compound discrete memoryless channel (DMC). The transmitter and warden are subject to distortion constraints. We address the potential loss in communication performance due to the perfect-security requirement. This loss is the same as the loss obtained under a weaker *order-1* steganographic requirement that would just require matching of first-order marginals of the covertext and stegotext distributions. Furthermore, no loss occurs if the covertext distribution is uniform and the distortion metric is cyclically symmetric; steganographic capacity is then achieved by randomized linear codes. Our framework may also be useful for developing computationally secure steganographic systems that have near-optimal communication performance.

***Index Terms*—Binning codes, capacity, error exponents, randomized codes, reliability function, secret communication, steganography, timing channels, universal codes, watermarking.**

I. INTRODUCTION

INFORMATION embedding refers to the embedding of data within a cover object (also referred to as *covertext*) such as image, video, audio, graphics, text, or packet transmission times [1]–[5]. Applications include copyright protection, database annotation, transaction tracking, traitor tracing, timing channels, and multiuser communications. These applications often impose the requirement that embedding only slightly perturbs the covertext. The name *watermarking* has been widely used to describe information embedding techniques that are perceptually transparent, i.e., the marked object (after embedding) is perceptually similar to the cover object.

In some applications, the presence of the embedded information should be kept secret (see applications below). Then perceptual transparency is not sufficient, because statistical analysis could reveal the presence of hidden information. The problem of embedding information that is hard to detect is called *steganography*, and the marked object is called *stegotext* [3], [4], [6]–[8]. Steganography differs from cryptography in that the very presence of the message needs to remain secret, rather than the value of the message. The dual problem to steganography is *steganalysis*, that is, detection of hidden information within a stegotext.

A classical model for steganography is Simmons' prisoner problem [9]. Alice and Bob are locked up in different cells but are allowed to communicate under the vigilant eye of Willie, the prison warden. If Willie detects the presence of hidden information in the transmitted data, he terminates their communication and subjects them to a punishment. Willie is a *passive warden* if he merely observes and analyzes the transmitted data. He is an *active warden* if he introduces noise to make Alice and Bob's task more difficult.

In the information age, there are several application scenarios for steganography.

1) Steganography may be used to communicate over public networks such as the Internet. One may embed bits into inconspicuous files that are routinely sent over such networks: images, video, audio files, etc. Users of such technology may include intelligence and military personnel, people that are subject to censorship, and more generally, people who have a need for privacy.
2) Steganography may also be used to communicate over private networks. For instance, confidential documents within a commercial or governmental organization could be marked with identifiers that are hard to detect. The purpose is to trace unauthorized use of a document to a particular person who received a copy of this document. The recipient of the marked documents should not be aware of the presence of these identifiers.
3) Timing channels can be used to leak out information about computers. A pirate could modify the timing of packets sent by the computer, encoding data that reside on that computer. The pirate wishes to make this information

Manuscript received February 17, 2007; revised October 2, 2007. This work was supported by the National Science Foundation under Grants CCR 02-08809 and CCR 03-25924. The material in this paper was presented in part at the 40th Conference on Information Sciences and Systems (CISS), Princeton, NJ, March 2004.

Y. Wang was with the Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA. She is now with Qualcomm Flarion Technologies, Bridgewater, NJ 08807 USA (e-mail: yingw@qualcomm.com).

P. Moulin is with the Beckman Institute, Coordinated Science Laboratory, and the Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA (e-mail: moulin@ifp.uiuc.edu).

Communicated by Y. Zheng, Guest Editor for Special Issue on Information Theoretic Security.

Color version of Figure 4 in this paper is available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIT.2008.921684

0018-9448/$25.00 © 2008 IEEE

Figure 4 Screenshot - Line Splitting

4. **Character Splitting**
    a. Work in process
5. **Character recognition**
    a. Work in process
6. **Consolidation into a PDF file.**
    a. Not started

Section 6 - Layout Analysis

Layout analysis is the process of identifying and labeling the regions of interest in the scanned image of a text document.

We need to identify the textual zones from non-textual ones and their arrangement in their correct reading order, so that the copy paste operations copy them in the right sequence.

For layout analysis, we use three algorithms to implement and try to find which one is better. All of our input is binary image which the text is black and background is white.

Algorithm 1:

This is the simplest one, albeit the results are not as good.

1. Set a threshold (240 for example).
2. Iterate the width, compute the average pixel value of each column, if the average pixel value is greater than 240, then we draw a split line. After that, we can get several columns divided by vertical split line.
3. Do the same thing- iterate the height. Compute the average pixel value of each row, find the average pixel value greater than threshold then draw a horizontal split line. However, before drawing a horizontal split line, make sure it is at least 25 pixels from the previous line above the threshold. This makes sure that this algorithm only separates blocks of text, not individual lines of text.

Pros: low computation time

Cons: Only satisfactory performance

Algorithm 2:

This is the same as algorithm 1, however, instead of using a threshold on the average value of pixels, we compute the FFT (Fast Fourier Transform) of a column or a row depending on whether we are doing a horizontal pass (i.e. Step 2 of algorithm 1) or vertical pass (i.e. Step 3 of algorithm 2). Then we apply a threshold on the magnitude of the high frequencies of the FFT. This detects large variation between the pixels which would be the case if we are dealing with text (there will be a lot of variation between black and white).

Pros: better performance, not very high computation time (FFT can be done very fast in modern day computers)

Cons: still not as good as algorithm 3

Algorithm 3:

The purpose of the algorithm is to group contiguous pixels under one label. Each block of text will be assigned a label.

First few steps are preprocessing.

1. Invert the image such that the text is white and background is black.
2. Apply dilation with a suitable kernel size to thicken the text.
3. Blur the image to remove little noise.
4. Convert the image to monochrome by thresholding.

Next, we move onto actual layout analysis.

5. Set two arrays both of size *WIDTH x HEIGHT*:
6. One should be boolean (0 or 1), showing whether the pixel has been grouped (labeled) or not. Let's call this *LABELED_OR_NOT*.
7. Another should not be binary. It should be integer. It should save the actual label (group) of the pixel. Let's call this *PIXEL_GROUP*. The label is also the label of a block of text.
8. Run the following algorithm.

```
Initialize i =1

Loop until LABELED_OR_NOT(x,y) contains as many ones as the number of white pixels in the
preprocessed image.

    Iterate from the top left pixel (x,y) in any direction*
    *see Section 8 for the actual implementation details

        If LABELED_OR_NOT(x,y)==0:

            Flood fill the pixels with label i
            Update all values PIXEL_GROUP(x,y) to i
            For all these pixels, also set  LABELED_OR_NOT(x,y) to 1

    Update i = i + 1 (so that in the next iteration for the next object, PIXEL_GROUP(x,y) will
be assigned the new value)
```

9. Find the 4 points of each block. We iterate the pixel in each block to find the maximum and minimum of x axis and y axis. Then *(min_x, min_y), (min_x, max_y), (max_x, min_y), (max_x, max_y)* is 4 corners of each block.
   The output of the algorithm is obtained the sets of 4 coordinates each for all possible blocks of text.

Pros: Excellent performance

Cons: High running time

We are working to reduce the running time of Algorithm 3 and use that in the final implementation alongside Algorithm 2 (providing the user a choice between the two).

Section 7 - Testing and Performance Measurement

We will test whether our algorithm can identify text in different formats (font, margin, line space, text direction), whether our algorithm can distinguish between words and pictures. We will test in two column and three column layouts.

The performance will be measured by the accuracy of text recognition and conversion. The task should be accomplished in a reasonable time frame (no more than a few seconds per page).

Section 8 – Challenges

In Algorithm 3 for layout analysis, the running complexity was large because for each round of iteration, we had to start from the first pixel. We took an alternate approach. We decided to randomly generate a pair of coordinates, and keep generating them, until we land at an unlabeled pixel.

It takes maximum 3 or 4 random guesses to land at an ungrouped pixel for the large blocks and maximum 20-30 guesses for the smaller blocks. It is hence more efficient than the iteration technique.

It takes significant trial and testing to devise the thresholds for line splitting and layout analysis. We choose thresholds that would be sufficient for major paper sizes like A4, letter, etc. scanned in 300 DPI to 600 DPI (the most common scanning resolutions).

Specifying other parameters like the size of dilation kernel, etc. also takes a lot of calibration.

Section 9 - Limitations

We assume that our input satisfies the following requirements:

- Paper should be placed on a flat surface. A simple projective transform should be enough to straighten the page.
- Should contain the entire content of the page (no missing parts)

References

[1] https://opencv.org

[2] https://www.tensorflow.org

[3] http://homepages.cae.wisc.edu/~ece539/project/f03/sarkar-ppt.pdf

[4] http://yann.lecun.com/exdb/lenet

[5] https://en.wikipedia.org/wiki/Document_layout_analysis