Learning how to Extract Information from Scanned Documents

by Natasha Consul

S.B., Massachusetts Institute of Technology (2017),
Computer Science and Engineering

Submitted to the
Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

Massachusetts Institute of Technology

September 2017

Author: ____
Department of Electrical Engineering and Computer Science
September 1, 2017

Certified by: ____
Regina Barzilay
Professor
MIT Computer Science and Artificial Intelligence Laboratory
September 1, 2017

Accepted by: ____
Dr. Christopher Terman, Chairman, Masters of Engineering Thesis Committee

Learning how to Extract Information from Scanned Documents

by

Natasha Consul

Submitted to the Department of Electrical Engineering and Computer Science
September 1, 2017
In Partial Fulfillment of the Requirements for the
Degree of Master of Engineering in
Electrical Engineering and Computer Science

ABSTRACT

In recent years, there has been a lot of interest in methodologies for extracting information from text-based documents. Specifically in the medical field, a recent challenge has been to extract information from different types of scanned medical documents, such as patient registration forms, prescription order forms, and medical history forms. The lack of structure and large variety of information across these documents makes it difficult to automate the process of retrieving data. Today, humans read the documents and manually record the key pieces of information.

This thesis focuses on the process of learning how to automate information extraction from a variety of scanned medical documents from a Computer Vision standpoint. We look at two different approaches: an *object-detection approach* and a *text-spotting approach*. In each method, we attempt to extract a subset of document fields correctly. We evaluate and compare the results for solving the problem at hand.

Acknowledgments

I would like to thank everyone who has supported me throughout my thesis. I would like to thank my supervisor Regina Barzilay for her guidance and for giving me the opportunity to work with her on this project, the other students in her lab, and my family and friends. I would also like to thank HCL Technologies for partially funding my thesis.

Table of Contents

List of Figures

Chapter 1

Introduction

Digitizing medical records dates back to the late 1960s, when Lockheed Corporation made the first efforts to create Electronic Health Records (EHRs) in order to electronically store a patient's medical data [3]. Since then, technology has advanced greatly from having mainframe computers, to people owning personal computers, to the launch of the Internet. Due to these technological advances, by the 2000s, the development of Electronic Health Records grew to be more usable, standardized, interoperable, and affordable [3]. Today, EHRs have become complete records of a patient's health, containing medical and clinical data derived from multiple hospital sites or health systems, accumulated overtime[1]. The data within the records allow healthcare providers to track patient data over time, identify patients who are due for preventive visits and screenings, monitor how patients measure up to certain parameters (such as vaccinations and blood pressure readings), and improve overall quality of care [2]. Digital records have many advantages in that they are universally accessible (not confined to a single health center), searchable (by keyword or phrase), and maintainable. Healthcare providers and insurers should be able to easily and securely access their clients' records.

A major challenge today is efficiently generating the electronic records. Many patient forms and clinical documents must be converted from paper documents to digital records. Today, documents are scanned and processed by hand. Humans read through the scanned documents and manually enter the important pieces of data. This process is slow, prone to errors, and labor intensive and can be improved through automation.

1.1. Problem Statement

There are many challenges with automatically extracting information from scanned medical documents. Three big challenges are:

(1) Documents are often processed as scanned images, in non-machine readable formats

(2) Documents contain different pieces of information to extract

(3) Documents vary greatly in format and structure

The first challenge has been tackled by using Optical Character Recognition (OCR) to convert images into machine-readable formats. OCR tends to be error-prone, thus requiring a lot of post-processing. This thesis explores methods that use Computer Vision techniques and neural network models to process the images. The second and third points above are the main challenges with building such models. Due to the large variety of information across the documents, it is difficult to train a model to extract all information. Additionally, the varied structure and format of the documents makes it challenging to build a model that can accurately extract information from all types of documents.

This thesis describes two methods for extracting information from the images: an *object-detection approach*, and a *text-spotting approach*. Both methods focus on extracting five pieces of information (Name, Date, Address, Date of Birth, and Phone Number). The accuracy of each method is evaluated based on their ability to identify each of these fields correctly. For example, if a patient's name in a document is John Smith, the system should tag the location of "John Smith" within the document as the patient's name. The thesis focuses on retrieving only five fields because they are commonly found in most medical documents.

1.2. Approaches

1.2.1. Object-Detection Approach

In the object-detection approach, each of the five fields of interest were treated as unique objects. A model following the Fully Convolutional Regression Network architecture was used to identify these different objects/fields. The network architecture contained several convolutional layers followed by a regression layer. The goal of the model was to be able to learn the features of the five field labels (Name, Date, Address, Date of Birth, and Phone Number), regardless of how they appeared. Then, the model could be used to tag the surrounding text as the appropriate field.

1.2.2 Text-Spotting Approach

In the text-spotting approach, a model following the Fully Convolutional Regression Network architecture mentioned above was pre-trained to spot text in noisy images. When applied to text-filled medical documents, the model could identify text from non-text, creating a collection of textual elements. Then, the system recognized the textual content of the elements, and from the resulting corpus of words, the fields of interest were localized and tagged as the appropriate field.

1.3. Thesis Organization

The remainder of this thesis will be organized with the background of the two approaches described above in Chapter 2, the first approach’s development and results in Chapter 3, the second approach’s development and results in Chapter 4, the future opportunities and future works in Chapter 5, and the conclusion in Chapter 6.

Chapter 2

Background

This section will provide background information surrounding the problem of extracting information from scanned documents and an overview of the systems described in the two approaches described above. We will describe related works of each approach and the architectures used in each of the systems.

2.1. Related Work

A lot of work has been done surrounding information extraction for medical documents. Most of the work has involved understanding the textual content of the documents by using rule-based techniques or Natural Language Processing [4][5], once the text is in a machine-readable format.

When hospitals digitize their medical documents, they need to be converted from image scans into machine-readable formats. There are different techniques that exist to convert the image scans into machine-readable formats, such as Optical Character Recognition (OCR). OCR uses pattern recognition to identify both typed and handwritten characters and words within a document. There has been work done to extract fields from biomedical documents using OCR [6][7]. However, there is a lot of post-processing manual labor needed to correct recognition errors and deduce the context or relations amongst the words. OCR tends to work better when digitizing content of structured line-based text documents [18]. These structured documents contain text of consistent fonts and sizes, and are usually split into horizontal lines throughout the document [18]. Medical documents tend to lack structure and have a variety of fonts and sizes of text within a single document. The complications brought by OCR can be avoided by using a method that does not use OCR at all (such as object detection and localization), or by using a method that reformats the input prior to applying OCR to make the process more efficient (such as text-spotting). There has not been much research done in these areas with regards to extracting information from medical documents. However, there has been research in these areas with regards to other applications, described in the following subsections.

2.2. Object Detection and Localization

2.2.1. Description

The goal of object detection and localization is to be able to simultaneously localize and classify objects within images. A lot of research has been done in this area, using Computer Vision techniques for pattern recognition, although very little pertaining to text recognition and localization within text-based documents. After learning about previous research in the area, I explored applying similar methods to detecting and localizing text.

2.2.2. Related Works

One of the earlier but more recent breakthroughs for object detection and localization was the development of the R-CNN architecture, created in 2014 [8]. This system has been the basis for many of object detection systems that came after it. The R-CNN system takes input images, determines region proposals for each potential object, resizes the region proposals to a fixed size, and runs the resized region proposals through a deep Convolutional Neural Network (CNN) to classify each of the objects, shown in Figure 2-1.

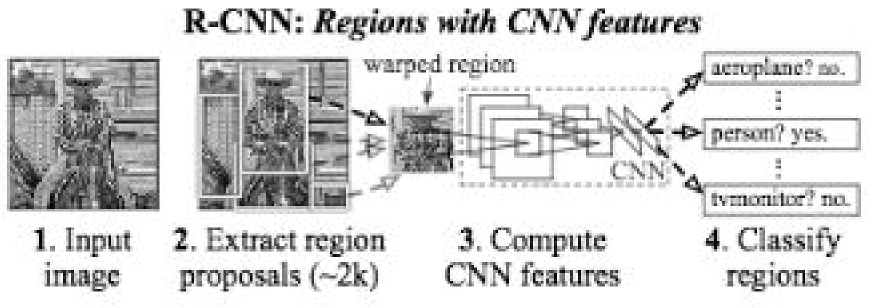


Figure 2-1: R-CNN system overview.
The system begins with (1) an input image, (2) generates 2000 region proposals, (3) computes features using a deep CNN model, and (4) classifies each region as an object [8].

The R-CNN architecture differed from systems that came before it because older systems focused mainly on SIFT [9] and HoG [10] architectures. SIFT and HoG systems break images into smaller cells and use histograms and sliding-window techniques for object detection. R-CNN utilized the idea of first generating region proposals using selective search [11], and then classifying each region using a CNN. The CNN was pre-trained under supervised learning on an ImageNet dataset [12] and evaluated on the PASCAL VOC datasets [13]. The ImageNet datasets contain scene-like images with cluttered backgrounds and varied poses/positions of the objects, similar in complexity as the target evaluation datasets. The objects in the images fall across 200 classes, all labeled with bounding box coordinates. The R-CNN architecture showed a 30%

improvement in the precision for detecting objects, compared to previous systems [8]. The results gave two insights: First, applying deep CNNs to region proposals worked well. And second, using a pre-trained CNN that was trained on a large ImageNet dataset solved the problem of having a smaller dataset to work with for the target problem of multi-object detection.

The R-CNN architecture provided a baseline for other object detection systems that came later: DenseCap (2015) [14], Faster R-CNN (2015) [15], and YOLO (2016) [16], which all aimed to detect, localize, and classify objects in datasets similar to the R-CNN datasets. The DenseCap system aimed to provide dense captions for images by first localizing objects, and then annotating the image with dense captions. The system used a Fully Convolutional Localization Network (FCLN) architecture, which was composed of a Convolutional Neural Network, followed by a Recurrent Neural Network language model for generating the dense captions. The use of a CNN for object detection was inspired by the R-CNN architecture [8], but in the FCLN design, there were no external region proposals necessary. Instead, the system used a Localization layer which proposed regions of interest by regressing bounding box coordinates on pre-determined anchors, where each anchor would be the center of the bounding boxes of various aspect ratios [14]. This system aimed to classify and localize objects, while also combining the classifications into complex captions through the RNN language model. An example of this is shown in Figure 2-2.

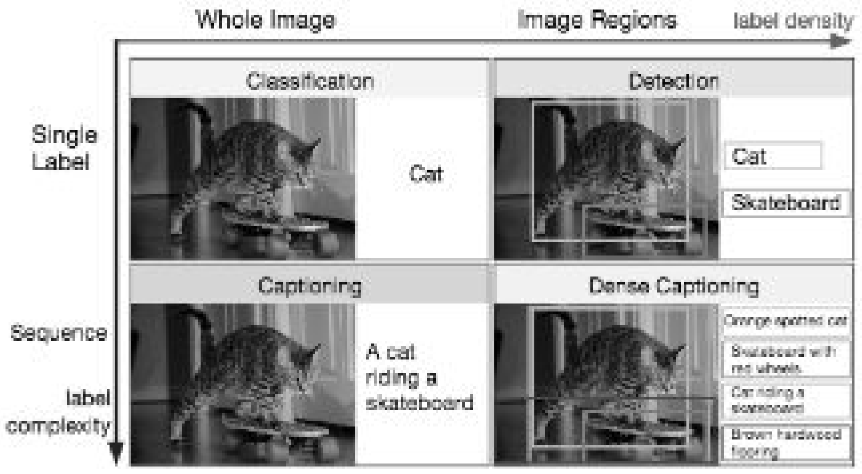


Figure 2-2: DenseCap Example
The figure depicts the object recognition, localization, and caption generation for a single input image [14].

The Faster R-CNN architecture makes improves upon the R-CNN architecture by using a Region Proposal Network (RPN) that shares features with the CNN, making the region proposal step cost-free [15]. This is shown in Figure 2-3. The region proposal step in R-CNN took as much computation as the detection step [8]. Faster R-CNN uses a single, unified network to operate two modules: the first module handles region proposals based on features from the CNN feature output layer, and the second module handles object detections on the region proposals. Similar to in DenseCap, anchors are used in the region proposal module. The module runs a sliding-window on the CNN output map and the center of each window is an "anchor", from which the system regresses bounding box coordinates for multiple boxes of different aspect ratios [15]. Sharing features between the RPN and classification modules overall makes the system more efficient than the original R-CNN design.

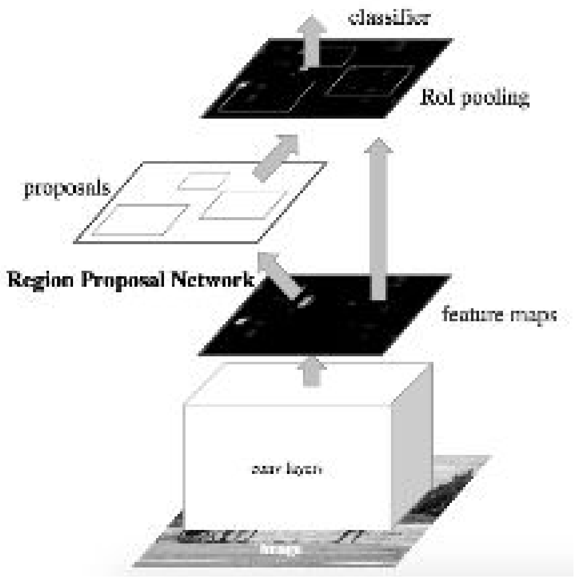


Figure 2-3: Faster R-CNN architecture breakdown
The Faster R-CNN architecture uses the feature mapping from the CNN for both, determining the region proposals (RPN) and for classifying each region as the respective object.

The YOLO architecture frames the object detection problem as a regression problem to determine object bounding box coordinates and class probabilities, rather than using classifiers [16]. While still using a deep CNN similar to the other architectures, YOLO simultaneously determines bounding boxes and object classes by processing the input image as a whole. Each input image is divided into a fixed number of grid cells. The system predicts a set number of bounding boxes centered each grid cell, along with their confidence scores as being objects. Then, if the confidence is high enough for containing an object, the system predicts the object's class. This is depicted in Figure 2-4. The YOLO system uses a single CNN, allowing it to determine image features, object bounding boxes, and object classes with a single network. This allows the network to perform faster than the other architectures above, making it suitable for real-time object detection.

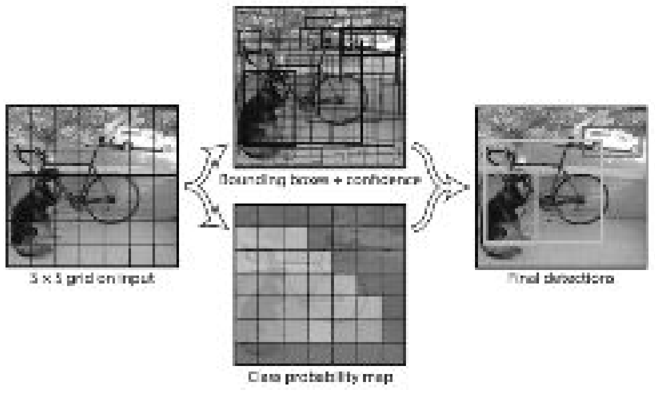


Figure 2-4: YOLO object detection and localization diagram [16]
The YOLO architecture computes the confidence of objects for several bounding boxes (higher confidences depicted with thicker bounding boxes) at each grid cell of the input image. Then, the class of the object is determined for high-confidence object detections.

2.2.3. Conclusion

After learning about the recent advances in multi-object detection through the different architectures described above, I attempted to apply similar architectures to the problem of detecting document fields. Each of the systems above helped determine how to approach the problem from an object-detection standpoint:

(1) R-CNN gave a baseline understanding for the more advanced, deep learning algorithms used for multi-object detection.

(2) DenseCap provided background in using NLP to understand relations amongst objects once detected. For detecting document fields, this would be useful for determining which textual elements pertain to which field.

(3) Faster R-CNN provided a more efficient algorithm than R-CNN. For detecting document fields, the convolutional neural network would have to be pre-trained on images of solely

document fields from across several documents, with not only different filled text, but also different formats and sizes.

(4) YOLO gave a base understanding of a regression-based approach that used a convolutional neural network trained on the full images, and not sub-regions.

All four architectures work well when detecting and classifying objects in the ImageNet dataset [12], but the medical document images fall under a completely different dataset. My goal of the object-detection approach was to explore how well a system similar to these would work with text detection and classification as a specific field. I did not bother with using OCR, since that would be applied after the field's location would be known. I describe my findings in Chapter 3.

2.3. Text-Spotting

2.3.1. Description

The goal of text-spotting is to detect and recognize text in images [17]. Text detection refers to localizing the text, and text recognition refers to translating the detected text into a string of characters. Recently, there has been a lot of research in this area using deep learning techniques, because images encode a large amount of textual information which often capture the semantics of the images [18]. A majority of the research aims to detect and recognize text in natural scene images, though it can be applied to any type of image. After learning about previous research in the area, I explored applying similar methods to detecting and recognizing fields in scanned medical documents.

2.3.2. Related Works

Traditionally, text detection has focused on structured text-based images, where OCR works well. However, when applied to natural scene images, the OCR techniques fail due to the added clutter and noise from the background objects, and the scattered text of various fonts, colors, and sizes [17]. Some systems for detecting text in natural scene images have been designed to recognize individual characters, while other systems have been designed to recognize patches of text or entire words. One system that was recently developed to detect recognize individual characters uses two convolutional neural networks along with a lexicon [20]. The first CNN detects text using a sliding window approach and determines whether there is a character centered at each window. This allows the system to separate lines of text. The next CNN classifies each character as the respective character. Non-maximal suppression is used to identify spaces between characters and in the lines of text. Then, using a lexicon (list of candidate words), characters are combined to create the strings of words that appear in the images. This system assumes that we can use prior knowledge of the image to constrain the search for words to certain words. However, this requires different images to have different lexicons, depending on their application.

Other systems detect patches of text and recognize entire words, rather than single characters. One of these systems is described in Figure 2-5. The pipeline starts by proposing bounding box regions through a combination of region proposal methods. Next, the system filters out the false-positive bounding boxes using a random forest classifier. Then, a CNN is used to recognize text in each region, and regions are merged based on proximity. In the last step, the system applies thresholding to determine the final text sequence [17].

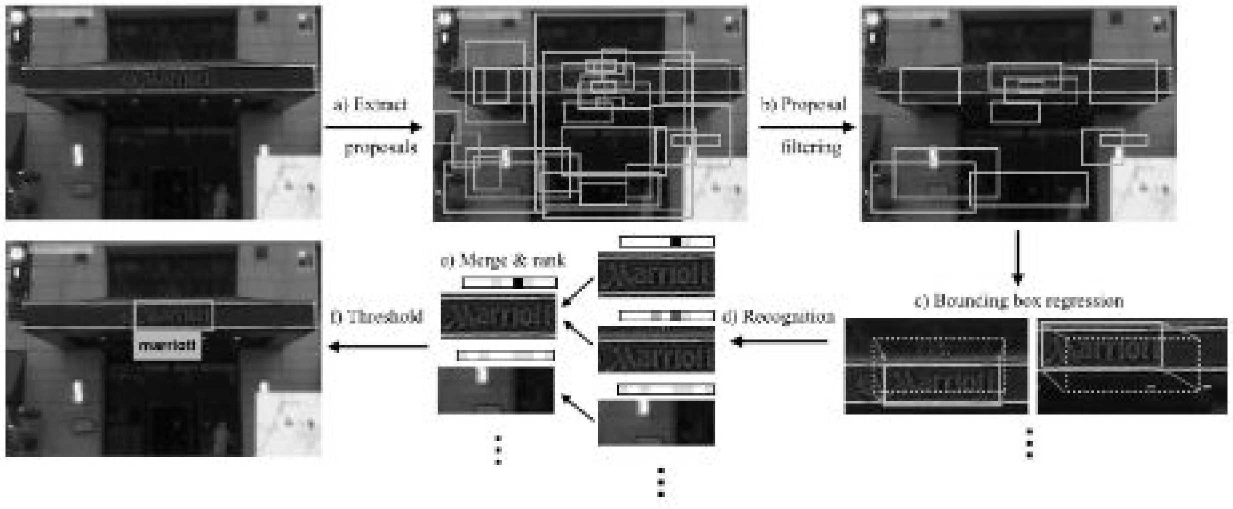


Figure 2-5: End-to-end pipeline for text extraction [17]
This example shows each step of the pipeline to detect and recognize text in the input image by extracting bounding box proposals, filtering the proposals, applying bounding box regression, and recognizing the text through merging results and applying thresholding.

The object-detection step of the pipeline was inspired by the R-CNN architecture described in Section 2.2.2 to use region proposals and resize the proposals prior to running them through the CNN. This system uses Edge Box proposals combined with aggregate channel feature detectors to propose bounding boxes. The Edge Box system takes advantage of contours to determine the bounding box containing the object [22]. Aggregate channel feature detectors aggregate pixels over the computed channels of the input images, resulting in lower resolution channels that can separate object from background [23]. The recognition step of the pipeline attempts to recognize whole words, rather than characters. The system runs each proposal through a deep CNN which has been trained to classify the word across a dictionary of ~90k words. This results in high accuracy because the synthetic dataset which the CNN was trained on contains images of words that are similar to the words in the real-world images that the system was built for.

Another system that also recognizes words in natural scene images uses a Fully Convolutional Regression Network (FCRN) architecture, based off of YOLO described in Section 2.2.2. [19]. The FCRN architecture makes improvements on the system in [17] described above. Similar to the flaws of R-CNN, the system described in [17] is inefficient in the region proposal step of the detection pipeline. The FCRN removes the need for region proposals by performing prediction at every location of the image. It not only predicts class labels at each location (text or no text), but it also predicts the bounding box coordinates of the word enclosed at each location, which is inspired by the YOLO system. The deep model begins with a CNN which determines the feature mapping of the input image at various scales. Then, at each scale, a dense text predictor is used to regress the confidence of containing text and the bounding box coordinates surrounding it. Although the FCRN architecture adopted YOLO's idea to use regression to determine the bounding box coordinates of text objects, the FCRN model is more space-efficient because the regression layer comes immediately after the CNN feature mapping. The YOLO system has two fully connected layers containing ~90% of its parameters prior to the regression step, which the FCRN architecture removes. Similar to the system in [17], the FCRN model is trained on a synthetic dataset. This dataset is generated by placing text from IDCAR and Google StreetView datasets onto images from the PASCAL VOC database. The text placement takes into account the image's texture and color in order to better-integrate the text into the image and make it appear more real.

2.3.3. Conclusion

After learning about the recent projects that used deep learning for identifying text in natural scene images, I attempted to apply similar architectures to detect and recognize the document fields. Each of the systems above helped determine how to approach the problem from a text-spotting standpoint:

(1) The first system described above introduced the use of CNNs for character-level detection and recognition. This method also used a lexicon to better identify words. With regards to medical documents, the lexicon could be extremely large, since many fields contain personal information (such as name and address) which can have endless possibilities.

(2) The second system described above showed how to use CNNs for word-level detection and recognition. Although it was not the most efficient system, it was similar to R-CNN in that it was a baseline for systems to come and for systems to build off of. This gave a good understanding of how CNNs can be used for text detection.

(3) The third system provided a regression-based deep network for word-level detection, more efficient than its predecessor. Since it was trained on a synthetic dataset, the same architecture could be used on any synthetic dataset, making this architecture seem promising.

As mentioned previously, these systems were created mainly to detect and recognize text in natural scene images, which have cluttered backgrounds and varied font sizes, and colors. The medical document scans fall under a different category where documents have less noise but may have varied fonts. My goal of the text-spotting approach was to see how well deep learning could be used to detect and recognize text. I describe my findings in Chapter 4.

Chapter 3

Object Detection and Localization Approach

This chapter describes how I applied the object detection and localization approach to extracting data from medical forms. The chapter begins with an overview of the system I built, and then describes the dataset I generated, the deep network architecture, the experiments, and the results.

3.1. System Overview

I decided to build a regression-based system that combined YOLO [16] and FCRN [19]. Both architectures used a deep CNN followed by a regression step, but the FCRN system did not have fully connected layers, which YOLO did. I also dismissed the fully connected layers in my model. I used the regression step described in the YOLO design, so that my system would not only determine the confidence of there being text at each location in the image, but would also classify the text as the type of field. If the model found that a text element was not a field or the confidence of being text was too low (representing white space), it would fall under a separate classification. Although the FCRN system was designed for text-spotting, my model followed it closely until the regression step, since it had a comparable accuracy to YOLO while reducing the number of parameters greatly. The system was built in Python and TensorFlow.

3.2. Dataset

My dataset consisted of 8,000 images of populated hospital patient information forms. I extracted and generated these images as follows:

(1) Downloaded hospital patient forms [23].

(2) Converted PDF files to images (JPGs).

(3) Used Photoshop to manually compute the (x, y, width, height) pose parameters for each field of interest in every form. The (x, y) coordinate was the top-left coordinate of the bounding box, and the width and height were the dimensions of the bounding box.

(4) Used the Python Image Library (Pillow) to populate the bounding boxes with random text for each of the 8,000 forms.

I generated the images using Pillow, which takes an empty image and the coordinates of the field as input. It then populates those fields with predetermined values. I used eight templates of hospital patient information PDFs and generated 1000 PDFs of each one, resulting in a dataset of 8,000 forms. The eight templates are shown in Figure 3-2.

Coordination of Benefits	Dental Patient Registration
Geriatric Assessment Form	Insurance Verification Sheet
Massage Client Intake Form	Medical Travel Form
Patient Information Form	Surgical Clearance Form

Figure 3-2: Form templates used to generate dataset for object-detection approach

I selected the eight forms in Figure 3-2 based on a couple factors. First, I wanted each form to contain a majority of the five target fields that I would focus on extracting. Second, I wanted each form template to be different in terms of format/structure. I chose to use only eight templates because it would be a good starting point for me to see how an object-detection network would perform on a dataset with less varied data.

3.3. System Architecture

In this section, I introduce the model I developed for the object-detection approach. I structured the convolutional layers of my CNN to follow the VGG-16 design [24] by using several convolutional layers with small dense filters. The architecture, shown in Figure 3-1 is composed of nine convolutional layers, each with 3x3 filters. The first three filters have a depth of 16, the next three filters have a depth of 32, and the last three filters have a depth of 64. Every third convolutional layer is followed by a max pool operation, with 2x2 filters and a stride of 2. Additionally, all nine convolutions employ zero padding to maintain tensor dimensionality.
I add a linear filter to regress values on the output feature map following the convolutional layers, completing the Fully Convolutional Regression Network design.

I model the deep convolutional layers of the network similar to how Gupta does in his FCRN system [19], but the regression step matches the YOLO design closely. I take the output feature map of the last convolutional layer and slide a 5x5 linear filter with a depth of 5 across it, so that the filter regresses 5 values at each position in the feature map *(c, x, y, w, h)*. These 5 values are:

c: object classification (field type)

x: x-coordinate of the top-left corner of the field’s bounding box

y: y-coordinate of the top-left corner of the field’s bounding box

w: width of the field’s bounding box

h: height of the field’s bounding box

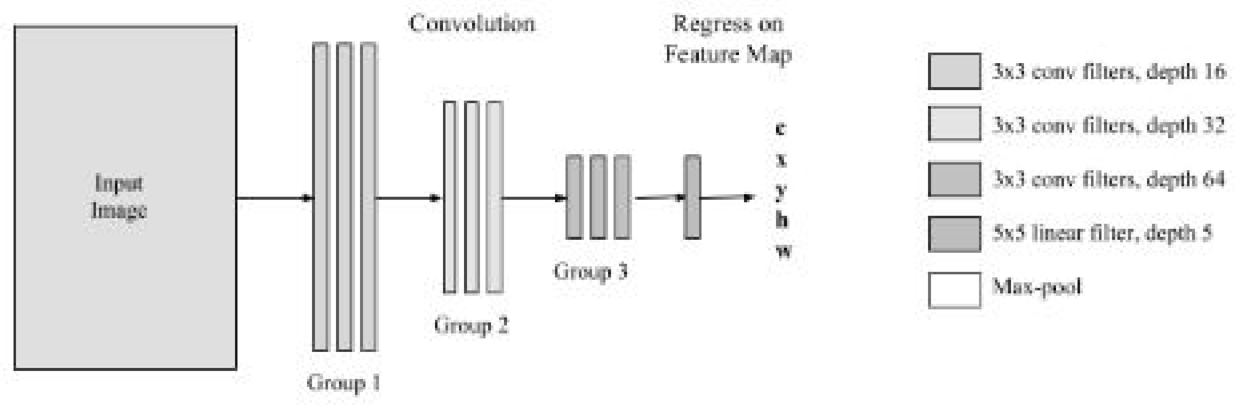

Input Image
Convolution
Group 1
Group 2
Group 3
Regress on Feature Map
c
x
y
h
w
3x3 conv filters, depth 16
3x3 conv filters, depth 32
3x3 conv filters, depth 64
5x5 linear filter, depth 5
Max-pool

Figure 3-1: Model Architecture
This figure shows the model architecture for the object-detection approach.

By applying a 5x5 filter at each position in the feature map, I remove the added complication of creating a Region Proposal Network (used in R-CNN and Faster R-CNN) and am able to systematically detect objects/fields in every section of my input images. If my input images have dimensions WxH, after I finish running them through the nine convolutional layers, the output feature mapping has dimensions $\frac{W}{8}$ x $\frac{H}{8}$ due to the three max pools. Thus, when I run my feature map through the linear filter for regression, I am able to detect objects/fields for every 8px by 8px region in my input images.

In the next section, I describe my experiments. The architecture described above is the model that I started with and modified throughout my experiments. Based on the results of each experiment, I tweaked my model's architecture to better fit the goal at hand: detecting and localizing the five fields of interest in hospital patient documents.

3.4. Experiments

The goals of my model were to look at each 8x8 section in the input images and determine:

(1) Whether or not there was a field in that section of the image

(2) The field name if the field exists in the respective region by outputting a number 1-5 (and outputting 0 if there was no field)

(3) The bounding box parameters of the field in the image

In this section, I describe the field localization experiments I ran to achieve the above goals and also present and discuss the results I obtained from each one.

3.4.1. Experiment 1 - Predicting the field's class and its pose parameters

In this experiment, I attempted to localize target fields in each image by having the model first predict whether a given 8x8 space contains a field (or portion of a field) or not. If the model determined a given space to be a field or portion of the field, it predicted the identity of this field. It additionally predicted the field's pose parameters (x, y, width, height), denoting the location and size of the bounding box enclosing the field.

My loss function for this experiment, which used Euclidean distance, took five arguments - field classification and the four pose parameters. The loss converged almost immediately -

within two epochs - to an extremely large number (~5900), indicating that the model was not learning. I concluded from this experiment that it was not necessary to train on the bounding box values, as the location of the field in the original image could be deduced based on the receptive field.

3.4.2. Experiment 2 - Predicting the field's class with (80,104,5) label matrix

Each input image had dimensions (638, 825, 1). After three max pools (one after each group of three convolutions), the final CNN layer produced an output feature map with dimensions (80, 104, 1). The original image's dimensions were reduced by a factor of 8 to produce the output feature map; each cell in the feature map had a receptive field of 8x8 in the original image.

I created a label matrix with one-hot encoding. In the 80x104x5 label matrix, each of the 80*104 cells had a five element list, with all but the *ith* elements being 0, where *i* is the cell's target field.

For this experiment, I computed loss using cross entropy; specifically, I used the tf.sigmoid_with_logits function, which allows classification of multiple non-dependent objects in a single image. I tuned two hyperparameters, learning rate and dropout, in order the find the best model.

3.5. Results

The results from running the third experiment by varying the dropout and learning rate are shown in Figure 3-3. The highest accuracy on the test set - **0.694** - was achieved with a dropout of 1.0 and a learning rate of 0.05. Since this value of dropout is the probability of keeping a given node, the network performed optimally without dropout.

Dropout	**Learning Rate**	**Training Accuracy**	**Test Accuracy**
1.0	0.01	0.674360	0.681452
1.0	0.05	0.684245	0.694294
1.0	0.1	0.687421	0.679188
0.9	0.01	0.657663	0.652849
0.9	0.05	0.672116	0.663387
0.9	0.1	0.672174	0.669237
0.8	0.01	0.658459	0.668347
0.8	0.05	0.680176	0.683636
0.8	0.1	0.672009	0.667283

Figure 3-3: Accuracies for Experiment #2 when varying the dropout and learning rate

I began to solve the problem at hand by first generating synthetic data (Section 3.2), and by creating a network (Section 3.3). After evaluating my initial results, I experimented by making various changes to the model: I modified the regression architecture, I changed the way that I computed loss, and I experimented with different hyperparameters. In the remainder of this section, I will analyze the results from the experiments.

3.5.1. Analysis

After my first experiment, I concluded that it is difficult to measure the accuracy for finding the bounding box coordinates and dimensions of a field **while also** classifying the field. When calculating the loss, I was unable to separate the calculations of loss due to incorrect classifications and loss due to incorrect bounding box parameters. Loss should be calculated differently when looking at classification errors vs. bounding box parameter errors. I had to either:

(1) Create two separate networks: one for labeling fields throughout the images, and one for determining the bounding box parameters

(2) Create a network to only label the fields throughout the images, and then deduce the bounding box parameters based off of the locations of the fields on the feature map and the receptive field

I decided to take the second approach by building a network that could classify the fields throughout the images. In the second experiment, I started seeing results that showed promise that the model was learning. At each epoch, the accuracy improved slightly, and eventually it would converge to some accuracy level between 65-70%. To improve the accuracy, I did not change the internal layers of the model, but I tried experimenting with different hyperparameters. Although the results changed when changing hyperparameters, it was difficult to make any claims regarding the effects of the hyperparameters, since all of the accuracies were very close.

When I looked closely at the results and the output field classifications, I realized that the model could identify non-text and label it correctly, but the model did not learn how to identify text elements as being fields, being portions of fields, or not being fields. Image cells containing text elements had mixed classifications, and seemed to be correct “by chance” every once in while. I feel that was why the accuracies varied slightly and could not improve beyond 65-70%.

The next point of analysis was why my model did not perform like the YOLO system or the FCRN system, even though it was modeled off of both of them. The goal of the YOLO system was to detect, localize, and classify multiple objects in images that were in the PASCAL VOC and ImageNet datasets. Those datasets had no overlap with the dataset I generated. Additionally, the datasets that YOLO was trained on were much larger than my dataset. The FCRN system aimed to differentiate between text and non-text, which is a simpler task than what I aimed to do. The idea of dividing the image into a grid and classifying each grid cell as text or nontext worked well. In my case, each cell would contain one of three things:

(1) No text (blank space)

(2) Subset of a field (few characters belonging to a field of interest)

(3) Subset of a non-field (few characters not belonging to a field of interest)

Although the model captured surrounding pixel information in the feature map (since a 5x5 linear filter was used), the model did not seem to learn how to differentiate amongst the fields. This is because a subset of one type of field, a subset of another type of field, and a subset of a non-field could all appear the same, while still having different classifications. Taking into account the training label matrices across all images, the variability of features across the different field classifications was most likely very low, preventing the model from learning how to differentiate fields and performing well in that aspect.

Chapter 4

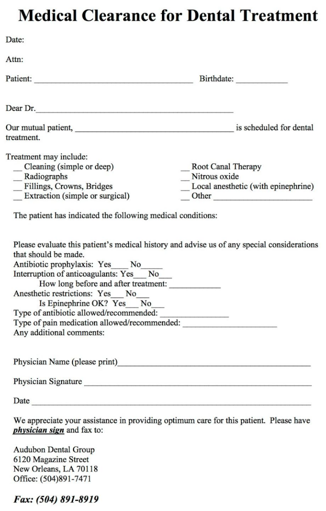
Text-Spotting Approach

This chapter describes how I applied the text-spotting approach to extracting data from medical forms. The chapter begins with an overview of the system I built, and then describes the dataset I generated, the system architecture, and the results.

4.1. System Overview

For the text-spotting approach, my goal was to build a system that could take a medical document image as an input, identify all of the patches of text or words along with their locations in the image, and recognize the character string in each patch/word. Then, from the collection of all words in the document, I would follow a rule-based approach to determine which text elements pertain to which fields.

As described in Section 2.3, there have already been several text-spotting systems created which can detect, localize, and recognize the text in images. These systems have been trained to identify text in natural scene images which have little textual structure and cluttered/noisy backgrounds. Medical document images lack structure (though not as much as text in natural scene images), but the backgrounds are usually plain and white with little noise. Additionally, the amount of text in medical documents generally far exceeds the amount of text in a natural scene image. Figure 4-1 provides examples of the different images.



Medical Clearance for Dental Treatment

Date:

Attn:

Patient: ______________________________ Birthdate: __________

Dear Dr.______________________________________

Our mutual patient, ______________________________ is scheduled for dental treatment.

Treatment may include:

__ Cleaning (simple or deep)	__ Root Canal Therapy
__ Radiographs	__ Nitrous oxide
__ Fillings, Crowns, Bridges	__ Local anesthetic (with epinephrine)
__ Extraction (simple or surgical)	__ Other ____________________

The patient has indicated the following medical conditions:

Please evaluate this patient's medical history and advise us of any special considerations that should be made.
Antibiotic prophylaxis: Yes____ No_____
Interruption of anticoagulants: Yes___ No___
How long before and after treatment: __________
Anesthetic restrictions: Yes___ No___
Is Epinephrine OK? Yes___ No___
Type of antibiotic allowed/recommended: ______________
Type of pain medication allowed/recommended: ___________________
Any additional comments:

Physician Name (please print)____________________________________

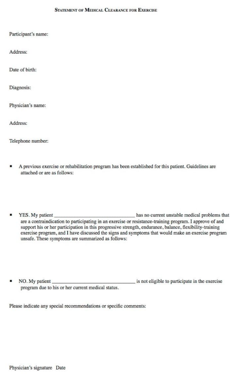
Physician Signature ____

Date ____

We appreciate your assistance in providing optimum care for this patient. Please have ***physician sign*** and fax to:

Audubon Dental Group
6120 Magazine Street
New Orleans, LA 70118
Office: (504)891-7471

Fax: (504) 891-8919



STATEMENT OF MEDICAL CLEARANCE FOR EXERCISE

Participant's name:

Address:

Date of birth:

Diagnosis:

Physician's name:

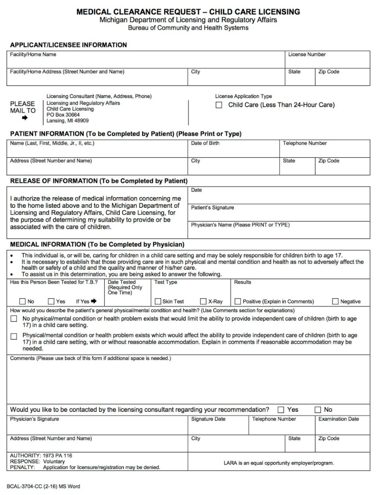
Address:

Telephone number:

- A previous exercise or rehabilitation program has been established for this patient. Guidelines are attached or are as follows:
- YES. My patient ____________________________ has no current unstable medical problems that are a contraindication to participating in an exercise or resistance-training program. I approve of and support his or her participation in this progressive strength, endurance, balance, flexibility-training exercise program, and I have discussed the signs and symptoms that would make an exercise program unsafe. These symptoms are summarized as follows:
- NO. My patient ____________________________ is not eligible to participate in the exercise program due to his or her current medical status.

Please indicate any special recommendations or specific comments:

Physician's signature Date



MEDICAL CLEARANCE REQUEST – CHILD CARE LICENSING
Michigan Department of Licensing and Regulatory Affairs
Bureau of Community and Health Systems

APPLICANT/LICENSEE INFORMATION

Facility/Home Name		License Number	
Facility/Home Address (Street Number and Name)	City	State	Zip Code

PLEASE MAIL TO
Licensing Consultant (Name, Address, Phone)
Licensing and Regulatory Affairs
Child Care Licensing
PO Box 30664
Lansing, MI 48909

License Application Type
☐ Child Care (Less Than 24-Hour Care)

PATIENT INFORMATION (To be Completed by Patient) (Please Print or Type)

Name (Last, First, Middle, Jr., II, etc.)	Date of Birth	Telephone Number	
Address (Street Number and Name)	City	State	Zip Code

RELEASE OF INFORMATION (To be Completed by Patient)

I authorize the release of medical information concerning me to the home listed above and to the Michigan Department of Licensing and Regulatory Affairs, Child Care Licensing, for the purpose of determining my suitability to provide or be associated with the care of children.	Date
	Patient's Signature
	Physician's Name (Please PRINT or TYPE)

MEDICAL INFORMATION (To be Completed by Physician)

- This individual is, or will be, caring for children in a child care setting and may be solely responsible for children birth to age 17.
- It is necessary to establish that those providing care are in such physical and mental condition and health as not to adversely affect the health or safety of a child and the quality and manner of his/her care.
- To assist us in this determination, you are being asked to answer the following.

Has this Person Been Tested for T.B.?	Date Tested (Required Only One Time)	Test Type	Results
☐ No ☐ Yes If Yes ➡		☐ Skin Test ☐ X-Ray	☐ Positive (Explain in Comments) ☐ Negative

How would you describe the patient's general physical/mental condition and health? (Use Comments section for explanations)

☐ No physical/mental condition or health problem exists that would limit the ability to provide independent care of children (birth to age 17) in a child care setting.

☐ Physical/mental condition or health problem exists which would affect the ability to provide independent care of children (birth to age 17) in a child care setting, with or without reasonable accommodation. Explain in comments if reasonable accommodation may be needed.

Comments (Please use back of this form if additional space is needed.)

Would you like to be contacted by the licensing consultant regarding your recommendation? ☐ Yes ☐ No

Physician's Signature	Signature Date	Telephone Number	Examination Date
Address (Street Number and Name)	City	State	Zip Code

AUTHORITY: 1973 PA 116 RESPONSE: Voluntary PENALTY: Application for licensure/registration may be denied.	LARA is an equal opportunity employer/program.

BCAL-3704-CC (2-16) MS Word

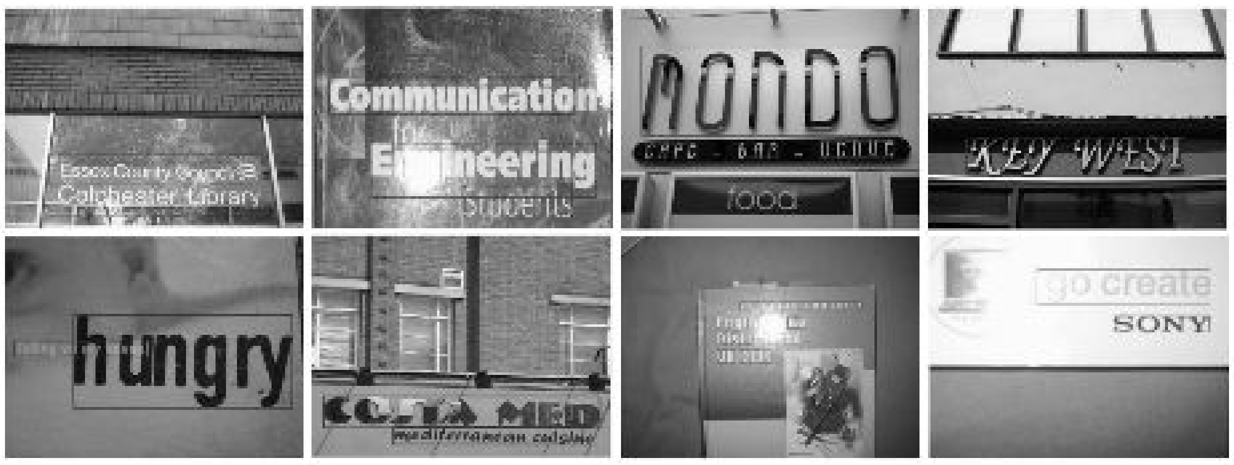
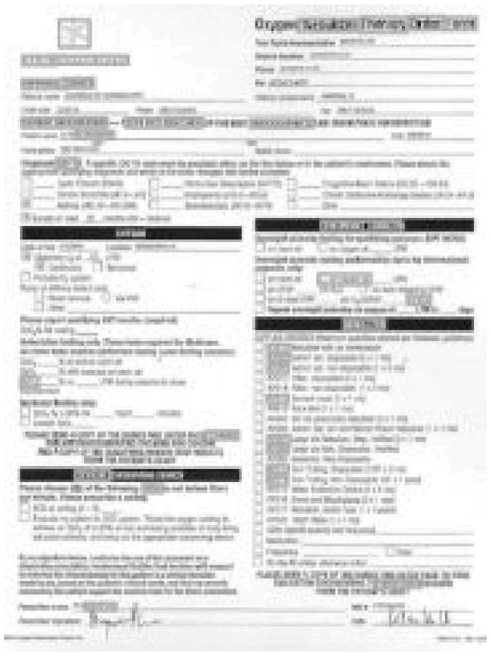
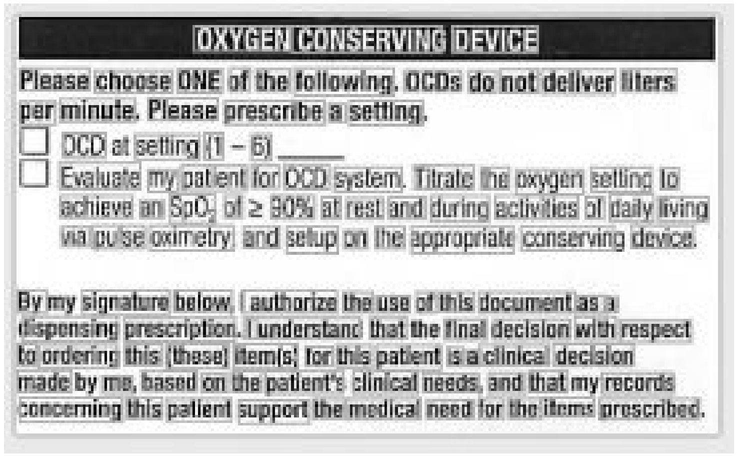


Figure 4-1: Sample Medical Document and Natural Scene Images [25][26]
The top row of images are sample medical scans, and the bottom two rows are natural scene images with text. The medical documents have many more words with consistent fonts and with little background noise, while the natural scene images have sparse text with varied fonts and colors and with cluttered/noisy backgrounds.

The text-spotting systems that already existed performed well when detecting and recognizing text in noisy images. I decided to test a text-spotting pipeline on some sample medical documents. The text-spotting system developed in [19] had an online demo, where I could upload images and see how well the system would perform on any image. I saw that on medical documents with sparse text (such as the second image in the top row of Figure 4-1), the system performed well, but on medical documents with heavy text (such as the third image in the top row of Figure 4-1), the system did not detect even 50% of the words. This was understandable, since the system was built to detect sparse text in noisy backgrounds. Next, I proceeded to test the pipeline on a text-heavy document, and a section of the same document. The pipeline could detect the text much better when only examining the image crop, as shown in Figure 4-2.



[illegible]



OXYGEN CONSERVING DEVICE

Please choose ONE of the following. OCDs do not deliver liters per minute. Please prescribe a setting.

☐ OCD at setting (1 – 6) _____

☐ Evaluate my patient for OCD system. Titrate the oxygen setting to achieve an SpO_2 of ≥ 90% at rest and during activities of daily living via pulse oximetry and setup on the appropriate conserving device.

By my signature below, I authorize the use of this document as a dispensing prescription. I understand that the final decision with respect to ordering this (these) item(s) for this patient is a clinical decision made by me, based on the patient's clinical needs, and that my records concerning this patient support the medical need for the items prescribed.

Figure 4-2: Text-spotting results on a full document scan (left) and on the bottom-left corner of the document (right)
The system took each image as an input and outlined the spotted text elements with light gray boxes. Using the full document as an input, the system *missed* over 75% of the text elements. Using the image crop (bottom-left section of the document) as an input, the system *detected* over 90% of the text elements.

This showed that the system could operate well on text-heavy inputs of smaller dimensions. This is because the system resizes inputs while maintaining their original aspect ratios, but if the original images are very large, the system will shrink them, making the text smaller and harder to detect/recognize. When input images are small, the system increase their dimensions, making the text larger and clearer to detect/recognize.

I designed my text-spotting system to use the pipeline developed in [19]. The system components were:

(1) *Image Splitting*: Splitting the documents into smaller sub-images so that the text within the image is large and of high resolution

(2) Text Detection and Recognition: Using the pre-trained text-spotting system in [19] to detect and recognize text. The location, character string, width, and height, for each text element would be written to an output xml file

(3) Rule-based Field Detection: Building a set of rules that could be used to identify the five fields of interest, based on the information in the output xml file (text element locations, dimensions, and character strings)

The majority of the system was developed in Python. The second component of the system, developed by the University of Oxford Visual Geometry Group, was created and trained in MatLab, but was executed as binaries.

4.2. Dataset

The dataset I used for the text-spotting system came from medical form templates found online. I downloaded 200 unique PDFs [23][25] and manually entered fake patient information into them. I chose a dataset different from the dataset described in Section 3.2. because in this case, I was using a pre-trained system and seeing how well it would perform on a variety of documents. Thus, variety and uniqueness of documents was important. Additionally, having a large variety of documents allowed the rule-based field detection (third system component described in Section 4.1.) to be more robust. After downloading the medical document templates and manually entering data into each PDF, I converted them to JPG files.

4.3. System Architecture

This section describes the architecture for the text-spotting system. As explained in Section 4.1., there were three components to the system: (1) Image Splitting, (2) Text Detection and Recognition, and (3) Rule-based Field Detection. In this section, I will describe each component's development and design in more detail.

4.3.1. Image Splitting

The first component of the system, Image Splitting, split the images into smaller sections. As explained in Section 4.1., the text-spotting system automatically resizes inputs to be larger (if they are small) or smaller (if they are large). The image crops would increase in size when passed to the text-spotting pipeline, which would make it simpler for the system to detect and recognize the text. I experimented with different methods for the splitting the images, described below.

The first method I used to split images followed an edge-detection approach. I aimed to separate the image into regions, where each region was a block of text. I used Python's OpenCV library to manipulate the input images. The edge-detection algorithm had four steps:

(1) Blurred the input so that each edge became thicker.

(2) Applied Inverse Adaptive Gaussian Thresholding to separate text from background, while reducing the background noise. All text would appear as white, and the background would appear black.

(3) Dilated the resulting image so that the white edges would grow larger, and text would bleed into nearby text, forming regions.

(4) Applied contours along between the dilated regions, where the image would be split.

Figures 4-3 and 4-4 demonstrate how this method worked well in some cases but not in others. These are potential reasons why the method did not always work:

(1) Different documents have different densities of text, so different levels of dilation were necessary depending on the image.

(2) Even after applying different levels of dilation, non-text elements affected the regions. For example, some medical documents are formatted using rectangular boxes, such as the third medical document example in Figure 4-1. These extra edges affected the dilation and contouring. This is shown in Figure 4-4.

(3) The edge-detection method could not focus only on textual edges and ignore non-textual edges, making it difficult to work with, given the various formats and text-density levels across all the medical documents.

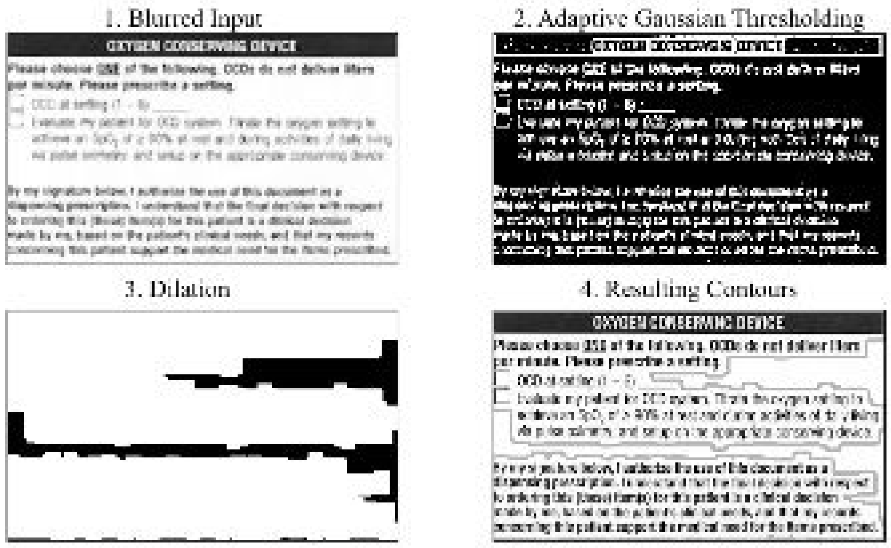


Figure 4-3: Edge-detection on dense-text input
This figure shows each step of the edge-detection algorithm on an input which has a large text density. The dilation level is relatively low since text elements are already close together, and the text regions have very little space between them. The fourth image (bottom-right) shows the resulting contours around the two separate regions.

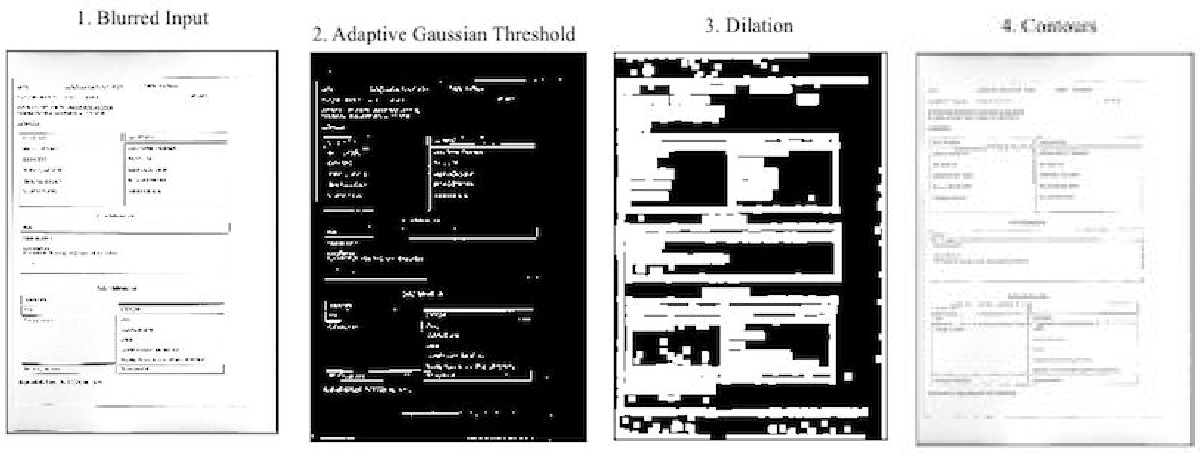


Figure 4-4: Edge-detection on input with textbox-formatting and sparse text
This figure shows each step of the edge-detection algorithm on an input with several outlined textboxes and a low text density. The dilation level is relatively high since the text elements are further apart. Additionally, the vertical and horizontal lines contribute to regions, making the regions difficult to split. This is shown in step 3. Step 4 shows contours that almost match the input, showing that the edge-detection algorithm fails on such inputs.

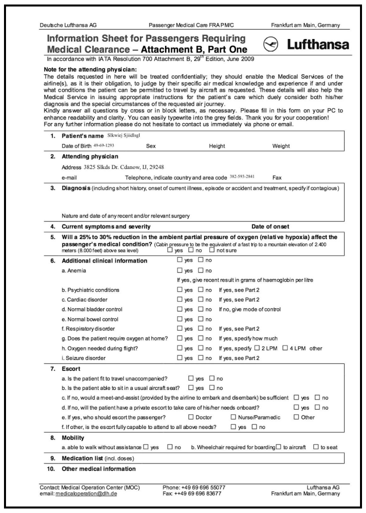
The edge-detection method for splitting the images seemed unreliable. It worked well in some cases but worked very poorly in other cases, as described above. I explored other methods to split the images.

The next method I implemented to split the images looked for large areas of white space and separated the images along those spaces. This approach was divided into two main steps:

(1) Determine document margins and remove the margins.

(2) Split the resulting image horizontally into smaller sub-images.

These two steps were separate because removing the margins depended on horizontal and vertical white spaces, while splitting the image only depended on horizontal components of the image. Additionally, document margins generally contained some level of noise (due to scanning or image quality) which should not have prevented the margins from being detected. When splitting the images, noise must be more carefully assessed so that the image is not split across textual content. The reason why the sub-images were the result of only horizontal splits and not vertical splits was because text-based documents tend to be structured in horizontal lines or groups of text, while vertical groups of text are less rare and less structured.



Deutsche Lufthansa AG Passenger Medical Care FRA PM/C Frankfurt am Main, Germany

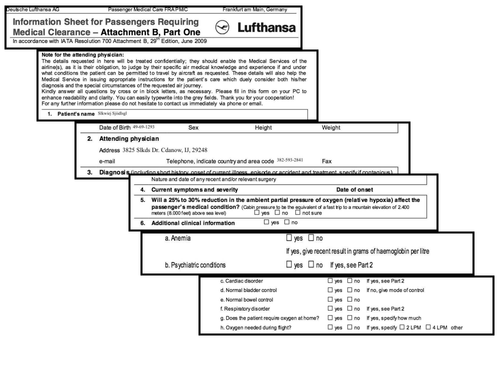
Information Sheet for Passengers Requiring Medical Clearance – Attachment B, Part One **Lufthansa**

In accordance with IATA Resolution 700 Attachment B, 29th Edition, June 2009

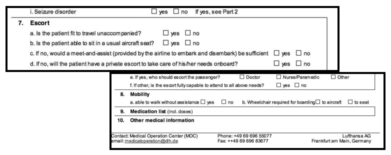
Note for the attending physician:
The details requested in here will be treated confidentially; they should enable the Medical Services of the airline(s), as it is their obligation, to judge by their specific air medical knowledge and experience if and under what conditions the patient can be permitted to travel by aircraft as requested. These details will also help the Medical Service in issuing appropriate instructions for the patient's care which duely consider both his/her diagnosis and the special circumstances of the requested air journey.
Kindly answer all questions by cross or in block letters, as necessary. Please fill in this form on your PC to enhance readability and clarity. You can easily typewrite into the grey fields. Thank you for your cooperation!
For any further information please do not hesitate to contact us immediately via phone or email.

1. **Patient's name** Slkwiej Sjiidlsgl
   Date of Birth 49-69-1293 Sex Height Weight
2. **Attending physician**
   Address 3825 Slkds Dr. Cdanow, IJ, 29248
   e-mail Telephone, indicate country and area code 382-593-2841 Fax
3. **Diagnosis** (including short history, onset of current illness, episode or accident and treatment, specify if contagious)
   Nature and date of any recent and/or relevant surgery
4. **Current symptoms and severity** **Date of onset**
5. **Will a 25% to 30% reduction in the ambient partial pressure of oxygen (relative hypoxia) affect the passenger's medical condition?** (Cabin pressure to be the equivalent of a fast trip to a mountain elevation of 2.400 meters (8.000 feet) above sea level) ☐ yes ☐ no ☐ not sure
6. **Additional clinical information** ☐ yes ☐ no
   a. Anemia ☐ yes ☐ no
   If yes, give recent result in grams of haemoglobin per litre
   b. Psychiatric conditions ☐ yes ☐ no If yes, see Part 2
   c. Cardiac disorder ☐ yes ☐ no If yes, see Part 2
   d. Normal bladder control ☐ yes ☐ no If no, give mode of control
   e. Normal bowel control ☐ yes ☐ no
   f. Respiratory disorder ☐ yes ☐ no If yes, see Part 2
   g. Does the patient require oxygen at home? ☐ yes ☐ no If yes, specify how much
   h. Oxygen needed during flight? ☐ yes ☐ no If yes, specify ☐ 2 LPM ☐ 4 LPM other
   i. Seizure disorder ☐ yes ☐ no If yes, see Part 2
7. **Escort**
   a. Is the patient fit to travel unaccompanied? ☐ yes ☐ no
   b. Is the patient able to sit in a usual aircraft seat? ☐ yes ☐ no
   c. If no, would a meet-and-assist (provided by the airline to embark and disembark) be sufficient ☐ yes ☐ no
   d. If no, will the patient have a private escort to take care of his/her needs onboard? ☐ yes ☐ no
   e. If yes, who should escort the passenger? ☐ Doctor ☐ Nurse/Paramedic ☐ Other
   f. If other, is the escort fully capable to attend to all above needs? ☐ yes ☐ no
8. **Mobility**
   a. able to walk without assistance ☐ yes ☐ no b. Wheelchair required for boarding ☐ to aircraft ☐ to seat
9. **Medication list** (incl. doses)
10. **Other medical information**

Contact: Medical Operation Center (MOC) Phone: +49 69 696 55077 Lufthansa AG
email: medicaloperation@dlh.de Fax: ++49 69 696 83677 Frankfurt am Main, Germany



Deutsche Lufthansa AG Passenger Medical Care FRA PM/C Frankfurt am Main, Germany
Information Sheet for Passengers Requiring Medical Clearance – Attachment B, Part One Lufthansa
In accordance with IATA Resolution 700 Attachment B, 29th Edition, June 2009
Note for the attending physician: The details requested in here will be treated confidentially; they should enable the Medical Services of the airline(s), as it is their obligation, to judge by their specific air medical knowledge and experience if and under what conditions the patient can be permitted to travel by aircraft as requested. These details will also help the Medical Service in issuing appropriate instructions for the patient's care which duely consider both his/her diagnosis and the special circumstances of the requested air journey. Kindly answer all questions by cross or in block letters, as necessary. Please fill in this form on your PC to enhance readability and clarity. You can easily typewrite into the grey fields. Thank you for your cooperation! For any further information please do not hesitate to contact us immediately via phone or email.
1. Patient's name Slkwiej Sjiidlsgl
Date of Birth 49-69-1293 Sex Height Weight
2. Attending physician
Address 3825 Slkds Dr. Cdanow, IJ, 29248
e-mail Telephone, indicate country and area code 382-593-2841 Fax
3. Diagnosis (including short history, onset of current illness, episode or accident and treatment, specify if contagious)
Nature and date of any recent and/or relevant surgery
4. Current symptoms and severity Date of onset
5. Will a 25% to 30% reduction in the ambient partial pressure of oxygen (relative hypoxia) affect the passenger's medical condition? (Cabin pressure to be the equivalent of a fast trip to a mountain elevation of 2.400 meters (8.000 feet) above sea level) ☐ yes ☐ no ☐ not sure
6. Additional clinical information ☐ yes ☐ no
a. Anemia ☐ yes ☐ no
If yes, give recent result in grams of haemoglobin per litre
b. Psychiatric conditions ☐ yes ☐ no If yes, see Part 2
c. Cardiac disorder ☐ yes ☐ no If yes, see Part 2
d. Normal bladder control ☐ yes ☐ no If no, give mode of control
e. Normal bowel control ☐ yes ☐ no
f. Respiratory disorder ☐ yes ☐ no If yes, see Part 2
g. Does the patient require oxygen at home? ☐ yes ☐ no If yes, specify how much
h. Oxygen needed during flight? ☐ yes ☐ no If yes, specify ☐ 2 LPM ☐ 4 LPM other



i. Seizure disorder ☐ yes ☐ no If yes, see Part 2

7. Escort

a. Is the patient fit to travel unaccompanied? ☐ yes ☐ no

b. Is the patient able to sit in a usual aircraft seat? ☐ yes ☐ no

c. If no, would a meet-and-assist (provided by the airline to embark and disembark) be sufficient ☐ yes ☐ no

d. If no, will the patient have a private escort to take care of his/her needs onboard? ☐ yes ☐ no

e. If yes, who should escort the passenger? ☐ Doctor ☐ Nurse/Paramedic ☐ Other

f. If other, is the escort fully capable to attend to all above needs? ☐ yes ☐ no

8. Mobility

a. able to walk without assistance ☐ yes ☐ no b. Wheelchair required for boarding ☐ to aircraft ☐ to seat

9. Medication list (incl. doses)

10. Other medical information

Contact: Medical Operation Center (MOC) Phone: +49 69 696 55077 Lufthansa AG
email: medicaloperation@dlh.de Fax: ++49 69 696 83677 Frankfurt am Main, Germany

Figure 4-5: Horizontal image-splitting (right) of original Medical Clearance document (left).

Figure 4-5 depicts the result of horizontally splitting an input image into smaller sub-images. The input image is shown on the left, and the sub-images are shown on the right in a staggered fashion to clearly depict each split. As visible in the input image, the sub-images could have resulted from both vertical and horizontal splitting. The fifth and sixth splits from the top could have been further split vertically down the middle, without cutting through any text. However, if the image had been horizontally split even 10 pixels above or below the position where it was split, vertical splitting would not be possible in an efficient manner. Additionally, building an efficient algorithm to determine both vertical and horizontal splitting simultaneously would have been unnecessarily complex, as the text-spotting algorithm performed well on the horizontal splits.

The algorithm that determined the input image's margins and horizontal splits followed a *moving standard deviation* approach. The algorithm first calculated the input image's *horizontal histogram* and *vertical histogram*. The image histograms represented the number of pixels in each row or column of an image . The horizontal histogram represented the sum of pixels across each row in the image, and the vertical histogram represented the sum of pixels across each column in the image. These plots are shown in Figure 4-6 and Figure 4-7.

As shown in Figure 4-6 and Figure 4-7, the minima in these graphs depicted horizontal and vertical regions of whitespace. However, little noise in these regions could prevent the white spaces from being identified. The moving standard deviation method helped with that.

The moving standard deviation is essentially a sliding-window calculation of the standard deviation for each histogram. By calculating the standard deviations, any sudden increases or decreases in the pixel count that would appear as an anomaly would be amplified, while any steady increases or decreases in pixel count would be condensed. By using a sliding window, or moving standard deviation, the anomalies in pixel counts would be diminished, while steady changes would be captured. The large anomalies in pixel counts are generally due to noise from scanning the document or poor image quality, and pixels from vertical or horizontal lines (boxes) around different text regions within the documents. This is shown in Figures 4-6 and 4-7.

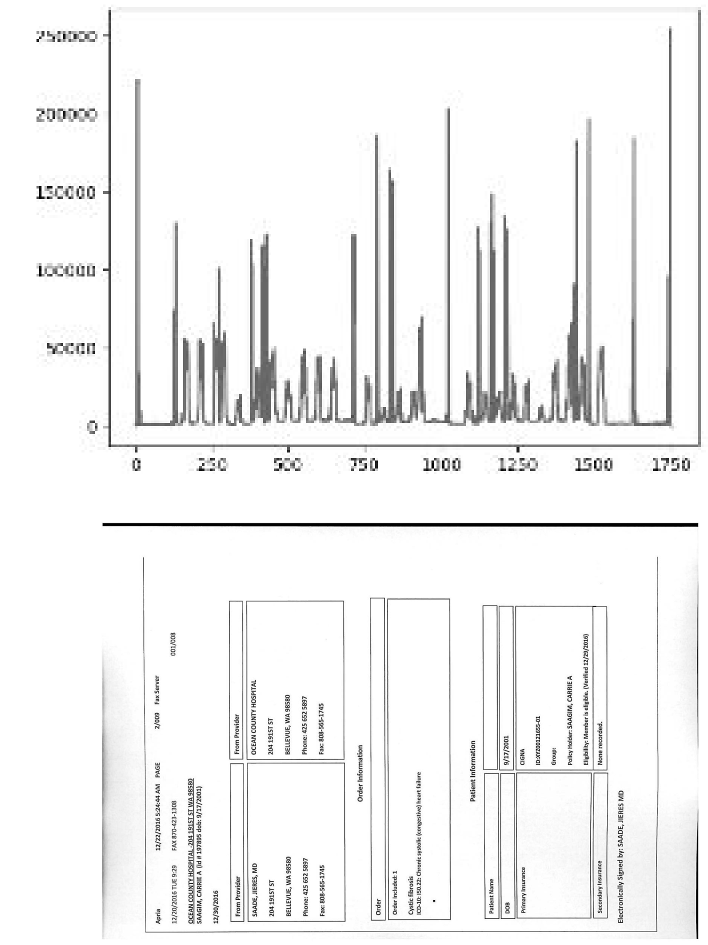
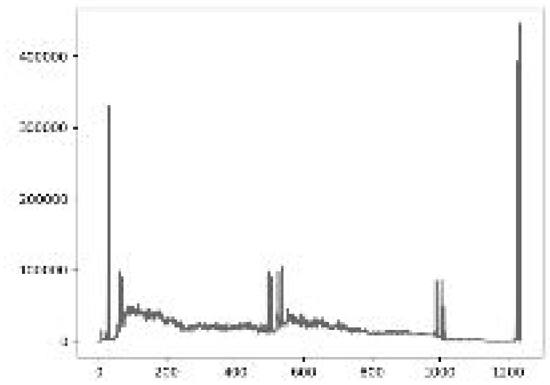
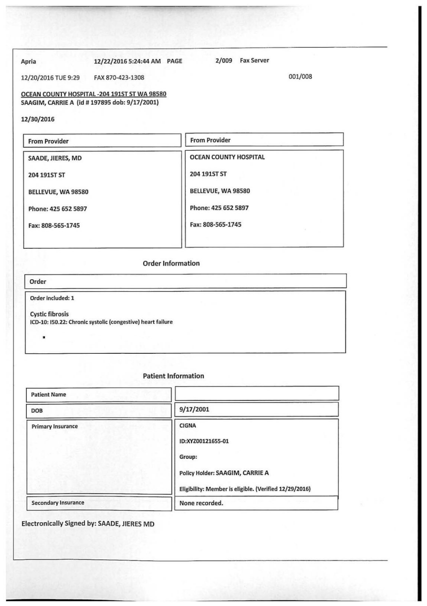


Figure 4-6: Horizontal histogram (top) of input image flipped horizontally (bottom)

The histogram shows the sum of pixels along each row of the original image, or along each column of the flipped image, shown above (bottom). As shown, the largest anomalies in pixel counts are along the margins and horizontal lines due to the text-boxes in the document.





Apria 12/22/2016 5:24:44 AM PAGE 2/009 Fax Server

12/20/2016 TUE 9:29 FAX 870-423-1308 001/008

<u>OCEAN COUNTY HOSPITAL -204 191ST ST WA 98580</u>
SAAGIM, CARRIE A (id # 197895 dob: 9/17/2001)

12/30/2016

From Provider	From Provider
SAADE, JIERES, MD 204 191ST ST BELLEVUE, WA 98580 Phone: 425 652 5897 Fax: 808-565-1745	OCEAN COUNTY HOSPITAL 204 191ST ST BELLEVUE, WA 98580 Phone: 425 652 5897 Fax: 808-565-1745

Order Information

Order
Order included: 1 Cystic fibrosis ICD-10: I50.22: Chronic systolic (congestive) heart failure ▪

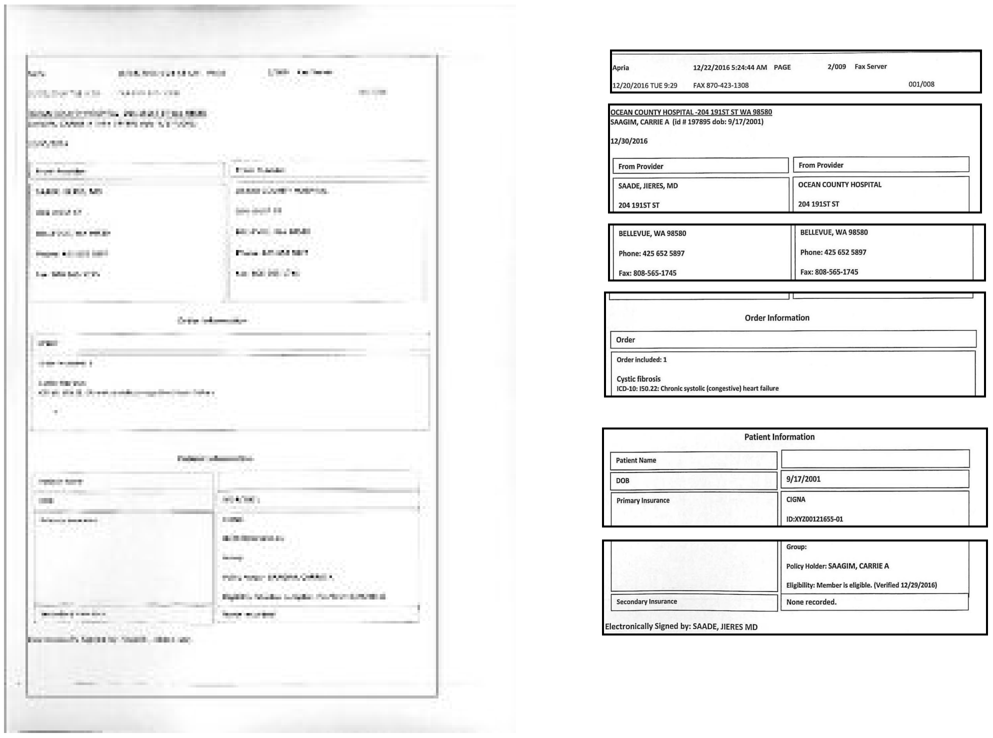
Patient Information

Patient Name	
DOB	9/17/2001
Primary Insurance	CIGNA ID:XYZ00121655-01 Group: Policy Holder: SAAGIM, CARRIE A Eligibility: Member is eligible. (Verified 12/29/2016)
Secondary Insurance	None recorded.

Electronically Signed by: SAADE, JIERES MD

Figure 4-7: Vertical histogram (top) of input image (bottom)
The histogram shows the sum of pixels along each column of the original image. As shown, the largest anomalies in pixel counts are along the margins and vertical lines due to the text-boxes in the document.

The resulting margins and sub-images for the input image in Figure 4-6 and Figure 4-7 are shown in Figure 4-8. By examining the moving standard deviation of the vertical and horizontal histograms, the algorithm could determine the margins to be close to the text boxes, surrounding all the text. The margins exclude the large vertical and horizontal lines on the image's border (due to poor scanning quality) and the spotty pixels on the top and bottom of the image (due to poor image quality). The horizontal splits only include regions of text, ignoring any whitespace or horizontal lines that were found between the text regions.



Apria 12/22/2016 5:24:44 AM PAGE 2/009 Fax Server
12/20/2016 TUE 9:29 FAX 870-423-1308 001/008

OCEAN COUNTY HOSPITAL -204 191ST ST WA 98580
SAAGIM, CARRIE A (id # 197895 dob: 9/17/2001)

12/30/2016

From Provider	From Provider
SAADE, JIERES, MD 204 191ST ST	OCEAN COUNTY HOSPITAL 204 191ST ST
BELLEVUE, WA 98580 Phone: 425 652 5897 Fax: 808-565-1745	BELLEVUE, WA 98580 Phone: 425 652 5897 Fax: 808-565-1745

Order Information

Order

Order included: 1

Cystic fibrosis
ICD-10: I50.22: Chronic systolic (congestive) heart failure

Patient Information

Patient Name	
DOB	9/17/2001
Primary Insurance	CIGNA ID:XYZ00121655-01
	Group: Policy Holder: SAAGIM, CARRIE A Eligibility: Member is eligible. (Verified 12/29/2016)
Secondary Insurance	None recorded.

Electronically Signed by: SAADE, JIERES MD

Figure 4-8: Margins and image-splits for input image shown in Figures 4-6 and 4-7
The image on the left contains a thin rectangular box surrounding all the text elements, representing the area of the image excluding the margins that the moving standard deviation algorithm calculated. The stack of images on the right represent the resulting sub-images after horizontally splitting the original image, pieced back together to form the original image. As shown by the small gaps between each sub-image, the algorithm removed the regions of whitespace that were between the splits.

When splitting each image horizontally, the algorithm searched for regions where the moving standard deviation calculations fell below some minimum threshold. This threshold was determined such that each image would have a maximum of eight sub-images. For example, the input image in Figure 4-8 resulted in six sub-images, while the input image in Figure 4-5 resulted in eight sub-images. The minimum threshold represented the minimum width for which the moving standard deviation results would count as whitespace. When applying a lower minimum threshold to an image with sparse text, there would be a large number of image-splits. When applying a higher minimum threshold to an image crowded with text, there would be only a few image-splits. Thus, the algorithm began with a small minimum threshold, calculated the image-splits, and depending on if there were many small sub-images, it would recalculate the image splits with a larger threshold value.

Furthermore, in order to make sure that the splits were as evenly spaced throughout the image, the algorithm searched for each split within a specific eighth of the image. For example, if there were eight resulting sub-images (seven splits), each had to be within a separate eighth of the image. If there were three resulting sub-images (two splits), one could be in the top one-eighth of the image, while the other could be in the bottom one-eighth of the image. By restricting the location of the horizontal splits, the algorithm ensured that the resulting sub-images would be as even in size as the density of the textual content permitted. This allowed for optimal performance of the text-spotting algorithm.

This image-splitting algorithm worked well across all inputs, producing at least five sub-images for each image across the entire dataset. The algorithm appeared to work well for the following reasons:

(1) For each image, the sub-images were fairly evenly-spaced, resulting in optimal performance by the text-spotting algorithm.

(2) Using both vertical and horizontal histograms to calculate the margins and remove them from the image helped remove unnecessary whitespace along the image borders.

(3) Calculating the moving standard deviations of the histogram values allowed the algorithm to ignore noise and sudden anomalies (such as text boxes or other noise) in the images.

Overall, this algorithm for splitting images performed well compared to the initial edge-detection approach. By combining histograms with moving standard deviations, all types of images could be optimally split into sub-images, regardless of their document structure, textual content, or text density.

4.3.2. Text Detection and Recognition

The second component of the system, Text Detection and Recognition, applied the text-spotting system from the Oxford Visual Geometry Group to the sub-images resulting from the Image Splitting component of the system [19][. As described in Section 2.3, the text-spotting system had been pre-trained on a synthetic dataset. Although the medical scans did not match the synthetic dataset that the system was trained on, there were two key points that allowed the system to perform well:

(1) The synthetic dataset which the text-spotting system was trained on consisted of natural scene images with small amounts of text content (varying in size, color, shape, font) in any part of the image. This meant the system was robust to image noise and to lack of structure when detecting and recognizing text.

(2) The medical document scans consisted of images with varying amounts of text (in different sizes and fonts) with no specific document structure, and with little background noise (no background scenery).

Although the dataset I was using was text-heavy and lacked textual structure, the text-spotting system was robust to identifying text in noisy images that lacked structure, making it fit for the task at hand.

The text-spotting system was packaged into MatLab binaries. The program's step-by-step process was as follows:

(1) The program would read a .txt file containing the file paths for each input image.

(2) The program would run the text detection and text recognition program on each input image.

(3) The program would write to a .xml file containing the text recognition results for each image.

At the first step, the system would locate all of the sub-images to run on. If the image dimensions are of large disproportion (image width is 10x image height), the system will error and will not perform the second and third steps.

At the second step, the system would locate and recognize the text elements in the document. The system had three different modes it could run in:

(1) Precise mode: the main focus is to limit false positive text detections. For my use case, this mode resulted in many missed text detections.

(2) Most mode: this mode maximized the number of text detections by also detecting very small text that could lie in the background of an image. For my use case, this was ideal since all text (large and small) had to be detected.

(3) Opt mode: this mode was a balance between the Precise and Most modes. For my use case, this mode performed better than the Precise mode, but not as well as the Most mode.

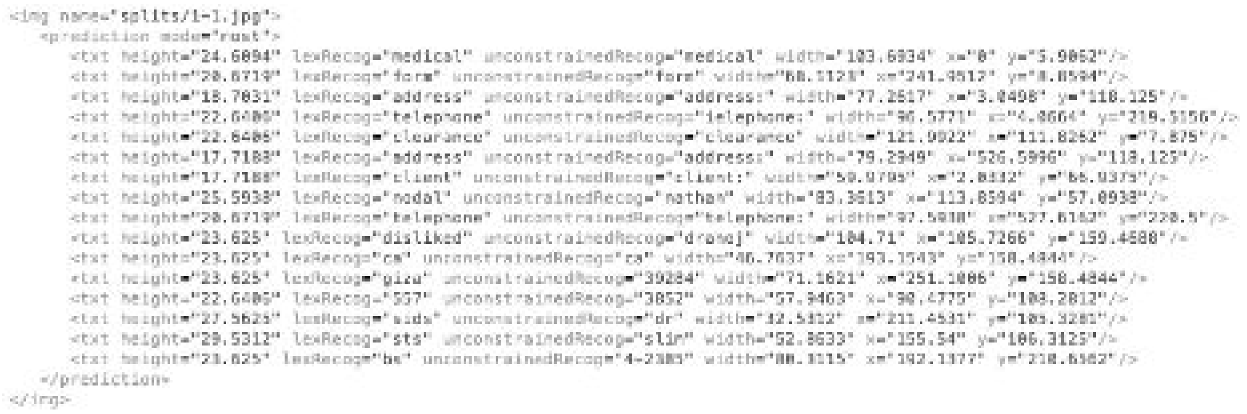
I used the Most mode when executing the binaries to ensure optimal text detection across all documents.

The final step of the text detection and recognition program was to write the output to an XML file. The output contained all the detected text elements and their data for each input image. The data for the text elements was as follows:

- *x*: the x-coordinate of the top-left corner of the text element
- *y:* the y-coordinate of the top-left corner of the text element
- *width*: the width of the text element, in pixels
- *height:* the height of the text element, in pixels
- *unconstrainedRecog*: the text element's undeciphered string of alphanumeric characters
- *lexRecog*: the text element's deciphered string of alphabet characters

A sample of the XML output is shown in Figure 4-9.

As described above, after detecting and recognizing all of the text, the program would decipher each text element into two key-value pairs. The *unconstrainedRecog* key mapped to the string of detected alphanumeric characters that made up the element, and the *lexRecog* key mapped to the string of alphabet characters (excluding numbers and other symbols) that the text element contained. Because a large amount of text in medical documents contained alphanumeric characters and punctuation marks, I used the string corresponding to the *unconstrainedRecog* key first before looking at the text from the *lexRecog* key. Figure 4-9 depicts that the *unconstrainedRecog* text tends to be more accurate than the text in *lexRecog* for my use case. For example, the third text element is found to contain the text "address:" in the undeciphered *unconstrainedRecog*, but only "address" in the deciphered *lexRecog*. With this element, the extra punctuation (or lack of) does not make a difference when the third and final component of the system makes meaning out of the text elements. However, looking at the bottom-most element, we see that the *unconstrainedRecog* contains the string "4-2385", while the *lexRecog* deciphers the string as "bs". This element pertains to a portion of the patient's phone number. Because the *lexRecog* piece of data only uses alphabet characters to encode the data, any numerical information would be missed. Thus, I initially looked at the



```
<img name="splits/1-1.jpg">
    <prediction mode="rust">
        <txt height="24.6094" lexRecog="medical" unconstrainedRecog="medical" width="103.6934" x="0" y="5.9062"/>
        <txt height="20.6719" lexRecog="form" unconstrainedRecog="form" width="68.1128" x="241.4517" y="8.8594"/>
        <txt height="18.7031" lexRecog="address" unconstrainedRecog="address:" width="77.2517" x="3.0498" y="118.125"/>
        <txt height="22.6406" lexRecog="telephone" unconstrainedRecog="ielephone:" width="96.5771" x="4.0664" y="219.5156"/>
        <txt height="22.6406" lexRecog="clearance" unconstrainedRecog="clearance" width="121.9022" x="111.8262" y="7.875"/>
        <txt height="17.7188" lexRecog="address" unconstrainedRecog="address:" width="79.2949" x="526.5996" y="118.125"/>
        <txt height="17.7188" lexRecog="client" unconstrainedRecog="client:" width="59.9795" x="2.0332" y="66.9375"/>
        <txt height="25.5938" lexRecog="nodal" unconstrainedRecog="nathan" width="83.3613" x="113.8594" y="57.0938"/>
        <txt height="20.6719" lexRecog="telephone" unconstrainedRecog="telephone:" width="97.5938" x="577.6162" y="220.5"/>
        <txt height="23.625" lexRecog="disliked" unconstrainedRecog="dramoj" width="104.71" x="105.7266" y="159.4688"/>
        <txt height="23.625" lexRecog="ca" unconstrainedRecog="ca" width="46.7637" x="193.1543" y="158.4844"/>
        <txt height="23.625" lexRecog="giza" unconstrainedRecog="39284" width="71.1621" x="251.1006" y="158.4844"/>
        <txt height="22.6406" lexRecog="557" unconstrainedRecog="3852" width="57.9463" x="90.4775" y="108.2812"/>
        <txt height="27.5625" lexRecog="aids" unconstrainedRecog="dr" width="32.5312" x="211.4531" y="105.3281"/>
        <txt height="29.5312" lexRecog="sts" unconstrainedRecog="slin" width="52.8633" x="155.54" y="106.3125"/>
        <txt height="23.625" lexRecog="bs" unconstrainedRecog="4-2385" width="88.3115" x="192.1377" y="218.6562"/>
    </prediction>
</img>
```

unconstrainedRecog pieces of data.

Figure 4-9: Sample of XML output for single sub-image
The output in this figure contains the six pieces of data described above for all the detected text elements in a single sub-image. The full XML document contains the data in this format for each sub-image.

In addition to textual data, the XML file also includes positional data for each text element (x, y, width, height). The *x* and *y* values are measured within the sub-image and not for the original image. For example, if a textual element was located at the top-left corner of sub-image #5 of some input document, the *x* and *y* coordinates of the element would be 0,0. The positional data needed to be recalculated, based on the position of the sub-image within the original image. When running the Image Splitting algorithm described in Section 4.3.1., I recorded the (x,y) coordinates of the top-left corner for each sub-image with respect to the original image in a .pickle file. Then, when parsing the XML file, I retrieved these coordinates by unpickling the file. For each text element, I added the respective sub-image's (x,y) coordinates to the *x* and *y* values of the text element to determine its position within the original input image.

At this point in the process, I had detected and recognized each of the text elements in the medical document scans. The next step would be to identify the document field content for the five fields of interest. I accomplished this in the third component of the system, the Rule-based Field Detection.

4.3.3. Rule-based Field Detection

The final component of the text-spotting system was a rule-based algorithm used to identify the fields of interest (Name, Phone Number, Address, Date on the document, and Date of Birth). After analyzing different documents, I came up with some document-based heuristics to use in order to identify the different fields.

When tagging text elements by their corresponding field, I assumed that they would be in the same vicinity as their field label. The first step was to find the field labels within the documents. When searching for the field label, I checked to see if the *unconstrainedRecog* values contained a substring matching one of the following: name, date, dob, birthdate, birth, phone, and address. Many documents did not have all five fields of interest, but had some combination of three or four of the fields. Some documents also had multiple values for the same field. For instance, some documents may have had an address for both the patient and the physician, or some documents may have had multiple dates for past surgeries. In either case, I attempted to locate all instances of the field labels for the documents, assuming that the field labels were expressed in one of the ways listed above.

After finding all field labels, I proceeded to check their vicinity for their corresponding field content. I realized that in most cases, the field content could be located in one of three positions relative to the field label:

(1) In-line with field label: the field content could come after the field label, along the same *y*-coordinate
(2) Above the field label: the field content could be written above the field label, in which case the font size of the field content tended to be larger than the field label itself
(3) Below the field label: the field content could be written below the field label, in which case the horizontal spacing after the field labels (before the next textual elements) tended to be large

I followed the heuristics above to locate the field content, under the assumption that the same relative positioning would apply to all fields within a given document. This assumption was important to keep in mind when analyzing my results because once I determined the location of some field's content relative to its label, I extrapolated that to the other fields of interest and searched for their content in the same relative location.

The next step was to determine the relative positioning of the field content with the field label for a document. The challenge was that the documents had text elements above, below, and next to field labels, so it could be difficult determining which elements actually corresponded to the field's content. However, since some fields of interest were numerical-based, I decided to use those fields as the predictor for the document's relative positioning scheme. If a document had a field for Phone Number, Date, or Date of Birth, I selected the string containing numbers that surrounded the field, and I used its relative positioning as the relative positioning for the rest of the fields within the document. In the cases where I couldn't use the numerical-based scheme, I relied on the assumption that text elements in one relative location to the field label were closer than text elements in the other relative locations, in which case I set the relative positioning based on the closest set of text elements.

In addition to relative positioning of field content and field labels, I closely examined the positional data of the text elements. I looked at the height of text elements, the distance from the field labels (both horizontal spacing and vertical spacing), and the spacing from other nearby text elements. This allowed me to identify the group of text elements that corresponded to a single field. In the case where there were multiple text elements corresponded to a single field, I would join the elements into a single element and group them together as that field. In other words, I calculated the new positional data (x and y coordinates, and overall width and height) of the elements as a group, and joined their *unconstrainedRecog* text values into a single string of text. As I will explain in the results, this became an issue when I faced fields that had non-relevant information (neither field labels nor field content) following the field labels, before the field content.

In order to evaluate the accuracy of the field extraction process, I measured the correctness of the positional data. I manually recorded the ground-truth positions (x, y, width, height) of the field content for each field of interest within the documents, and I compared those values with the positional data given by the algorithm. This is described in more detail in the next section.

4.4. Results

The results from running the text-spotting pipeline are shown in Figure 4-10. I calculated the accuracy by measuring the percentage of *true positives*, *false positives*, and *false negatives* for labeling bounding box locations as fields for different *threshold* values. The true positives were the percentage of regions that were labeled as pertaining to the correct field. The false positives were the percentage of regions that were labeled as some field when they were not part of any field of interest. The false negatives were the percentage of missed fields, or regions that were not labeled as fields when they should have been. I used a threshold value to determine how strict the accuracy measurement should be: a threshold value of 0.9 (strict) meant that the predicted region for a given field should overlap with the actual field content by at least 90%, and a threshold value of 0.1 (lenient) meant that the predicted region for a given field should overlap with the actual field by at least 10%.

As shown in Figure 4-10, the lower threshold values (lenient accuracy measurement) resulted in higher true positive percentages and lower false positive and false negative percentages. This increase in true positive values and decrease in false positive values shows that the system was able to detect field content locations with less precision, while not increasing its detection mistakes. In section 4.4.1., I analyze why this may be the case, and where the system can be improved upon.

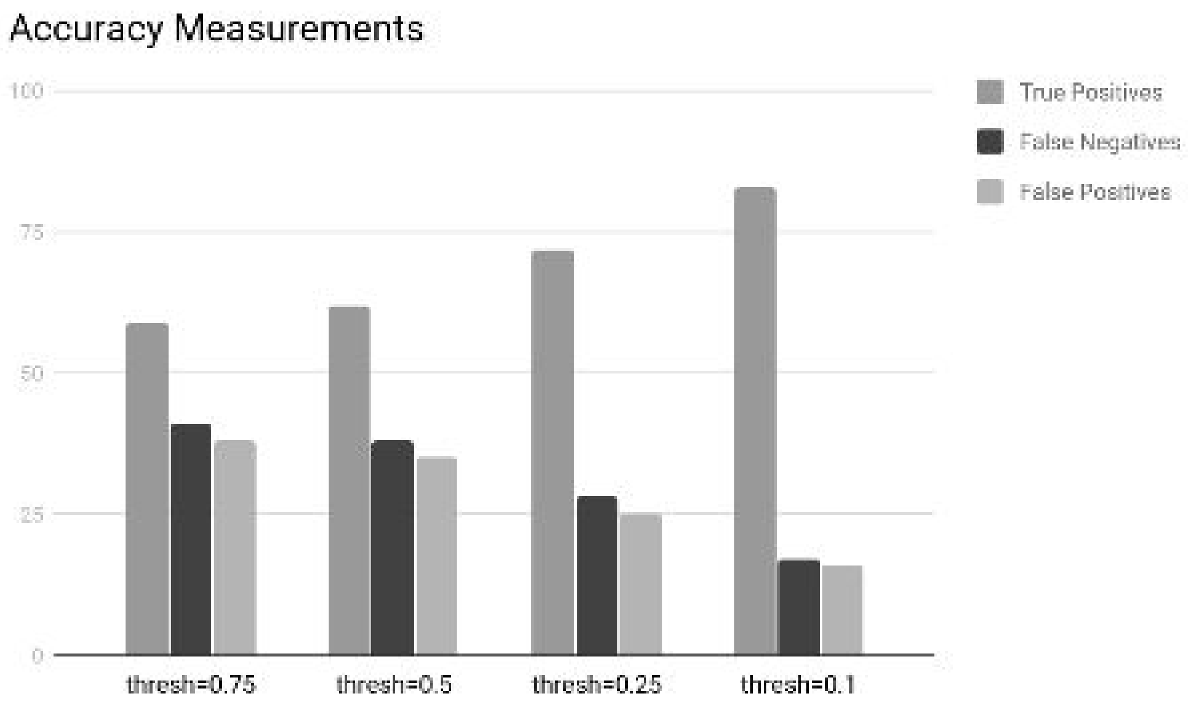


Figure 4-10: Accuracy measurements for different threshold values
The plot above shows the percentage of true positives, false negatives, and false positives at different threshold values. As the threshold becomes more lenient (less overlap necessary between actual field content and predicted field content), the percentage of true positives increases, and percentages of false negatives and false positives decrease.

4.4.1. Analysis

Overall, the text-spotting system performs well but with low precision. The threshold values essentially reflect on precision, with high accuracies at low threshold values reflecting on the system's low precision. This may be due to the fact that the rules for field detection in the last component of the system are too general. The following list of reasons explain why the rules may be too general.

(1) An entire cluster of text elements near a field label will be predicted to be the field content. This is dangerous because in many cases, documents have field descriptions after the label and before the content. For example, with a "Name", a document may say "Name (First and Last): _________" and the system should be able to determine that "(First and Last):" do not pertain to the field's content. However, my system did not differentiate field content from irrelevant text, and thus would predict everything after "Name" to be the field content.

(2) The same relative-location scheme was applied for all fields in a document. This would have contributed to both the false positive and false negative rates. If half the fields in a document were written in-line with their field labels, and the other half were written below, my system would miss half of the fields, depending on the relative-location scheme it sets.

(3) For the five fields of interest, the field content could have followed field labels that were outside of the fixed set that I provided. After looking closely at my dataset, I saw there were several documents that used "Patient" or "Client" instead of "Name".

The first point I listed was the main contributor for low accuracy at high threshold values. If I could have differentiated irrelevant data (field label descriptors) from actual field content, I believe that I would be able to improve the precision and overall performance by removing the irrelevant data from my predictions. The second and third points are also important to address. When deciding my rule-based heuristics, I made these assumptions regarding the general document structure, and the document field nomenclature. However, even within a set of 200 documents, there was a large level of variety in both the document structure (how field label positioning varies with respect to field content), and in the document field naming scheme.

5. Future Work and Conclusion

After learning about recent work in object-detection and text-spotting, and experimenting with both approaches with regards to extracting information from medical documents, I realized that there is a lot of room for improvement and I have only scratched the surface.

With regards to the object-detection approach described in Chapter 3, implementing a system that can detect and recognize specific words in various documents that vary in font, size, and structure would require a very large synthetic training set. My results showed that the model could differentiate text from non-text, but it could not classify different textual elements as being a field or a part of a field. An object detection model that would be able to accomplish this would need to be trained on a synthetic dataset, similar to how the text-spotting system in [19] was built and trained.

With regards to the text-spotting approach described in Chapter 4, improving upon it and building a highly accurate system with high precision requires a stronger rule set. My rule set that I used to recognize the field content was too general and did not perform equally well across all documents. However, by addressing the three points I listed in Section 4.4.1., I believe that the text-spotting system can perform with higher accuracy and precision for any given input. In addition, as more fields of interest are extracted, the system may improve with the increased amount of data from looking at more of the document's textual content.

In this thesis I describe two different approaches to extracting fields from medical forms: an object-detection approach and a text-spotting approach. Both of these methods have been applied to different applications, and I explored how to apply them to extracting information from scanned medical documents. I found that the object-detection approach resulted in a 65-70% accuracy for classifying document content as the correct field or not a field at all, but a majority of the accuracy came from classifying whitespace as whitespace, so it did not perform very well overall. The text-spotting approach seemed to have more promise, but the challenge was in building a ruleset that would be robust to any input. In the end, my ruleset performed well at low thresholds. With a threshold of 0.1, the system achieved 83% accuracy. The next step to improving this system will be to build a stronger ruleset.

References

[1] Wasserman, R. C. (2011). Electronic Medical Records (EMRs), Epidemiology, and Epistemology: Reflections on EMRs and Future Pediatric Clinical Research.

[2] Hussain, F., & Qamar, U. (2016). Identification and Correction of Misspelled Drugs' Names in Electronic Medical Records (EMR). Proceedings of the 18th International Conference on Enterprise Information Systems.

[3] Masthead. (1975). The Journal of Organic Chemistry,40(23)

[4] Deléger, Louise, Cyril Grouin, and Pierre Zweigenbaum. "Extracting medical information from narrative patient records: the case of medication-related information." Journal of the American Medical Informatics Association 17.5 (2010): 555-58. Web.

[5] Meystre, Stéphane, and Peter J. Haug. "Natural language processing to extract medical problems from electronic clinical documents: Performance evaluation." Journal of Biomedical Informatics 39.6 (2006): 589-99. Web.

[6] Rasmussen, L. V., P. L. Peissig, C. A. Mccarty, and J. Starren. "Development of an optical character recognition pipeline for handwritten form fields from an electronic health record." Journal of the American Medical Informatics Association 19.E1 (2012): n. pag. Web.

[7] Bergeron, Bryan. "Clinical Data Capture: OMR and OCR and Your Flatbed Scanner." MedGenMed 7.2 (2005): n. pag. Web.

[8] Ross B. Girshick and (2013). Rich feature hierarchies for accurate object detection and semantic. CoRR, abs/1311.2524, .

[9] D. Lowe. Distinctive image features from scale-invariant keypoints. IJCV, 2004.

[10] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In CVPR , 2005.

[11] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. IJCV, 2013.

[12] Olga Russakovsky*, Jia Deng*, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. (* = equal contribution) ImageNet Large Scale Visual Recognition Challenge. arXiv:1409.0575, 2014.

[13] The PASCAL Visual Object Classes Homepage, host.robots.ox.ac.uk/pascal/VOC/.

[14] Justin Johnson and (2015). DenseCap: Fully Convolutional Localization Networks for Dense Captioning. CoRR, abs/1511.07571, .

[15] Shaoqing Ren and (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal. CoRR, abs/1506.01497, .

[16] Joseph Redmon and (2015). You Only Look Once: Unified, Real-Time Object Detection. CoRR, abs/1506.02640, .

[17] Max Jaderberg and (2014). Reading Text in the Wild with Convolutional Neural Networks. CoRR, abs/1412.1842, .

[18] Jaderburg, Max. “Deep Learning for Text Spotting.” Robotics Research Group Department of Engineering University of Oxford, 2015.

[19] Gupta, Ankush, Andrea Vedaldi, and Andrew Zisserman. "Synthetic Data for Text Localisation in Natural Images." 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016): n. pag. Web.

[20] Wang, Tao. “End-to-End Text Recognition with Convolutional Neural Networks.” Stanford University.

[21] Zitnick, C.L., Doll ár, P.: Edge boxes: Locating object proposals from edges. In: Computer Vision–ECCV 2014, pp. 391–405. Springer (2014)

[22] Doll ár, P., Appel, R., Belongie, S., Perona, P.: Fast feature pyramids for object detection (2014)

[23] Medical Office Forms, www.freeprintablemedicalforms.com/category/forms.

[24] Simonyan, Karen, and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Visual Recognition." Visual Geometry Group Home Page. N.p., 8 Oct. 2014. Web. 7 May 2017.

[25] “27+ Sample Medical Clearance Forms.” 27+ Sample Medical Clearance Forms | Sample Forms

[26] prir.ustb.edu.cn/yin/image/samples_icdar_compressed_part.jpg.

[27] “Vggdemo / textspot_standalone.” GitLab, gitlab.com/vggdemo/textspot_standalone.