

# **Final Project report**

Neural network based math equation Recognition,  
Recovery and automatic Calculation

Group member:

Agarwal Mohit	Long Yun
Ma Meng	Wang Hongyang

School of Electrical and Computer Engineering  
Georgia Institute of Technology  
04/26/2015

## Project summary

We developed an automatic mathematical equation recognition system which is capable to convert a scanned image of printed mathematical equation in machine-readable format along with its LaTeX code and solution.

Text recognition systems which converts printed/handwritten data in machine readable format are widely known as Optical Character Recognition (OCR) systems<sup>[1]</sup>. Current OCR systems works quite perfect, achieves accuracy up to 99% for printed text and 95% for handwritten text<sup>[2][3]</sup>. However, when it comes to scientific papers including complex mathematical equations and complex symbols, they do not perform very well. It happens due to different nature of mathematical equations, they comprise two-dimensional information (subscripts/superscripts, division, summation etc) which is more complex to recognize as compared to textual OCRs.

Along with OCR of mathematical equation, we have added the functionalities of generating LaTeX code of equation and solution of equation (if possible to solve). We have also designed an Android Application to make it easy for users to transfer the clicked photos from phone to computer.

Our project is divided into 4 important tasks:

- 1. Character segmentation:**  
Segment the complex equation into images of separate symbols.
- 2. Character recognition:**  
Recognize each image of a symbol and assign it a particular label out of a 60 symbol set. Neural Networks and SVMs are used for this task.
- 3. Mathematical expression recovery**  
Structure of the equation is analyzed using position and size information.  
Recovers the equation, generates LaTeX script and solves the equation as well.
- 4. App and GUI design and development.**  
An android application using TCP-based socket communication to connect the client phone and server computer. A MATLAB GUI to put all blocks together as a whole software package.

For now, we have accomplished the majority of our goals.

# Detail description

## I . Introduction and Overview

Optical Character Recognition (OCR) is the electronic conversion of images of handwritten or printed text into machine-encoded text <sup>[1]</sup>. For example, given an image of a scanned document as an input, an OCR system will give you the text in machine-readable format.

Our project is OCR for mathematical equations where our efforts are in direction of expanding OCR horizons. We developed a software package which can recognize printed equations, generates LaTeX code and solves them as well.

Figure 1 is an overview of our OCR system. The input of our system can be both screenshots from computers and photos taken by our phones. Once we designate the path for image file, the system loads the picture and the segmentation block will work on this image to extract separate symbol from the image. The output of segmentation block is a series of 20×20 pixels image containing a single character. Then, we use the neural network to analysis these separate symbols and assigns them a label. Recovery part receives the recognition result and recovers the equation according to the positional information of every symbol along with generating LaTeX script. It also solves the equation if it is solvable. The last block combines all the above steps with an interactive GUI developed in MATLAB.

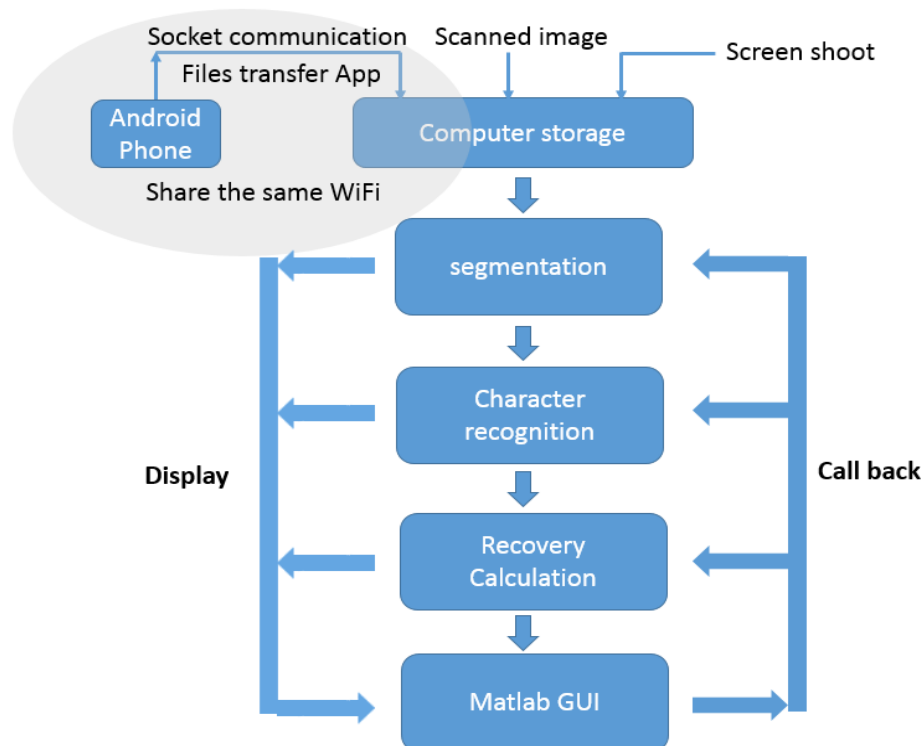


Figure 1. Schematic of our system

## II. Background and Motivation

As we know, there are a lot of data in this world in printed/handwritten format whose machine-equivalent text copies are not available. OCR saves us from the manual work of transforming this data to machine-readable format by typing. Research on OCR has been quite focused from past 60 years and now it has reached a saturation level giving the accuracy to maximum of 99% for common text <sup>[2]</sup>. However, most of them fails if tested on scientific papers or journals which involves a lot of mathematical symbols and equations.

Due to the limitation of current OCR software on solving mathematical equation, our group's goal is to design a system that can recognize, recovery, and calculate the math expression accurately and efficiently. In this OCR system we have implemented learning algorithms like Neural Network and SVM introduced in this course.

## III. Technical details and analysis

*Here we illustrate this system from 4 different part:*

1. *Equation Segmentation* (Hongyang Wang, Meng Ma)
2. *Character Recognition* (Mohit Agarwal, Yun Long)
3. *Recovery and Calculation* (Meng Ma, Hongyang Wang)
4. *Android App and MATLAB GUI* (Yun Long, Mohit Agarwal)

*\*We planned to put equal contribution for each of the tasks, and assignning each task to a separate individual as to hold responsibility of that particular task. That way, we ensured every part was completed timely as well as perfectly.*

### 1. Equation Segmentation

#### Goal

In this part, steps like denoising and binarization are made to the input image. Then a segmentation algorithm is performed to separate out every single character/symbol from the original image and normalize it to a 20-by-20 matrix.

#### Preprocessing

Since a big part of images used in our system are taken from cellphone, so speckle noise may exists due to light, orientation, and camera quality. To eliminate noise dots, we can take advantage of their smaller size compared to normal character components to effective suppress most of them. The most obvious and simple technique would be morphological open operation, i.e. morphological erosion followed by dilation.

As for binarization, Ostu's method, which optimizes the threshold by minimizing the variance between black and white pixels, is employed. It can automatically select the optimal threshold to convert grayscale image to binary image with varying lighting conditions and effectively separate character from background at the same time.

#### Segmentation

The segmentation part seems to be very straightforward with a perfect input binary image, since a simple binary labeling process that labels each connected region with a unique number, can possibly solve this problem. Well, this problem is not that simple

as it looks like. Careful consideration would reveal many practical problems, for example, how to associate the dot and the main part of character “i” and “j” in various situations, or how to distinguish minus sign “-” from fraction sign. Situation could be even harder if complex operators are involved such as summation, product, square root, and so on. What’s worse, there may be some expression under big operators like square root, which makes it difficult to extract the operator alone <sup>[4]</sup>. Algorithms that address all these problems can be found in appendix 2.

Figure 2. Segmentation result

## 2. Character Recognition

### Goal

This step assigns a label to each of 20×20 image matrix, which can be identified as a symbol.

### Symbol Set

We have 60 symbols in our database including digits (0-9), alphabets (a-z), few mathematical operators and some Greek symbols. We have assigned each mathematical symbol a unique numeric label (from 1-60). Mapping is present in Symbol Table (Table 1 of Appendix).

### Training Data

We weren’t able to find training dataset on internet for printed symbol recognition. A lot of repositories of handwritten training symbols was present but only few for printed ones with mathematical symbols. So, we decided to prepare our training data by ourselves. We knew that it’s a lot of hectic manual work, but it was the prime requirement of project.

Our targeted training data size was 40 samples for each symbol (i.e. 2400 training sample size). 40 samples were comprised of 10 symbols of each font namely, Computer Modern, Helvetica, Comic Sans and Verdana.

To prepare the dataset, we printed 10 symbols on one paper with a single font and took a 10 pictures of them using mobile. We repeated the previous step to cover all possible symbol and font combinations. So, in total 240 pictures were sufficient to prepare the 2400 sample dataset. Next step was to run character segmentation on all of the images and checking manually that every symbol is segmented perfectly and assigning label to them.

We used website <http://www.codecogs.com/latex/eqneditor.php> for symbol writing.

### ML Algorithms

We implemented 2 different supervised learning algorithms for symbol recognition, Neural Networks and SVM (Support Vector Machine) <sup>[5]</sup>. Motivation behind implementing two different algorithm was to have extensive knowledge of both

algorithms along with their implementation. We didn't use any library for implementation of Neural Networks, but due to time constraints we used svmLib for implementation of SVM.

## 1). Neural Networks

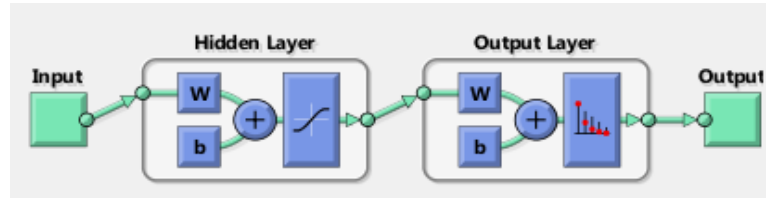


Figure 3. Neural Network

Neural Networks was implemented by calculating the Cost function and its gradient [6]. We used '*fmincg*' function to minimize the cost function resulting in optimum weights of Neural Network. We didn't use standard MATLAB NN library.

Architecture:

Input Layer Size: 400 (Translated from 20x20 image matrix)  
Hidden Layer Size: 100 Neurons  
Output Layer Size: 60 (as 60 symbol set is present)

## 2). SVM (Support Vector Machine)

SVMs are binary classifiers. To implement them for multi-class classification problem we either have to do OneVsOne strategy or OneVsAll. For our instance, we have 60-class classification problem. OneVsOne will result in 60C2 binary SVM classifiers to differentiate b/w 2 classes out of 60. Their results can be combined on metric of distance from hyperplane. On the other hand, OneVsAll strategy result in designing 60 classifiers. We can see, that OneVsOne have more computational complexity as compared to OneVsAll, but it performs much better than OneVsAll.

We used 'libsvm' library that supports multiclass classification using SVMs. It has OneVsOne implementation of it.

### Validation

For testing purposes, we divided our whole training dataset in two parts shuffled randomly. Split was 70% for learning model parameters (training) and 30% for testing. Results obtained for both the algorithms are as follows:

Accuracy	Training Set (70%)	Test Set (30%)
Neural Networks	99.1667%	98.9286%
SVM	98.4722%	98.6111%

We received almost same performance in terms for accuracy for both algorithms. We decided to go with Neural Networks for our final software because of relatively less computational complexity.

Symbol recognition part outputs the label no. corresponding to each input image provided to it by character segmentation algorithm (Figure 4). Note that the black boxes result from characters like dots, minus signs and fractions, which need to be further distinguished using the information of their sizes and positions in the next part.

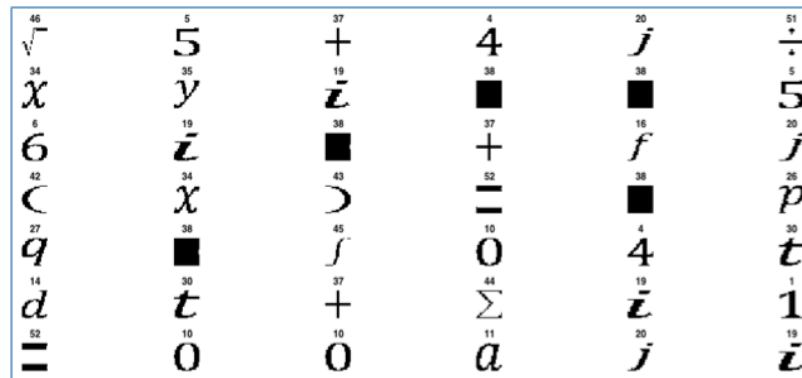


Figure 4. Recognition result

### 3. Recovery and Calculation

#### Goal

Given a sequence of characters with their labels and bounding boxes, recover the whole equation and generate a LaTeX statement for the equation. If the equation is computable, perform the calculation and output the result.

#### Recovery algorithm

Using the information of the position and size provided by the bounding box, we could handle subscripts and superscripts. Thus we can deal with the basic problem i.e. to recover a line of simple equation.

The equation may have some special structures, like square root, fraction, summation and product. However, we can convert it to the basic problem: when there is a square root, the expression under it is a simple equation; when there is a fraction/ summation/ product, the expressions over and under it are simple equations.

The most complicated scenario is the combination of the three special structures. We can use a recursive way to solve this problem as is shown by figure 5. Thus we can deal with really complex equations.

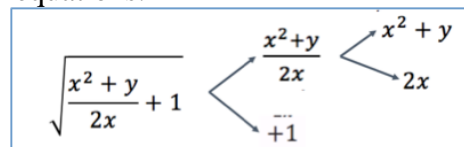


Figure 5. Recover complex equations in a recursive way

#### Calculation

Given a line of LaTeX expression, we can convert it to Matlab expression according to LaTeX syntax and then do simple calculations using Matlab. Our OCR system supports simple arithmetic calculations as well as equations solving.

#### Results

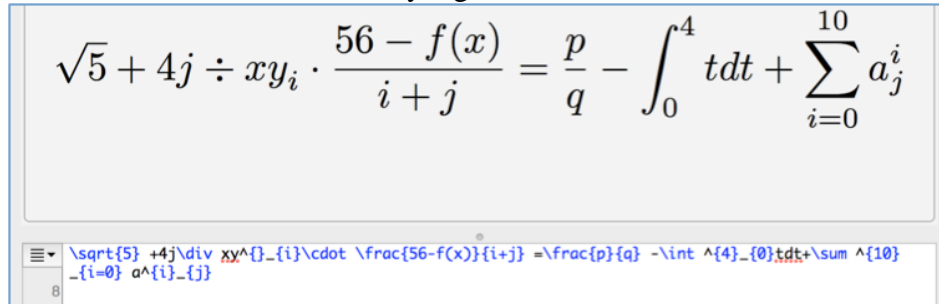
Given result show by figure 2 and 4, the recovery algorithm will analyze the structure

of the original equation and generate a LaTeX expression for the equation.

```
latex =
\sqrt{5} + 4j \div xy^{\{i\}} \cdot \frac{56 - f(x)}{i + j} = \frac{p}{q} - \int_0^4 t dt + \sum_{i=0}^{10} a^{\{i\}}_{-j}
```

Figure 6. Output: a LaTeX expression for the original equation

If we execute the LaTeX expression in LaTeXiT, it will show the equation, which verifies the correctness of the recovery algorithm.



$$\sqrt{5} + 4j \div xy^{\{i\}} \cdot \frac{56 - f(x)}{i + j} = \frac{p}{q} - \int_0^4 t dt + \sum_{i=0}^{10} a^{\{i\}}_{-j}$$

Figure 7. Use LaTeXiT to verify the result

#### 4. Android App and MATLAB GUI design

##### Goal

This App can automatically detect the IP of the client phone. After copying this IP to the browser on the server computer, a connection will be built between the phone and computer. Then, customers can browse the SD card of phone and select the files to be uploaded.

For MATLAB GUI, It can call back every block of the system, includes loading in image, segmentation, equation recovery and calculation, and Latex script generation. Moreover, the initial image and execution result can be shown on the GUI.

##### Communication mechanism and APP development

The connection between the client Android phone and server computer is based on the network socket. An Internet Socket is characterized by at least two parts: (1) Local socket address including local IP address and port number (We used port 9400 for test); (2) Protocol, a transport protocol (We used TCP for test).

To be more specific, an internet socket communication is like a bridge that connects devices in the same local internet. The protocol for information transport is TCP and the address for identity verification is IP. In order to make the whole system conveniently and more efficiently, we also modified a web page to access the SD card directly according to an online web page template. The template of this web page can be found here: <http://blog.csdn.net/mad1989/article/details/9147661>





Figure 8. Basic communication model of internet socket

## MATLAB GUI

At last, a simple MATLAB GUI was designed for better customer experience. With this GUI, users can select and load the image containing math equation, as well as segmentation, recognition, recovery and calculation.

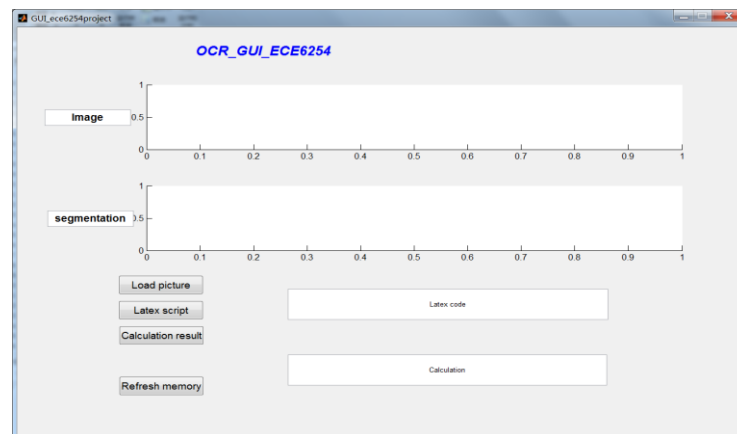


Figure 9. MATLAB GUI

## IV. Conclusion and Acknowledgement

An efficient and accurate OCR system has been developed for complicated mathematical equation recognition based on neural network and SVM classifier. Besides recognition, our system can also realize character segmentation, equation recovery, and equation calculation. Furthermore, in order to achieve better user experience and higher efficiency, we also developed an Android App for image uploading and MATLAB GUI for easy and convenient usage.

For now, we have accomplished the majority of our goals. Our next step will focus on recognition of handwritten equation and enhancement of calculation capability.

During this project, we learnt a lot not only about the knowledge of machine learning, but also knowledge of JAVA development, MATLAB, group management and cooperation. Thanks to every team members, we worked together to accomplish this meaningful project. Thanks to Professor Mark, you gave us a good opportunity to learn such interesting stuff.

## Reference

- [1]: [http://en.wikipedia.org/wiki/Optical\\_character\\_recognition](http://en.wikipedia.org/wiki/Optical_character_recognition)
- [2]: Holley, Rose. "How good can it get? Analysing and improving OCR accuracy in large scale historic newspaper digitisation programs." *D-Lib Magazine* 15.3/4 (2009).
- [3]: Smith, Ray. "An Overview of the Tesseract OCR Engine." *ICDAR*. Vol. 7. No. 1. 2007.
- [4]: Lu, Yi. "Machine printed character segmentation—; An overview." *Pattern Recognition* 28.1 (1995): 67-80.
- [5]: Hastie, Trevor, et al. *The elements of statistical learning*. Vol. 2. No. 1. New York: springer, 2009.
- [6]: Rajavelu, A., Mohamad T. Musavi, and Mukul Vassant Shirvaikar. "A neural network approach to character recognition." *Neural Networks* 2.5 (1989).

## Appendix

Table 1. Chart of labels for symbols

character	label	character	label	character	label
1	<b>1</b>	k	<b>21</b>	]	<b>41</b>
2	<b>2</b>	l	<b>22</b>	(	<b>42</b>
3	<b>3</b>	m	<b>23</b>	)	<b>43</b>
4	<b>4</b>	n	<b>24</b>	$\Sigma$	<b>44</b>
5	<b>5</b>	o	<b>25</b>	$\int$	<b>45</b>
6	<b>6</b>	p	<b>26</b>	$\sqrt{\quad}$	<b>46</b>
7	<b>7</b>	q	<b>27</b>	/	<b>47</b>
8	<b>8</b>	r	<b>28</b>	*	<b>48</b>
9	<b>9</b>	s	<b>29</b>	$\pi$	<b>49</b>
0	<b>10</b>	t	<b>30</b>	$\Pi$	<b>50</b>
a	<b>11</b>	u	<b>31</b>	$\div$	<b>51</b>
b	<b>12</b>	v	<b>32</b>	=	<b>52</b>
c	<b>13</b>	w	<b>33</b>	<	<b>53</b>
d	<b>14</b>	x	<b>34</b>	>	<b>54</b>
e	<b>15</b>	y	<b>35</b>	$\lambda$	<b>55</b>
f	<b>16</b>	z	<b>36</b>	$\alpha$	<b>56</b>
g	<b>17</b>	+	<b>37</b>	$\beta$	<b>57</b>
h	<b>18</b>	- (line)	<b>38</b>	$\gamma$	<b>58</b>
i	<b>19</b>	$\times$	<b>39</b>	$\delta$	<b>59</b>
j	<b>20</b>	[	<b>40</b>	$\epsilon$	<b>60</b>

## Code 2. Matlab code for segmentation (core code)

```
BEGIN
    LABEL each connected component
    FIND all the dot and minus sign
        IF area/bounding rectangle area > 0.8 AND width = height
            THEN label it as dot
        ELSE IF area/bounding rectangle area > 0.8 AND width >> height
            THEN label it as minus sign
        END
    FIND all 'i' and 'j'
        FOR each dot
            FIND the nearest component
            IF the nearest component is under this do
                THEN combine these two parts, label as 'ij'
            END
        END
    FIND all the minus and fraction
        FOR each minus sign
            FIND above components and under components
            IF no above
                LABEL it as minus sign
            ELSE label it as fraction
            END
    EXTRACT all the character image
        IF character is square root
            FIND all the components that inside this operator
            SUBTRACT all the inside components
        ELSE extract the content of bounding rectangle
        END
END
```