Pre-algebra Arithmetics Autograder

Guangting Yu
University of Michigan
Ann Arbor, Michigan, USA
Email: yugtmath@umich.edu

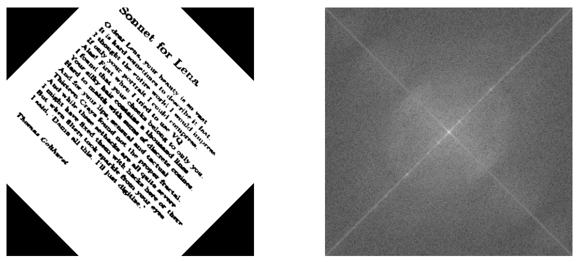
Chuxi Wei
University of Michigan
Ann Arbor, Michigan, USA
Email: weichuxi@umich.edu

Dazhi Wang
University of Michigan
Ann Arbor, Michigan, USA
Email: dazhi@umich.edu

***Abstract*—In this report, we will introduce our pre-algebra arithmetics autograder. The solution we are going to propose utilizes computer vision knowledge to make an autograder for grading simple arithmetic problems on the working sheets. We will go through the methdology we use in every step of the process, and demonstrate the results of our product. We will also discuss about the next steps to imporve this product.**

$253 + 564 =$	$30 + 210 =$	$11 \times 78 =$
$8 \times 44 =$	$183 + 141 =$	$300 \div 4 =$
$417 \div 1 =$	$283 + 553 =$	$9 \times 60 =$
$700 - 425 =$	$310 \div 310 =$	$202 \div 1 =$
$564 - 225 =$	$5 \times 17 =$	$63 \times 1 =$
$327 \div 3 =$	$745 \div 5 =$	$65 \times 6 =$
$648 \div 648 =$	$52 \times 15 =$	$959 - 793 =$
$839 - 127 =$	$6 \times 58 =$	$48 \times 1 =$
$114 \div 114 =$	$52 \times 10 =$	$13 + 684 =$
$316 - 221 =$	$740 \div 37 =$	$22 \times 35 =$

Figure 1: An Example of the Working Sheet



(a) Rotated Image, Spacial Domain (b) Rotated Image, Fourier Domian

Figure 2: Fast Fourier Transform Effect

1. Introduction

When primary school students start to learn addition, subtraction, multiplication and division operations, they need to practice a lot. Doing exercises on the working sheets with many simple arithmetic problems, as shown in [fig1], can help them quickly master their skills. However, this brings lots of grunt work to the teachers, who have to check the answers one by one on every question in every student's working sheet, which is very frustrating and boring.
The questions on the working sheets follow relatively strong patterns and the background is just pure white, which brings us possibilities to deprecate the human work and automate the process. With our product, teachers will only need to scan the sheets with students' answers without providing standard solution or even the blank sheet. Our auto-grader will recognize the questions, calculate the right answers, and grading the students' working sheets automatically.

2. Procedure

2.1. Preprocess

2.1.1. Determine Rotation. A real image of the work sheets may be inclined, which makes it hard to perform the further recognition. Thus, we want to rotate the image to make the lines horizontal. Fast Fourier transform can achieve the goal by finding the principal responses.
By choosing the dominant respons, we can determine the rotation of the image and rotate it back to be horizontal orientation.

2.1.2. Segmentaion of Pixels and Labeling. We have the basic assumption that every character on the work sheet should be separated. Although this assumption usually fail for the handwritten parts, we still apply it as if users can provide handwritten digits legible enough. Thus, we assign unique labels to the connected regions, which are digits or symbols in our project.

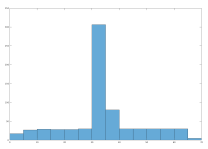
2.1.3. Noise Removal. Noises can also occupy the labels, which can cause false responses in further recognition. Thus, we implement a method to remove them. Our current algorithm is to calculate the average areas of the connect regions and remove those with areas less than 1/10 of the mean.

2.2. Sign Recognition

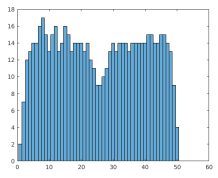
2.2.1. Subtraction Sign. This is the easiest recognition. Subtraction sign has the unique feature that the height-width ratio is low, which is usually 0.1. However, this rubric alone is not enough, since both equality signs and division signs have the same locol feature.

2.2.2. Equaltiy and Division Sign. Both signs have multiple disconnected regions. So we select a square window whose side length is equal to the width of either part of equality signs. A true equality sign will have another subtraction-sign-like segment above or below it, while a division sign will have two dot-like segments above and below it.

2.2.3. Addition and Multiplication sign. Both of the signs have the feature that the height-width ratio is approximately 1. Furthermore, both of them have 4 symmetrical axis and are both rotationally symmetric. Thus, we generate 7 variants of the original candidate: left-right flipped, top-down flipped, transposed (flipped along the diagonal) and flipped along the vice diagonal, rotated in 90, 180 and 270 degrees. If the difference between them is large, we reject the candidate as any of the signs. To distinguish these signs from ech other, we apply the histogram of the coordinates.



(a) +



(b) ×

Figure 3: Histogram of Coordinates

It is obvious that the variance of the histogram is larger on addition sign. Furthermore, a rotation of 45 degree can alternate the two symbols to eahc other.

2.3. Equation Parsing

At first, we want to only use histogram to figure out the number of columns in the working sheet, because we assume that all the questions on the same questions are aligned perfectly. However, we found that the equations are not always aligned perfectly, and even if they are aligned visually, misalignment of several pixels could occur due to the resolution of the scanner. In this case, three columns may have more than 3 bins in the histogram, for which a three-column working sheet could be recognized as 4 or 5 columns.
To solve this issue, we come up with some addtional methods to enhance the hist function and make the equation parsing correct in any situations. First, after using hist function, we group those equations by checking their horizontal distances with each other. If the distances are within a certain threshold, we consider them as a group, i.e. a column. In this method, we eliminate the error caused by misalignment of equations. After we have groups of equations, we sort all equations in a single group by their vertical coordinates. By doing that, we got the vertical order of the equations in a column. Combining the above two methods, we know the exact order of every equation in the working sheet, which make it easy for us to scan through every equation in order.

2.4. Digit Recognition

Accurate digit recognition is the key to the high-quality grading. To complete this task, we use the idea of KNN, K nearest neighbors based on the features extracted from PCA, principal component analysis.

KNN classifier finds the nearest K neighbors to the test data and classifies the test data as the majority class of these K training data. The distance between two data is the euclidean norm of the difference of the two feature vectors. The feature vector of the data is extracted by PCA. PCA transforms the raw high-dimensional data to a new coordinate system that captures the greatest variances, where the greatest variance is projected on the first coordinate, the second greatest variance is projected on the second coordinate, etc.
In our project, digit recognition has two main parts: the recognition of the printed digits and the recognition of the handwritten digits.

2.4.1. Printed Digit Recognition. For printed digits, we first print digits from 0 to 9 of 80 different fonts, respectively, on a single sheet. After scanned and segmented, each image of digit is resized into 32×32-pixel image and the 32×32 matrix is further vectorized into a 1024×1 column vector. We then merge these column vectors into a 1024×800 matrix, where the first 80 columns are vectorized images of 0's of 80 different fonts, the second 80 columns are vectorized images of 1's of 80 different fonts, etc. Denote the matrix as X, and the column mean of the matrix as μ. In order to improve the affine invariance of the recognition, we subtract μ from the 1024×800 matrix, from which we get $\bar{X}$. Then we find the covariance matrix of $\bar{X}$ from

$$\mathrm{Cov}(X) = \frac{\bar{X}\bar{X}^T}{n}$$

where n is the number of data, i.e. 800. We then find the 30 dominant eigenvectors of the covariance matrix by using the Matlab built-in function *pcacov*, which are visualized in [fig4].

As can be observed from [fig4], the first several eigenvectors capture greater variances while the last several eigenvectors capture finer details. Next, we project the training data onto the new coordinate system by

$$projection\_train = \text{basis}^T \bar{X}$$

where the basis is a 1024×30 matrix whose columns are the eigenvectors, each column of projection_train is the

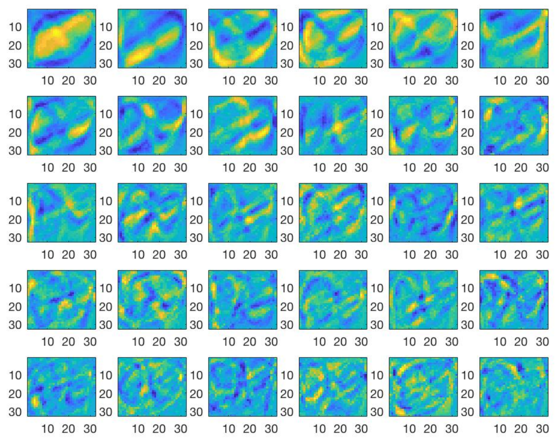


Figure 4: Visualization of the 30 basis. The eigenvectors are ordered row-wise

transformed coordinates of the training data $\bar{x}_i$.
So far we have built the infrastructure for the KNN. To recognize a test digit, we resize it to 32×32, vectorize it to a 1024×1 column vector and subtract it by the mean vector μ so that is is compatible with the training data. Denote this processed test data as $t\bar{es}t_i$. Then, we project this high-dimensional data to the sub-space spanned by the eigenvectors in the basis matrix by

$$projection\_test = \text{basis}^T \text{t}\bar{\text{es}}\text{t}_i$$

where each element in the column vector projection_test is the projection onto the corresponding coordinate.
Next, we calculate the distances between the test digit and each of the training digits and find the K training digits of the minimum distances, where the distance is the Euclidean norm (l2-norm) of the difference of the two vectors. Suppose one of the K nearest neighbors is the i_th column of the matrix projection_train, we can find the label of the i_th training data simply by

$$label_i = (i + 1)/80$$

since the labels of the first 80 columns of the training data matrix X are 0's, the second 80 columns are 1's, etc, which save the memory since we do not need an extra matrix to store the labels.
From the K nearest labels, we take the majority and classify the test digit as this label, which is more robust than just taking the nearest neighbor (NN). Note that we should carefully select K since a small K can make the classifier not robust to outliers and a large K can blur the detail of the classifier.

2.4.2. Handwritten Digit Recognition. The recognition of the handwritten digit basically follows the same flow, except that the training set is now the handwriting of the specific user. To assure the accuracy of the grading, our auto-grader first asks the student to write the 10 distinctive digits on a worksheet, 12 times for each digit, which is the training data for the student. After collecting all the training data, we can build a database for all the students so that we can use the training set for the specific student once auto-grader is grading that student's worksheet. The idea of customized training set for the students is critical since every student has a different handwriting style. Classifying all the students' handwritten digits using a single training set can result in low accuracy of recognition.

3. Difficulties in Digit Recognition

We encounter some cases of handwritten digits that are challenging to recognize for our auto-grader, which are discussed below.

3.1. Digits with Separated Segments

When a digit is separated in mutiple segments in the segmentation, our auto-grader will fail to recoginze it . Take digit "5" an example. As shown in [fig5], it is very likely for people to not connect the upper part (the bar) with lower part of "5". In this case, our auto-grader will segment them into to 2 labels, which tends to be recognized as two digits, instead of "5".

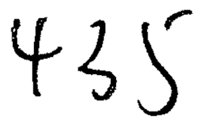


Figure 5: The "5" in "435" is a digit with separated segments

3.2. Accidentally Connected Digits

When mutiple digits are accidentally connected in handwriting, it is almost impossible for our auto-grader to recoginze. As shown in [fig6], two numbrs, "2" and "0" in this case, are accidentally connected, which happens a lot in handwriting. When our auto-grader encounters such situations, it will segment them into one single label, and make it totally unrecognizable.

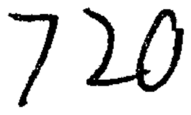


Figure 6: The "2" and "0" in "720" are accidentally connected

3.3. Non-legible Scribbled Handwriting

When people scribble and wirte fast, the digits look completely different from the standard style. Take [fig7] as an example, The lower circle of digit “8” is written very small and narrow, which makes our auto-grader recognize it as “9” or “7”. There are many cases of scibbled handwriting, and all of them make the digits very hard to recognize.

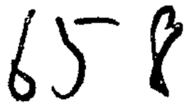
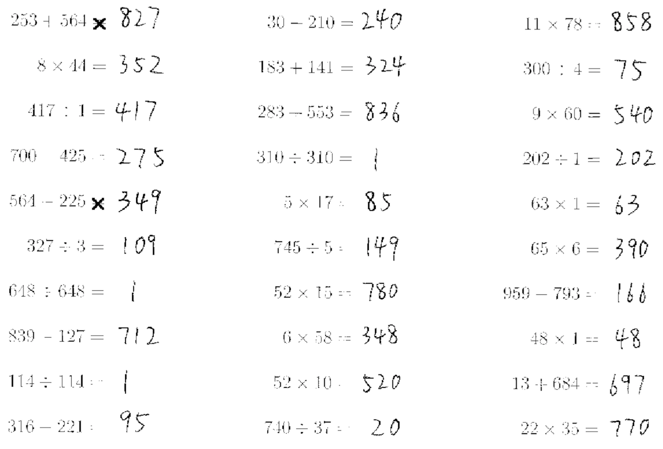


Figure 7: The “8” in “658” are scibbled

4. Result

As shown in [fig8], our auto-grader can grade the working sheet like a scantron. It will display crosses on the questions with wrong answers.
For the part of segmenting all digits and symbols, our program has a very good performance, which operate well in any conditions.
For the part of recognizing machine-printed digits and symbols, our program performs perfectly as long as the printing and the scanning are qualified, so that the printed digits and symbols do not have disconnected parts within itself.
For the part of recognizing handwritten digits, the quality of training set has a great impact on the correctness. If we use the training set written by the person to grade the same person’s working sheet, the handwritten recognization is of high accuracy, nearly 100% correctness.



$253 + 564$ ✗ 827	$30 - 210 = 240$	$11 \times 78 = 858$
$8 \times 44 = 352$	$183 + 141 = 324$	$300 : 4 = 75$
$417 : 1 = 417$	$283 - 553 = 836$	$9 \times 60 = 540$
$700 \quad 425 = 275$	$310 \div 310 = 1$	$202 \div 1 = 202$
$564 - 225$ ✗ 349	$5 \times 17 = 85$	$63 \times 1 = 63$
$327 \div 3 = 109$	$745 \div 5 = 149$	$65 \times 6 = 390$
$648 \div 648 = 1$	$52 \times 15 = 780$	$959 - 793 = 166$
$839 - 127 = 712$	$6 \times 58 = 348$	$48 \times 1 = 48$
$114 \div 114 = 1$	$52 \times 10 = 520$	$13 + 684 = 697$
$316 - 221 = 95$	$740 \div 37 = 20$	$22 \times 35 = 770$

Figure 8: Graded Working Sheet with Crosses Showing the Questions with Wrong Answers

5. Discussion

This section includes two main parts. The first part is the possible methods for improving the efficiency. The second part is the potential next steps for improving the practicality and the functionality.

5.1. Improving the efficiency

First, we can use the idea of parallel computing since some calculations or execution of the processes can be carried out simultaneously. There are two levels of parallel computation. The first is the processing of several worksheets. The second is the processing of the questions within one worksheet. The Matlab parallel toolbox supports such operations.
Second, we can leverage the GPU to enhance the performance in matrix operations, especially in the principle component analysis, where we need to find the eigen-decomposition of large matrices. The Matlab CUDA core supports such operations.

5.2. Improving the practicality and the functionality

First, our auto-grader should be developed to deal with the case where the user corrects his or her answer by drawing a cross on the undesired answer. This can happen a lot in real-life scenarios. However, recognizing the crossed answer and locating the real answer can be challenging if the writing has no regularized pattern.
Second, we may improve the accuracy in the recognition of the digits by deep learning or neural networks.

6. Related Works

The methods and algorithms in this project are derived by ourselves. We have not referred to any paper nor used any open-source code. The idea for determining the orientation of the scanned worksheet is from Fast Fourier Transform. The idea for PCA is from the eigen-face analysis in the course.

7. Acknowledgement

We would like to thank Prof. Jason Corso for determing the scope of the project and giving technical suggestions.
We would also like to show our gratitude to the GSIs, Siyuan Chen and Byungsu Kyle Min for the assitance in choosing the topic of the project.