Math Runner

Rupesh Chavan, Tejas Dond, Pawan Gidwani, Vineet Hariharan, Manoj Sabnis

***Abstract*— In this paper aims to address the need of solving mathematical equations quickly and help students to cross check their solutions. "Maths runner" is a application that will help us to take photos of mathematical equation and solve it within limited time. As some of the maths equation are very complex and hard to type. It can take a lot of time in typing those equations in calculator. "Maths Runner" will help us to solve those equations efficiently and quickly. We often face a situation where we need to add dozens of numbers eg. Grocery bills, marks of students in the class etc. In such situations we can use "Maths runner" to solve these problems. We will use techniques such of Optical Character recognition (OCR) to convert the detected text into characters. In order to this emerging domain, this study explores the image processing, dynamic programming and efficient algorithms.**

Keywords*— *Equations, Optical Character Recognition, image processing

I. INTRODUCTION

The term Optical Character Recognition refers to conversion of images of various types such as handwritten or printed text into machine-encoded text, whether from document which is scanned or a photo. It is one of the widely used as a form of information entry from printed paper data records, computerized receipts, printouts of static data or any suitable document. It is a field of artificial intelligence and pattern recognition. We tried to expand this concept and used it with efficient mathematical algorithms to solve various cases of equations. As problem may arise that there are many fonts available but in 1960s, a special font called **OCR-A** was developed that could be used as a standard. Check-printers too use this font and OCR can easily recognize it. Also OCR programs recognize letters written in a number of common fonts such as Times, Helvetica, Courier and so on. Hence OCR could recognize quite a lot of printed text. The advantages of OCR that we used in our application are instantly editing text after conversion, getting custom font that matches printout such as document or image.

After converting image (equation of linear, quadratic, cubic type) to text it will be given as an input to java code which consists of different cases to identify type of equations. We have used different combination of algorithms to solve equations.

The structure of rest of this paper is as follows: Section II discusses the motivation behind the research, followed by

Section III which describes system overview. Further, Section IV provides the brief description of our proposed system followed by literature survey. Finally, Section VI ends up with some conclusions and future research scope in application.

II. MOTIVATION

They say math is the language of the universe but there are many for whom it might seem gibberish. Be it helping your child with his homework, verifying your answers or even chatting on an assignment (though we don't recommend it) this app can help you do all that. The future possibilities are endless. It could be developed to even solve complex equations and help with daily activities like tallying a grocery or restaurant bill to even filing his taxes.

III. SYSTEM OVERVIEW

Our goal is to develop a mobile based android application that captures image of equation, solves it and displays solution on mobile itself. The general approach for solving an equation contained in an image as follows:

1) **Capture image:**
   Overall system consists of 2 blocks. The client side IDE is an android application that captures image using camera. It is recommended to use 300 dpi resolutions for OCR. The reason is that all ABBYY technologies are tuned for that resolution.

2) **Preprocess image:**
   For preprocessing we have used java in-built classes which increase accuracy of conversion.

3) **Selection/cropping image manually:**
   Image cropper is used to crop a part of image which user wants to convert into text or user can manually crop image and then browse from gallery or internal memory.

4) **Text Detection:**
   Tesseract OCR (Optical Character Recognition) is used for text detection which is widely used in most of the applications.

5) Equation identification:

Regular expressions are used for matching particular pattern containing alphanumeric characters as well as mathematical expressions. It is nothing but a sequence of characters that define a search pattern. Usually this pattern is used by searching algorithms. Hence we wrote our own algorithm to identify various categories of equations.

6) Solve equation:

For solving equation we used various methods such as Cramer's rule, quadratic formula method, etc. which provides efficient and accurate result.

7) Display results:

Result of equations is displayed in the form of variable values or root values on the screen after fraction of seconds.

IV. PROPOSED SYSTEM

The proposed system works in 2 phases:

Mobile Side Coding:

1. Android GUI:

Android studio is the IDE (Integrated Development Environment) for android devices. Android studio is freely available under Apache License2.0. It is available to download on Windows, macOS, Linux, replaced Eclipse Android Development Tools (ADT) as Google's primary IDE for native Android application development.

2. Tesseract OCR:

Tesseract is an optical character recognition engine for various operating systems[1]. It is freely available tool under Apache License. Tesseract was in top three OCR engines in terms of character accuracy in 1995. It is available for Linux, Windows, and MacOS X. Tesseract is widely used for backend and can be used for more complicated OCR tasks including layout analysis. Tesseract performs well for fonts that exist in its database, with accuracy in the range of 85-90%. The notable exceptions were the characters '0',' -', '=', and 'a'. The character '0' was at times mistakenly identified as' ()' (prior to the removal of' ('and ')' from the dictionary). The character 'a' was at time mistakenly identified as '3'or '8'. The characters '-' and '=' were at times completely absent from the detection, likely due to the down-sampling of the camera image to 640x480 pixels. This problem was not observed for well-focused images of equations written in font sizes larger than 16. In fact, varying font sizes are not a factor if the image remains in focus and fills most of the camera's viewfinder. Tesseract output will be very poor quality if the input images are not preprocessed to suit it: Images (especially screenshots) must be scaled up such that the text x-height is at least 20 pixels, any rotation or skew must be corrected or no text will be recognized, low-frequency changes in brightness must be high-pass filtered, or Tesseract's binarization stage will destroy much of the page, and dark borders must be manually removed, or they will be misinterpreted as characters. We have used TessOCR because computer printed expressions are processed with Tesseract Optical Character Recognition engine originally developed by HP Labs and currently maintained by Google. Currently we restrict research till 3 variables but we can expand this to include more variables in the future.

PC side coding:

1. MATLAB:

The MATLAB platform is used for solving most of the engineering and scientific problems. MATLAB has matrix-based language is the world's most natural way to express computational mathematics. The first step is image acquisition which acquires the scanned image followed by noise filtering, smoothing and normalization of scanned image, rendering image suitable for segmentation where image is decomposed into sub images.

2. Java Program:

It consists of various modules such as Equation identification using Regex pattern matching technique and solving equations using algorithms such as Cramer's rule, formulas, etc. Regex Matcher is used to match pattern of equation and then we selected 18 cases categories as:

1. Quadratic Equation cases: 9
2. Linear Equation cases: 4
3. Arithmetic Calculation cases: 4
4. Trigonometric Equation cases: 1

We can extend those cases in many ways because it has no limit. Also we have used image cropper to crop a part of equation or paragraph of document to convert it into text. So that user can copy it and use it anywhere.

Data Flow Diagram:

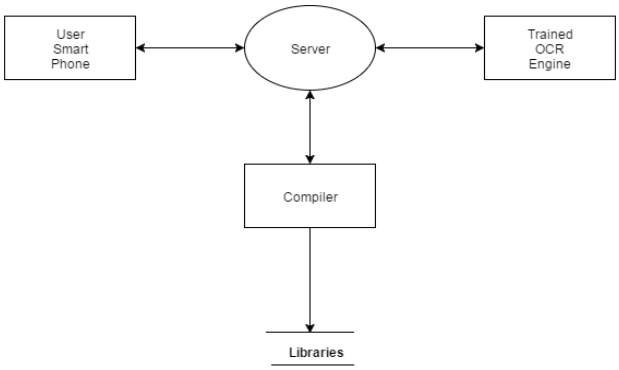

Fig 1.1 Block Diagram-1

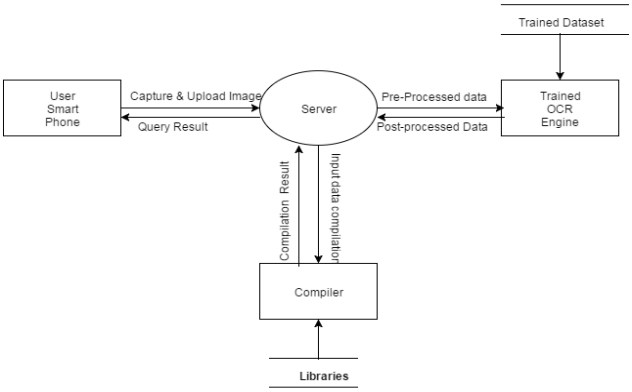

Fig 1.2 Block Diagram-2

A. Text Recognition

Computer generated text is easier to recognize than recognizing handwritten text on paper because everyone has different way of writing letters or numbers. Due to the predictable structure (font style) of the characters, a template matching algorithm can be used. The Tesseract OCR was chosen for printed text recognition. The accuracy of optical character recognition improved by converting color image into black and white image using ColorMatrix and ColorMatrixColorFilter class in java. Bitmap classes are used for storing and manipulating images to increase accuracy. Bitmap objects stores the color of each pixel of an image as a 32-bit number. It is configured as 8 bits each for red, green, blue and alpha. It is nothing but a value between 0 to 255, with 0 representing no intensity and 255 representing full intensity. Alpha component specifies the transparency of the color: 0 is fully transparent and 255 are fully opaque.

Hardware Requirements:

1. A smartphone with good quality camera.
2. System for application development: A computer system with web browser enabled in it for application development.
3. Processor: Above 2 GHz
4. Hard Disk: 50 GB
5. RAM: 4 GB

Software Requirements:

1. MATLAB (**mat**rix **lab**oratory).
2. Android Studio.

V. LITERATURE SURVEY

The first paper [2] that is, "Text Detection and equation solving in scene images with unsupervised feature learning" by Adam Coates, Blake Carpenter, Carl Case, Sanjeev Satheesh, Bipin Suresh, Tao Wang, David J. Wu, Andrew Y, provides a general overview of how to perform detection of text and identification in scene images. Our results point out that it may be possible to achieve high performance using a more automated and scalable solution. With more scalable and sophisticated feature learning algorithms currently being developed by machine learning researchers, it is possible that the approaches pursued here might achieve performance well beyond what is possible through other methods that rely heavily on hand-coded prior knowledge.

The second paper [2] i.e. "Linear equation solver in android using OCR" by Amey Chavan, Asad Naik provides an efficient method to process image before actually converting it to text. The image is capture, cropped and then blurred to denoise image. Text recognition is done by Google Tesseract. The detected text is sent back to the text file. The confirm equation is then solved with parser and Cramer rule and results are displayed.

The third paper [4] i.e. "Optical Character Recognition Techniques: A Review" by Er. Neetu Bhatia provides a study on accurate recognition. It is directly depending on the nature of material to be read and by its quality. But as result says current research is not concern about various type of handwriting such as cursive handwriting and child handwriting which will require high supervised system. In this paper we studied that selection of relevant classification and feature extraction techniques plays an important role in performance of character recognition. In this paper we have studied various papers with variety of algorithms. But still there are many premature problems, when multiple variables exist in equations. Also it's hard to detect variable such as x^{10} .for eq. variables having 2 digits in raise to position. So, in future there is lots of work to remove drawbacks.

The fourth paper [5] i.e. "An overview of the Tesseract OCR engine" by Ray Smith provides a brief idea about Tesseract and how it works with android. Also paper includes other experiences about Tesseract and where it can be useful. Paper include techniques such as Line and word finding, Baseline fitting, fixed pitch detection and chopping, Proportional word finding, Word recognition, etc. Some of which we might implement in future. The line finding algorithm is designed so that a page which is skewed can be recognized without having de-skew, thus it saves lot of image quality. The key parts of the process are blob filtering and line construction. Support for a number of new image formats was added using the Leptonica library. Tesseract can detect whether text is monospaced or proportional.

VI. FUTURE WORK

According to current results, handwriting recognition is an area where we need to make it more accurate. We are trying to generate noise cancelling algorithm in preprocessing image. The idea can also be extended to text recognition on a tablet device with a stylus, which would provide seamless workflow for the user. Also we can add more equation forms such as complex exponential and differential equations. Also if a company needs a java program to calculate the boiling

point of an acid based on its ph value. The formula for calculating ph value is complex. So they write java program but instead of executing a java program again and again on computer they can use our app. They can click picture of values and it will solve it in a second. So according to requirement of a company needs we can make changes to the code.

VII. CONCLUSION

In this paper, we describe an Android-based system for capturing and solving an image containing a mathematical expression. The system consists of two parts - an Android application for capturing the camera image, and for displaying the recognized text and solution, and a Server for performing the image processing, text recognition, and equation solving algorithms. The camera image of a mathematical expression is sent from the Android phone to the server where it is first processed using Bitmap. For printed text text-recognition is performed with the Tesseract OCR engine. The detected text is sent back to the Android device for confirmation by the user. The confirmed equation is then solved with MATLAB on the server, and the solutions are displayed on the Android device.

VIII. References

[1]Tesseract Optical Character Recognition Engine. Available: http://code.google.com/p/tesseract-ocr/.

[2] Adam Coates, Blake Carpenter, Carl Case, Sanjeev Satheesh, Bipin Suresh, Tao Wang, David J. Wu, Andrew Y, " Text Detection and equation solving in scene images with unsupervised feature learning", Computer Science Department Stanford University.

[3] Amey Chavan, Asad Naik, "Linear equation solver in android using OCR", Department of Information Technology, Trinity College of Engineering, Pune, India.

[4] Er. Neetu Bhatia, "Optical Character Recognition Techniques: A Review", International Journal of Advanced Research in Computer Science and Software Engineering.

[5] Ray Smith, "An overview of the Tesseract OCR engine",