QUADRATIC EQUATION SOLVER

H S RAJATH[1], HEMACHANDRA SAGAR S[2], RAKESH H R[3], SANDEEP R[4]
[1]DEPARTMENT OF ELECTRONICS AND COMMUNICATION
[2]VIDYAVARDHAKA COLLEGE OF ENGINEERING
MYSURU. KARNATAKA, INDIA

(E-mail: rajathgowda100@gmail.com, hemachandra.sagar.s@gmail.com, rakeshhassan83@gmail.com, Sandeep.ece@vvce.ac.in,)

Abstract— In this paper we demonstrate a prototype of quadratic equation solver using machine learning that allows user to write the coefficient of a quadratic equation using mouse on the window screen and displays the solution. It provides both real and imaginary roots of the equation. The prototype makes use of Convolution Neural Network (CNN) for object detection and Support Vector Machine (SVM) for image classification. Tools used in propose system are Python IDLE. The collection of datasets is from MNIST and handwritten digit images are captured and fed. The efficiency of the proposed prototype is around 90%-95%.

Keywords—Convolution Neural Network (CNN), Support Vector Machine (SVM), Histogram of Oriented Gradient (HOG), Modified National Institute of Standard and Technology (MNIST).

Introduction

The quadratic equation is an equation with second degree meaning it contains at least one term that is squared. The standard form of the quadratic equation is ax2+bx+c = 0, where a, b, and c being constants and x is an unknown variable. Quadratic equations are a ubiquitous part of student life. Generally, one has to make use of a pen and paper or calculator to solve quadratic equations. In pen and paper approach it is difficult to solve complex equations and to get imaginary roots. Where as in calculator the equation needs to be entered in a specific format to get roots of real and imaginary and it is time consuming. Our goal is to develop a prototype that bridges the gap between the technology and the traditional pen and paper approach in an impulsive manner. It allows user to write the coefficients of the quadratic equation using mouse on the display of laptop and solution will be calculated and displayed. In general handwriting recognition is classified into two types as off-line and on-line character recognition. Off-line character recognition involves automatic conversion text into an image into letter codes which are usable within computer and text processing applications. Off-line handwriting recognition is more difficult as different people have different handwriting styles. But, in the on-line system on-line character recognition takes place once the user writes the coefficient of equation the digit gets captured as output image and get recognized. Initially, we limit the scope of the task to solve equation of degree 2. Mobile applications for solving such equations do exist. However, these apps are essentially templates for pre-defined equations where users enter numbers in static text boxes. While the constrained interface ensures robust capture of the input, it is cumbersome for the user, who must click multiple times to input an equation. A more intuitive approach would be the use of prototype to snap a picture of the written and text expression to get the respective roots.

I. LITERATURE SURVEY

A. Camera Based Equation Solver for Android Devices

In the existing method, they demonstrate an Android based mobile application equation solver that takes a camera image of an equation and displays the solution. The program is capable of solving simple arithmetic equations (addition, subtraction, and multiplication) and systems of linear equations.
The approach for solving an equation contained in an image is as follows:
Image Capture: Within the mobile application, the user is given two options **-** "Solve Printed" or "Solve Hand Written" which capture an image and initiate two different Hypertext Preprocessor (PHP) scripts on the server.

Binarization and Segmentation:
Binarization and segmentation are crucial steps in identifying the regions of interests. Otsu's adaptive thresholding was initially used. However, the processing time was fairly slow and the binarization was unsatisfactory for poor lighting conditions. Instead, the image is binarized using Maximally Stable Extremal Regions (MSER). Regions that are either too small or too large are excluded from the candidate list. Regions in the binarized image are labeled, and their centroids and bounding boxes calculated. Individual lines (a single equation or expression) are segmented out by plotting the y-centroids of the regions in a histogram with a bin size of 20 pixels. The average y-centroid of each line is assumed to be the maximum of the well-separated peaks. The midpoint

between each line is then calculated and used to segment the binarized image into images of individual lines.

Recognize Text: Text recognition is divided into two separate classes- computer printed expressions and hand written expressions. Computer printed expressions are processed with Tesseract [3], a free OCR Handwritten expressions are processes with a SVM prediction model.

Computer Text Recognition: Recognizing computer generated text is considerably easier than recognizing handwritten text. Due to the predictable structure (font style) of the characters, a template matching algorithm can be used. The free OCR engine Tesseract was chosen for printed text recognition. The accuracy of optical text recognition algorithms improves significantly if perspective distortion is first corrected. The topline and baseline of the regions in the segmented expression are found. The segmented line image is shown in Figure 1. (a). The RANSAC MATLAB Toolbox is used to compute RANdom SAmple Consensus (RANSAC) fits to the top-line and baseline as shown in Figure 1. (b). A bounding box for the distorted text is found, and the four corner points identified. Using the leading edge of this bounding box, the desired, corrected bounding box is then calculated, as shown in Figure 1. (c). A projective affine transformation that maps the four corner points of the distorted box to the desired box is then used to correct for the perspective distortion. The resulting image after correction, as shown in Figure 1. (d), is then passed to the Tesseract OCR engine.

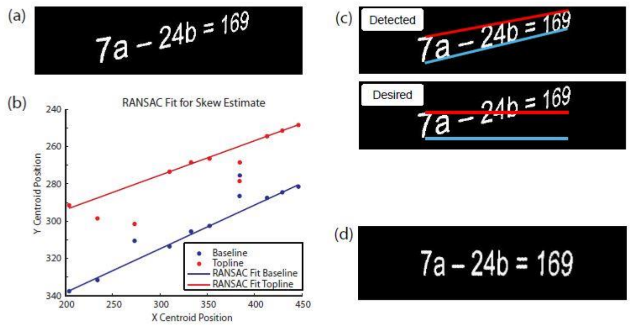


Fig 1. (a) Segmented line image showing perspective distortion. (b) RANSAC fits to the top line and the baseline. (c) Top: The detected, distorted bounding box. Bottom: The desired, corrected bounding box. (d) Corrected image after projective affine transformation is applied to the distorted text.

Handwritten Text Recognition: Though Tesseract performed well for printed text, with a detection rate better than 85% for fonts within its database. However, its detection rates for handwritten text is below 50%, likely due to size variations in the writing and a lack of matching fonts in its database. A machine learning algorithm based on SVM was applied instead. SVM is a supervised learning method that analyzes data and recognizes patterns. It is often used for classification and regression analysis. The prediction model was created using libSVM, a set of tools for training and modeling SVM developed. In order to create a training dataset, character images must first be converted into vector form. After line segmentation, region labels are used to determine the bounding. A small amount of padding is added to the border, as shown in Figure 2. (a). The segmented character is down sampled to 32 × 32 pixels (by deleting the alternative rows and columns) and further divided into 64 4 ×4 regions, with region 1 at the top left and region 64 at the bottom right. The count in each region is the determined vector value, as shown in Figure 2. (b). This conversion thus results in a 64 dimensional vector for each character image. Initially, the optical recognition of handwritten digit data set was used as the training set. The 64 dimensional vector conversion of this data set was obtained. After training libSVM, a test of their handwritten digits resulted in a low prediction rate of between 20-30%. So, they decided to create a new training data set based on our own handwriting. A simple application for taking a camera image of a single character, segmenting and down sampling the character, calculating the corresponding 64 dimensional vector, and writing it to an output text file, was created to generate a training dataset. Approximately 20 entries for each character in the dictionary was entered as a training data set.

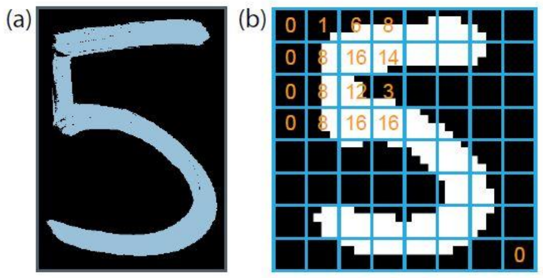


Fig 2. a) Segmented character from input image. (b) Character down sampled to 32×32 pixels. The image is then further divided into 64 4×4 regions, with region 1 at the top left and region 64 at the bottom right. The count in each region is the vector value.

User confirmation and solve equation:
Even for a system with a single character recognition rate of 95%, the probability of detecting a set of equations with 12 characters completely correctly is only slightly above 50%. It is thus critical to provide the user with a means to confirm or to edit the recognized text. Upon confirmation, the expression is sent back to the server where it is solved. Once the user confirms the equation, the equation is converted into a MATLAB-readable expression by inserting missing operators. For example, the equation ‘2a + 5b = 6’ is converted to ‘2a + 5b = 6’and the equation ‘a2 + a = 3’ is converted to ‘a2+a = 3’. After parsing, MATLAB functions *eval()* and *solve()* are invoked to solve the given expression. *eval()* is used for expressions with no variables such as ‘123_4563+789’ and *solve()* is used for a system of equations. The resulting solution is then sent back to the
Android device.

Experimental Results:
Tesseract OCR by limiting the number of alphabets to `a’, `b’, and ‘x’, the spell-check function of Tesseract is essentially circumvented. Segmenting and detecting an entire line of text was actually more accurate than segmenting and detecting individual characters. Tesseract performs well for fonts that exist in its database, with an accuracy in the range of 80-85%.

The notable exceptions were the characters ‘0’, ‘□’, ‘=’, and ‘a’. The character ‘0’ was at times mistakenly identified as ‘()’. The character ‘a’ was at times mistakenly identified as ‘3’or ‘8’. The characters ‘□’ and ‘=’ were at times completely absent from the detection.
SVM handwriting recognition with SVM has a detection rate of 80-85% for our handwriting. It is expected that the accuracy will be considerably lower for individuals not in training dataset. A few outliers are noted here, the characters ‘0’, ‘3’, and ‘1’ are at times mistakenly identified as ‘a’ or ‘6’, ‘5’, and ‘7’ respectively. Problems detecting the mathematical symbols ‘-’ and ‘+’ appear to arise from the fact that their size is quite different from other characters within the dictionary

II. EQUATIONS

In elementary algebra, the quadratic formula is the solution of the quadratic equation. There are other ways to solve the quadratic equation instead of using the quadratic formula, such as factoring, completing the square, or graphing. Using the quadratic formula is often the most convenient way.
The general quadratic equation is

$$ax^2 + \mathrm{bx} + c = 0$$

Here x represents an unknown, while a, b, and c are constants with a not equal to 0. One can verify that the quadratic formula satisfies the quadratic equation by inserting the former into the latter. With the above parameterization, the quadratic formula is:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Each of the solutions given by the quadratic formula is called a root of the quadratic equation.

III. PROPOSED METHOD

In order to overcome the above drawback, present in existing method. The quadratic equation solver prototype is developed. The process involved are:

A. Capture an image

The webcam present in laptop /local desktop is used to capture the sample image of any digit of selecting appropriate resolution of their choice.

B. Conversion of color image to gray scale image

There are two methods for converting color image to greyscale image: Weighted method or luminosity method. We have seen the problem that occur in the average method. Weighted method has a solution to that problem. Since red color has more wavelength of all the three colors, and green is the color that has not only less wavelength then red color but also green is the color that gives more soothing effect to the eyes. It means that we have to decrease the contribution of red color, and increase the contribution of the green color, and put blue color contribution in between these two. So the new equation that form is:

grayscale image = $((0.3\times R) + (0.59\times G) + (0.11\times B))$.
According to this equation, Red has contribute 30%, Green has contributed 59% which is greater in all three colors and Blue has contributed 11%. Applying this equation to the image, we get the image as shown in Fig 4.

C. Gaussian Blur

During this conversion from color image to graysacale image, the image loses the characteristics like sharpness, edges and shadow and also presence of Gaussian noise. Therefore we use Gaussian filter to eliminate the Gaussian noise and smoothens the edges. There are many ways to perform the edge detection.

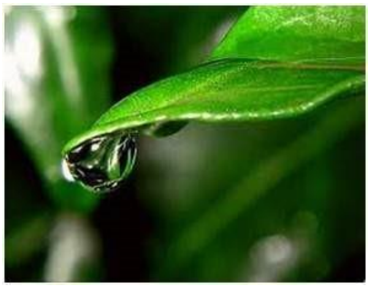


Fig 3. Original image

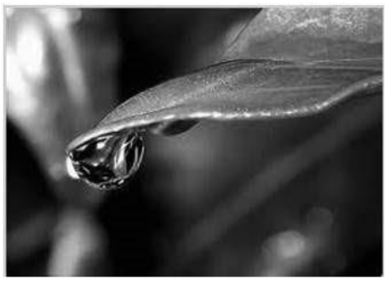


Fig 4. Grayscale image

However, it may be grouped into two categories, that are gradient and Laplacian. The gradient method detects the edges by looking for the maximum and minimum in the first derivative of the image. The Laplacian method searches for the zero crossings in the second derivative of the image to find edges. However, in calculating 2nd derivative is very sensitive to noise. This noise should be filtered out before edge detection. To achieve this, “Laplacian of Gaussian” is used. This method combines Gaussian filtering with the Laplacian for edge detection. In Laplacian of Gaussian edge detection uses three steps in its process. The First one is filter which is the image object. Secondly, it enhances the image object and finally detects. Here, Gaussian filter is used for smoothing and the second derivative is used for the enhancement step. In this detection criteria, the presence of a zero crossing, In the second derivative with the corresponding large peak in the first derivative. In this approach, at first the noise is reduced by convoluting the image with a Gaussian filter. The isolated noise points and small structures are filtered out. However, the edges are spread with smoothing. Those pixels are locally maximum gradient which are considered as edges by the edge detector in which the zero crossings of the second derivative are used. The zero crossings are only insignificant edges to avoid the detection that corresponds as the first derivative is above some thresholds, which are selected as edge points. In this the LoG mainly uses two methods which are mathematically similar. At first, let us convolve the image object with Gaussian smoothing filter and then compute with the Laplacian result. Secondly, we shall convolve the image object with the linear filter which is the Laplacian of the Gaussian filter. This is also the case in the LoG. Smoothing (filtering) is performed with a Gaussian filter. The enhancement is done through transforming edges into zero crossings and the detection is done by detecting the zero crossings for the various samples.

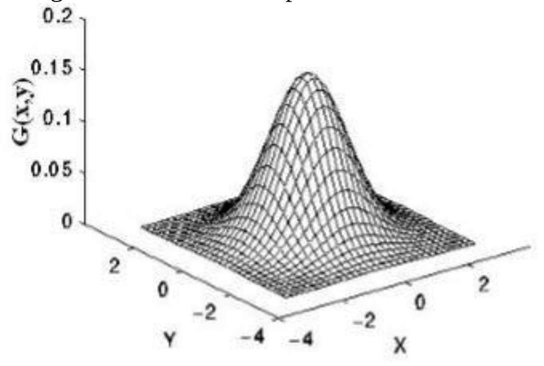


Fig 5. The two dimension Laplacian of Gaussian (LoG)

D. Applying Contour

A contour is a closed curve of points or line segments, representing the boundaries of an object in an image. The input to the contour finding process is a binary image, which we will produce by first applying thresholding and / or edge detection. active contours (also known as snakes or deformable contours) and active surfaces (also known as deformable surfaces). The active models deform on the image domain and capture a desired feature by minimizing an energy functional subject to certain constraints. The energy functional usually contains two terms: an internal energy, which constrains the smoothness and tautness of the model, and an external energy, which attracts the elastic model to the features of interest (FOI). The contoured image is then segmented by threshold algorithm.

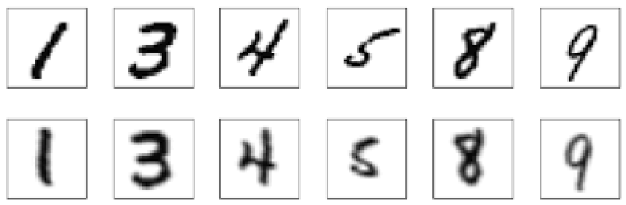


Fig 6. Example of applying contour

E. Image segmentation

Image segmentation is an important processing step in many image, video and computer vision applications.

Extensive research has been done in creating many different approaches and algorithms for image segmentation, but it is still difficult to assess whether one algorithm produces more accurate segmentations than another, whether it be for a particular image or set of images, or more generally, for a whole class of images. To date, the most common method for evaluating the effectiveness of a segmentation method is subjective evaluation, in which a human visually compares the image. segmentation results for separate segmentation algorithms, which is a tedious process and inherently limits the depth of evaluation to a relatively small number of segmentation comparisons over a predetermined set of images.

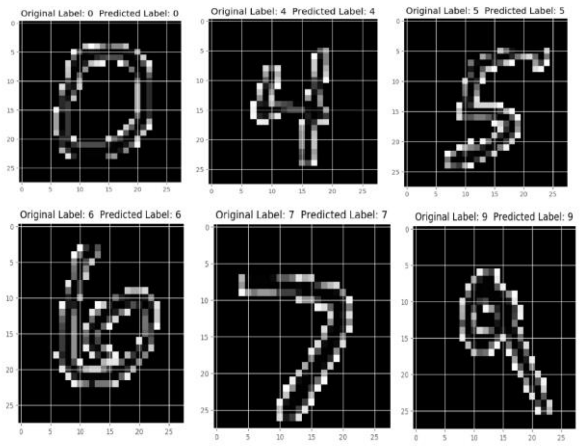


Fig 8. The segmentation process

Another common evaluation alternative is supervised evaluation, in which a segmented image is compared against a manually segmented or pre-processed reference image. Evaluation methods that require user assistance, such as subjective evaluation and supervised evaluation, are infeasible in many vision applications, so unsupervised methods are necessary. Unsupervised evaluation enables the objective comparison of both different segmentation methods and different parameterizations of a single method, without requiring human visual comparisons or comparison with a manually-segmented or pre-processed reference image. Unsupervised methods are crucial to real-time segmentation evaluation, and can furthermore enable self-tuning of algorithm parameters based on evaluation results.

F. Convolution Neural Network

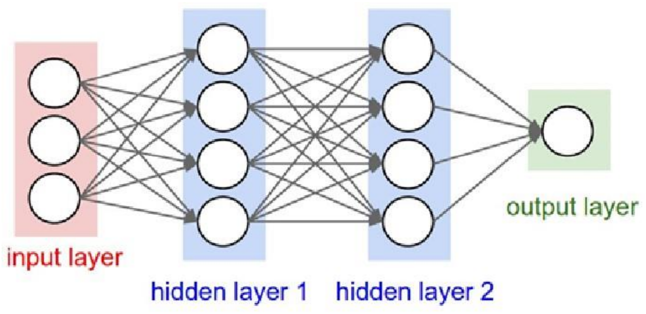


Fig 9. The layers of convolution neural network

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of deep neural networks, most commonly applied to analyzing visual imagery. CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually refer to fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks make them prone to overfitting data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. However, CNNs take a different approach towards regularization, they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics.

G. Support Vector Machine

.

Support vector machines are based on the Structural Risk Minimization principle from computational learning theory. The idea of structural risk minimization is to find a hypothesis h for which we can guarantee the lowest true error. The true error of h is the probability that h will make an error on an unseen and randomly selected test example. An upper bound can be used to connect the true error of a hypothesis h with the error of h on the training set and the complexity of H (measured by VC-Dimension), the hypothesis space containing h. Support vector machines find the hypothesis h which (approximately) minimizes this bound on the true error by effectively and efficiently controlling the VC-Dimension of H. SVMs are very universal learners.

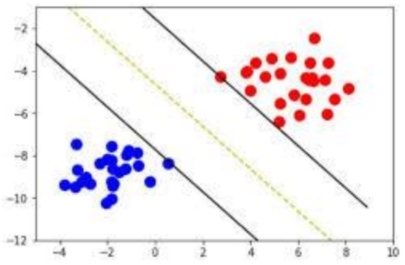


Fig 10. SVM Classifier.

H. TRAINING AND TESTING MODELS

The image obtained after classification get stored in respective datasets and this dataset present in the database get trained for about 8-10 epochs. Here training model consists of 60% of original dataset. It is categorized into two parts:

1. To build up our prediction algorithm and adjust weight on neural network.
2. Our algorithm tries to tune itself to the features of the training dataset.

Finally, the trained datasets are compared with testing dataset. The testing model consists of 20% of original dataset. This is done to determine how well the algorithm is trained and to test the final output in order to confirm the actual predictive power of the network.

I. Graphical User Interface

Graphical User Interface is a form of user interface that allows users to interact with electronic devices through graphical icons and visual indicators such as secondary notation, instead of text based user interface, typed command labels or text navigation, which require commands to be typed on a computer keyboard. Here GUI is developed to enter the coefficients of the quadratic equation. Since the quadratic image is captured initially and digits or coefficients will be recognized and be placed in the respective slot.

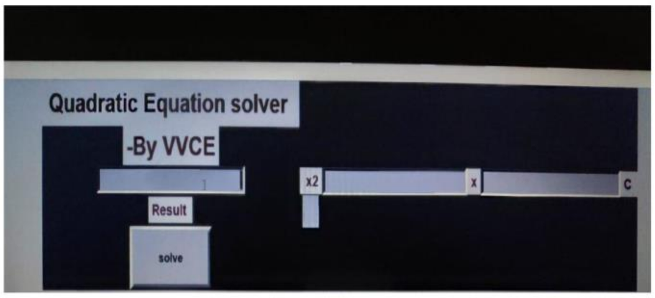


Figure 11. The hierarchy of segmentation evaluation methods. Our emphasis in this project is on the unsupervised objective evaluation.

IV. RESULTS AND DISCUSSION

The detection rate of individual characters in the dictionary dictates the overall probability of correctly identifying an equation. For testing, two sets of equations were used, one printed from the computer and the other handwritten, resulting in approximately 1000 instances of each character within the dictionary.

The real and imaginary roots obtained from solving equation is displayed on output window is shown
in following Figure.

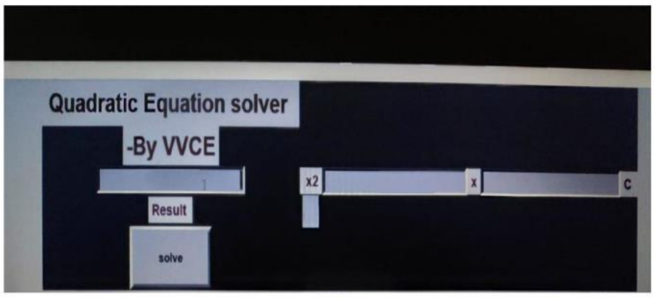
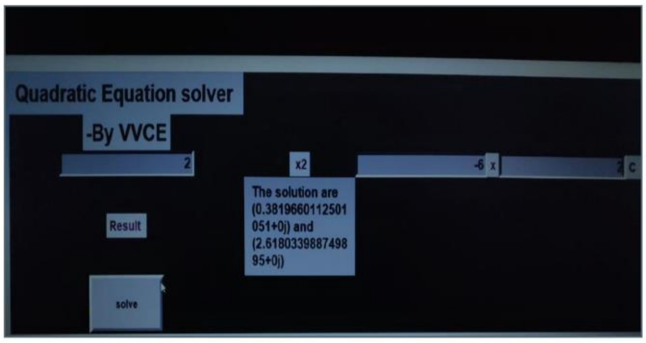
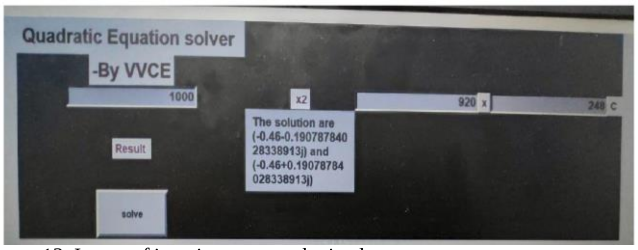


Figure 11. Initial output screen.



The solution are (0.3819660112501051+0j) and (2.618033988749895+0j)

Figure 12. Image of real roots obtained.



The solution are (-0.46-0.190787840 28338913j) and (-0.46+0.19078784 028338913j)

Figure 13. Image of imaginary roots obtained.

.

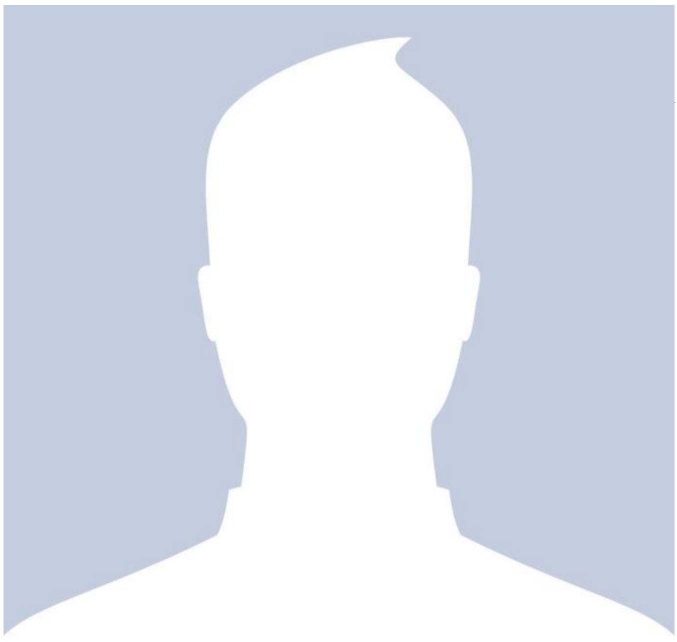
V. APPLICATIONS

Quadratic functions are more than algebraic curiosities they are widely used in science, business, and engineering. The U shape of a parabola can describe the trajectories of water jets in a fountain and a bouncing ball, or be incorporated into structures like the parabolic reflectors that form the base of satellite dishes and car headlights. Quadratic functions help forecast business profit and loss, the course of moving objects, and assist in determining minimum and maximum values. Most of the objects we use every day, from cars to clocks, would not exist if someone, somewhere hadn't applied quadratic functions to their design. We commonly use quadratic equations in situations where two things are multiplied together and they both depend of the same variable. For example, when working with area, if both dimensions are written in terms of the same variable, we use a quadratic equation. Because the quantity of a product sold often depends on the price, we sometimes use a quadratic equation to represent revenue as a product of the price and the quantity sold. Quadratic equations are also used when gravity is involved, such as the path of a ball or the shape of cables in a suspension bridge.

VI. CONCLUSION AND FUTURESCOPE

In this project, we describe a prototype for solving an image containing quadratic equation using machine learning. The system captures the image containing quadratic equation both in computer font and handwritten form. Then system performs the image processing, text recognition, and equation solving algorithms. Once the image is captured, the color image is converted into grayscale image using luminosity method. Feature extraction takes place using HOG, noise and interference are removed by using Gaussian filter, contour is applied to the filtered image and image segmentation is performed using unsupervised segmentation algorithm, then the segmented image is converted into binary image by applying threshold. The recognized image is classified using SVM model and the recognized coefficient numbers are interfaced to output GUI window to obtain the roots of the equation.

The prediction rate of the SVM model is highly dependent on the training dataset. Currently, thetraining dataset only contains our handwriting. Ideally, a larger training dataset from multiple individuals should be collected, to accommodate for wide ranging styles in writing. An extension would be for the system to train itself. Each time the user sends the confirmation for an equation, the corresponding 64 dimensional vectors for the detected characters can be added to the existing training dataset. This would allow the training dataset to grow quickly and efficiently without the need for separate training. The second improvement would be to make the application a stand-alone device by porting the existing server code to OpenCV on the mobile device. The idea can also be extended to text recognition on a tablet device with a stylus, which would would provide a seamless work flow for the user.



HEMACHANDRA SAGAR S RAKESH H R H S RAJATH