DIP Project Mid-Evaluation Report

LaTeX Code Generation from Printed Equations

Team Escher

Kritika Prakash[1] and Karthik Chintapalli[2]

[1]20161039
[2]201501207

2nd November 2018

1 GitHub Repository

`https://github.com/Kritikalcoder/Latex-Generation-from-Printed-Equations`

2 Main Goals

To create a system that takes a photograph, scan or screenshot of a printed mathematical equation and produces a valid LaTeX markup code representation to be able to generate the equation.

3 Problem Definition

Working with lengthy involved mathematical equations can be cumbersome, tedious and error-prone. Mathematical equations in the printed form are not easily reproducible in new LaTeX documents. This is because, once a LaTeX document has been rendered, the underlying producer's code to recreate it is inaccessible.

3.1 Approach

The Latex Code Generation from Printed Equations Project aims to automatically generate valid LaTeX expressions for a photograph, scan or a screenshot of a printed mathematical equation.

3.2 Scope

The scope is defined by the kind of mathematical operations and expressions the system will be able to recognize and recreate. The equation is assumed to be the primary data in the input image, as opposed to extraction of the mathematical equation from an entire page full of content other than the equation.

4 Current Progress

The following modules have been built so far:

4.1 Page Optimization

4.1.1 Binarization

The input RGB image is first converted into a binary image for processing. The input images considered are either photographs or screenshots. Photographs generally contain shadows and hence, are more challenging to threshold. In order to differentiate between the two image types, an image is first converted to grayscale and the proportion of pixels that are gray (having an intensity value between 15 and 240) is extracted. If this proportion is less than 0.1, the image are treated as scanned images or screenshots. The other images are treated as photographs. The binarization method is described for each of the two categories of images.

- Screenshots and Scanned Images: It is assumed that these images do not require lighting correction. Hence, it suffices to perform Otsu's thresholding on these images.
- Photographs: Photographs generally contain uneven lighting and various page imperfections. Hence, adaptive thresholding is performed, followed by additional noise removal.

  First, high-resolution images are blurred with a Gaussian filter to smoothen edges and remove high frequency noise. This pre-filtering is very beneficial for character recognition. in high-resolution images. However, this is not done for low-resolution images as it does not preserve sufficient details due to the low number of total pixels in the image. The threshold for high resolution is taken to be 2000×1000 pixels, and images which are above this threshold are smoothed using a 10×10 Gaussian filter with standard deviation of 3. Adaptive thresholding is performed for the binarization in order to compensate for uneven lighting. The window size used is equal to $1/60^{\text{th}}$ of the smaller dimension of the image. The window size is chosen such that it can handle uneven lighting while still being small enough to preserve the characters.

  After the binary image is obtained, it is inverted so that the foreground is now the text. Noise removal is performed using a morphological opening

operation. Then, in order to close gaps in character edges, a closing operation is performed. Finally, hole filling is performed to fill the gaps within characters. The hole size threshold is chosen carefully so as to avoid filling holes that are actually part of characters. The final binarized output is obtained by restoring the original parity.

4.1.2 Skew Correction

The next step is to correct the orientation of images so that the equation is horizontally aligned. In order to do this, the dominant orientation needs to be determined. To compute the dominant orientation, the Hough transform of the image is computed. Generally, most equations have multiple horizontal lines, such as fraction bars, equal signs, and negative signs. Based on this fact, it is reasonable to assume that the dominant orientation is given by the correct, horizontal orientation. To prevent large magnitude peaks given by long diagonal lines, such as the diagonal division bar, from being considered as the dominant orientation, the mode of the top four magnitude Hough peaks is chosen.

There are some other details to be considered while implementing this. While an image could be rotated by a small positive angle, the orientation in Hough space is given as the angle between the normal and the horizontal axis, which is the complement of the required angle. This can be corrected by subtracting $90°$. After deskewing, edge softening is performed to smoothen edges that are jagged due to rotation. Edge softening is performed using an opening operation.

4.2 Character Recognition

4.2.1 Character Segmentation

Because characters are matched individually, the characters are first extracted from the derotation algorithm output. We use centroids and bounding boxes of edge maps for simplicity and for the ability to extract characters surrounded by others (such as under a square root). First, the edge map is obtained by eroding the inverted image and then XORing that with the original inverted image, resulting in a white edge map on a black background. Then, for each edge, we extract its centroid, bounding box and convex hull. This successfully extracts many characters, except some edge cases. If a character's convex hull is fully contained within another convex hull, we examine the edge map to determine if the outer character fully surrounds the inner character. If so, we discard the inner segmentation. This additional edge check is necessary before discarding the inner segmentation for proper behavior on equations with a square root symbol where several characters underneath the square root are completely within its convex hull. Finally, we extract each bounding box region and keep the largest single character within each region to extract only the character of interest in cases like the square root where other characters are contained in its bounding box.

5 Results

5.1 Binarization

The binarization procedure described works well for most cases. However, there are issues with the hole filling procedure. In cases where the equation occupies a smaller fraction of the entire images, the holes within characters are similar in size to holes due to noise. Hence, they are identified as holes and are filled.

5.2 Skew Correction

The skew correction procedure works well for high resolution images as there is sufficient evidence for the dominant direction. However, for low-resolution images where there are dominant directions other than the desired direction, the results are not as expected.

5.3 Character Segmentation

The character segmentation procedure is working well on most characters except characters like the infinity symbol where it is subdividing the character into two parts.

6 Future Work

6.1 Improvements

Due to the problems in the skew correction approach, a PCA based approach is being considered. The idea is that the direction of maximum variance in the image should give us an approximate measure of the dominant direction.

6.2 Remaining Modules

- Character Identification & Matching
  Characters and tokens have already been segmented from the cleaned input image. We need to match them with an existing database of characters. We will use Hu Invariant Moments and Circular Topology to identify characters against a database of characters. We will classify each character using Nearest Neighbour Classification.

- LaTeX Compilation
  We will assemble the corresponding LaTeX code from the identified sequence of input characters.

7 Tasks

S.No	Stage		Task	Member
1	Baseline Model	Page Optimization	Image Thresholding	Karthik
2			Binarization	Karthik
3			Skew Correction	Kritika
4		Character Recognition	Character Segmentation	Karthik
5			Character Identification	Kritika
6			Character Matching	Kritika
7		LaTeX Compilation	Equation Assembly	Both
8	Improvement		Testing	Both
9			Improving Baseline Model	Both
10			Comparative Study	Both
11	Presentation		Preparing Presentation	Both