Segmentation and Contour Detection for handwritten mathematical expressions using OpenCV

[1]Sakshi[0000-0002-8757-4001]
Chitkara University Institute of Engineering and Technology,
Chitkara University, Punjab, India
sakshi@chitkara.edu.in

[2]Vinay Kukreja[0000-0002-9760-0824]
Chitkara University Institute of Engineering and Technology,
Chitkara University, Punjab, India
vinay.kukreja@chitkara.edu.in

2022 International Conference on Decision Aid Sciences and Applications (DASA) | 978-1-6654-9501-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/DASA54658.2022.9765142

Abstract: Contour Detection owes its own significance in performing semantic segmentation and object classification. It is possible to recognize the edges of objects and readily locate them inside an image by employing contour detection. It is frequently the first step or achievable milestone in a varied range of exciting applications, such as extracting the foreground object from an image, classical-image segmentation methodology, identification or detection of the object, and many more. In this study, we have endeavored to perform contour detection using the OpenCV Adaptive Threshold method on the Aida handwriting math recognition dataset, containing ten batches of 10K images. Our experimentation successfully segmented the inputted mathematical expressions with an accuracy of 94.3% on the acquired dataset.

Keywords: handwritten mathematical expression, OpenCV, Adaptive Thresholding, contour detection, segmentation

1. Introduction

Segmentation has been one of the most trivial phases in the recognition task [1]. In part, this is because, while we are familiarised with writing mathematical expressions containing multivariable calculus, proportions, decimals, exponentials, or indices by hand, there is a dearth of human-adaptable methods for inputting or digitalizing these expressions into a digital system [2]. Even though mathematical equations have two-dimensional structures, handwriting analysis is convenient [3]. However, there are two problems to solve: Recognition of symbols and analysis of structural relationships [4]. The entire recognition story has a significant call for effective segmentation as a mathematical expression, and its content is usually complicated and nested at times. Thus, to accomplish the task of recognition of entire expression, we need first to perform effective segmentation of individual and constituting symbols integrated into the mathematical expressions [3]. In this study, we aim to reach out to an effective segmentation methodology to perform the overall recognition task efficiently.

1.1. Motivation

As OpenCV has been widely deployed for easy availability and compatibility reasons. The authors in the study are motivated to probe into and scrutinize its application for the segmentation of mathematical characters from whole complex mathematical expressions. These mathematical expressions are taken in handwritten form. Also, exploring the potential of varied thresholding techniques is one of the motivating factors inspiring this work. Therefore, the author has performed comparative analysis further in the study while experimenting with various thresholding techniques.

2. Background

2.1. Defining contours

Contours could be thought of as a plot or kind of curve structure that connects all of the perpetual and consistent points (mostly visualized along the border of a digital image) that seem to be similar colors or intensity and are associated together. The contours are a handy tool for various applications, including structure analysis, object segmentation, and later recognition, amongst many others.

2.2. Role of contours in the segmentation process

Contours are defined areas inside an image that define the area of interest. A contour is a collection of points that has been subjected to interpolation. Segmentation is a technique for describing, delineating, and classifying regions of interest. Segmentation is a method that uses a picture to determine the area, curvature, or contour of the desired target region. The segmentation of a digital image is dependent on several different features and parameters.

It is possible to build image segmentation models based on curve flow, curvature, and contour to obtain the accurate target zone or segmented part in an image for subsequent analysis and processing by using the specific contour models. Contour models can also be used to process images of different and multiple modalities, a technique known as multimodal image processing. Active contours define zones with the required pixel intensities based on the energy forces and situations that exist in the scene. Segmentation is accomplished through the use of a variety of active contour models.

2.3. Contour Properties and Features

2.3.1. Contour Properties

Contour generation and detection are dependent on its internal attributes, termed as properties of the contours. Some of the significant properties of contours are described below:

1. **Aspect Ratio:** It measures the ratio of width to height of the bounding box rectangle formed after an object for detection.

Aspect Ratio= Width of bounding box/Height of bounding box

2. **Extent:** The ratio of the contour area to the bounding rectangle area is known as the extent.

Extent= Object Area/Bounding Rectangle Area

3. **Solidity:** The solidity of a shape is defined as the ratio of its contour area to its convex hull area.

Solidity= Contour Area/Convex Hull Area

4. **Equivalent Diameter:** It is the circumference of a circle whose area is equal to the contour area.
5. **Orientation:** The orientation of an object is defined as the angle at which it is directed.
6. **Pixel points and mask value:** The pixel points are generally the coordinates that help to mark the region or area of contour. The mask value of contour enables us to extract patches and specific regions of absolutely random shape from images.
7. **Mean Color or Mean Intensity:** In grayscale mode, it might represent the average intensity of the object.
8. **Extreme Points**: Extreme Points are the points on an object that is at the very top, very bottom, very right, and very left of the object. It is basically with extreme coordinates of the object.

These properties are frequently accessed properties of contours. More of them can be read from documentation of MatLab region props.

2.3.2. Contour Features

Some of the significant contour features are usually extracted and accessed using macros, and functions provided by the OpenCV library are mentioned as:

1. **Moments:** Image moments assists in calculating some properties of an item, such as the center of mass of the object, the area of the object, and so on.
2. **Contour Area**: The area of contour usually uses the cv.contourArea() function.
3. **Contour Perimeter:** It is sometimes referred to as arc length. It can be determined by calling the cv.arcLength() function. When passed True, the second argument specifies whether the shape is a closed contour or simply a curve.
4. **Contour Approximation:** Depending on the precision, the selection is made accordingly. In contour approximation, we convert a contour form to a shape with fewer vertices. It is based on the implementation ideology of the Douglas-Peucker algorithm.
5. **Convex Hull**: Contrary to popular belief, Convex Hull is not a contour approximation. Correct convexity faults are found in a curve. It's generally accepted that convex curves are constantly bulging out or are at least flat. Convexity defects, on the other hand, refer to an internal bulge.
6. **Convexity Defects:** A function is equipped in cv2, named convexitydefects(). When given the contour and its corresponding hull indexes, this function provides an array containing the convexity flaws as an output.
7. **Bounding rectangle:** It is the kind of structure drawn from the edges of the acquired object for segmentation. There are two categories of these rectangles, as mentioned in figure 1.

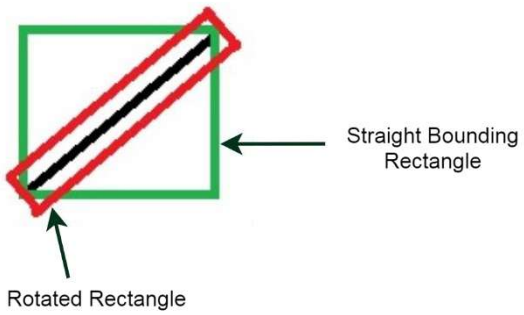

Figure 1: Types of Bounding Rectangle

3. Related works

In this section, the authors have reviewed the existing works performed on the segmentation of handwritten mathematical expressions. The study by Nomura [5] focused on detecting and segmenting the touching characters of mathematical expressions. They use two stages: the detection stage and the segmentation stage; in the detection phase, they focus on detecting the connected components and are segmented as single character images. Eleven journals have been taken by the author, and from these eleven journals, 21

scanned math documents are considered, including 391 pages. A total of 2978 are touching characters out of 14K, which is approx. 2% and they achieved a 95.1% improved rate from the proposed methodology.

Human-computer interaction is widespread or needs of today's society. In the paper by Gurav [6], OpenCV has been deployed for segmentation on real-time finger tracking and contour detection. This study examines the critical mode of exchange: gestures, as we can control many household applications and other daily use items through gestures. The author uses Adaboost based hand pose detectors in their experiment. The system is trained on various features based on local contour sequence. It is used for pre-processing like rotation, translation, and scaling of images. Two algorithms are considered: convex hull with 92% accuracy and AdaBoost with 70% accuracy.

In the study by Li [7], the author proposes a traffic control system that presents a video-based solution for pre-recorded traffic video using cameras— in further the combined technique of virtual detector and blob tracking is used. The study includes openCV and visual c++ methods for detection, tracking, and counting purposes. This is used for moving vehicles. To detect the shadow of moving vehicles, otsu's thresholding method is used by the author.

A strategy proposed by Kanahori and Suzuki [8] discussed how to segment data using symbol bounding boxes, the distance between them, semantic properties, and other techniques. It was proposed in [1] to use a modified projection profiling method. Rows and columns with a more precise separation from one another are first selected in the proposed way. For the duration of the procedure, this creates a transient cluster of strokes in one matrix dimension. The second matrix dimension is then determined by evaluating all grouped entries. In the study presented by Qashlim [9], the authors proposed an android based smartphone technology that deploys openCV for segmentation of fish detection resulting in images in the form of length, width, and weight that could be helpful in determining the cost and sale. At the conclusion of the procedure, the elements are segmented one last time. Instead of relying on symbols, this technique relies on a preliminary study of the matrix structure and input strokes.

4. Materials and Methods:

Our research methodology at a glance looks lucid and self-explanatory from figure 2. The foremost step is extracting the dataset which has been downloaded from [10]. Thereafter, pre-processing has been performed by resizing the images. Later adaptive thresholding has been perceptibly performed, and later contour generation phase is initiated, and there comes contour detection phase. After this second phase of contour detection, properly segmented images can be obtained as the final outcome.

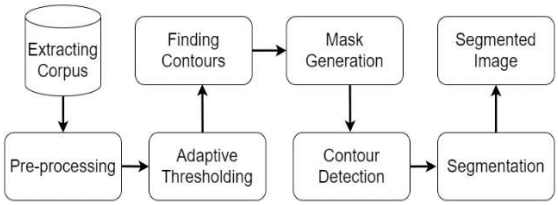

Figure 2: Research Methodology

4.1. About the Dataset:

There are a total of 100,000 photos in the Aida Calculus Math Handwriting Recognition Dataset, which is divided into ten batches. On each image is a snapshot of a handwritten calculus math statement (particularly related to the issue of limits), which was written with a dark tool on plain paper and then captured. The ground truth math equation in LaTeX is included with each image, along with bounding boxes and pixel-level masks for each character. All of the images have been created synthetically. A glimpse of some mathematical expressions of the dataset has been presented in figure 3.

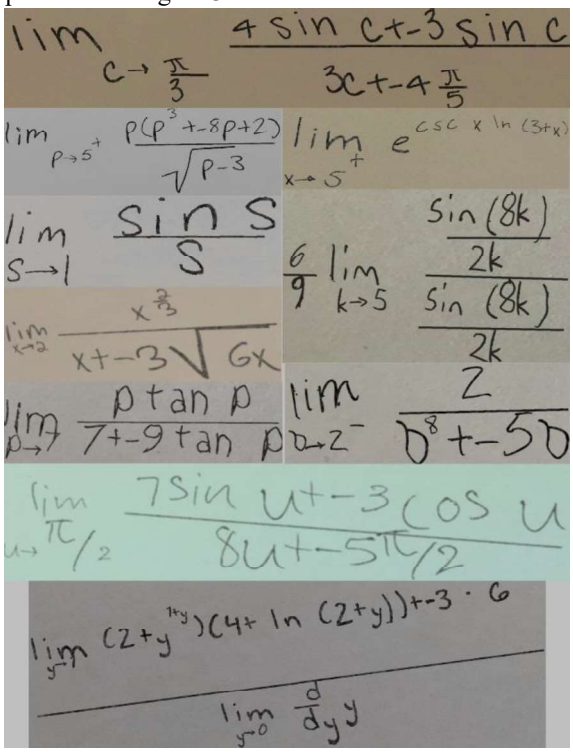

Figure 3: Sample Corpus containing mathematical expressions

4.2. Experimentation:

The experimentation has been performed on the system with disk space specifications of 108 GB and internal memory of 12.69 GB. The NVIDIA

tesla k80, based on two Kepler GPU, has been used in the process. The Microsoft Windows 10 Pro is the OS installed on the implemented system. The intel ® core ™ i5-8250U processor with eight logical processors is the built-in processor of the system used for experimentation. The implemented code construct for our acquired module has been executed on Google Collab, its in-built GPU. The dataset of 100k images from the Aida Calculus dataset has been downloaded after generating and uploading API tokens from kaggle[10]. The jpg format files have then been extracted and pre-processed at the onset. This downloaded dataset has been imported to the internal directory of Google Collab. Resizing the image to a specific width of 600px has been accomplished. Then Adaptive Thresholding has been applied to the resized images.

4.2.1. First Stage Finding Contours and Mask Generation

The contours after Adaptive Thresholding has been identified and extracted from the inputted images. Then masks have been generated as of the same shape and size as that of the image. The contours have then been drawn from the processed image due to Thresholding. These drawn contours are then masked. The stepwise working of our entire segmentation experiment has been presented in figure 4.

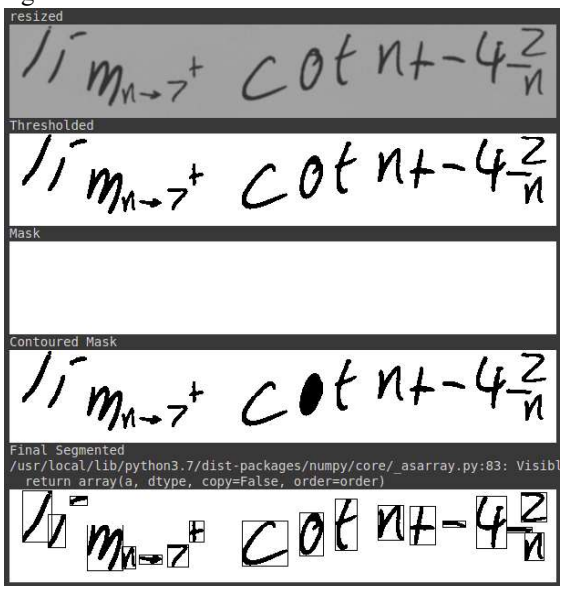

Figure 4: Stepwise illustration of the segmentation process.

4.2.2. Second Stage Contour Detection

The contours have been looked on from the inverted mask image. Again, identifying areas for contours has been accomplished, and thereafter, the areas of contours are found and retrieved. These areas are then being sorted in proper order. The in-built functions of OpenCV have been thoroughly deployed for drawing the rectangles around the segmented extracts of expressions. The list of all used functions of the OpenCV library has been tabulated in table 1. Thereby, the drawn contours over the image have squared. Later, each rectangle's independent and comfortable cropping has been performed to achieve the segmented symbols from the expressions at the end.

cv2.imread()	for reading image from a directory
cv2.resize()	to resize the image
cv2.imshow()	to display the image
cv2.adaptiveThreshold()	to binarize image
cv2.boundingRect()	to draw an approximate rectangle around the binarized image
cv2.findContours()	to extract contours from the binarized image
cv2.drawContours()	to draw extracted contours on image
cv2.contourArea()	to calculate the area of contours
cv2.rectangle()	To draw a rectangle with the passed coordinates as parameters
cv2.imwrite()	To save image in directory

Table 1: List of all OpenCV libraries deployed in our methodology

5. Predominance of our proposed model

Our proposed model holds the benefits of using segmentation via OpenCV and Adaptive thresholding as its strengths.

Preponderance of using OpenCV

The factors that convince and inspire us to use OpenCV for deployment are as mentioned below:

- Compatibility with other frameworks
- Clean and Compact Code
- Extending output for potential exploration
- Community Support
- Support by faster platforms like google Collab

Preponderance of using Adaptive Thresholding

The authors have experimented with varied thresholding techniques before finalizing to work on adaptive thresholding. We have experimented with adaptive thresholding, local thresholding, global thresholding, and Otsu binarization. Outcomes have been compared on the grounds of pixels' distribution by generating histograms, as displayed in figure 5.

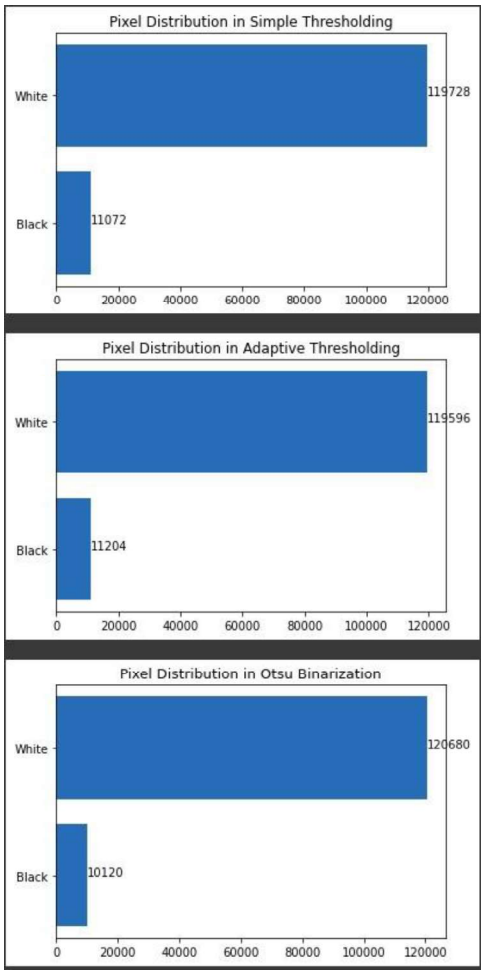

Figure 5: Comparative Analysis of varied thresholding techniques.

While performing thresholding on the inputted images, the sole focus was on keeping the strokes intact. Thus, on experimenting with adaptive thresholding, it has been observed that the intensity of most of the black pixels is the highest. Thus, the performance analysis of varied thresholding techniques witness the best outcomes with the adaptive threshold, so it has been considered part of the experiment.

6. Results and Conclusion:

The author detects handwritten mathematical characters and accomplishes the process of localizing them and extracting them for further process. The handwritten characters are segmented and removed to be analyzed further for usage like feature extractions, local distortion for data augmentations, recognition, etc. The performance of the model is evaluated by the number of images it segments and saves concerning time. To support the choice of our research methodology, a comparative analysis of varied thresholding techniques has been presented in figure 5. Next, to summarize, our method reads the image in grayscale. It resizes it to a width of 600 pixels while keeping the aspect ratio, after which it applies OpenCV Adaptive Thresholding to achieve the desired result. Adaptive thresholding has been deployed as it works well even with diverse lighting conditions. As presented in figure 6a) it has been captured in varied light modes. Performing the entire methodology on such images results with a final segmented image as shown in figure 7.

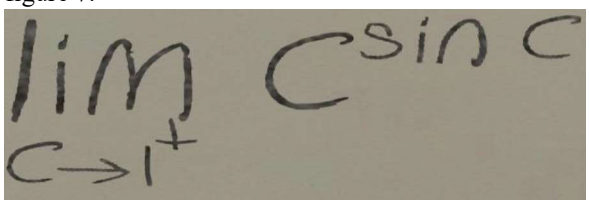

Figure 6a): Sample Input Image

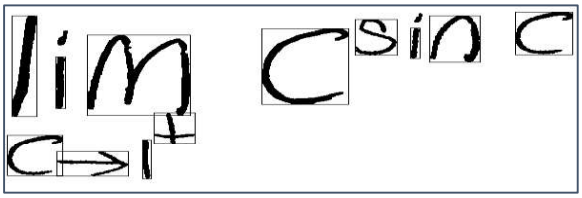

Figure 6b): Segmented extracts

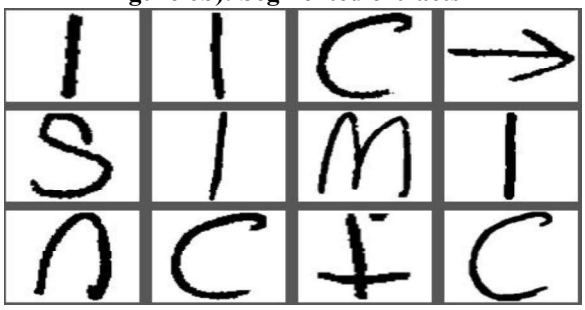

Figure 7: Final Segmented Images

Our segmentation process using the OpenCV findContours function discovers contours in the image and filters out irrelevant contours (or noise) based on their regions. The outline of each phrase is captured by the contours in the illustration (figure 6b). We extract each phrase from the image with the help of contours and preserve the results, achieving 94.3% accuracy.

Acknowledgment

The author would like to acknowledge the support received by Mae Fah Luang University, Thailand.

References:

[1] O. Yakovchuk, A. Cherneha, D. Zhelezniakov, and V. Zaytsev, "Methods for lines and matrices segmentation in RNN-based online handwriting mathematical expression recognition systems," *Proc. 2020 IEEE 3rd Int. Conf. Data Stream Min. Process. DSMP 2020*, pp. 255–261, 2020, doi: 10.1109/DSMP47368.2020.9204273.

[2] R. Yamamoto, S. Sako, T. Nishimoto, and S. Sagayama, "On-Line Recognition of Handwritten Mathematical Expressions Based on Stroke-Based Stochastic Context-

Free Grammar," in *Tenth International Workshop on Frontiers in Handwriting Recognition*, 2006.

[3] Sakshi and V. Kukreja, "A retrospective study on handwritten mathematical symbols and expressions : Classification and recognition," *Eng. Appl. Artif. Intell.*, vol. 103, p. 104292, 2021, doi: 10.1016/j.engappai.2021.104292.

[4] Sakshi, V. Kukreja, and S. Ahuja, "Recognition and classification of mathematical expressions using machine learning and deep learning methods," 2021, pp. 1–5, doi: 10.1109/icrito51393.2021.9596161.

[5] A. Nomura, K. Michishita, S. Uchida, and M. Suzuki, "Detection and segmentation of touching characters in mathematical expressions," *Proc. Int. Conf. Doc. Anal. Recognition, ICDAR*, vol. 2003-Janua, pp. 126–130, 2003, doi: 10.1109/ICDAR.2003.1227645.

[6] R. M. Gurav and P. K. Kadbe, "Real time finger tracking and contour detection for gesture recognition using OpenCV," *2015 Int. Conf. Ind. Instrum. Control. ICIC 2015*, no. Icic, pp. 974–977, 2015, doi: 10.1109/IIC.2015.7150886.

[7] D. Li, B. Liang, and W. Zhang, "Real-time moving vehicle detection, tracking, and counting system implemented with OpenCV," *ICIST 2014 - Proc. 2014 4th IEEE Int. Conf. Inf. Sci. Technol.*, pp. 631–634, 2014, doi: 10.1109/ICIST.2014.6920557.

[8] T. Kanahori and M. Suzuki, "Detection of matrices and segmentation of matrix elements in scanned images of scientific documents," *Proc. Int. Conf. Doc. Anal. Recognition, ICDAR*, vol. 2003-Janua, pp. 433–437, 2003, doi: 10.1109/ICDAR.2003.1227704.

[9] A. Qashlim, B. Basri, H. Haeruddin, A. Ardan, and I. Nurtanio, "Smartphone Technology Applications for Milkfish Image Segmentation Using OpenCV Library," *Int. Assoc. Online Eng.*, 2020, doi: 10.3991/ijim.v14i08.12423.

[10] Pearson, "https://www.kaggle.com/aidapearson/ocr-data.".