Text Generated Through Hand Gesture Recognition Using Machine Learning

September 5, 2019

Pratyay Kumar
ISM2016002

Dhvit Mehta
ICM2016003

Sudhanshu Sakya
ISM2016004

Abstract

This paper presents a real-time video-based pointing method that allows the sketching and writing of English text over the air in front of a camera. The proposed method has two main tasks. First, to track the fingertip in the video frame, and second, to apply character segmentation and recognition over segmented images to recognize the written characters. The proposed method provides a natural human–system interaction without requiring a keypad, stylus, pen, glove, or any other device for character inputs. For tracking fingerprints and constructing the image thereafter, an application was developed using OpenCV with the Python language. The proposed system is a software-based approach and is relevantly very simple, fast, and easy to use. It does not require sensors or any hardware other than a camera and can be made applicable to all disconnected languages. It has one issue though, it is sensitive to moving objects in the background before the start of and during the character writing which can lead to false results.

1. Introduction

1.1. Motivation

Object tracking is considered an important task within the field of computer vision. The invention of faster computers, the availability of inexpensive and good-quality video cameras, and the demand for automated video analysis have made object tracking techniques quite popular. Generally, the video analysis procedure has three major steps: detecting the object; tracking its movement from frame to frame, and analyzing the behavior of the object. For object tracking, three different issues are taken into account: selection of a suitable object representation, feature selection for tracking, and object detection. In the real world, object-tracking algorithms are the primary components of different applications, such as automatic surveillance, video indexing, and vehicle navigation systems.

Another application of object tracking in human-computer interaction. Different researchers have proposed many algorithms, which are categorically divided into two main approaches: the image-based approach and the glove-based approach. The image-based approach requires images as inputs to recognize the hand (object) movements. On the other hand, the glove-based approach requires specific hardware, including special sensors. Such applications are beneficial for disabled people.

Every technology becomes limited when it cannot be used by a section of society. An activity as basic as typing on the keyboard cannot be performed by several people. Amputees and disabled people find it difficult to write using a pen. It is very difficult for people who turn develop blindness over their lifetime to type on the keyboard. Students tend to write notes instead of typing because it establishes a stronger conceptual understanding. A major population of this country doesn't know how to type on a keyboard either.

Therefore this project aims to deliver a solution for the aforementioned problems. It explores and applies the fundamental aspects of machine learning to generate text using hand gesture recognition. The problem is solved by providing a real-time method that allows the user to write text in front of a web camera and generate the respective text on the system.

1.2. Aim

This project aims to deliver a real-time video-based pointing method that allows the sketching and writing of English text over the air in front of a camera. The proposed method has two main tasks. First, to track the fingertip in the video frame, and second, to apply character segmentation and recognition over segmented images to recognize the written characters. The proposed method is a software-based approach, a departure from almost all the existing finger-tracking-based text recognition systems, which require extra hardware. Furthermore, previous approaches carry out comparisons to recognize the input, whereas, in the proposed system, machine learning is applied for text recognition, considerably reducing the computational time.

2. Literature Review

Automatic object tracking has many applications, such as computer vision and human-machine interaction. Generally, the object can be a text or a person that needs to be tracked. In our case, we need to track the fingertip and then use it for recognition of the text written by it. In the literature, different applications of the tracking algorithm are proposed.

Hannuksela[JH07], Asano[TA10], and Honda and Vikram[SV13] presented finger tracking-based character recognition systems. In [JH07], the author presents a motion-based tracking algorithm that combines the Gaussian filtering and expectation-maximization techniques for estimating the motions of the finger. The estimation is based on the motion features, which are effectively computed from the scene for each image frame. Their main idea is to control a mobile device simply by moving a finger in front of a camera. In [TA10], the authors discuss a visual interface that recognizes the Japanese characters in the air. For tracking the hand gesture, they used an LED pen and a TV camera. They converted the movements of the pen into direction codes.

In [SV13], a new recognition system for gesture and character input in the air is presented. For detecting the finger positions, a 3D capturing device called the leap motion controller is used. In the method proposed in this paper, the dynamic time warping distance technique is used for searching a similar written character from the database. For character recognition, a database consisting of 29,000 recordings was created, in which simple pre-written characters are present. The data set has two parts: the candidate character and the data time series character data sets. The candidate character data set contains upper- and lowercase characters, and each character is replicated five times.

Another proposed methodology that captured our interest was presented by [FB13]. This was a real-time video-based pointing method consisting of the following steps. It tracks the motion of the colored fingertip, finds its coordinates, and plots such coordinates. After plotting the coordinates, OCR is applied to the plotted image, the output is matched with the trained database for OCR, and the most possible match is achieved and displayed.

One of the classic methods for optical character recognition was the use of a Hidden Markov Model. This approach uses letters as a state, which then allows for the context of the character to be accounted for when determining the next hidden variable [Yan16]. Several other projects used different methods for recognizing characters from texts. proposed the use of MLPs to recognize features from the character images and then classify them to their respective classes.

To detect fingertips from a video from a camera using OpenCV, a histogram-based approach is used to separate out the hand from the background frame. Thresholding and Filtering techniques are used for background cancellation to obtain optimum results. Finally, for fingertip detection, the point farthest from the centroid of the largest hull is considered [Pan17].

Another interesting approach specified in [Iza16] was to capture the background initially and then create a mask of it so as to use it during the real-time video capturing. Once, the mask is constructed, it is used for background removal by erosion and bitwise AND. Once the

background is removed, we are left with the moving hand in greyscale. Morphology and smoothing are applied to get a proper hand shape out of the frame. A threshold converts this into a binary image. One of the datasets that looked suitable for our project was proposed by [GCvS17]. The paper introduced a variant of the full NIST dataset, which was called Extended MNIST (EMNIST), which follows the same conversion paradigm used to create the MNIST dataset. The result is a set of datasets that constitute more challenging classification tasks involving letters and digits, and that shares the same image structure and parameters as the original MNIST task, allowing for direct compatibility with all existing classifiers and systems.

3. Proposed Methodology

3.1. Approach Followed

After a thorough reading of all the resources and building on the concepts discussed in the section above we concluded that we should start by recognizing fingertips in real-time by background detection followed by background subtraction and contour detection. We then locate the fingertip movement and construct an image using these coordinates and preprocess it to the format required by the neural network. We use a CNN for feature extraction and then fully connected layers to classify the features into respective classes.

3.2. Datasets Used

Our main resource for training and testing our handwriting recognizer was the EM NIST Dataset. This dataset contains data in five classes: Balanced, By class, By merge, Digits, and Letters. We have used the EMNIST By class dataset for our classification. The EMNIST By Class contains the full 814,255 characters contained in the original NIST Special Database. It contains the full 62 classes (26 uppercase characters + 26 lowercase characters + 10 digits) for classification.

The EMNIST dataset by default provides separate training and testing sets. The number of training samples in the EMNIST By class dataset is 697,932. The number of testing samples is 116,323.

This database given its breadth, depth, and quality tends to serve as the basis for many handwriting recognition tasks and for those reasons motivated our choice of the EMNIST Dataset as the source of our training and test data for our models.

3.3. Dataset Preprocessing

Usually, the images from the dataset do not have the size of 28x28, therefore we resize it to a width of 28 or a height of 28. Finally, we center the image to remove redundant white spaces from the image in order to enable the CNN to recognize features effectively.

3.4. Image Formation and Preprocessing

The real-time video-based pointing method proposed here consists of the following steps. First, the motion of the fingertip is captured and then its coordinates are plotted in an empty image. This image is processed and sent to the neural network where it will be classified into its respective classes. The various steps used are :

3.4.1. Background Removal

First, a background subtractor is created by capturing the background from the frame. Once the background is captured, it is removed from the live feed during fingertip capturing. This is done by performing a bitwise and of the background mask and the current frame.

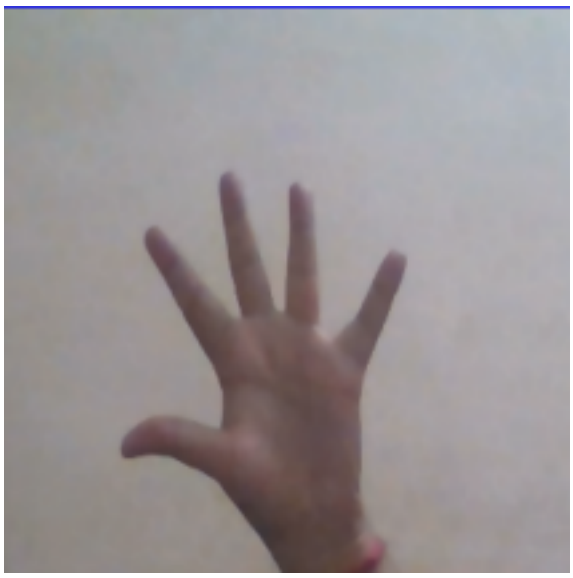

Figure 1: Original

Figure 2: After Background removal

3.4.2. Smoothing

It is a well-known fact that the video frames captured from any camera have noise in them at least to some extent, especially when the ambient light around the sensor is low [15–17]. If the

frames are extracted from a compressed video clip instead of from a camera, they usually contain undesired artifacts in addition to noise [18,19]. The smoothing process attenuates the sensor noise and reduces the effects of the artifacts, resulting in lesser false edges in the subsequent operation. Hence, the image is first converted to grayscale and then Gaussian blurring is applied to it.

Figure 3: After Smoothing

3.4.3. Thresholding

Thresholding is performed to remove any residual false edge and convert the image into a binary image. This image can then be used for further processing as it preserves the hand data almost accurately.

3.4.4. Contour Detection and fingertip identification

Finally, fingertips are detected from the resulting binary image using the convex hull of the largest contour. First, the defects are captured, and then using that the number of fingers is calculated. The highest y-coordinate specifies the fingertip location.

3.4.5. Construction of image and preprocessing

Finally, the coordinates of the fingertip are stored and plotted in an empty image. Thereafter, the image is segmented character-wise. Each character is resized to a size of 28x28. Any redundant white spaces are removed. Finally, all the segmented images are sent for detection to the model.

Figure 4: After Thresholding

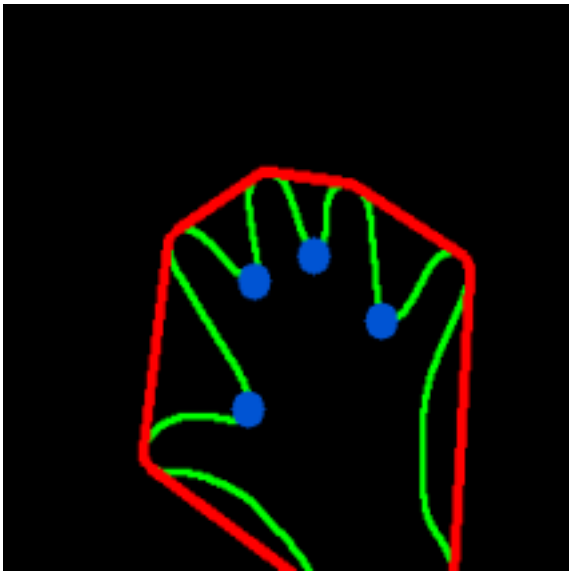

Figure 5: Convex Hull and Defects

Figure 6: Image to be segmented and sent to Model for classification

3.5. Network Model

We have an input layer with 32 convolutional filters and kernel size 3x3. The input is of volume (28,28,1). This layer will produce an output of volume (28,28,32). This layer is followed by another layer of 32 filters which constructs a feature map of shape (28,28,32). Then, we used a max pool layer with pool size (2,2) which downsamples the feature map to (14,14,32). Next, a dropout layer with a probability of 0.25 is added to reduce overfitting. Hence, our final feature map of size (14,14,32). ReLu activation is used in both layers.

This feature map is then used to classify each image into its respective characters using fully connected layers. We have used 2 dense layers with 512 neurons in the first one and with 62 neurons in the second one(62 is the total number of classes). The first layer uses Relu activation whereas the second layer uses softmax activation. The loss function used is categorical cross-entropy since the input needs to be classified into several classes and this appears to be the best suited for the above task. The optimization function used is Adagrad which uses adaptive learning rates for faster convolution.

This model is trained for 10 epochs with a batch size of 256 on the training data and saved for future use.

Hence, each image is classified in its respective classes after getting probabilities of each character from the model and selecting the maximum probability.

4. Workflow

The following implementation plan is followed.

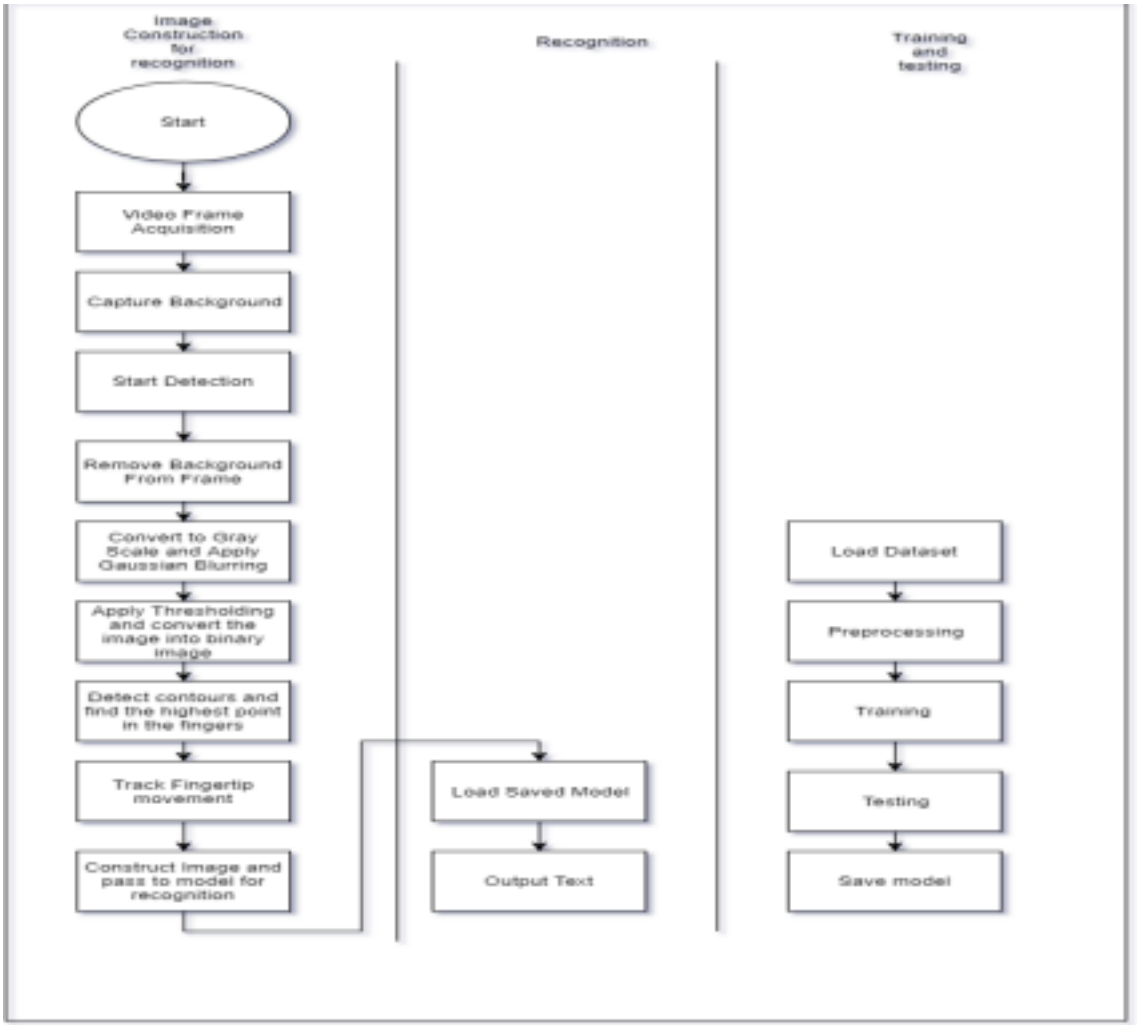

Figure 7: This is the flow of work inside the project

5. Results and End Product

The model was trained on the EMNIST dataset By class training dataset(697,932 images) and tested on the test dataset(116,323 images). The overall accuracy obtained was about 86 percent.

The end product of this project is software that uses the device's camera that detects fingertips and processes it to generate text on the device efficiently. The software can be used on any device that has a camera and satisfies the software's requirements.

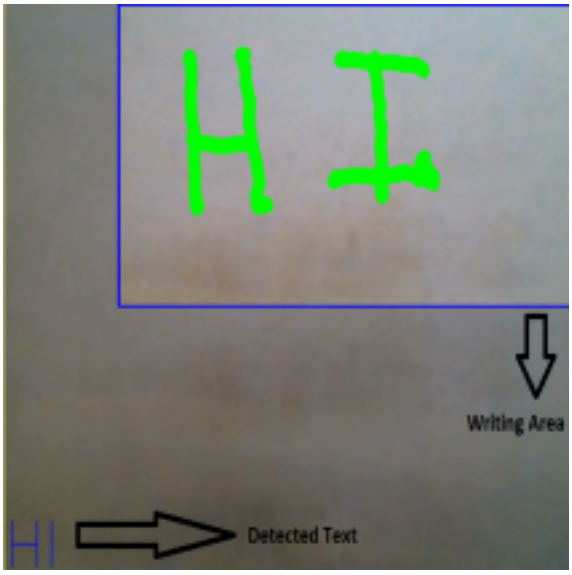

Figure 8: Sample Working.

6. Refrences

[FB13] Saira Begc Faisal Baiga, Muhammad Fahad Khanb. Text writing in the air. 1, 2013.

[GCvS17] Jonathan Tapson Gregory Cohen, Saeed Afshar, and Andre van Schaik. Emnist: an extension of mnist to handwritten letters. 1, 2017.

[Iza16] Izane. Fingers detection (or gesture recognition) using OpenCV and python. 1, 2016.

[JH07] P. Sangi J. Heikkila J. Hannuksela, S. Huttunen. Proceedings of the 4th European conference on visual media production. 1:1–6, 2007.

[Pan17] Amar Prakash Pandey. Finger detection and tracking using OpenCV and python. 1, 2017.

[SV13] S. Russell S. Vikram, L. Li. Chi 2013 extended abstracts. 1, 2013.

[TA10] S. Honda T. Asano. Ieee international symposium on robot and human interactive communication principe di piemonte. 19:56–61, 2010.

[Yan16] Lisa Yan. Recognizing handwritten characters. 1, 2016. 13