CS 5330: Handwriting Gesture Detection and Recognition

Malhar Mahant
MS Computer Science
Northeastern University
Boston, USA
mahant.ma@northeastern.edu

Kruthika Gangaraju
MS Robotics
Northeastern University
Boston, USA
gangaraju.k@northeastern.edu

Sriram Kodeeswaran
MS Robotics
Northeastern University
Boston, USA
kodeeswaran.s@northeastern.edu

***Abstract*—Working on projects involving Object detection and Deep Networks, seemed an interesting topic to explore more. One particular field which we looked into was object tracking and hand gesture recognition. Object detection involves detecting points on a finger, tracking its movement from frame to frame, and finally creating a pattern along the trajectory of the tracked points. Our project is an application that combined object tracking and hand gesture recognition to develop a real-time system that could track a user's finger movements and use them to draw on a virtual canvas. This project was a fascinating exploration of the capabilities of deep networks and their potential applications in computer vision. Ongoing research focuses on how to develop an accurate model and reduce the processing time. We believe that this approach has enormous potential and could lead to many exciting innovations in the future such as creating wearable devices which can easily convert gestures into meaningful text. It could also benefit hearing-impaired individuals to communicate easily with others.**

***Index Terms*—Object detection, Air Canvas, Deep Learning, Handwriting Recognition**

I. Motivation and Introduction

The conventional art of writing is being replaced by digital forms in today's world. Developing technologies to keep up with the changing form of communication becomes crucial. This gave birth to the idea of developing a code for creating an environment for writing on air. The idea was to develop a system that allows the user to move their finger in front of a camera and the system would track its motion and use it to draw on a virtual canvas. As part of our curriculum, we were introduced to object detection using OpenCV. Object detection works on the Viola-Jones algorithm which is based on machine learning. The first step involves training a cascade function with a large amount of negative and positive labeled images. After training the classifier, identifying features, known as the HAAR features, are extracted from the training from these training features. HAAR features are rectangular features with bright and dark pixels. Each of the feature's values is calculated as the sum of differences of pixels under bright regions and dark regions. The classifier is trained with pre-labeled data set to extract useful features to get minimum errors by applying appropriate weights to each feature.

We explored the integration of object detection, tracking, and hand gesture recognition in this project in order to create a real-time system that could follow a user's finger movements and use them to draw on a virtual canvas. Just writing something in the virtual environment does not prove to be of any use if we do not find a method to collect it as useful data which can further be used in required ways.

Text recognition is one of the most researched problems for the digitization of documents. There are various methods to tackle this problem but after an extensive literature review, the best approach for our project seemed to be TrOCR (Transformer based Optical Character Recognition). Text recognition is usually treated as an encoder-decoder problem where CNN based encoder is used for understanding the image and RNN based decoder is used for text generation. Recent advances have proven that using transformer architecture can vastly improve this process. However, most of the hybrid encoder-decoder methods still use the CNN backbone. TrOCR is an end-to-end Transformer based OCR model with pre-trained CV and NLP models. This project could revolutionize the methods for digital expression in the form of art and writing. It can be used for virtual sketching, interactive whiteboards, and gaming.

II. Literature Review

In order to proceed with our project, we had to gather existing research on the ways of implementing air canvas. After going through 12 papers, we could deduce certain details to proceed with our project. [1] discusses the challenges in using finger tracking, such as the major problem of the program not being able to detect a break required while writing something. It could often lead to the creation of chaotic figures. After discussing the challenges, the authors developed a method to tackle the issues by capturing 2-second videos of a person's hand in different environments and training it. They developed different hand gestures which could allow the user to easily navigate and edit the text being written on the screen. Single Shot Detector (SSD) and Faster RCNN pre-trained models were used to train their dataset.

In [2] and [4], Mediapipe is used for object detection. Mediapipe has a special library that can specifically detect hands that are used for the application of painting or writing in the air. These and some additional papers which covered similar topics of interest, we collected sufficient information for finger tracking and writing on air. The next step was to

explore different methods to convert the handwritten text to typed text. [6] talks about using multidimensional recurrent neural networks (MDRNNs) to replace the single recurrent connection found in standard recurrent networks with as many connections as there are spatiotemporal dimensions in the data. These connections allow the network to create a flexible internal representation of the surrounding context, which is robust to localized distortions. On the other hand, recurrent neural networks (RNNs) share many properties with DNNs and were applied to small handwriting recognition tasks with great success. However, the training time of RNNs is much larger compared to the mini-batch training of neural network architectures with pure forward connections due to the interdependence of the observations in the sequence [6] [8] [9]. We also looked into convolution neural networks (CNN) for our application. [10] talks about using the MNIST dataset for the purpose of handwritten text recognition.

III. Methodology

Our project can be understood as two separate problem statements: the first task is to create a canvas and detect and track the motion of a fingertip to write something on the canvas; the second is to extract useful data from the canvas and recognize the handwritten text using pre-trained datasets.

A. Preprocessing

Preprocessing is the initial step in our pipeline. To begin, we use a camera to capture the input gesture as an image. We first invert this image to get the drawing in white on black background. Then the acquired image is converted to grayscale because color information is not required for our objective. Following that, we add a Gaussian blur filter to the image to remove any noise created during the capturing process. We also use thresholding to transform the image into a binary image. Finally, we use morphological processes like erosion and dilation to smooth out the image and remove any minor artifacts.

B. Creating an air canvas

We have developed a tool that allows the user to write in the air, clear the canvas and save the image. It provides a user-friendly interface that can be easily interpreted by any user. We used MediaPipe Hands and OpenCV to develop this tool. MediaPipe Hands provides a Machine Learning model that can infer 21 3D landmarks on a hand. It has proven to be highly accurate in tracking hand movements. Fig 1. shows the 21 3D coordinates of hand knuckles as detected by the program. Using OpenCV, we provide real-time input using the webcam. Each frame is flipped which makes it easier for the user to write something on the canvas without worrying about the text being inverted. Using the MediaPipe libraries, we first retrieved the landmark of the detected fingertip. Tracking the trajectory of the detected landmark, we can connect these points to create a line as the finger traces a path on the screen. We also set a limit to the motion of the finger, i.e. if the finger hasn't moved a certain distance or for a certain number

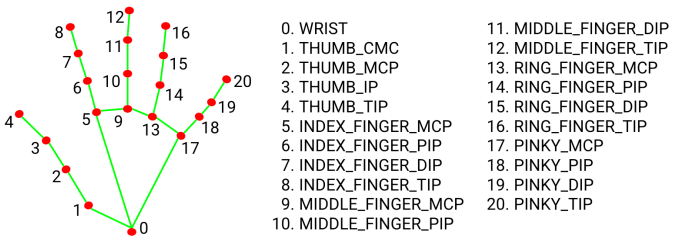

Fig. 1. 21 3D landmarks on hand

of frames, the program stops tracking the motion. The user can also clear the canvas, restart the drawing or even save the image as required.

C. Handwritten text recognition

1) Preprocessing: The preprocessing steps we employ on the image from canvas include inverting the image so that the drawing is white on a black background, creating a grayscale image, and calculating contours. In cases where there are multiple contours in the image, as in the case of the letter "i", we use contour detection algorithms to separate the text components. We then compute the oriented bounding box to determine the angle of the text and rotate the image to correct any tilt. Finally, we recalculate the bounding box on the rotated image and crop the image accordingly. These preprocessing steps are applied during both the dataset creation phase and the inference phase to ensure consistent and accurate results.

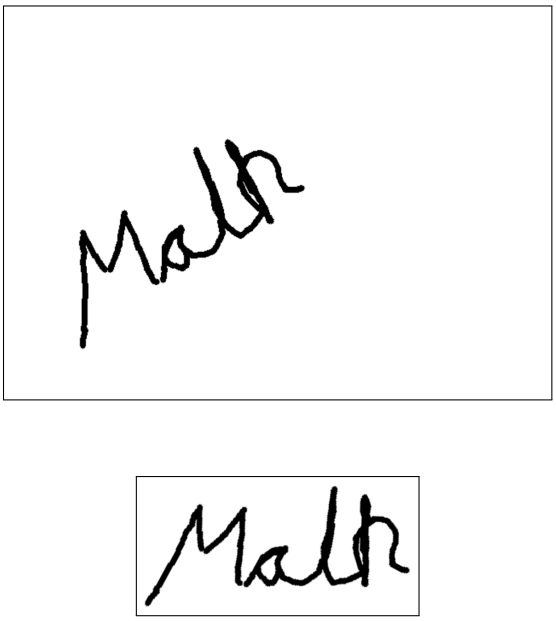

Fig. 2. Canvas Before and After Pre-processing

2) TrOCR Model: After the user writes something on the canvas, the written text can be recognized by character recognition techniques using pre-trained datasets. Transformer-based Optical Character Recognition [12] is one simple and effective method of text detection and text recognition that

does not use CNN as the backbone. TrOCR uses the pre-trained image Transformer and text Transformer models, which take advantage of large-scale unlabeled data for image understanding and language modeling, with no need for an external language model. It does not require any convolutional network for the backbone and does not introduce any image-specific inductive biases, which makes the model very easy to implement and maintain. Finally, experiment results on OCR benchmark datasets show that the TrOCR can achieve state-of-the-art results on printed, handwritten, and scene text image datasets without any complex pre/post-processing steps. Fig 3. describes the architecture of TrOCR.

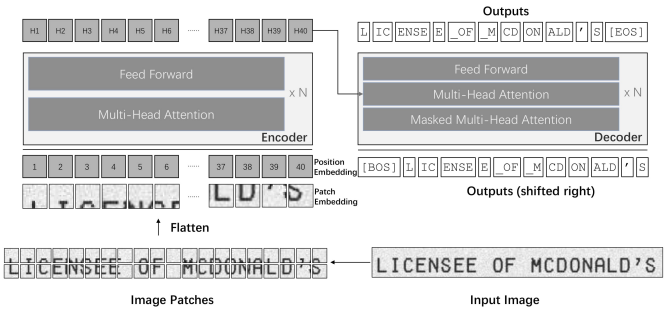

Fig. 3. TrOCR architecture

3) Evaluation metrics: The model is tested by calculating the Character Error Rate (CER). Character Error Rate (CER) is a metric used to evaluate the performance of Optical Character Recognition (OCR) systems, particularly for text recognition tasks. It measures the percentage of characters that were incorrectly recognized by the OCR system. CER is calculated on test datasets before and after training the model with the datasets described in the next section. After training with the datasets, we plot the training loss for each epoch. The CER is then calculated again to measure how much the performance has improved.

D. Flowchart of program

The first step is to open a webcam connected to the computer which is done to capture real-time video frames using OpenCV. The preprocessing steps calculate the angle of text, rotate the image to fix the tilt and then crop it using the bounding box. Once we have the frames, we use the Mediapipe library to detect the landmarks of the hand in the frames (here it detects the fingertip). These landmarks are specific points on the hand that can be used to detect various hand gestures and movements. After detecting the landmarks, we get the coordinates of the fingertips from the detected landmarks which is done to identify which finger is used to perform the gesture.

Based on the fingertip gesture, we use a trained transformer model to recognize the text written on the virtual canvas. These transformer models are a type of neural network architecture that shows great performance in natural language processing tasks. We then finally display the recognized text on the screen to the user. This provides an interactive way to recognize handwritten text in real-time using a webcam and a computer. Fig 4. best describes the steps taken to create our project.

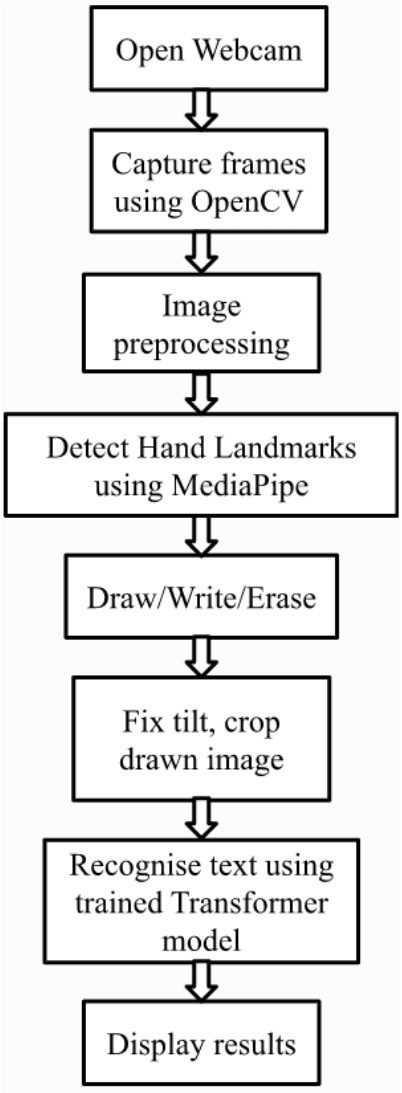

Fig. 4. Flowchart

IV. DATASETS

We make use of the TrOCR pre-trained model, the model is trained on from the IAM and SROIE datasets. However, the pre-training is done on document-level and scene-level datasets. For the purpose of our downstream task, we need the model to be trained on the word-level or character-level data. This need is demonstrated by running an evaluation of the data as described in the experiments section. We use the IAM word-level dataset to fine-tune the pre-trained model. The IAM Handwriting Database 3.0 is structured as follows:

- 657 writers contributed samples of their handwriting
- 1'539 pages of scanned text

- 5'685 isolated and labeled sentences
- 13'353 isolated and labeled text lines
- 115'320 isolated and labeled words

The split for the IAM word dataset is as follows:

- Number of training examples: 53841
- Number of test examples: 17616
- Number of validation examples: 7899

Since the IAM dataset consists of writing on paper, it has some differences from handwriting in a digital format. Handwriting in digital format has pixel edges, the lines and curves are not perfectly smooth, and other such flaws. Therefore there is a need to train the model to identify characters regardless of these discrepancies. We created a custom dataset of digital characters using our application and pre-processing described earlier. The custom dataset has 5-10 images of each alphabet in capital and small case and digits from 0-9. The split for the custom dataset is as follows:

- Number of training examples: 384
- Number of validation examples: 97
- Total number of examples: 481

Fig 5 shows a few examples from the IAM dataset.

Fig. 5. Sample of IAM dataset text

Fig 6. shows a few examples from the custom dataset.

V. Experiments and Results

A. TrOCR Model

We utilized the pre-trained TrOCR model, which has been trained on the IAM and SROIE datasets at the document and scene levels. However, as our focus was on word or character-level data because of the limit of our canvas size, we needed to tune the pre-trained model accordingly. Therefore, we performed additional training and fine-tuning to adapt the model for our specific downstream task.

We start off by evaluating the pre-trained model on the word-level IAM dataset. Our evaluation metric is Character Error Rate (CER). The CER achieved on the testing set of 17,616 unseen images of words was 0.47. To improve the CER, we train on the training split of the word-level IAM dataset and use a separate validation set. There are 53,841 examples in the training set and 7,899 in the validation set. Since the dataset is huge, we trained it only for 1 epoch as the data is fairly vast and also computationally expensive to

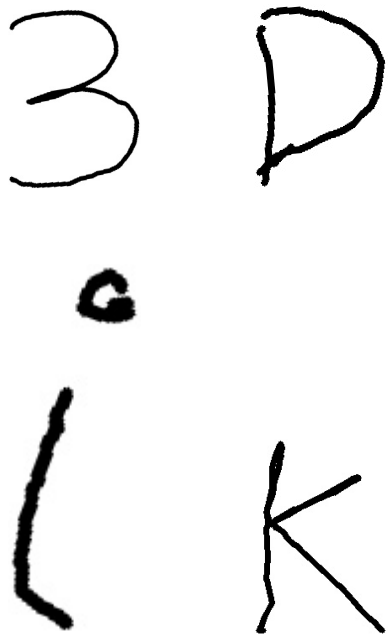

Fig. 6. Sample of custom dataset text

train. After training the CER achieved on the unseen test set was 0.16. Fig 7 shows the training loss for the IAM dataset.

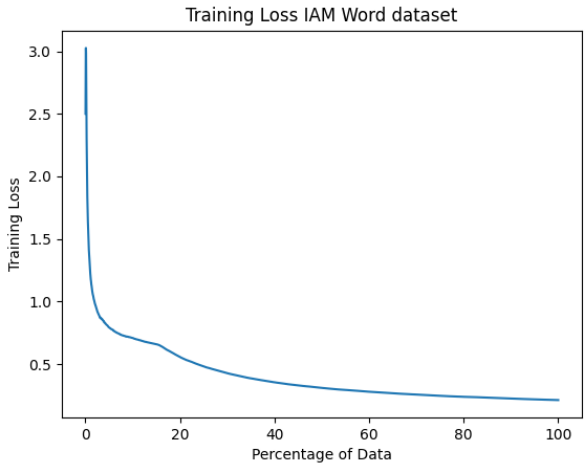

Fig. 7. Training loss for IAM word dataset

However, the model still had trouble recognizing handwritten text in digital form. Handwriting on paper and handwriting in digital form can look different to an ML model due to various factors. Handwriting in digital format has jagged edges depending on the resolution. The finger gesture tracking also results in squiggly or jagged lines. While writing on paper is more uniform with cleaner lines. This is demonstrated by evaluating the CER of the model on our custom data test set and on the whole dataset.

We use our limited dataset and train the model for 30 epochs till the loss plateaus. We recalculate CER on both datasets to

evaluate improvements. We see a significant improvement in the CER for both datasets. Fig 8 and Fig 9 show the training losses and validation CER across epochs respectively.

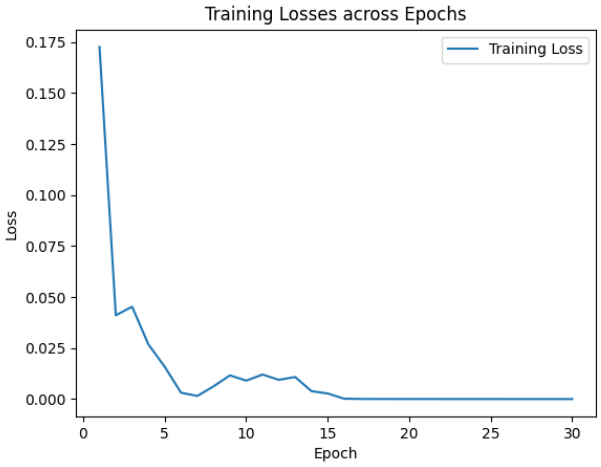

Fig. 8. Training loss across Epochs for Custom Dataset

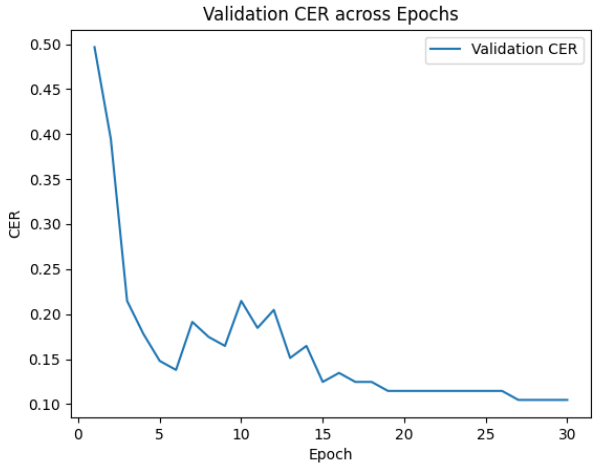

Fig. 9. Validation CER across Epochs for Custom Dataset

In conclusion, even though we used a pre-trained TrOCR model to start, we fine-tuned our model using a word-level dataset. However, there are some difficulties in handwriting recognition since finger-tracing drawn handwriting has jagged edges and lines than the one written on paper. Most of the pre-trained models are handwritten data on paper. We create our own dataset of digital handwritten samples to train and improve the model. The model can be further improved by creating a larger dataset to avoid potential overfitting and improve generalizations to different writing styles.

Table 1 shows the CER rates for each IAM and custom dataset test set and the whole dataset train and test set.

TABLE I
CER RATES BEFORE AND AFTER TRAINING

Training Dataset	**CER Rates**	
	Before	***After***
IAM Word Dataset test set	0.47	0.16
Custom Dataset test set	1.71	0.10
Whole Custom Dataset	0.40	0.029

B. Application

On running the program, the real-time feed is generated by the webcam, and another window is opened which acts as the canvas. As soon as the webcam feed is turned on, finger tracking starts. When the motion of the finger is detected, the user can write or draw anything in the air, which will appear both on the real-time feed as well as the canvas.

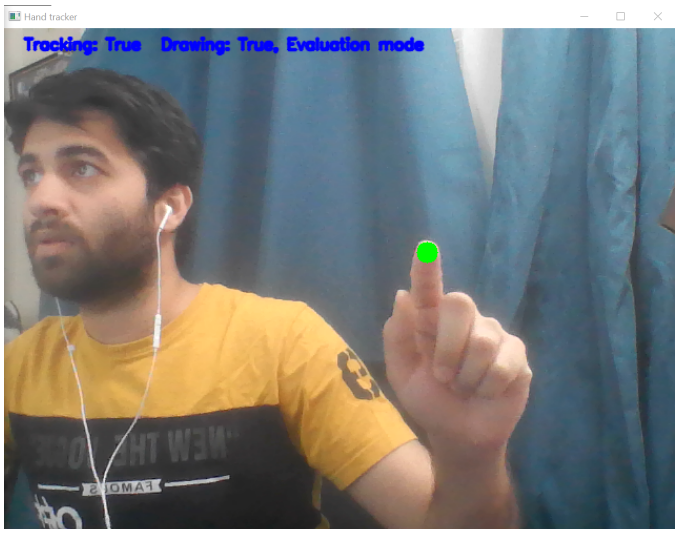

Fig. 10. Hand Tracking Window

We have provided flexibility to the user to clear the screen whenever required and restart drawing on the virtual canvas. Another useful feature is a pointer that shows up both on the canvas and live video for ease of use.

Once satisfied with the written text, the user can also save the image as training data with desired label and the label will be added as an entry to the CSV file generated while running the program. This is useful in expanding our own database in addition to using a pre-existing database, resulting in better text recognition.

The user can enable evaluation mode. The model that was trained previously is used to infer text from drawing on user command.

VI. CONCLUSION

The development of technologies to interface a computer-controlled environment with hand gestures is still in its early stages. There have been various approaches to deal with this problem. This project explores one of the ways in which finger tracking and hand gestures can be used to navigate

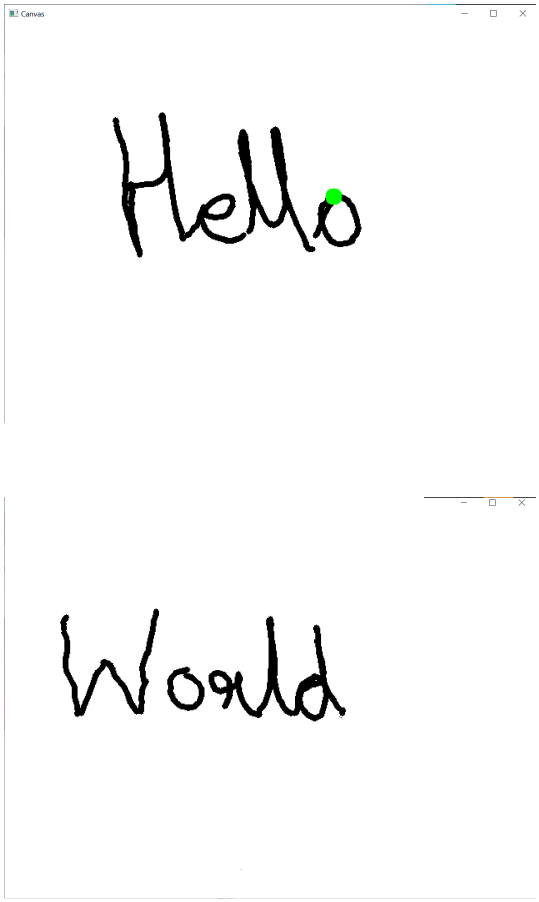

Fig. 11. Canvas Window

Fig. 12. Training Data Creation Window

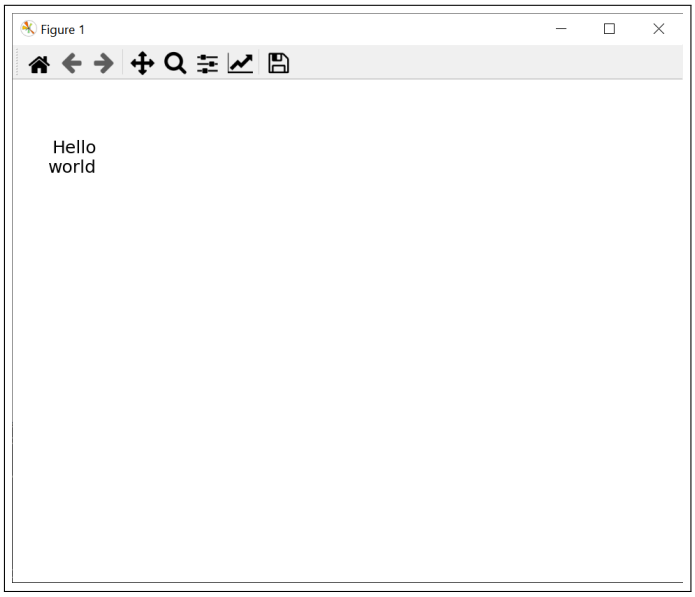

Fig. 13. Inference Window

something in the virtual environment; even if it starts with just writing and drawing in the air. This could potentially be further developed into a more accurate and complex environment. Now, writing in the air is an interesting concept but of no use if it cannot be extracted as useful data. There have been numerous pieces of research to recognize handwritten texts to digitize everything with ease. As a way to cover these two vast and widely researched topics, we came up with the project which is a straightforward conjugation of writing in the air and recognizing the handwritten text. The implications of such applications are in uses that predominantly use digital handwriting input such as note-taking on digital screens or digital signatures.

ACKNOWLEDGMENT

Firstly we would like to thank Prof. Bruce Maxwell for introducing us to the important concepts of computer vision and guiding us through each and every project. His insights have been very useful to us in coming up with our project idea and without his guidance we would not have been able to achieve the completion of our project. Next, we would like to thank the TAs for the class and to provide assistance at places we could not proceed further. They were available at all times to aid us in our projects. Lastly, we would like the authors of all the papers we referred to. They provided us with the necessary details to move forward with our project and presented their findings in a very simple yet effective manner which made our work easier. Without their work, we wouldn't have been able to make the necessary choices in choosing the path to take for the completion of this project.

REFERENCES

[1] Saoji, S. U., Dua, N., Choudhary, A. K., and Phogat, B. (2021). Air canvas application using Opencv and numpy in python. IRJET, 8(08).

[2] RamachandraH, V., Balaraju, G., Deepika, K., and Sebastian, S. R. (2022, November). Virtual Air Canvas Using OpenCV and Mediapipe. In 2022 International Conference on Futuristic Technologies (INCOFT) (pp. 1-4). IEEE.
[3] Salina, A., Kaivalya, K., Sriharsha, K., Praveen, K., and Nirosha, M. Creating Air Canvas Using Computer Vision.
[4] Gulati, S., Rastogi, A. K., Virmani, M., Jana, R., Pradhan, R., and Gupta, C. (2022, February). Paint/Writing Application through WebCam using MediaPipe and OpenCV. In 2022 2nd International Conference on Innovative Practices in Technology and Management (ICIPTM) (Vol. 2, pp. 287-291). IEEE.
[5] Pardeshi, S., Apar, M., Khot, C., and Deshmukh, A. Air Doodle: A Realtime Virtual Drawing Tool.
[6] Graves, A., and Schmidhuber, J. (2008). Offline handwriting recognition with multidimensional recurrent neural networks. Advances in neural information processing systems, 21.
[7] Oh, I. S., and Suen, C. Y. (2002). A class-modular feedforward neural network for handwriting recognition. pattern recognition, 35(1), 229-244.
[8] Doetsch, P., Kozielski, M., and Ney, H. (2014, September). Fast and robust training of recurrent neural networks for offline handwriting recognition. In 2014 14th international conference on frontiers in handwriting recognition (pp. 279-284). IEEE.
[9] Graves, A., Liwicki, M., Bunke, H., Schmidhuber, J., and Fernández, S. (2007). Unconstrained on-line handwriting recognition with recurrent neural networks. Advances in neural information processing systems, 20.
[10] Zamora-Martínez, F., Frinken, V., España-Boquera, S., Castro-Bleda, M. J., Fischer, A., and Bunke, H. (2014). Neural network language models for off-line handwriting recognition. Pattern Recognition, 47(4), 1642-1652.
[11] U. Marti and H. Bunke. The IAM-database: An English Sentence Database for Off-line Handwriting Recognition. Int. Journal on Document Analysis and Recognition, Volume 5, pages 39 - 46, 2002.
[12] Li, M., Lv, T., Chen, J., Cui, L., Lu, Y., Florencio, D., Zhang, C., Li, Z. and Wei, F. (2021). Trocr: Transformer-based optical character recognition with pre-trained models. arXiv preprint arXiv:2109.10282.
[13] Nitin Kumar, R., Vaishnavi, M., Gayatri, K. R., Prashanthi, V., and Supriya, M. (2022). Air Writing Recognition Using Mediapipe and Opencv. In Ubiquitous Intelligent Systems: Proceedings of Second ICUIS 2022 (pp. 447-454). Singapore: Springer Nature Singapore.