Online Handwriting Recognition using Depth Sensors

Rajat Aggarwal*, Sirnam Swetha*, Anoop M. Namboodiri, Jayanthi Sivaswamy and C. V. Jawahar
International Institute of Information Technology, Hyderabad
Email: {rajat.aggarwal@students, sirnam.swetha@research}.iiit.ac.in, {anoop, jsivaswamy and jawahar}@iiit.ac.in

***Abstract*—In this work, we propose an online handwriting solution, where the data is captured with the help of depth sensors. Users may write in the air and our method recognizes it in real time using the proposed feature representation. Our method uses an efficient fingertip tracking approach and reduces the necessity of pen-up/pen-down switching. We validate our method on two depth sensors, Kinect and Leap Motion Controller. On a dataset collected from 20 users, we achieve a recognition accuracy of 97.59% for character recognition. We also demonstrate how this system can be extended for lexicon recognition with reliable performance. We have also prepared a dataset containing 1,560 characters and 400 words with the intention of providing common benchmark for handwritten character recognition using depth sensors and related research.**

Fig. 1: User writes freely in front of a depth sensor (here Kinect). Our approach processes the trajectory and recognizes it in real time.

I. Introduction

As computing devices get seamlessly integrated into our daily lives, the need for natural and intuitive interfaces (HCI) to interact with them becomes critical. These devices rarely use a mouse or keyboard and employs natural interfaces like gesture and speech instead. Natural interfaces using depth sensors like kinect are getting popular for gesture recognition [1, 2, 3, 4], sign language recognition [5], signature based authentication systems [6], and computer games. Wachs *et al.* [7] also talks about vision based hand-gesture applications, which include medical systems and assistive technologies, crisis management and disaster relief, entertainment and human-robot interaction. The main challenge in such interfaces lies in terms of responsiveness, accuracy, user adaptability and intuitiveness. Using fingers to interact with a computer is probably the most natural way in these applications that does not force users to touch a specific device or to wear special sensors, while allowing unrestricted use.

Interfaces that use air based writing has evolved a lot in the recent years. Bunke *et al.* [8] and Fink *et al.* [9] used multiple cameras, whereas [4, 5, 10, 11] used depth sensors for gesture and handwriting recognition. Lee and Lee [12] proposed the use of a skin color model to track the hand and fingertips whereas Oikonomidis *et al.* [13] models the hand using a multi-camera setup. Liwicki and Everingham [5] have worked on recognizing words from video, where words are finger spelled using the British Sign Language(BSL). Encouraged by the success of these initial approaches, researchers are now tackling the problem of unrestricted writing in air without the use of any sign language. Tian *et al.* [6] proposed KinWrite, which is a handwriting based authentication system using Kinect.

Feng *et al.* [10] has proposed a real time fingertip tracking system, which is the most relevant work to our proposed solution. They used Gaussian Mixture Model (GMM) and K-means on each input frame for fingertip tracking. They used elastic mesh features for each sample and a Modified Quadratic Discriminant Function (MQDF) to recognize the character. Zhang *et al.* [11] proposed an approach that uses a combined model for depth, background and skin color for fingertip tracking. Murata and Shin [14] implemented an alphanumeric character recognition system based on Kinect writing. Their recognition is based on palm's graffiti. However, these systems either use external character recognition devices or lack real-time support. Most of these systems require pen-up/pen-down switching. Many researchers who worked on these kind of systems proposed supervised indication methods that explicitly commands the system to open, write or close the system. Murata and Shin [14] used hand gestures whereas Lee and Lee [12] have proposed fingertip actions to command Kinect to start or stop.

The main contribution of this work is a representation of the handwritten data that is robust in presence of the noise in the depth images. The proposed normalization approach makes the features independent of user's writing speed and scale. With this, we eliminate the lag between an event occurrence and the system response, while using minimal number of frames for processing. By capturing the system properties like speed and direction, the system is made more natural and intuitive. Using these properties we overcome the necessity of pen-up/pen-down gestures to determine the beginning and end of a meaningful input. We evaluate the effectiveness of our representation using SVM and DTW based classifiers. We also demonstrate that our method can be used as a basis for many real world applications by providing them with reliable recognition performance. Our method is tested on the dataset collected from 20 users, with varied experimental conditions and two different depth sensors.

We now describe the fingertip segmentation technique (Sec. II), followed by the character recognition solution. Section IV provides the details of the dataset with recognition results.

*Equal Contribution

(a) (b) (c) (d)

Fig. 2: Detection of fingertip from depth image: (a) Depth map captured by Kinect, (b) Segmented user from the depth image, (c) Segmented arm region and (d) Segmented finger-hand part.

II. Segmentation

Fingertip segmentation is the first step to form a trajectory from the user's hand motion. This involves a series of steps beginning with user segmentation from the depth image. Depth images are captured with the minimum assumption that there is no obstacle between the sensor and user body. For user segmentation, Feng *et al.* [10] used an user ID map provided by OpenNI to remove surrounding background. Their method is based upon the common observations on writing and hand-torso relationships. A histogram with a single peak indicates that the hand is close to body and one with two peaks models the hand and torso when user writes with hand holding up front. In [10], to segment the hand region, the depth histogram is characterized into each of the above models using single Gaussian model and two-component Gaussian mixture model respectively. Since hand region is always in front of the body while writing, hand pixels belong to the Gaussian component with smaller mean.

In the segmented hand region, K-means clustering algorithm is used to separate the finger-hand and hand-arm parts. It is a basic assumption that hand is in the front region of entire body during writing. Hence, the hand-arm cluster should have the largest depth value (that is far away from depth camera). In this way, cluster with the maximum depth sample is regarded as the hand-arm cluster and other one as the finger-hand cluster. The K value is 2 because the main interest is to separate the finger part from non-finger part. As shown in Fig 2(d), finger-hand part is the cluster centered at the least depth.

For user segmentation, we perform image slicing with known depth threshold. Unlike Feng *et al.* [10] we do not distinctly apply different Gaussian fitting for hand region segmentation on all the frames. This segmentation is only required for the first frame. Since length of arm is sufficiently high, the difference in the depth of the hand region and the body region is sufficient enough to be used as a threshold for the consecutive frames. For the consecutive frames, depth can be used to segment the hand region directly as shown in Fig 2(c). Inspired by hand-forearm segmentation in [10], we employ a clustering algorithm, K-means, to identify finger part. Feng *et al.* [10] applied K-means on all the collected frames. Applying K-means on all the frames to segment the finger-hand part is computationally expensive. We apply K-means on first frame to calculate the maximum depth in which finger-hand or the hand-arm part lies. For each consecutive frames, we use this depth as the threshold to segment the finger-hand part.

$$\chi_{fh} = \{x : \ x \in \chi_b \quad \text{and} \quad D(x) \leq x_{max}\} \tag{1}$$

where χ_{fh} is the set of points lying in the finger-hand cluster, χ_b is the set of all points in the body-region, $D(x)$ is the depth value of point x and x_{max} is the one of the points from hand-arm cluster with maximum depth.

After getting finger-hand part we are interested in fingertip. During writing, index finger always points out. Therefore, fingertip is the farthest point in finger-hand region from the central point in the hand-arm region. We then connect these points to form a trajectory which represents a meaningful input. We discuss the trajectory formation from these data points in section III.

(a) (b) (c)

Fig. 3: (a) Motion of the segmented hand while writing, (b) Red marker shows the bunch of points recognized as fingertip(c) Each straight line represents a vector.

III. Recognition

For air based writing system, we need a representation which can be adapted to use the state-of-the-art techniques to recognize the character with this system also. For most of the online handwriting character recognition systems, features are generated as time series data. With this motive, we use the spatial information of the time series data as the generalized feature representation for the character recognition techniques which are discussed in detail in following sections.

A. Character Formation

Fingertip point found from the above step is not a single point but a set of points as shown in Fig 3(b). This results in a sparse representation of the writing, if the points are connected to form a trajectory. This is due to the reason that bunch of points lie at same depth from x_{max} due to low resolution images provided by Kinect. We solve this problem by selecting only the mean point among the bunch of points. These set of points result in a discontinuous trajectory. Separation between the two points and the level of discontinuity strongly depends upon the speed of writing and the frame capturing rate of the kinect. We overcome this problem by using the vector algebra for the points. For each sample, we now get only spatial location to which fingertip is pointing which is represented as (x_i, y_i). We draw the trajectory by combining all these points as shown in Fig 3(c). Points are joined by drawing a vector between the adjacent points such that trajectory contains N vectors starting from frame $f = 0$ to $f = N$ with vectors $\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_N}$. This trajectory is represented as $T = \{\mathbf{v_i} : \mathbf{v_i} = (x_i - x_{i-1}, y_i - y_{i-1})\}$. The length of these vectors depend upon the writing speed and the frame rate.

Fig. 4: Character samples from the dataset using Kinect. The dataset contains lowercase English alphabets and words. Each character is written as a continuous trajectory without pen-up/pen-down gestures.

Angle between the vectors depends upon the writing style of the user. Using these vectors we calculate the speed of the user as

$$s_t = \sum_{m=1}^{M} ||\mathbf{v_m}|| \tag{2}$$

where $M+1$ is the number of frames per second and s_t is the length of trajectory traversed between time $t-1$ to t. We call the portion of the trajectory as meaningful input if the speed rises from zero and ends at zero. This eliminates the necessity of pen-up/pen-down gestures as discussed in Section I.

B. Data Sampling

After converting the points in the form of distinct N vectors, samples are equidistantly sampled into n_s vectors, where n_s is the sampling rate. This way we solve the problem of the variable length data from different users with different speeds. We use d_s as the sampling distance and find the set of k consecutive vectors such that

$$|\mathbf{v_1}| + |\mathbf{v_2}| + \ldots + |\mathbf{v_k}| \leq d_s < |\mathbf{v_1}| + |\mathbf{v_2}| + \ldots + |\mathbf{v_{k+1}}| \tag{3}$$

These k vectors are then added up to obtain $\mathbf{v_{si}}$

$$\mathbf{v_{si}} = \mathbf{v_1} + \mathbf{v_2} + \ldots + \mathbf{v_k} \tag{4}$$

Vectors $\mathbf{v_{s1}}, \mathbf{v_{s2}}, \ldots, \mathbf{v_{sn_s}}$ are n_s sampled vectors. These vectors represent character features.

C. Normalization and Spatial Features

It is a common observation that people write differently with different characteristics, some write very short and some very long characters. After sampling we have n_s vectors with variable lengths because $|\mathbf{v_{si}}|$ is not equal to the sampling distance d_s. To bring all these data to one common scale, we normalize the vectors such that each vector has equal length and reconstruct the trajectory with new spatial coordinates. To normalize the data, we first find the angles between all the adjacent vectors as $\theta_i = \arccos(\hat{\mathbf{v}}_\mathbf{i} \cdot \hat{\mathbf{v}_\mathbf{i-1}})$ and get a sequence of $N+1$ angles. We then start with the spatial location x_0, y_0 and reconstruct the character with vectors of length l_0. We get the sequence of $N+1$ spatial coordinates as $x_i, y_i = x_{i-1} + l_0\cos(\theta_i), y_{i-1} + l_0\sin(\theta_i)$

Each character is different, since we start every character with common point x_0, y_0. The center of gravity is inconsistent as it varies with the shape of the character. To place the data on the common bounding box, we translate the spatial coordinates by x', y' such that

$$X_c, Y_c = \frac{1}{N}\sum_{n=1}^{N}(x_n, y_n) \tag{5}$$

$$x', y' = X_c - X_0, Y_c - Y_0 \tag{6}$$

where X_0 and Y_0 are the fixed center of gravity for the system. We translate each spatial coordinate by x', y' such that their center of gravity becomes X_0, Y_0. We then construct the feature vector $\mathbf{F}$ of length $2(N+1)$ such that

$$\mathbf{F_i} = \begin{cases} x_{\frac{i}{2}} & \text{if } i \text{ is even} \\ y_{\frac{i-1}{2}} & \text{otherwise} \end{cases} \tag{7}$$

These features are used for character recognition using standard classifiers like Support Vector Machine (SVM) and Dynamic Time Warping (DTW).

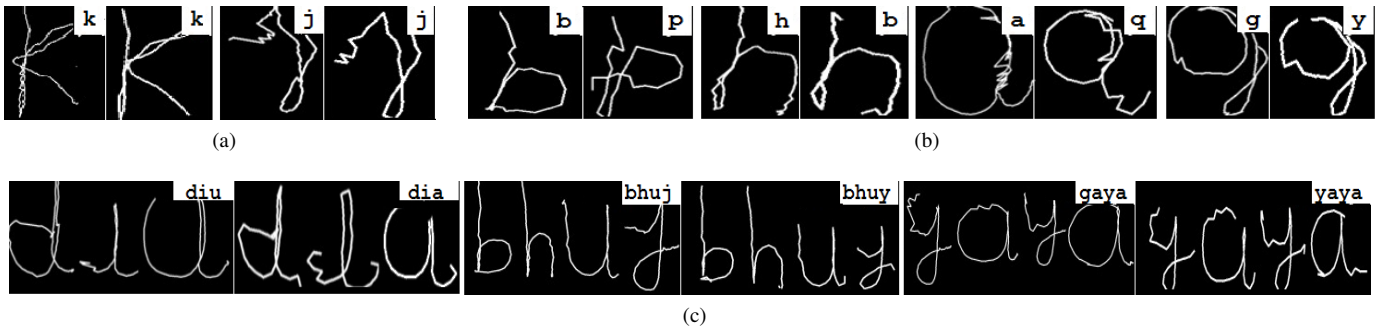
Trajectory formation can be easily extended from single input to multiple inputs. We extend the character trajectory formation to word trajectory. Each word is written with an assumption that the system recognizes each character separately. For this, we assume that the user writes each character discontinuously as shown in Fig 3(b). Ending of a character and beginning is detected when speed $0 < s_t < \epsilon$, where ϵ is some constant. Similarly, word end is detected when $s_t = 0$. We then extract features for each character separately as discussed earlier. We then use edit distance approach to find the closest match between the predicted word from all words in the pre-defined lexicon.

IV. Experimental Results and Discussion

A. Experimental Data

To evaluate the performance of our approach, we created a dataset, 'Dataset for AIR Handwriting'(DAIR)[1] which consists of lowercase English alphabets (as shown in Fig 4). Microsoft Kinect sensor is used to capture the color and depth image having 480×640 resolution at 30 fps. The dataset is created using 20 subjects, where each user stands straight in front of the sensor and writes in the air with one finger out. Users are allowed to write at their own speed and writing style. Dataset contains two sections DAIR I and DAIR II. DAIR I consists of 1248 character samples from 16 users by taking 3 samples per character per user. DAIR II consists of words from a lexicon of length 40. Words in the lexicon are taken from the names of most populous cities and vary in length from 3 to 5. It contains 400 words which totals to 1490 characters.

[1] http://cvit.iiit.ac.in/resources/OnlineHandwritingRecognition



(a) (b) (c)

Fig. 5: Labels represent the predicted and expected characters/words in each pair. Each pair has the input trajectory and trajectory after normalization respectively. (a) samples correctly classified, (b) mis-classified samples, (c) correctly classified words.

We also experimented our approach using Leap Motion Controller(LMC). 312 character samples have been collected from 4 other users on LMC. Data from left-handed user has been captured to analyze different users, which was only a part of testing dataset. The distance between the user and the Kinect is typically in the range of [1.5, 2.5] meters. We alternated two Kinect sensors for data collection and did not differentiate samples collected by different Kinect sensors to validate that our algorithm is insensitive to individual Kinect sensors. Summary of dataset division is shown in Table I.

	DAIR I	DAIR II(Words)	DAIR II(char)	Leap Motion
Samples	1248	400	1490	312

TABLE I: Summary of the experimental dataset(DAIR).

B. Recognition

For user segmentation, depth threshold is set to 1500 in each sample. X_0, Y_0 is taken as origin for normalization. Average time to write a character is 1-2 seconds. Sampling rate is set to $n_s = 32$ which is consistent with the frame rate at which depth image is captured. We take normalized length of vectors, $l_0 = 100$. l_0 scales the spatial coordinates and hence an increase in l_0 increases the inter-class variation. Thus, l_0 should be very high than the sampling distance d_s so that it ensures the separation between the close characters distinctively after normalization. It gives 66 dimensional feature vector $\mathbf{F}$ for each sample.

	Trained on all subjects	Trained on 10 subjects	Trained on Left-handed subject	Words
LIBSVM	97.59	96.36	96.15	97.76
DTW	97.60	95.94	96.15	99.61
MQDF	91.35	89.95	88.4	96.05
Feng[10]	73.72	61.75	73.07	73.82

TABLE II: Character and Word recognition results. We found that features proposed in this work give reliable performance with most of the state-of-the-art-classifiers. We also compare our method to the method proposed by Feng *et al.* [10].

In first experiment, DAIR I is randomly divided into training and testing set by 2:1 for each user, such that it contains 832 and 416 samples respectively. We trained a SVM classifier (LIBSVM implementation) with RBF kernel (γ = 1.0). A 5-fold cross validation has been performed on the data, which resulted in an accuracy of 97.59%. Recognition accuracy for this dataset with different approaches is shown in Column 2 of Table II. In second experiment, the training data had samples from 10 subjects and testing was done for rest of the users. Accuracy for this division is shown in Column 3 in Table II.

Some of the character recognition results are shown in Fig 5(a). Our method uses a constant length for normalizing the trajectory and reconstructs bad samples into good ones. It is observed that the user's writing style in air is different from the normal handwriting. It is difficult to trace the same line on character in air. Our approach tends to improve these type of characters by normalizing with constant length and reconstructing them into good samples. As shown in Fig 5(a), shape of sample 'k' improved after normalization. Similarly the sample labeled as 'j' looks close to some of the 'y' samples before normalization due to the writing style of the user in air. After normalization, the reconstructed character is correctly classified as 'j'.

The depth images provided by Kinect sensor are usually noisy and low quality. Due to this reason, characters which are written very fast are distorted due to errors in fingertip positions. Samples which are written in less than 2 seconds are shown in Fig 5(b). Due to these distortions, total length of trajectory increases. Hence, sampling rate goes wrong and reconstructed character takes the shape of the other nearest character. As shown in Fig 5(b), 'b', 'h', 'a' and 'y' are mis-classified to their closest match as 'p', 'b', 'q', 'g' respectively. Suitable superior depth sensors could alleviate this problem in the future.

To demonstrate the utility of our handwriting recognition solution in a specific domain, we performed word level recognition where each sample in DAIR II is tested against the training sample of DAIR I. Each character in the word is recognized independently using the proposed approach. column 5, Table II shows the character level accuracy of words using different classifiers. We then find the closest match for the predicted word from the lexicon. We have achieved 100% accuracy for word recognition as the vocabulary prepared is limited in size and specific to a domain. As shown in Fig 5(c), characters predicted for the written word 'diu' is 'dia' but

since we have an advantage of pre-defined lexicon, it correctly identifies its nearest match by edit distance approach as 'diu' with edit cost as 1. Similarly, the words 'bhuj' and 'gaya' are predicted as 'bhuy' and 'yaya' respectively using character recognition, but are classified correctly by finding their closest match.

Noise in depth images from kinect tend to effect the accuracy of the method. Therefore, we experimented our method on the dataset collected using LMC. It is accurate for small finger movements and its high resolution differentiates it from Kinect like devices which are more suitable for whole-body tracking. These kind of sensors tend to provide comfort to the user while writing. These samples are tested on DAIR I and reported an accuracy of 98.72%. Fig 6. shows the comparison between SVM classification results for each character for both the devices. It is clear from the figure that the proposed approach is independent of the device and gives high accuracy in both cases. As it is observed that the air handwriting style is different from the normal handwriting, we test our method with the samples taken from a left-handed user, where the training dataset DAIR I, does not contain any left hand user. Accuracies are reported in column 4, Table II.

C. Discussions

We compare our approach to the method proposed by Feng *et al.* [10] and Zhang *et al.* [11]. They used mean filter on the fingertip trajectory and then applied MQDF character classifier on 512 dimensional feature obtained after elastic meshing method. We implemented the methods in [10, 11] which reported an accuracy of 73.72% on our dataset. As shown in Table II our approach increases the character and word recognition accuracy by a promising amount.

Our approach resulted in an efficient implementation such that it uses less time for pre-processing. It takes 6.89 seconds which is significantly less than the approach in [10, 11] which takes 76.36 seconds to recognize. The response time is in coherence with the user writing speed which makes the system more interactive. Thus, it enables users to write at their own speed without any lag. It indirectly reduces the number of frames required to process for character formation. In this way, the character written at its natural speed will use minimal number of frames to be processed. It is clear from the recognition results that the character level solution can be extended into word level recognition systems with 100% accuracy.

V. Conclusion

Our work proposes a novel feature representation for the handwritten data captured using depth sensors. Our objective is to demonstrate a handwriting recognition solution on an emerging HCI modality. There are many open issues both in theory and usability that needs to be addressed in the future. We also introduce a new dataset for air handwriting character recognition research. It is developed with the intention of providing a common benchmark for air handwriting character recognition and allied research. The main characteristics of dataset are the variety subjects like right handed and left handed and diversity in terms of handwriting as we capture

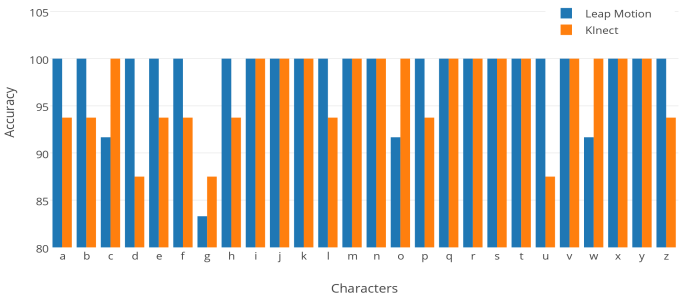


Fig. 6: Accuracy comparison of characters using Kinect and LMC.

multiple samples of each character from every subject. We also use leap motion controller to capture samples from a few subjects. Based on the dataset prepared, the proposed algorithm is tested and compared with existing approaches. By making this database available to the research community, we hope to encourage the exploration of many problems in the domain of HCI.

References

[1] F. Guo and S. Chen, "Gesture recognition techniques in handwriting recognition application," in *ICFHR*, 2010.

[2] M. Bouillon, P. Li, E. Anquetil, and G. Richard, "Using confusion reject to improve (user and) system (cross) learning of gesture commands," in *ICDAR*, 2013.

[3] P. Li, N. Renau-Ferrer, E. Anquetil, and E. Jamet, "Semi-customizable gestural commands approach and its evaluation," in *ICFHR*, 2012.

[4] X. J. Jin, Q. F. Wang, X. Hou, and C. L. Liu, "Visual gesture character string recognition by classification-based segmentation with stroke deletion," in *ACPR*, 2013.

[5] S. Liwicki and M. Everingham, "Automatic recognition of fingerspelled words in british sign language," in *CVPR*, 2009.

[6] J. Tian, C. Qu, W. Xu, and S. Wang, "Kinwrite: Handwriting-based authentication using kinect," in *NDSS*, 2013.

[7] J. P. Wachs, M. Kölsch, H. Stern, and Y. Edan, "Vision-based hand-gesture applications," in *Communications of the ACM*, 2011.

[8] H. Bunke, T. Von Siebenthal, T. Yamasaki, and M. Schenkel, "Online handwriting data acquisition using a video camera," in *ICDAR*, 1999.

[9] G. A. Fink, M. Wienecke, and G. Sagerer, "Video-based on-line handwriting recognition," in *ICDAR*, 2001.

[10] Z. Feng, S. Xu, X. Zhang, L. Jin, Z. Ye, and W. Yang, "Real-time fingertip tracking and detection using kinect depth sensor for a new writing-in-the air system," in *ICIMCS*, 2012.

[11] X. Zhang, Z. Ye, L. Jin, Z. Feng, and S. Xu, "A new writing experience: Finger writing in the air using a kinect sensor," in *IEEE MultiMedia*, 2013.

[12] D. Lee and S. Lee, "Vision-based finger action recognition by angle detection and contour analysis," in *ETRI*, 2011.

[13] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, "Efficient model-based 3d tracking of hand articulations using kinect." in *BMVC*, 2011.

[14] T. Murata and J. Shin, "Hand gesture and character recognition based on kinect sensor," in *International Journal of Distributed Sensor Networks*, 2014.