2015

Gesture Recognition with the Leap Motion Controller

Robert McCartney

Jie Yuan

Hans-Peter Bischof

Gesture Recognition with the Leap Motion Controller

R. McCartney[1], J. Yuan[1], and H.-P. Bischof[1,2]
[1]Department of Computer Science, Rochester Institute of Technology, Rochester, NY, USA
[2]Center for Computational Relativity and Gravitation, Rochester Institute for Technology, Rochester, NY, USA

Abstract—*The Leap Motion Controller is a small USB device that tracks hand and finger movements using infrared LEDs, allowing users to input gesture commands into an application in place of a mouse or keyboard. This creates the potential for developing a general gesture recognition system in 3D that can be easily set up by laypersons using a simple, commercially available device. To investigate the effectiveness of the Leap Motion controller for hand gesture recognition, we collected data from over 100 participants and then used this data to train a 3D recognition model based on convolutional neural networks, which can recognize 2D projections of the 3D space. This achieved an accuracy rate of 92.4% on held out data. We also describe preliminary work on incorporating time series gesture data using hidden Markov models, with the goal of detecting arbitrary start and stop points for gestures when continuously recording data.*

Keywords: Gesture recognition, CNN, HMM, deep learning

1. Introduction

There was a time when communication with programs like ‘vi’ [1] was done via keyboard only. The keyboard was used to input data and to change the execution behavior of the program. The keyboard was a sufficient input device for a one-dimensional system.

At the moment that operating systems moved to GUI’s the use of a mouse became handy to switch between graphical applications and exert control over them. The first notable applications making use of a mouse came when Microsoft introduced a mouse-compatible Word version in 1983 and when Apple released Macintosh 128 with an updated version of the Lisa mouse in 1984 [2].

Visualizations and games moved parts of computing into 3 dimensional spaces. The visualizations of these 3D worlds are either projected onto the 2D space of your screen or experienced with stereoscopic viewing devices. The control of these 3D worlds is not easy, some would say extremely unnatural, with a 2D mouse. The availability of 3D input devices allows for better control of this 3D world, but requires gesture recognition algorithms in order to use such devices in a natural way. This paper evaluates different gesture recognition algorithms on a novel dataset collected for such purposes.

2. Problem Description

The inputs coming from a mouse or a keyboard are discrete and have limited interpretation. A mouse down event is a single event at a given position on the screen, and dependent on the environment carries information with it about the position of the mouse pointer, time of the click, and so on. A double or triple click is an event over time, and will only count as such if the click events happen within a predefined window. Apple’s Magic Mouse [3] somewhat opened the door to 2D gestures, allowing users to swipe between pages or full screen applications and to double tap for access to mission control.

A keyboard or mouse sends an event only if a key or button is pressed or the mouse is moved. They do not start to send events as your hand approaches the device. In contrast, 3D input devices, like the Leap Motion controller[1], start to send frames as soon as they are turned on. These devices send a series of positions in space over time of whatever they detect in their views. The problem becomes to convert the output of these devices into something meaningful.

The output from motion sensing devices comes in two flavors: high-level and low-level. Low-level output is a series of frames where each frame contains information on what the device has sensed, such as the number of fingers, finger tip positions, palm position and direction, etc. The frame rate depends on the user settings and compute power, but 60 or more frames per second is typical. High-level output is the interpreted version of the raw frame data. This allows users or application developers to be informed when a particular predefined gesture is recognized.

We are interested in gesture recognition algorithms. Therefore, we are interested in the low-level information in order to interpret this into high-level information for others. The next section will describe the device we have used as our sensor, a relatively new and inexpensive motion sensing device. Then, we will discuss the gestures used and the dataset we captured for such purposes. Following that, we will discuss the particular form of dimensionality reduction and normalization we used on this data. The last sections will discuss the different gesture recognition algorithms we used as well as their results.

[1] https://www.leapmotion.com/

3. Leap Motion Device

There are many motion sensing devices available in the marketplace. The Leap Motion controller was chosen for this project because of its accuracy and low price. Unlike the Kinect, which is a full body sensing device, the Leap Motion controller specifically captures the movements of a human hand, albeit using similar IR camera technology.

The Leap Motion controller is a very small ($1.2 \times 3 \times 7.6cm$) USB device [4]. It tracks the position of objects in a space roughly the size of the top half of a 1m beach ball through the reflection of IR light from LEDs. The API allows access to the 'raw' data, which facilitates the implementation of gesture recognition algorithms. A summary of the specifications of the API: Language support for Java, Python, JavaScript, Objective C, C# and C++; data is captured from the device up to 215 frames per second; the precision of the sensor is up to 0.01mm in the perception range of 1 cubic feet, giving it the ability to identify 7×10^9 unique points in its viewing area.

The SDKv2 introduced a skeletal model for the human hand. It supports queries such as the five finger positions in 3 dimensional space, open hand rotation, hand grabbing, pinch strength, and so on. The SDK also gives access to the raw data it sees. Here we use this device to implement and analyze different gesture recognition algorithms from a dataset collected by this API.

4. Previous Work

One commonly used method of recognition involves analyzing the path traced by a gesture as a time series of discrete observations, and recognizing these time series in a hidden Markov model [5]. Typically, the discrete states are a set of unit vectors equally spaced in 2D or 3D, and the direction of movement of the recorded object between every two consecutive frames is matched to the closest of these state vectors, generating a sequence of discrete directions of movement for each gesture path [6], [7], [8]. Hidden Markov models have also been used to develop online recognition systems, which record information continuously and determine the start and stop point of a gesture as it collects data in real time [8], [9].

Another class of methods for recognition of dynamic gestures involves the use of finite state machines to represent gestures [10], [11]. Each gesture can be represented as a series of states that represent regions in space where the recorded object may be located. The features of these states, such their centroid and covariance, can be learned from training data using methods such as k-means clustering. When evaluating a new gesture, as the recorded object travels through the regions specified by these states, these sequences of states are fed into finite state machines representing each of the trained gestures. In this way, gestures whose models are consistent with the input state sequences are identified.

Neural networks have typically been used to recognize static gestures, but recurrent neural networks have also been used to model gestures over time [12], [13]. One of the main advantages of this type of model is that multiple inputs from different sources can be fed into a single network, such as positions for different fingers, as well as angles [13]. Additionally, convolutional neural networks and deep learning models have been used with great success to recognize offline handwriting characters [14], which can be considered analogous to hand gestures under certain representations as shown in this paper. A similar problem domain of gesture recognition, although in a lower dimensional space, is that of handwritten text recognition, where long-short term memory networks are the current state of the art [15], [16], [17].

5. Dataset

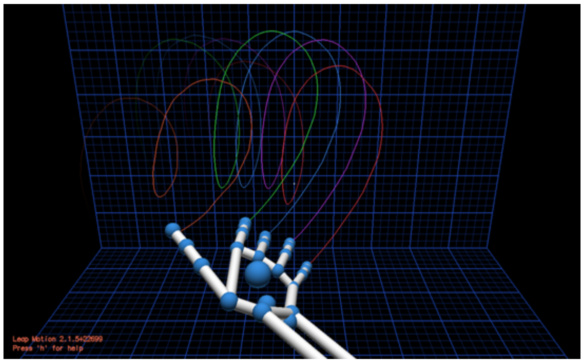

Fig. 1: Leap Motion Visualizer

In order to examine various machine learning algorithms on gestures generated through the Leap Motion controller, we needed to have a dataset that captured some prototypical gestures. To this end, a simple GUI was created that gave users instructions on how to perform each of a chosen set of 12 hand gestures and provided visual feedback to the participant when the system was in the recording stage. All gestures were performed by holding down the 's' key with the non-dominant hand to record and then using the primary hand to execute the gesture at a distance 6" to 12" above the top face of the controller. The code for this capture program is located online[2].

Students and staff on the RIT campus used the GUI to record their versions of each of 12 gesture types: one finger tap, two finger tap, swipe, wipe, grab, release, pinch, check mark, figure 8, lower case 'e', capital 'E', and capital 'F'. The one and two finger taps were vertical downward movements, performed as if tapping a single or set of keys on an imaginary keyboard. The swipe was a single left to right movement with the palm open and facing downwards, while the wipe was the same movement performed back and forth

[2] https://github.com/rcmccartney/DataCollector

several times. The grab motion went from a palm open to a closed fist position, while the release was performed in the opposite direction. Pinch was performed with the thumb and forefinger going from open and separated to touching. The check mark was performed by pointing just the index finger straight out parallel to the Z axis, then moving the hand in a check motion while traveling primarily in the X-Y plane. The figure 8, lower case 'e', capital 'E', and capital 'F' were all similarly performed by the index finger alone, in the visual pattern indicated by their name in the plane directly above the Leap Motion controller. The native Leap Motion Visualizer shown in Figure 1 was available for each subject to use alongside of our collection GUI while performing the gestures if so desired, providing detailed visual feedback of the user's hand during motion.

As each gesture was performed, the Leap Motion API was queried for detailed data that was then appended to the current gesture file. The data was captured at over 100 frames per second, and included information for the hand such as palm width, position, palm normal, pitch, roll, and yaw. Positions for the arm and wrist were also captured. For each finger 15 different features were collected, such as position, length, width, and direction. In all, we collected 116 features for each frame of the recording, with the typical gesture lasting around 100 to 200 frames, although this average varies greatly by gesture class. Files for each gesture are arranged in top-level folders by gesture type, inside which each participant in the study has an anonymous numbered folder that contains all of their gesture instances for that class. Typically, each user contributed 5 to 10 separate files per gesture class to the dataset, depending on the number of iterations each participant performed.

In all, approximately 9,600 gesture instances were collected from over 100 members of the RIT campus, with the full dataset totaling around 1.5 GB. The data is hosted online for public download[3]. Individual characteristics of each gesture vary widely, such as stroke lengths, angles, sizes, and positions within the controller's field of view. Some users had used the Leap Motion before or were comfortable performing gestures quickly after starting, while others struggled with the basic coordination required to execute the hand movements. Thus, there is considerable variation within a gesture class, and identifying a particular gesture performed given the features captured from the Leap Motion device is not a trivial pattern recognition task.

6. Image Creation

In its raw form, the varying temporal length of each gesture and large number of features make it difficult to apply traditional machine learning techniques to this dataset. Thus, a form of dimensionality reduction and normalization is needed for any learning technique to be effectively applied. For the convolutional neural network (CNN) that we employ in Section 7, this dimensionality reduction took the form of converting each instance of real-valued, variable length readings into a fixed-size image representation of the gesture.

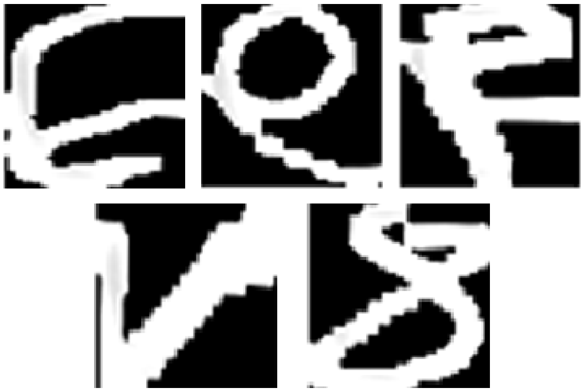

Fig. 2: One instance example of each of the gestures used for the CNN experiment

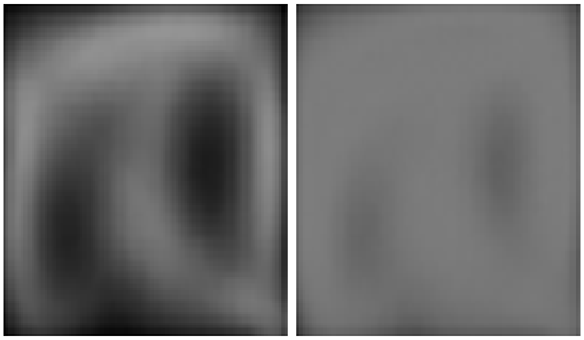

Fig. 3: The mean image of the dataset on the left and the standard deviation on the right used for normalization

CNNs traditionally operate on image data, using alternating feature maps and pooling layers to capture equivariant activations in different locations of the input image. Due to the complex variations that are nevertheless recognizable to a human observer as a properly performed gesture, CNNs offer a way to allow for differences in translation, scaling, and skew in the path taken by an individual's unique version of the gesture. To transform each gesture into constant-sized input for the convolutional network, we created motion images on a black canvas using just the 3 dimensional position data of the index finger over the lifetime of the gesture. That is, for each frame we took the positions reported in the Leap coordinate axes, which varies approximately from -200 to 200 in X and Z and 0 to 400 in Y, and transformed those coordinates into pixel space varying from 0 to 200 in three different planes, XY, YZ, and XZ. For each reported position, the pixels in the $5x5$ surrounding region centered on the position were activated in a binary fashion. From this

[3] http://spiegel.cs.rit.edu/~hpb/LeapMotion/

point, each of the three coordinate planes are separately or jointly able to be used as image input data in the learning model. However, for this first experiment on the dataset we kept only the XY plane of index finger positions and concentrated on those gestures that mainly varied in that plane, as explained below.

Despite being equivariant across feature maps, CNNs still have some difficulty in classification over widely varying positions and orientations of input activations. Thus, we cropped each image to fit the minimum and maximum indices of nonzero activations, and then sampled the resulting pixels to resize each image to a constant $50x50$ input size. After resizing, the pixel activations were normalized by subtracting the mean and standard deviation for that pixel across the entire training set, rather than using the statistics within a single image. Note that using only the XY positions of the index figure is a significant simplification of the data contained in an instance of the Leap dataset, but it served to show the applicability of computer vision techniques to the task of gesture recognition. As a result of this, we kept only those gestures that varied in the XY planes for the CNN experiments, namely the check mark, lower-case 'e', capital 'E', capital 'F', and figure 8. Since this subset of the gestures are guided by the index finger in the XY plane they appear rather well-formed there, but appear as mostly noise in the other two planes as their appearance in those projections largely depends upon unconscious movements of the hand. Expanding this representation to all three planes of movement for all gesture classes should be sufficiently expressive to broaden the learning algorithm to the entire dataset, and will be explored in future work. An example of each of the gesture classes in this representation after preprocessing is shown in Figure 2. Figure 3 shows the normalization factors used for the dataset, with the mean image on the left and the standard deviation on the right.

Note that there are other possibilities for generation of the images here that we did not do, such as removing skew and including motion history into gray-scale representations. There are still present many forms of variation in the input activations that are inherent to the users, such as the left-handed version of the check mark shown in Figure 4. While such differences as this and other examples of allowable variance in gestures from a given class are easily and unconsciously accounted for by humans, for instance by two people conversing in American Sign Language, they pose a significant challenge to the classification models that we discuss further in Section 7 and must be accounted for when training such classifiers.

7. Models

We have chosen our initial experiments on this dataset using two diverse models for classification of temporal sequences. The first is to convert the data into a fixed image representation as discussed above and use a CNN for classification. The second is to use a hidden Markov model to aid in a time series recognition task.

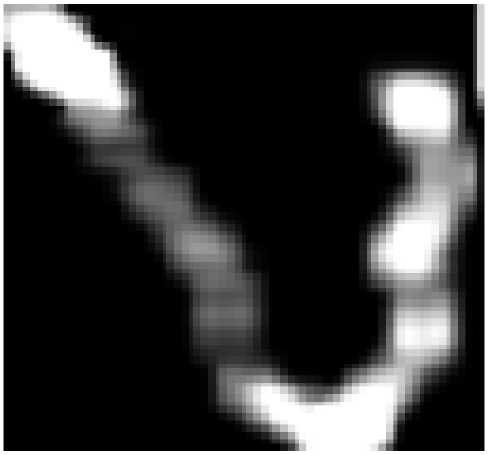

Fig. 4: A left-handed check mark after cropping, sampling, and normalization

7.1 Convolutional neural network

Convolutional neural networks are powerful models in computer vision due to their ability to recognize patterns in input images despite differences in translation, skew, and perspective [14], [18], [19], [20]. They can be effective at finding highly complex and nonlinear associations in a dataset. They do so in the context of supervised learning, by allowing the model to update parameters dynamically so as to minimize a cost function between a target value and the observed output of the model. An advantage they have over traditional, fully-connected neural networks is that the learned feature maps are applied with the same parameters to an entire image, drastically reducing the number of parameters required to learn without seriously degrading the capacity of the model [14]. This allows for more complex and deeper architectures to be employed without as serious a risk of overfitting the training data.

Human gestures are highly complex, nonlinear, and context-dependent forms of communication with both considerable overlap and great divergence between gesture types. People often perform the same gesture class in highly unique and differing ways, yet to the human brain these are easily recognized as constituting the same meaning. Further, very subtle and small differences exist between gestures that impart greatly differing meanings to the separate classes, yet such differences are not easily defined or separated. Given this type of data, convolutional neural networks have the advantage of learning good features as part of the classification task itself. Thus, we do not need to handcraft features of each valid gesture but allow the model to learn them as a product of minimizing the loss function. The model can thus learn to classify gestures based off of the

		Truth				
		E	✓	e	F	8
Prediction	E	28	0	1	0	0
	✓	1	62	2	0	1
	e	2	0	30	1	4
	F	1	2	1	40	0
	8	1	0	1	0	34

Table 1: Confusion matrix for CNN without dropout

		Truth				
		E	✓	e	F	8
Prediction	E	28	0	0	0	1
	✓	0	63	1	0	1
	e	2	0	30	0	3
	F	2	1	1	41	0
	8	1	0	3	0	34

Table 2: Confusion matrix for CNN with dropout

complex interactions between learned features that may not otherwise be easily discerned or discovered.

The convolutional neural networks used in these experiments came from MatConvNet, a toolbox for Matlab developed in the Oxford Visual Geometry Group [21]. All experiments were run on a GeForce GTX 960 GPU, with 1024 CUDA cores and 2 GB memory. In addition, NVIDIA's CUDA Deep Neural Network library (cuDNN)[4] was installed as the convolution primitives inside the MatConvNet library. The network consisted of alternating convolutions and max pooling layers, as depicted in Figure 5, followed by two layers of a fully-connected neural network with a softmax output. All neurons were rectified linear units, as they can be trained faster than their sigmoid counterparts [18]. The model was trained both with and without dropout, following the techniques described in [22], [23], [24]. See Tables 1 and 2 for the results of training this network with the 5 input gesture classes. The code for this network and for image creation is hosted online[5]. Overall the network produced a 92.5% recognition rate on held-out data after training to perfectly fit the input data, with a very modest improvement seen from using dropout with a rate of 50% on the two fully connected layers. This modesty may be due to the fact that dropout was not applied to the convolutional layers, which in the future could lead to greater improvements in generalization. A few of the misclassified gesture image representations can be seen in Figure 6.

7.2 Time series recognition with HMMs

Though the convolutional neural network performs well on images of the whole gesture, it does not take into account temporal information such as the order in which the strokes are performed. This can be addressed by modeling individual or groups of points as discrete states in a hidden Markov model. However, one of the principle challenges of definition and recognition of arbitrary gestures in 3D space is the high variability of gestures within the sequence space. For example, many traditional dynamic gesture recognition models have used translations between pairs of consecutive frames to generate a sequence of observations by fitting the translations to the closest of a set of evenly-distributed discrete vectors [6]. These methods work well in 2D, but suffer in 3D because 3D motions tend to be more varied and uncontrolled. Any portion of a single curved motion may be represented by slightly different vector sequences, and these sequences may result in highly distinct observations sequences even though they represented the same intended movement.

To solve this high-variability problem, we propose a method to process a sequence of frames of positional data and summarize them to a shorter and more generalized sequence of lines and curves, which are then fed into a hidden Markov model as discrete states. This method involves first identifying line segments in the sequence of frames by calculating average vectors of consecutive points âĂŞ from the sequence of average vectors those within a minimum angle distance are combined into one growing line segment. This line segment is then fit to one of 18 discrete observation states represented by vectors pointing away from the origin distributed equally in 3D space. Next those sequences of points that do not satisfy the criteria above but are of some minimum length of frames are likely curved segments. These sequences of points are fit to a sphere using a least squares approximation method [25]. The sphere then defines the centroid of the curve's rotation. To discretize the curve, the normal vector of the rotation is found by taking the cross product of the vectors emanating from the discovered centroid to the two end points of the curve. This normal vector is then fit to a set of six state vectors (clockwise and counterclockwise for each of roll, pitch, and yaw). The sequence of discovered lines and points then serves as the observation sequence, which is much shorter and more invariant to individual differences between training examples.

The performance of this model was relatively poor, at around 50% recognition for the specified gesture set. We believe this time series model is less robust to sources of error in the data, specifically the combination of very small and very large drawn gesture examples, as well as examples containing large disjointed spaces between consecutive segments of points due to sampling or user error. We hope to address these errors in the future by experimenting with

[4] https://developer.nvidia.com/cuDNN

[5] https://github.com/rcmccartney/LeapDeepLearning

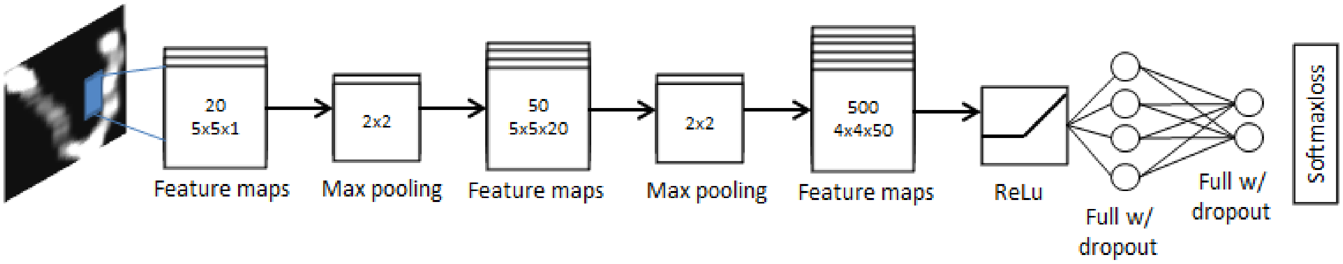

Fig. 5: A depiction of the CNN topology used

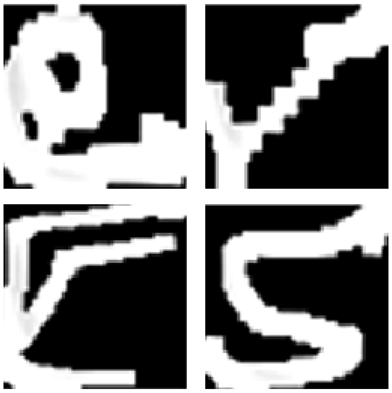

Fig. 6: Examples of misclassified gestures

rescaling and re-sampling training gesture paths.

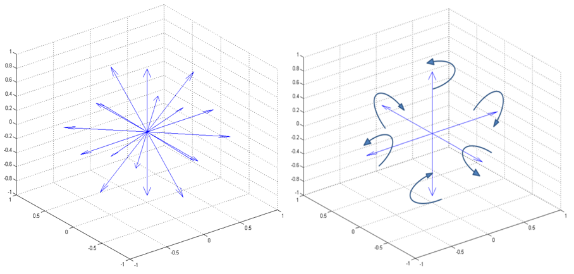

Fig. 7: The set of discretized states describing the motion of consecutive groups of 3D points, including 18 line directions and 6 curve directions

8. Future Work

This experiment represents the first to use the novel dataset collected from the Leap Motion controller. There is still much to be explored with this dataset as well as with applying other forms of learning algorithms to our representations of the gestures. Different forms of dimensionality reduction, such as PCA or gradient-based methods, could be used to help deal with the large amount of features available per gesture instance. Recurrent neural networks-long short term memory models in particular-could prove effective at dealing with the varying temporal nature of human gestures. Future work will also expand the scope to encompass the segmentation task as well as the classification task. One particularly interesting avenue of research is in combining the models discussed in Section 7 into a single online recognition engine. The HMM could specialize in segmenting gestures as they occur, using the two hidden states of “in-gesture" and “between-gesture" to distinguish between when a human hand is trying to semantically communicate or just resting. Once segmented, the frames of data from the “in-gesture" state could then be sent to the CNN model for classification. Note that the requirement to segment actual communication from idling is not an issue when using other input devices such as a mouse, and arises here due to the inability to set these 3D devices into non-recording states.

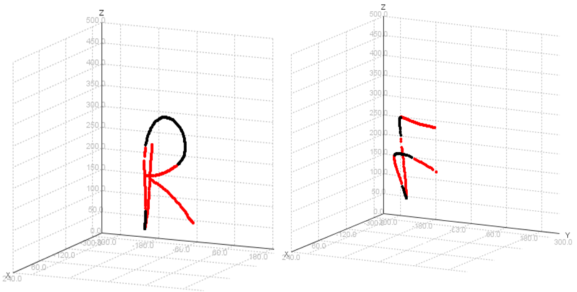

Fig. 8: Example gestures described as sequences of lines (black) and curves (red)

9. Conclusions

The Leap Motion controller is a promising device for enabling user-friendly gesture recognition services. Based on our results, the data generated by this device can be accurately classified by representing its 3D gesture paths as sets of 2D image projections, which can then be classified by convolutional neural networks. Here we limited the classification results to gestures performed in the XY plane, but the model can be extended to give equal consideration to all 3 planes of 2D projections, allowing for a wide variety of gesture representations. Despite its good performance, one of the limitations of this model is that it cannot provide online recognition of gestures in real time. As future work we look to incorporate an alternative model, such as a hidden Markov model, as a segmentation method to determine likely start and stop points for each gesture, and then input the identified frames of data into the CNN model for gesture classification.

References

[1] W. Joy and M. Horton. (1977) An introduction to display editing with vi. [Online]. Available: http://www.ele.uri.edu/faculty/vetter/Other-stuff/vi/vi-intro.pdf

[2] A. S.-K. Pang, "The making of the mouse," *American Heritage of Invention and Technology*, vol. 17, no. 3, pp. 48–54, 2002.

[3] R. Loyola, "Apple's magic mouse offers multitouch features," p. 65, 01 2010. [Online]. Available: http://search.proquest.com.ezproxy.rit.edu/docview/231461266?accountid=108

[4] F. Weichert, D. Bachmann, B. Rudak, and D. Fisseler, "Analysis of the accuracy and robustness of the leap motion controller," *Sensors*, vol. 13, no. 5, pp. 6380–6393, 2013. [Online]. Available: http://www.mdpi.com/1424-8220/13/5/6380

[5] L. Rabiner and B.-H. Juang, "An introduction to hidden markov models," *ASSP Magazine, IEEE*, vol. 3, no. 1, pp. 4–16, 1986.

[6] M. Elmezain, A. Al-Hamadi, J. Appenrodt, and B. Michaelis, "A hidden markov model-based continuous gesture recognition system for hand motion trajectory," in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, Dec 2008, pp. 1–4.

[7] T. Schlömer, B. Poppinga, N. Henze, and S. Boll, "Gesture recognition with a wii controller," in *Proceedings of the 2nd international conference on Tangible and embedded interaction*. ACM, 2008, pp. 11–14.

[8] H.-K. Lee and J.-H. Kim, "An hmm-based threshold model approach for gesture recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 21, no. 10, pp. 961–973, 1999.

[9] S. Eickeler, A. Kosmala, and G. Rigoll, "Hidden markov model based continuous online gesture recognition," in *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, vol. 2. IEEE, 1998, pp. 1206–1208.

[10] P. Hong, M. Turk, and T. S. Huang, "Gesture modeling and recognition using finite state machines," in *Automatic face and gesture recognition, 2000. proceedings. fourth ieee international conference on*. IEEE, 2000, pp. 410–415.

[11] R. Verma and A. Dev, "Vision based hand gesture recognition using finite state machines and fuzzy logic," in *Ultra Modern Telecommunications & Workshops, 2009. ICUMT'09. International Conference on*. IEEE, 2009, pp. 1–6.

[12] H. Hasan and S. Abdul-Kareem, "Static hand gesture recognition using neural networks," *Artificial Intelligence Review*, vol. 41, no. 2, pp. 147–181, 2014.

[13] K. Murakami and H. Taguchi, "Gesture recognition using recurrent neural networks," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 1991, pp. 237–242.

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[15] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 5, pp. 855–868, 2009.

[16] A. Graves and J. Schmidhuber, "Offline handwriting recognition with multidimensional recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2009, pp. 545–552.

[17] A. Graves, "Offline arabic handwriting recognition with multidimensional recurrent neural networks," in *Guide to OCR for Arabic Scripts*. Springer, 2012, pp. 297–313.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[20] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," *arXiv preprint arXiv:1405.3531*, 2014.

[21] A. Vedaldi and K. Lenc, "Matconvnet – convolutional neural networks for matlab," *CoRR*, vol. abs/1412.4564, 2014.

[22] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[23] G. E. Dahl, T. N. Sainath, and G. E. Hinton, "Improving deep neural networks for lvcsr using rectified linear units and dropout," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 8609–8613.

[24] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014. [Online]. Available: http://dl.acm.org/citation.cfm?id=2627435.2670313

[25] D. Eberly. (2015) Least squares fitting of data. Geometric Tools, LLC.