Towards Robust Tracking with an Unreliable Motion Sensor Using Machine Learning

J. Cecilia Wu
Media Arts and Technology
UC Santa Barbara. CA
93106, U.S.A.
cecilia@mat.ucsb.edu

Yijun Zhou
Mark Rau
Yun Zhang
CCRMA, Stanford University, CA
94305, U.S.A.
yjz@stanford.edu
mrau@ccrma.stanford.edu
yunzhang@stanford.edu

Matthew James Wright
CCRMA, Stanford University, CA
94305, U.S.A.
matt@ccrma.stanford.edu

ABSTRACT
This paper presents solutions to improve the reliability and to work around the challenges of using a Leap Motion™ sensor as a gestural control and input device in digital music instrument (DMI) design. We implement supervised learning algorithms, including k-nearest neighbors, support vector machine, binary decision tree, and artificial neural network, to estimate hand motion data, which is not typically captured by the sensor. Two problems are addressed: 1) the sensor is unable to detect overlapping hands 2) The sensor's limited detection range. Training examples included 7 kinds of overlapping hand gestures as well as hand trajectories where a hand goes out of the sensor's range. The overlapping gestures were treated as a classification problem and the best performing model was k-nearest neighbors with 62% accuracy. The out-of-range problem was treated first as a clustering problem to group the training examples into a small number of trajectory types, then as a classification problem to predict trajectory type based on the hand's motion before going out of range. The best performing model was k-nearest neighbors with an accuracy of 30%. The prediction models were implemented in an ongoing multimedia electroacoustic vocal performance and educational project named *Embodied Sonic Meditation* (ESM).

Author Keywords
Gesture mapping, supervised learning in DMI design, real-time vocal processing, Leap Motion™

ACM Classification
H.5.5 [Information Interfaces and Presentation] Sound and Music Computing. H.5.2 [Information Interfaces and Presentation] User Interfaces --- Input devices and strategies (e.g., mouse, touchscreen). I.5.2 [Pattern Recognition]: Design Methodology—Classifier design and evaluation

1. INTRODUCTION
1.1 The *Embodied Sonic Meditation*
Embodied Sonic Meditation (ESM) is an ongoing project in gestural control DMI design and sound education, inspired by Pauline Oliveros's "Deep Listening" practice [16] and the work of George Lakoff and his collaborators on embodied cognition [10]. ESM artistically explores the fact that we understand the world and abstract concepts such as music, art, and mathematics through our physical body and senses.

The ESM project applies seven ancient Buddhist hand gestures named *mudras* [9] that have the hands and fingers crossed or overlapped, as shown in Figure 1, to trigger corresponding sonic effects. Meanwhile, dynamic hand motions are also mapped to the audio/visual system to continuously control electroacoustic voice manipulations and a visualization of a 4-dimensional Buddhabrot fractal [7]. In spring 2017, ESM is being introduced to fifteen undergraduate students at the College of Creative Studies at University of California, Santa Barbara[1], as a teaching tool and a DMI design case study during a course named "Embodied Sonic Meditation – A Creative Sound Education."

NIME'17, May 15-19, 2017, Aalborg University Copenhagen, Denmark.

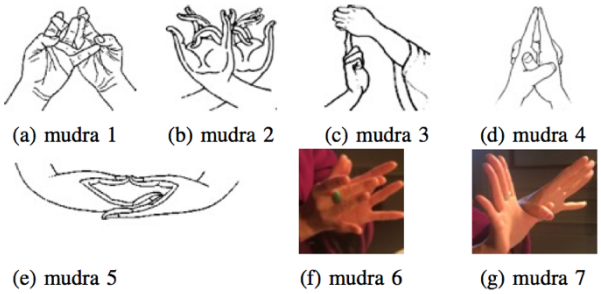

Figure 1. Example of input gestures and output label

1.2 Goals and Motivation
There are many musicians have had been using hand gestures to control live electronic music performance. The pioneer works include: *the Hands* [24] by Michel Waisvisz in 1984, *the Lady's Glove* [21] by Laetitia Sonami in the 1990s, the Sensorband [2] by Edwin van der Heide, Zbigniew Karkowski, and Atau Tanaka in 1994, as well as the Angry Sparrow [15] and Genoa by Chikashi Miyama in the late 2000's. In 2012, Thomas Mitchell et al [14] first used subtle finger gestures to real-time control and process live vocal performance. Instead of using sensors that are attached to a performer's hands, our project uses a non-attached optical sensor and touchless hand gestures to control a real-time system producing sounds and visuals, and processing voice.

1.2.1 The Sensor and Its Current Challenges
Because of its low price, light weight and portability, as well as a lower latency and higher frame rate compared to other sensors such as the Microsoft's Kinect™ [22], a Leap Motion™ infrared sensor is chosen as our non-attached tracking sensor to realize the gestural instrument for ESM project. The Leap Motion™ controller is a small USB device that uses two monochromatic IR cameras and three infrared LEDs to sense hand motion in 3D space. The sensor's field of view is an inverted pyramid with angles of 150° and a height of roughly 60cm, as shown in Figure 2. It provides information about the identity and motion of both hands and their corresponding fingers (position, velocity, normal vector of palm, etc.) in the form of a frame, which is easily accessible through a developer API[2]. The controller is well suited to music applications due to its performance in recognizing subtle hand and finger movements.

1 https://www.ccs.ucsb.edu/
2 https://developer.leapmotion.com/

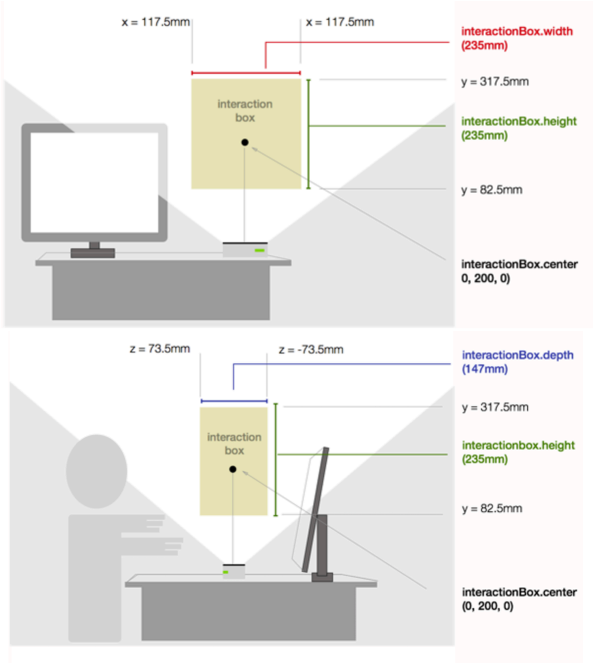

Figure 2. The coordinates of Leap Motion interactionBox

However, the sensor's small range greatly limits the expressiveness of a performer's natural hand motions during a live performance, impeding the exploration of a user's body-mind connectivity. The inverted pyramid shape of the sensor's tracking range makes it difficult for users to keep their hand movements in range. Moreover, the sensor cannot track any gestural input when two hands/fingers are close to each other, or overlap above the sensor. The sensor typically generates no output in these two situations, and is essentially "frozen."

These sensor limitations cause errors, which heavily interrupt the audio-visual system – either no data or discontinuous data will be sent to the audio-video engine, interrupting the smoothness and enjoyment of the user experience. In practice, the sensor is unreliable whenever the user's hands move outside the tracking range (such as with larger, more theatrical gestures) or overlap (for example, working with sign languages).

1.2.2 Research Goal

Since we require the ability to allow the above kinds of difficult-to-track gestures in our project, the ideal ESM system should be able to track hand movement and process continuous data without unpredictable output errors. Our research aims to predict the hands' trajectories when they are out of the sensing range, and to classify overlapping hand gestures in real-time, with minimum errors. Both instances are approached using machine learning and are framed as classification problems. Our optimization method should be able to predict a time series of what it guesses the user's hands might be doing while out of range, or overlapping. To the best of our knowledge, these challenges have not yet been explored within the NIME community for gestural control DMI design. We hope that our preliminary research shows the general potential of applying machine learning to create robust DMIs using accessible but unreliable sensors combined with machine learning.

2. BACKGROUND

2.1 The Leap Motion™ Controller at NIME

Due to the limited space, the large corpus of musical applications cannot be recapitulated completely here. We mainly pinpoint the closest related works that evaluate the sensor's tracking capability/challenges of gestural input and control in DMI design and live performance. Silva et. al [19] and Han et. al [8] evaluate the sensor's performance and implementations in keyboard instruments, and found that the sensor has limitations and tracking difficulties when fingers are close to each other. Occlusion plays a great part in the errors. In a NIME2015 concert, Yemin Oh performed *Space in Hands*, where he used the sensor to track hand position to control the array of surrounding speakers. In order to reduce sensor errors, he installed an LED light under the sensor to optimize the optical environment. This indeed can improve the tracking data when hands move slowly and finger/hand gestures are simple enough; but does not solve the overlap or the out of range problems. Fiebrink et al. developed *Wekinator* [5], a cross-platform, open source free software that applies machine learning to artists' and musicians' work in real-time; Leap Motion™ is one of the selected gestural interface that suits the system. One of the uses of Wekinator is to make accurate classifications from noisy sensors. However, Wekinator does not support offline data processing, visualisation, segmentation, feature comparison, etc. It is necessary to do the analysis for our project.

2.2 Related Work in Machine Learning

We treat the hands-out-of-range issue as a motion-tracking problem. Choi and Hebert use Markov models to predict a pedestrian's trajectory based on past movement [17]. Our case is subtler because of the intricate finger and hand movements involved. Vasquez and Fraichard used a cluster-based technique, including a learning algorithm and an estimation algorithm to solve the problem [4]. The cluster-based method is an effective way to predict the trajectory of a moving object. In addition, Pierre Payeur approached the prediction problem with neural networks, which are well suited for simultaneous anticipation of position, orientation, velocity, and acceleration of an object [11]. We have adapted these algorithms for the prediction of out-of-range hand motions.

We treat the hands-overlapping issue as a classification problem: predict the true hand gesture when the sensor cannot detect both hands. A robust approach using a novel combination of features is critical to the performance of the prediction. Marin shows that fingertip altitudes and angles are good indicators in the prediction [6]. Funasaka applies the K-Nearest Neighbor method to classify 26 gestures, and the recognition rate is 72.78% [3]. The use of SVM improves the recognition rate to 79.83%. We apply these methods for this project's prediction model.

3. SYSTEM OVERVIEW

The performer gives the system two inputs: vocals via a microphone, and hand gestures and motions via a Leap Motion™ sensor. The OSC protocol enables communications between Chuck [25] software for audio processing, and Python software for sensing and visual processing. Recognition of one of the 7 *Mudras* triggers the corresponding sound, while continuous hand motions control 2 sound filters and 4 effects for real-time vocal processing, as well as the visualization of a 4-dimensional Buddhabrot's trajectory deformation. Finally, the resulting sounds and graphics are amplified and projected into the performance space. Figure 3 shows the overall system architecture.

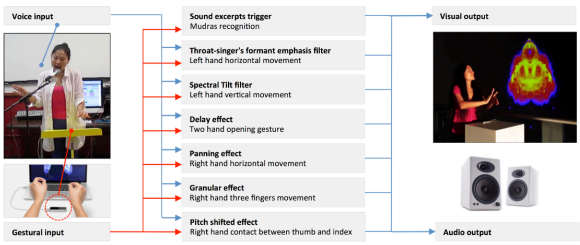

Figure 3. The general overview of the system

4. GESTURAL DESIGN AND MAPPING

Mapping the 7 *Mudras* gestures to trigger sonic outputs was originally impossible for the ESM project before we added the machine-learning component for gesture recognition. When the classifier recognizes a particular *Mudra*, it sends the detected type to Chuck, which plays the corresponding sound clip.

There are also six ways that hand motions are mapped to continuously manipulate the real-time vocal processing:

a) For the right hand:

1) Three-fingers' vertical movement -> granular effects;

2) Horizontal movement -> spatial panning;

3) Vertical movement -> roll-off of a Spectral Tilt filter [20]

b) For the left hand:

1) Horizontal movement -> frequency of a second-order "peaking equalizer" aka "throat-singer's formant emphasis filter." This filter efficiently simulates the vocal technique of Tibetan throat singing that emphasizes frequencies corresponding to particular harmonic overtones. When the left hand is at the most left from the sensor's center, this filter emphasizes the 2nd harmonic partial; when the left hand is closest to the sensor's center position, the filter emphasizes the 13th harmonic partial.

2) When the thumb and index finger touch (a "pinch" gesture), the vertical movement shifts the pitch of the simulated Tibetan throat singing – higher distance controls higher pitch, and vice versa.

3) The horizontal distance between two hands controls a delay effect's parameter. The delay time is proportional to distance; when two hands are closest to each other, all effects are switched off.

5. DATA SET AND FEATURES

5.1 Data Acquisition

Our project is not attempting to predict all human hand motion in the general case; we train our machine learning models to predict and recognize only a certain subset of gestures and hand positions relevant to this specific project. Therefore we acquired our training dataset by recording the first author's hand movements in context using the Leap Motion. Although hand movements are continuous, the Leap Motion sensor discretizes them with sampling rate 60 frames per second.

5.1.1 Hands Out of Range

As detailed in Section 1.2.1, the Leap Motion sensor has only a limited range that is often too constrained for the desired performance practice. When the hand leaves this range we would like to predict its motion until it comes back into range. So in order to record the hand's true motion while out of the sensor's range, we used a second Leap Motion placed 34 cm to the side, as shown in Figure 4. The master Leap Motion mimicked the sensor used for performances, while the slave Leap Motion accurately senses hand information while it is out of the sensing range of the master Leap Motion. We assume left/right symmetry of the possible hand movements in the performance, so for the sake of simplicity we recorded data for only right-hand out-of-range gestures.

To reconcile the data from the two Leap Motion devices, we synchronized the two corresponding computers with ntplib, a Python module that offers a simple interface to query NTP servers[3]. We were thus able to start the control programs of the two devices simultaneously and match the time stamp. We used the last 15 data frames of the master Leap Motion before it detected the right hand going out of its sensing range and all the data frames of the slave Leap Motion from the time that the right hand went outside the sensing range of the master Leap Motion to the time that it came back into the sensing range or went outside the sensing range of the slave Leap Motion™.

[3] https://pypi.python.org/pypi/ntplib/

Our dataset contains 292 examples extracted from recorded training data; we split them randomly into training set (80% of examples) and test set (20% of examples).

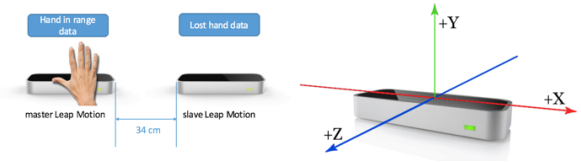

Figure 4. Leap Motion system setup (left) and right-handed coordinate system (right)

5.1.2 Overlapping Gestures

The Leap Motion sensor's outputs become inconsistent when two hands overlap above it. Since many of the mudras used in real performance involve overlapping hands, we would like to use the recorded data using a single Leap Motion before two hands overlap to predict and classify different mudras. If any of the following criteria are satisfied, then we regard the current hand positions as overlapping:

(a) The distance between the two palms is less than 10cm.
(b) The Leap Motion cannot detect left hand.
(c) The Leap Motion cannot detect right hand.
(d) The Leap Motion can detect neither hand.

We used the last 10 data frames of hand movements before the Leap Motion detects overlapping. Our dataset contains 435 examples (around 60 examples for each mudra) extracted from recorded training data; we split them randomly into training set (70% of examples) and test set (30% of examples).

5.2 Features

5.2.1 Hands Out of Range

Our training examples contain frames of raw data recorded by the two Leap Motion sensors. Within each time frame, we selected 19 features: hand position on x, y and z axes, hand velocity on x, y and z axes, hand palm normal on x, y and z axes, finger altitude for each of the five fingers, and finger pan for each of the five fingers. Since we would like to learn a pattern of hand movements, each training example is a time series consisting of 15 frames of these 19 features, represented as a vector of 285 real numbers.

5.2.2 Overlapping Gestures

We selected features for overlapping gesture classification by plotting examples in Matlab for the seven mudras and choosing the features which had relatively apparent differences over the seven categories. We selected these 8 most important features: hand palm normal on x, y and z-axes, and finger altitude for each of the five fingers. These plots showed that the last 10 frames of data were relatively consistent because both hands tried to maintain the gesture, so we preprocessed the data by taking a moving average over the last 10 frames. We also tried batch normalization, but in our experiments it did not improve test accuracy. For SVM model that will be discussed in Section 6.2, we reduced the number of features to only 3 (palm normal on x, y and z axes) to relieve over-fitting problem.

6. METHODS

6.1 Movement Prediction for Hands Out of Range

6.1.1 Trajectory Clustering

The out-of-range training data consist of multiple different trajectories. We used an approach taken by Vasequez, which assumes that there are a certain number of typical trajectories taken by the hand [4]. This reduced the problem to grouping these trajectories into categories using an unsupervised method, predicting

the category for each new trajectory, and applying the average trajectory of the predicted category to use for the audio-video synthesis.

The trajectories were clustered using a k-means algorithm, which is as follows:

1. Randomly initialize the k cluster centroids $\mu_1,..,\mu_k \in R^n$
2. Repeat until convergence {

For each trajectory i, set

$$c^{(i)} := \underset{j}{\operatorname{argmin}} ||x^{(i)} - \mu_j||^2$$

For each cluster j, set

$$\mu_j := \frac{\sum_{i=1}^{m} 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)} = j\}}$$

}

where $\{x^{(1)}, \ldots, x^{(m)}\}$ are the training data, μ_j represents each cluster centroid (in other words, the cluster's average trajectory, which will ultimately be the output of the movement prediction), and k is the number of clusters. The algorithm was implemented using Matlab's Statistics and Machine Learning Toolbox[4].

The number of clusters was determined using the elbow method, a standard practice which looks at the percentage of variance explained as a function of the number of clusters [23]. This plot usually looks like an elbow, and the bend represents a typically good number of clusters. The elbow method plot is shown in Figure 5; it suggests k=10 clusters. This also corresponds with the performer's introspective analysis that there are 5 to 10 main types of trajectories.

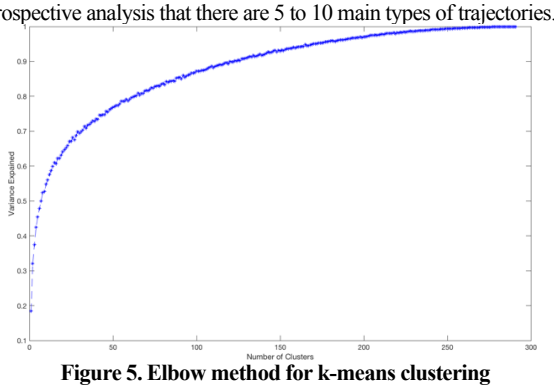

Figure 5. Elbow method for k-means clustering

6.1.2 *Trajectory Classification*

Once the training trajectories were clustered, we tried three different classification algorithms to assign new trajectories to the appropriate cluster: support vector machine (SVM), binary decision tree, and K-Nearest Neighbors (KNN). The algorithms were implemented for testing using Matlab's Statistics and Machine Learning Toolbox, and the best one was incorporated into the Leap Motion system using the Python library scikit[5].

Support vector machine (SVM) is a powerful and popular machine learning technique for multi-class classification. It first does an implicit mapping of data into a higher dimensional feature space. Secondly, it finds a linear separating hyperplane with the maximal margin to separate data in this higher dimensional space. For a training set

$$(x_i, y_i),\ i = 1,..., l$$

where

$$x_i \in R^n$$

and

$$y_i \in 1,-1$$

and, l is the number of training examples. Then a test example x is classified by:

$$f(x) = sgn(\sum_{i=1}^{l} \alpha_i y_i K(x_i, x) + b)$$

where α_i are Lagrange multipliers of a dual optimization problem that describe the separating hyperplane, $K(\cdot,\cdot)$ is a kernel function, and b is the threshold parameter of the hyperplane. The training samples x_i with $\alpha_i > 0$ are called support vectors, and SVM finds the hyperplane that maximizes the distance between the support vectors and the hyperplane. Given a non-linear mapping Φ that embeds the input data into the high-dimensional space, kernels have the form of $K(x_i,x_j) = (\Phi(x_i) \cdot \Phi(x_j))$. SVM allows domain-specific selection of the kernel function. Though new kernels are being proposed, the most frequently used kernel functions are the linear, polynomial, and Radial Basis Function (RBF) kernels. Since SVM only makes binary decisions, the classification is fulfilled by only using the one-only technique – to train classifiers to discriminate one expression from other expressions [26].

There are advantages of using SVM to solve this problem. First, it has a strong founding theory based on gradient descent and binary decision-making. In our classifying problem, the yes or no decision-making is all we need for each prediction. Second, global optimum is guaranteed in SVM algorithm. Third, we do not need to worry about choosing proper number of parameters for SVM model [3].

K-nearest neighbors (KNN) were chosen, as it is a fairly simple classification algorithm that can be used for quick predictions. KNN works by storing all available cases and will classify new cases based on a similarity measure known as the distance function. A case is compared to its k nearest neighbors, and is classified to be the classification that is most common among its neighbors. We used simple Euclidean distance

$$d = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2}$$

where d is the distance, x is the trained example, y is the test point, and m is the number of features, in our case, m=285. The relationship between the number of nearest neighbors, k, and the test/training error (Figure 6) was investigated to determine the ideal k value [1]. We chose k=20 for a good balance between low test error and model complexity.

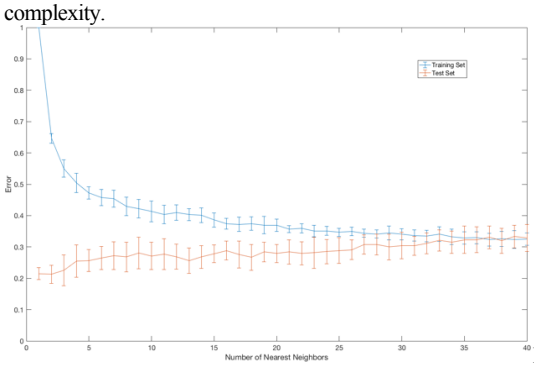

Figure 6. Train and test error vs. the number of nearest neighbors (k)

6.2 Overlapping Gesture Classifications

To classify overlapping gestures, we trained three classifiers including multi-class Support Vector Machine (SVM), k-nearest Neighbor (k-NN) in scikit-learn and neural network in Fast Artificial Neural Network (FANN) Library. We did experiments to find out the best classification model with the highest test accuracy.

For the SVM, we took "one-against-one" approach [18] for multi-class classification. If n is the number of classes to be classified into, then n*(n-1)/2 classifiers are built and each classifier discriminates between two classes. We used the Radial Basis Function (RBF) kernel: $K(x,x') = exp(-\gamma|x - x'|2)$, in which γ defines how much influence a single training example has and is selected automatically. For decision function, each classifier puts in a vote as to what the

[4] https://www.mathworks.com/products/statistics/
[5] http://scikit-learn.org/stable/

correct answer is and the output returns the class with the most votes.

We used k-NN for supervised nearest neighbor classification. The key hyper-parameter of k-NN is the predefined number *k* of training examples closest in distance to the new point and the predicted label is based on the majority votes of its *k* nearest neighbors. We did experiment on how the choice of *k* would affect training and test accuracy and found that *k=8* generated the highest test accuracy, as shown in Figure 7.

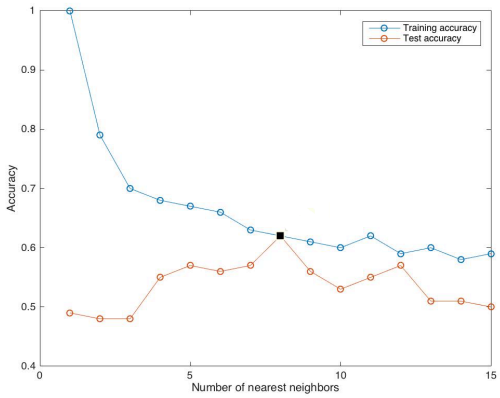

Figure 7. Train and test accuracy vs. the number of the nearest neighbors k

Furthermore, we developed a neural network classification model. Because the prediction of *mudras* needed to be in real-time, we used Fast Artificial Neural Network (FANN) Library[6]. FANN is a free open source neural network library, which implements multilayer feed forward neural networks in C using back-propagation training. FANN is very easy to use, versatile, and fast, making it a good choice for real-time classification. We implemented the classification model using FANN's Python binding. We implemented the neural network structure with 3 layers (input, hidden, and output) and 64 hidden neurons. We set the hyper-parameters to connection rate = 1, learning rate = 0.7, and we trained during 10000 epochs using sigmoid as the activation function and RPROP as the training algorithm.

7. RESULTS

7.1 Movement Prediction for Hands Out-Of-Range

The classification models described Section 6.1.2 were implemented in Matlab and tested using holdout cross validation. The training and test accuracy are shown in Table 1 as an average of 10 different holdout sets. Additionally, the average prediction time over 10 predictions (all in Matlab on the same computer) is shown for comparison. KNN was chosen as the best prediction method as it provided the best test accuracy of 30 ± 4%, where chance is 10%. KNN is also sufficiently fast and performed in this case in under 1/60ms, the refresh time of the Leap Sensor. The test accuracy is low for all methods, likely due in large part to the quality of the recorded training data. The training examples were difficult to sort through and many may have been inconsistent with typical trajectories. However, the nature of this task is such that, if a trajectory prediction is incorrect, the chosen trajectory is likely similar to the ideal case. This will mean that the music generation will not be that far off from what it should have been. Ultimately, any prediction is better than dropping motion data.

	Train Accuracy	Test Accuracy	Prediction Time (s)
SVM	0.49 ± 0.09	0.21 ± 0.08	0.038
BDT	0.80 ± 0.03	0.20 ± 0.03	0.0043
KNN	0.36 ± 0.02	0.30 ± 0.04	0.0069

Table 1. Test accuracy of SVM, BDT and KNN models

7.2 Overlapping Gesture Classification

We implemented all three models mentioned in Section 6.2 and tried to get the best accuracy on the test set, which is summarized in Table 2. Since the prediction time is also an important component to be considered in real-time performance, we timed each model and the result is summarized in Table 3. Our best k-NN model with k = 8 achieves the best result of 62% average test accuracy. The prediction time of k-NN model is 3.90ms, which is short enough for real-time performance. The k-NN predictor is sufficiently fast to achieve real-time performance and is the most accurate so it seems to be the best choice. The Neural Network likely performed poorly because our date set only includes 435 examples, lower than most successful typical Neural Network implementations.

We visualized the normalized confusion matrix of the best k-NN model to examine its performance, as shown in Figure 8. The color shading corresponds to the discrete of prediction, while the numbers within the matrix correspond to the normalized prediction. From the confusion matrix, we can see that some classes are harder to classify than others. For example, the algorithm misclassified a large portion of mudra 6 as mudra 1. This is pretty reasonable, as the relative position of each finger is relatively similar in mudra 6 and mudra 1. However, the diagonal of the confusion matrix is fairly high, suggesting an acceptable correct prediction ratio.

	SVM	NN	k-NN
Mean	0.52	0.43	**0.62**
Mudra 1	0.86	0.25	**0.49**
Mudra 2	0.55	0.50	**0.57**
Mudra 3	0.50	0.25	**0.51**
Mudra 4	0.73	0.60	**0.59**
Mudra 5	0.32	0.58	**0.69**
Mudra 6	0.31	0.50	**0.77**
Mudra 7	0.52	0.50	**0.70**

Table 2. Test accuracy of SVM, k-NN and NN models

	SVM	NN	k-NN
Predict time(ms)	5.48	0.03	3.90

Table 3. Prediction time of SVM, k-NN and NN models

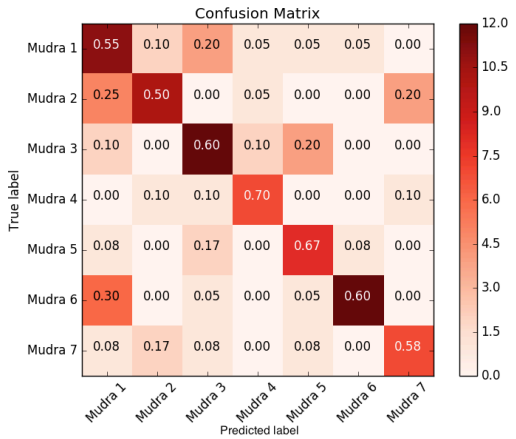

Figure 8. Confusion matrix of the best k-NN model

8. DISCUSSION AND FUTURE WORK

We attempted to solve two problems that arise when a Leap Motion™ sensor is used as part of a musical instrument. First, the sensor cannot detect overlapping hand or finger motions, which were important for musical applications. We implemented three different

[6] http://leenissen.dk/fann/wp/

algorithms to predict hand gestures while the hands were overlapped and ended up choosing to use KNN as it gave an average test accuracy of 62%. KNN is ideal for the predictions as it can provide fast predictions needed for the real-time audio-visual generation.

As future work, despite the fact that Leap Motion™ cannot detect hands when they are overlapped, it still can record the raw images by using infrared stereo cameras as tracking sensors. It is possible to classify the mudra that is using these camera images as input. As the input data are lower level and more complete, it can give a better performance on classification. However, this approach will make the problem even more complicated and run into another research topic and expertise – Computer Vision.

Additionally, the sensor has a limited range, so the lost trajectory data were predicted using a combination of k-means for clustering and KNN for classification. The algorithm only predicted the correct trajectory type 30% of the time, suggesting that there is room for improvement. Future work would look in to more carefully recording and processing the training data, as well as tweaking the classification algorithm parameters. This is a major problem that can generally affect every user of the sensor. For the specific ESM system, without our system optimization solutions, gestural input data from Leap Motion would be dropped when the sensor cannot detect the hand(s) or fingers, thus interrupting the coherence and smoothness of the real-time audio-visual output during the live performance or user experience; with our system optimization solutions, the trajectory is predicted. Instead of nothing, at least there are some data to be processed, improving the coherence and wholeness of the performance. In other words, even a made-up trajectory is better than having the data completely drop out. Additionally, the music generation algorithms could be chosen such that prediction mistakes are not obvious.

We believe we can improve the training results by increasing our training database, using extra slave sensors, and further observing typical movements by the performer. This would open up the full potential of a neural network approach.

Moreover, since our analysis provide methods to choose the good features from the mudra movement, it seems likely that an interactive machine learning approach like Wekinator's could be applied for the classification components of this problem. This could lead to a more accurate and useful system than the current offline machine learning approach. Specifically, an interactive approach would allow the performer to identify types of gestures the system is misclassifying, then immediately provide corrective training examples to help fix these errors.

At the time of writing, Leap Motion has just newly built their next-generation system Leap Motion Mobile[7], designed to be mounted on a VR headset, with a 180×180° field of view. This can already partially solve the out-of-range problem at the hardware level. It is reasonable to expect that we will achieve better results with the new equipment. We look forward to running tests and experiments on the Leap Motion Mobile in the near future and evaluating its performance specifically in music applications and DMI design.

9. ACKNOWLEDGMENTS

Thanks to UCSB/CREATE, Curtis Roads, and Clarence Barlow as well as Stanford/CCRMA and Chris Chafe for the facilities and funding support. Thanks to the UCSB/MAT writing club. Thanks to Julius Smith and Donghao Ren for their collaborations on realizing the ESM project. Leap Motion™ is a trademark of Leap Motion Inc.

10. REFERENCES

[1] Alex Smola and S.V.N. Vishwanathan. "Introduction to Machine Learning." Cambridge, UK. Cambridge University press. 2008.

[2] Bongers, Bert. "An interview with Sensorband." Computer Music Journal 22, no. 1 (1998): 13-24.

[3] Caruana, Rich, and Alexandru Niculescu-Mizil. "An empirical com- parison of supervised learning algorithms." Proceedings of the 23rd international conference on Machine learning. ACM, 2006.

[4] D. Vasquez and T. Fraichard, Motion prediction for moving objects: a statistical approach, Proc. IEEE International Conference on Robotics and Automation, vol. 4, no. April, pp. 39313936, 2004.

[5] Fiebrink, Rebecca, Dan Trueman, and Perry R. Cook. "A Meta-Instrument for Interactive, On-the-Fly Machine Learning." *NIME. 2009.*

[6] Giulio Marin. "Hand Gesture Recognition with Leap Motion andKinect Devices". Univeristy of Padova. 2014.

[7] GREEN, Melinda. "The Buddhabrot Technique." Superliminal Software(2012).

[8] Han, Jihyun, and N. E. Gold. "Lessons Learned in Exploring the Leap Motion™ Sensor for Gesture-based Instrument Design." Goldsmiths University of London, 2014.

[9] Hirschi, Gertrud. *Mudras: Yoga in your hands*. Weiser Books, 2016.

[10] Lakoff, George, and Mark Johnson. Philosophy in the flesh: The embodied mind and its challenge to western thought. Basic books, 1999.

[11] Liorand Rokach and Oded Maimon. "Data Mining and KnowledgeDiscovery Handbook." Boston, MA. Springer. 2010.

[12] Mahesh Pal. "Multiclass Approaches for Support Vector Machine Based Land Cover Classification." arXiv preprint arXiv:0802.2411. 2008.

[13] Makiko Funasaka et. al.. "Sign Language Recognition Using LeapMotion Controller". Int'l Conf. Par. and Dist. Proc. Tech. and Appli.. 2015.0.51

[14] Mitchell, Thomas J., Sebastian Madgwick, and Imogen Heap. "Musical interaction with hand posture and orientation: A toolbox of gestural control mechanisms." (2012).

[15] Miyama, Chikashi. "Angry Sparrow." In NIME, p. 326. 2009.

[16] Oliveros, Pauline. Deep listening: a composer's sound practice. IUniverse, 2005.

[17] Patrick. Choi. "Learning and Predicting Moving Object Trajectory: APiecewise Trajectory Segment Approach".Robotics Institute, CarnegieMellon University. 2006.

[18] Pierre Payeur. "Trajectory Prediction for Moving Objects Using Arti-ficial Neural Networks". IEEE Transactions on Industrial Electronics,VOl. 42, No. 2. 1995.

[19] Silva, Eduardo S., et al. "A preliminary evaluation of the leap motion sensor as controller of new digital musical instruments." Recife, Brasil (2013).

[20] Smith, Julius Orion, and Harrison Freeman Smith. "Closed Form Fractional Integration and Differentiation via Real Exponentially Spaced Pole-Zero Pairs." arXiv preprint arXiv:1606.06154 (2016).

[21] Sonami, Laetitia. "Lady's glove." Online demonstration as part of Paradiso (1997).

[22] Tormoen, Daniel, Florian Thalmann, and Guerino Mazzola. "The Composing Hand: Musical Creation with Leap Motion and the BigBang Rubette." *NIME*. 2014.

[23] Trupti Kodinariya and Prashant Makwana. "Review on determining number of Cluster in K-Means Clustering." International Journal of Advance Research in Computer Science and Management Studies.Volume 1, Issue 6. 2013.

[24] Waisvisz, Michel. The hands: A set of remote midi-controllers. Ann Arbor, MI: Michigan Publishing, University of Michigan Library, 1985.

[25] Wang, Ge. The chuck audio programming language. a strongly-timed and on-the-fly environ/mentality. Princeton University, 2008.

[26] Zhao, Xiaoming, and Shiqing Zhang. "Facial expression recognition based on local binary patterns and kernel discriminant isomap." Sensors 11.10 (2011): 9573-9588

[7] http://blog.leapmotion.com/mobile-platform/