Embodied Music Meditation: A Real-time Interactive Audio-Visual System for Buddhist Mudras Exploration

Mark Rau (mrau), Yun Zhang (yunzhang), Yijun Zhou (yjz)

I. Introduction

Our group collaborated with J. Cecilia Wu, a PhD Candidate of the Media Arts and Technology Department at the University of California, Santa Barbara, on her ongoing project, "Embodied Music Meditation"[1]. The goal of the project is to develop an interactive audio-visual system that captures the hand gestures of Buddhist Mudras: spiritual gestures performed with hands and fingers. Hand movements and gestures of a performer can be mapped in real time to control the interactive graphics and sonic manipulation, which help add to the expressiveness of the Mudra and vocal live performance.

The aim of our project is to improve the performance of certain areas of Cecilia's Audio-Visual system. There are currently two existing problems that we are going to approach using machine learning techniques. A Leap Motion infrared sensor (Section II-B) is used to track the hand and finger movements. The sensor has a limited sensing range and it is not able to correctly track the gestures when two hands overlap above the Leap Motion. These two problems occur frequently and will interrupt the audio-visual system because either no data or incorrect data will be sent to the audio-video engine. We used machine learning techniques to predict the motion of the hands when they are out of the sensing range, and to classify overlapping hand gestures in real-time. For predicting hand movements out of the sensing range, the input to our algorithm is frames of selected raw data recorded by two Leap Motions during training. We then implemented k-NN, SVM and binary decision tree to output predicted trajectory of the hand out of range. For classifying overlapping mudras, the input is averaged frames of raw data from a Leap Motion, and we used k-NN, SVM and neural network models to classify seven mudras.

II. Background

A. Related Work

As for the out-of-range problem, it is a motion tracking problem essentially. Patrick Packyan Choi and Martial Hebert use Markov model to predict moving object trajectory based on its past movement. However, it predicts much more complicated movements than our case, the movements of pedestrians[14]. Dizan Vasquez and Thierry Fraichard used the cluster-based technique, including learning algorithm and estimation algorithm to solve the problem [4]. Complete-Link algorithm performs the best in the specific scene. Cluster-based method is an effective way to predict the trajectory of a moving object. In addition, Pierre Payeur introduce neural network to the prediction model, which is well suited for simultaneous anticipation of position, orientation, velocity, and acceleration of the object [15]. We will fine-tune these algorithms and change some parameters to improve the prediction of out-of-range hand motions.

As for the overlapping problem, it is a typical classification problem. A robust approach using a novel combination of features is critical to the performance of the prediction. Giulio Marin shows that fingertips altitudes and fingertips angles are good indicators in the prediction [16]. In order to classify 26 gestures, Makiko Funasaka applies the k-nearest neighbor method, and the recognition rate is 72.78% [17]. The use of SVM improves the recognition rate to 79.83%. These methods give us a rough and ready plan of the prediction model.

B. Introduction to Leap Motion

The Leap Motion controller is a small USB device which uses two monochromatic IR cameras and three infrared LEDs to sense motion. The sensor field of view is an inverted pyramid with angles of 150° and a height of roughly 60cm. The controller is ideal as a musical tool because it is inexpensive and easily accessible. The sensor 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 frame, which is easily accessible through a developer API [2]. We define the sensing range of the Leap Motion using the interaction box mentioned as shown in Figure 1. As long as the hands stay in the the interaction box, they are guaranteed to be inside Leap Motion field of view.

Fig. 1: The coordinates of the Leap Motion interactionBox

III. Dataset and Features

A. Data Acquisition

In order to implement prediction models that fits Cecilia's performance better, we acquired the dataset by recording Cecilia's hand movements in real performance using the Leap Motion. Although hand movements are continuous, they are discretized by the Leap Motion sensor with a sampling rate 60 frames per second.

1) Movement prediction for hands out of range: As mentioned in the previous section, the Leap Motion sensor only has a limited range which is often too constrained for the desired performance practice. In order to predict the lost data, we would need to know which trajectory to use. To extend the range and obtain the information of the lost hand, we aligned two Leap Motions as seen in Figure 2. We used a master Leap Motion and a slave Leap Motion, side by side with a separation of 34 cm. The master Leap Motion mimicked the sensor used for performances, while the slave Leap Motion recorded hand information when it moved out of the sensing range of the master Leap Motion.

For the slave Leap Motion to know when the hand was lost from the master Leap Motion, we synchronized two devices with ntplib, which is a Python NTP library. This module offers a simple interface to query NTP servers from Python [3]. Using NTP query, we were able to start the control programs of the two devices simultaneously and matched the time stamp. Furthermore, because most of the possible hand movements in Cecilia's performance are symmetric, to reduce complication, we dealt with one hand (right hand). We used the last 15 data frames of the master Leap Motion before it detected 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. The later case was included as there was a desired sound effect when the hands left the sensor and did not return.

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

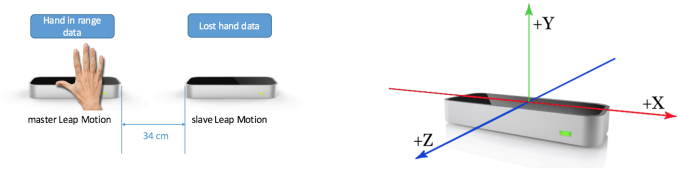


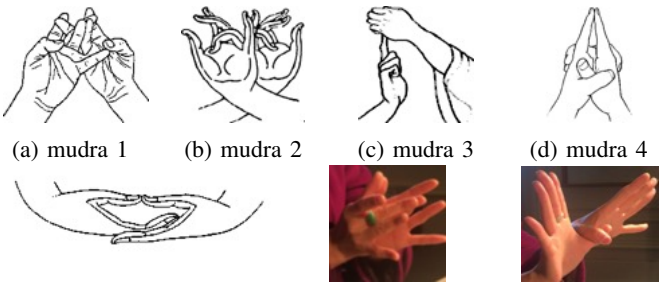
Fig. 2: Leap Motion system setup (left) and right-handed coordinate system (right)

2) Overlapping Gesture Classification: The Leap Motion sensor becomes 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 one of the following criteria is satisfied, then we regard the hand gesture as overlapping:

(a) The Euclidean Distance between two hands palm was 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. Because all seven mudras used in performance involve two hands and are symmetric, to reduce complication and reduce the number of features, we only used the data of right hand.

Our dataset contains 435 examples (around 60 examples for each mudra) extracted from recorded training data and we split them into training set (70% of examples) and test set (30% of examples.)



(a) mudra 1 (b) mudra 2 (c) mudra 3 (d) mudra 4

(e) mudra 5 (f) mudra 6 (g) mudra 7

Fig. 3: Example of input gestures and output labels

B. Features

1) Movement prediction for hands out of range: As mentioned in the previous part, our training examples contain frames of raw data recorded by both the master Leap Motion and the slave Leap Motion. Within each time frame, we selected 19 features including hand position on x, y and z axis, hand velocity on x, y and z axis, hand palm normal on x, y and z axis, 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, we used multiple frames of these 19 features in one training example.

2) Overlapping Gesture Classification: We selected features for overlapping gesture classification by plotting examples in Matlab for the seven mudras and choosing the features which had relatively large differences over the seven categories. We finally chose 8 most important features, including hand palm normal on x, y and z axis and finger altitude for each of the five fingers. From the Matlab plots, we noticed 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 average over the last 10 frames. We also tried batch normalization but our experiments showed that it didn't help to increase test accuracy. For SVM model that will be discussed in Section IV-B.1, we reduced the number of features to 3, including palm normal on x, y and z axis to relieve overfitting problem.

IV. Methods

A. Movement prediction for hands out of range

1) Trajectory Clustering: The out of range data consisted of multiple different trajectories. We used an approach taken by Vasequez which assumed that there are a certain number of typical trajectories taken by the hand [4]. This reduced the problem to grouping these trajectories using

an unsupervised method, predicting them, and taking an average of the trajectory data to use for the audio-video synthesis. The trajectories were clustered using a k-means algorithm, which is a follows:

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

$$\text{For every i, set } \; c^{(i)} := \underset{j}{\operatorname{argmin}} ||x^{(i)} - \mu_j||^2$$

$$\text{for each j, set } \mu_j := \frac{\sum_{i=1}^{m} 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)} = j\}}$$

}

where $\{x^{(1)}, ..., x^{(m)}\}$ is the given data, μ_j represents the cluster centroids, and k is the number of clusters. The algorithm was implemented using Matlab's Statistics and Machine Learning Toolbox [6].

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 [13]. 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 4. We also knew the general grouping of trajectories as Cecilia tended to train similar examples together. Using knowledge of watching the training, and using the elbow method, we choose to use 10 clusters which would be assigned as the classifications for the trajectory prediction.

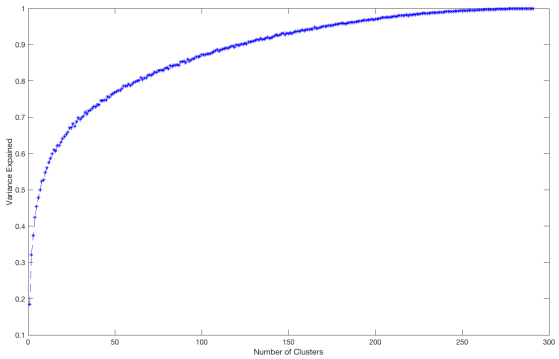


Fig. 4: Elbow method for k-means clustering

2) Trajectory Classification: Once the trajectories were clustered, we tried three different classification algorithms: 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].

i) 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 (sometimes probably infinite). 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

$$y_i \in 1, -1$$

. Then a test example x is classified by:

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

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 sample x_i with $\alpha_i > 0$ is 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, kernel 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 [9].

The advantages of using SVM to solve this problem includes the following. First of all, 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 image. Secondly, global optimum is guaranteed in SVM algorithm. Thirdly, we do not need to worry about choosing proper number of parameters for SVM model [8].

ii) Binary decision tree: A binary decision diagram is data structure use to represent the Boolean function as a tree. For Decision tree learning, a decision tree is used as a predictive model which maps observations about an item (represented as the branches) to conclusions about the item's target value (represented as the leaves) [12]. This type of prediction model can be used for multiclass prediction as well, and is ideal because of it's simplicity, resulting in a fast run time.

iii) KNN: K-nearest neighbors was chosen as it is a fairly simple classification algorithm which 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 which is most common among it's neighbors. The distance function chosen in out case was Euclidean metric as defined below:

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

where d is the distance, x is the trained example, y is the test point, and m is the number of features.

The relationship between the number of nearest neighbors, k, and the test/training error (Figure 5) was investigated to determine the ideal k value [11]. It was chosen to be 20 as it gave a good balance between low test error and model complexity.

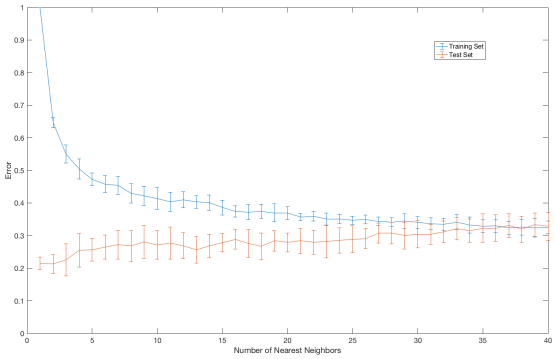


Fig. 5: Train and test error vs. number of nearest neighbors k

B. Overlapping Gesture Classification

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.

1) SVM: The reason of choosing SVM for multi-class classification has been mentioned in the previous part. Under this case, we took "one-against-one" approach [10] 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 train data from two classes. For kernel function, we used the Radial Basis Function (RBF) kernel: $K(x,x') = exp(-\gamma|x-x'|^2)$, in which gamma 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 correct answer is and the output returns the class with the most votes [5].

2) k-NN: The reason of choosing k-NN has also been mentioned previously. In our case, we used KNN 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 [5]. 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 the following figure.

3) Neural Network: We also tried neural network classification model. Because the prediction of mudras needed to be in real-time, we used Fast Artificial Neural Network (FANN) Library. FANN is a free open source neural network library, which implements multilayer feedforward neural network in C using back-propagation training [7]. FANN is very easy to use, versatile and fast, which is a good choice for real-time classification. We implemented the classification model using FANN python binding. The neural network structure we implemented including 3 layers (input layer, hidden layer and output layer) and 64 hidden neurons. The hyper-parameters of the neural network connection rate = 1, learning rate = 0.7 and was trained over 10000 epochs using sigmoid function as activation function and RPROP as the training algorithm.

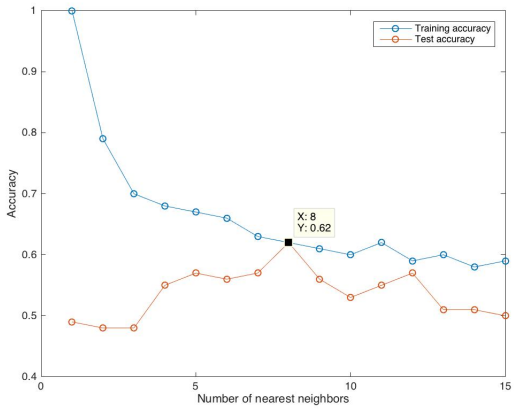


Fig. 6: Train and test accuracy vs. number of nearest neighbors k

V. Results and Discussion

A. Movement prediction for hands out of range

The classification models described in IV-A.2 were implemented in Matlab and tested using hold-out cross validation. The training and test accuracy are shown in Table I as an average of 10 different hold-out sets. Additionally, the average prediction time over 10 predictions is shown. These computations were all performed on the same computer using Matlab so can be used as a comparison, but not as absolute truth. KNN was chosen as the best prediction method as it provided the best test accuracy of $30 \pm 4\%$, where chance is 10%. KNN is also sufficiently fast and performed in this case in under $1/60$ms, the refresh time of the Leap Sensor. While the test error on all of the methods is not great, this was 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. In any case, having some kind of prediction is better than dropping motion data.

B. Overlapping Gesture Classification

We implemented all the three models mentioned in Section IV-B and tried to get the best accuracy on test set, which is summarized in Table II. Since the prediction time is also an

	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 I: Test accuracy of SVM, BDT and KNN models

important component to be considered in real-time performance, we timed each model and the result is summarized in Table III. Our best k-NN model with $k = 8$ achieves the best result of 62.0% test accuracy. The prediction time of k-NN model is 3.90 ms, which is short enough and desirable in real performance. We visualized the confusion matrix of the best k-NN model to examine its performance, as shown in figure 7. From the confusion matrix, we can see that some classes are harder to classify than other. For example, the algorithm misclassifed 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.

	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 II: Test accuracy of SVM, k-NN and NN models

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

TABLE III: Prediction time of SVM, k-NN and NN models

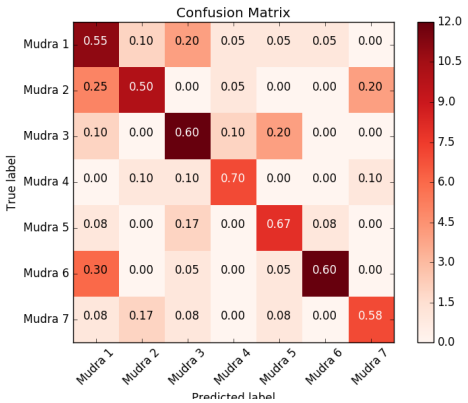


Fig. 7: Confusion matrix of the best k-NN model

VI. Conclusion

We attempted to solve two existing problems that arose while a Leap Motion sensor was used as a musical instrument. The sensor has a limited range so the lost trajectory data was predicted using a combination of k-means for clustering and KNN for classification. The algorithm only predicted the correct trajectory 30% of the time, but this could likely be improved by collecting more training data and further observing typical movements by the performer. Additionally, the sensor cannot detect overlapping hand motions which were important for artistic reasons. We implemented three different algorithms to predict hand gestured while the hands were overlapped and ended up choosing to use KNN as it gave an average test accuracy of 62%. KNN is also ideal for the predictions as it can provide fast predictions needed for the real-time audio-visual generation. If we were to continue working on this project, it would be beneficial to observe more performances and learn more about the typical motions. We could then gain more accurate, and a larger amount of training data which would likely lead to greater prediction accuracy, and would open up the full potential of a neural network approach.

References

[1] "Embodied Music Meditation," [Online]. Available: http://ceciliawu.com/html/research-meditation.html

[2] "Leap Motion Developers," 2016. [Online]. Available: https://developer.leapmotion.com/

[3] "ntplib 0.3.3": "Python Package," [Online]. Available: https://pypi.python.org/pypi/ntplib/

[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] "scikit-learn: machine learning in Python," [Online]. Available: http://scikit-learn.org/stable/

[6] "Statistics and Machine Learning Toolbox," [Online]. Available: https://www.mathworks.com/products/statistics/

[7] "Fast Artificial Neural Network Library (FANN)," [Online]. Available: http://leenissen.dk/fann/wp/

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

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

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

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

[12] Liorand Rokach and Oded Maimon. "Data Mining and Knowledge Discovery Handbook." Boston, MA. Springer. 2010.

[13] 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.

[14] Patrick Choi. "Learning and Predicting Moving Object Trajectory: A Piecewise Trajectory Segment Approach".Robotics Institute, Carnegie Mellon University. 2006.

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

[16] Giulio Marin. "Hand Gesture Recognition with Leap Motion and Kinect Devices". Univeristy of Padova. 2014.

[17] Makiko Funasaka et. al.. "Sign Language Recognition Using Leap Motion Controller". Int'l Conf. Par. and Dist. Proc. Tech. and Appli.. 2015.