Enhanced Appearance-based Finger Detection and Tracking Using Finite State Machine Control

Noorkholis Luthfil Hakim[1], Timothy K. Shih[1], Lin Hui[2]

[1] Department of Computer Science and Information Engineering, National Central University, Taiwan
[2] Department of Innovative Information and Technology, Tamkang University, Taiwan
{koliskol, timothykshih}@gmail.com, amar0627@gms.tku.edu.tw

Abstract

Real-time finger detection and tracking systems have been growing rapidly in the past decade. Among those methods, Appearance-based and Model-based methods have produced excellent results. However, the occlusion issue is one of the main challenges in this field. In this study, we address this issue by considering the repeating-finite gestures of a guitar-strumming or a hand puppet and, represent using a Finite State Machine model. Also we proposed a novel finger pose tracking system using FSM Model combining with the appearance-based method.. The proposed system consists of two parts: FSM-FT builder creates the FSM hand, and the FSM-FT runner controls the FSM-FT system. Empirically, we conducted an experimental study involving one sample repeating hand gesture and our approach achieved a significance recognition rate of 82% in the testing phase.

Keywords: Finger detection, Finger tracking, Human-computer Interaction, Finite State Machine

1 Introduction

Finger detection and tracking recently has grown to be more and more popular and is an essential feature of many computer vision applications such as in [22]. Among th previous studies, most of them were proposed under wearable devices-based [1-3], model-based [4-6] or hand-pose-based [23], and appearance-based or visual-based [7-9]. Wearable devices may have the best accurate result of detecting the finger position. However, due to its "uncomfortable" and "expensive" devices, it cannot be used in real-life applications. On the other hand, model-based finger estimation works very well in estimating the finger position and some of them can work in a real-time system However, to implement and use such a system in real-time home applications is an expensive and extensive resource allocation process. Even though the appearance-based method does not have better accuracy compared to the other methods, it can be easily implemented in real-life applications without allocating many resources and a powerful machine. Besides, the appearance-based system also has the speed that needed in the fast-moving finger problem due it simplifies the structure of the model.

Some finger actions used in the real-life situation often has a repeatable structure. Guitar strumming, doll playing manipulation such as Bu-da-chi doll or Finger-like doll are the example of such previous mentioned actions. The problem of detecting and tracking such a gesture may be solved using Hand pose or Model-based system by estimate the finger position given the hand data to the model. However, the estimated finger position's result may have a significantly differ position with the real world finger position and it will become a problem. In 3D animation for 3D doll or sound producing by the finger movement in virtual guitar is an example for that. Thus, leave the real-world position of the finger as it originated in the application system could solve the problem. To do such work, model the real world finger trajectory in repeatable action in the form of FSM without changing its position is possible. Combining the expensive model-based system with FSM control can be a problem in the speed of finger action in the future, thus using the speed-strength with acceptable accuracy of the appearance-based method is the possible option. Using the FSM model finger trajectory in an appearance-based method could also solve many problems that the appearance-based method has. Such as missing fingers by occlusion, finger jittering, miss labeling, and miss detection.

Based on the facts, we proposed a novel finger pose tracking system using FSM Model by combining with the appearance-based method. The proposed solution consisted of two parts: FSM-FT builder creates the FSM hand, and the FSM-FT runner controls the FSM-FT system. The first part is employed to generate the FSM trajectory model of finger actions. The second part is employed to produce a finger position given by the FSM trajectory model and a new hand input data. In this study, we call the first part as FSM-Builder and the second one as FSM-Runner. The purpose of FSM-

*Corresponding Author: Noorkholis Luthfil Hakim; E-mail: koliskol@gmail.com
DOI: 10.3966/160792642020072104017

Builder is to represent the whole trajectory of finger actions into several states. It clusters similar hand gestures into one class, or we name it as Hand state. One action should have more than one hand state and it depends on how smooth the finger has produced by the system. Besides, one particular FSM trajectory model expresses one repeatable action. Each hand state has a representation of five 3D finger positions in the 3D world that collected by real-world data. In the second part, the FSM-Runner concept makes the FSM trajectory model produce the finger position by moving from one hand state to the next hand state. This is done based on the transition function and the new hand input data. The hand input data consist of the 3D finger position from the appearance-based method that we build and its RGB hand image. The FSM-Runner choose the correct state and the next state so, the system could produce the correct finger position accordingly.

In order to validate the proposed system, we conduct an experimental study involving one sample repeating hand gesture. Our approach achieved a significant recognition rate during the testing phase. The rest of this paper is structured as follows. Section 2 discusses some related works and Section 3 presents a detail description of the implementation of the proposed work. Section 4 explains the experimental steps including results, and finally Section 5 concludes the study.

2 Related Work

2.1 Finite State Machine

Finite State Machine is one of the mature methods that has been used for many purposes. One of them uses to model the trajectory of movements of the gestures. As mentioned in [10] FSM mostly used as the recognizer. Given the input data, the feature vectors such as trajectories will decide whether to stay at the current state of the FSM or jump to the next state. When reaching the final state, we can say that the gesture completely recognized. For example, hand or finger to recognize the gestures [11-14]. These works give us the general idea that some repeating gestures could be modeled by this Finite State Machine, and by using the gesture recognition classification problem, we are able to go through the states in the FSM model to retrieve the finger representation from each state.

2.2 Finger Detection and Tracking

Finger detection and tracking method divided into several kinds. There is appearance/visual based finger detection. Such as Ren et al. works in [15] proposed the infamous distance matrix called FEMD (Finger-Earth Movers Distance). Then improve their work in [16] by using the K-curvature Algorithm to find the finger positions. Li [17] proposed to utilize the Graham Scan algorithm for generating the convex hulls to detect the fingers. Our previous work in [18] is inspired us to create our enhanced finger detection method used in this work as one of the features.

The other kind of method is a model-based approach (generative pose). This method works by finding the minimum error of the hand model with its angle DOF parameters given the input depth image or a Point cloud. These kinds of methods have produced better accuracy results of finding the fingers compared to the appearance model. Starting from Oikonomidis work [4] in 2011, presented to analyze the orientation and full articulation of human hand captured from Kinect sensor using PSO. Then Qian [5] extends it by adding a guided PSO step. Another example works are [19] driven by a physics solver to generate 3D pose estimation. While this model based or generative based finger pose estimation (or hand pose estimation) are have a good result, their work is too complex and need a big amount of resource for the real-time application. That's why appearance based research still growing until know.

3 Methodology

As seen in Figure 1, the complete FSM Control system has two main algorithms: FSM-Builder and FSM-Runner. The first algorithm is composed to create the model of representation finger trajectory based on sample finger action. On the other side, the latter is used to control the position and label of the finger based on the new input of the new data regarding the trajectory model created by FSM-Builder. Before discussing the two main systems in detail, we explain the preprocessing module in the following section. During this section, we have presented the way we collect and preprocess the sample data to be used for the FSM-Builder algorithm. Then, the second module about the appearance Finger detection and Tracking method that we proposed has explained as well.

3.1 Hand Sample Data Collections

To model the finger trajectory into FSM, one needs to sample the specific action or gesture. In this work, we do sample several sets of actions with all the possible gestures needed to be act. These samples extracted from the people whose expertise in the field. As an instance, if we want to create the FSM trajectory control model for guitar strumming or picking, at least we may need people that could perform those skill well. The data recorded consists of several parts. Those data are the RGB images *(Irgb)* and depth images *(Idepth)*. The Point cloud data (*Pc*) can be extracted from the given data and the point cloud needed in order to get the real representation of the finger position in the 3D world. The finger position extracted from the point

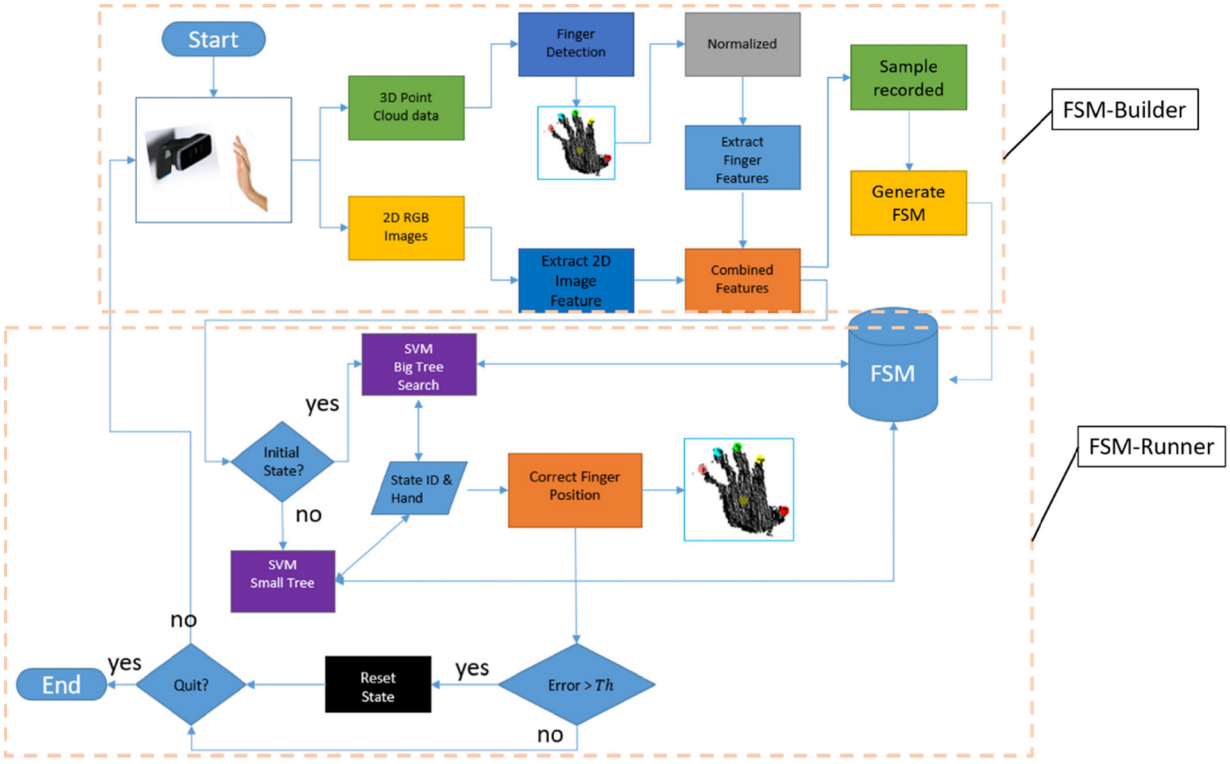


Figure 1. System Architecture, consist of 2 main algorithms, FSM-Builder, and FSM-Runner

cloud using our proposed appearance-based finger detection has explained in detail within the next section. Both point cloud and finger position generated in 3-dimensional Euclidean space consists of *x*, *y*, and *z* points. Point cloud and finger are in the form of *.ply* files and *.txt* files, respectively. In addition, the finger files consist of a finger position in *x*, *y*, *z*, and its label.

During the next step, all of these finger position samples and the label will be fixed by the help of a human, to produce the correct labeling sample and finger position. Because of the inability of producing accurate results, with the proposed appearance-based finger detection and tracking, semi-manual labeling by the human is essential. The 3D finger selector application has been built to help for marking and fixing the fingers and the results of marking are in Figure 2. Apart from that, Figure 3 presents the interface application of 3D finger selector and this helps the user to mark or edit the finger position faster.

(a) Point Cloud (b) Raw Finger Tracking (c) Fixed Finger

Figure 2. Example of fixing finger position in the labeling section

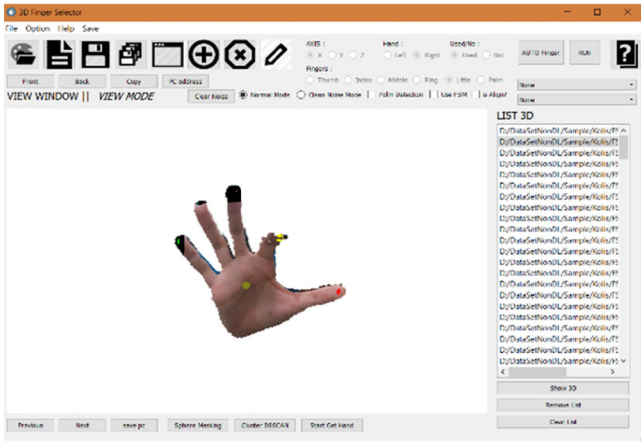


Figure 3. The interface of 3D finger selector application that used to fix the finger

3.2 Hand Segmentation and Arm Removal

In this section, we present the proposed appearance based 3D finger detection and tracking from the point cloud data that we use to extract the finger position feature. The first step of the system is to separate hand from the rest of the body and the background. For this task, we used a Realsense Depth Camera and we apply a common depth threshold value to separate the hand. This method supports to filter out the unnecessary points in point cloud and pixels in the RGB image from the predefined minimum and maximum distance. Moreover, the background and the body of the user can be easily removed by considering the hand as the closest object in the range. However, occasionally the arm is still available in the range of a predefined area.

To extract the hand from the arm, the central point or middle point of the hand is needed. It computes by averaging all of the available 3D points after removing the background, as seen in Figure 4(a).

Let $Pc^{hand} = \{p_1, p_2 \ldots p_n\}$ as the point cloud of the hand, while p_i is the single point of the point cloud with the n total number of the hand point cloud and p_{mid} is the middle point that we need to extract using the equation (1).

$$p_{mid} = \frac{\sum_{i=1}^{n} p_i}{n} \quad \textbf{(1)}$$

The direction of hand detection is needed to remove the arm region. Thus we assume that the direction of the hand is predefined as "Up" direction, as seen in Figure 4(a) and the opposite direction of the hand direction will be removed. Given the palm point position and direction "UP", the reference point p_{ref} iscalculated by: $p_{ref} = p_{mid} \cdot y + s_c \cdot y$ where, s_c = scalar. Any point in the hand point cloud bellow p_{ref}, is removed as the condition in the equation (2).

$$Pc^{rem} = A_{rm}(p_i) = \begin{cases} p_i \cdot y > p_{ref}, p_i * 0 \\ otherwise, p_i * 1 \end{cases} \quad \textbf{(2)}$$

The reference point p_{ref}, in Figure 4(b) with the red line as the visual representation, while the final result of the arm removal is shown in Figure 4(c). This preprocessing step results in the new point cloud Pc^{rem} that can be used for the next finger extraction algorithm.

(a) (b) (c)

Figure 4. Arm removal strategy

In the next frame, to track the direction of the hand and it is calculated by finding the rotation degree of the average detected fingers compared to the rotation degree in the previous frame. The detail explanation about the direction tracking is presented in the next following sections.

3.3 Finger Detection

To extract the finger position from the point cloud after the preprocessing step the system first needs to check whether the state of the hand belongs to the "detection" state or "tracking" state. When the hand in the "detection" state, the system will extract the finger position by detection algorithm. After successfully detect the finger position, the state of the hand will change into the "tracking" state in the next frame. In this section, we will talk about our proposed finger detection method briefly.

The proposed finger detection has three steps: remove the palm's point area, clustering finger's point candidates, and finding the fingertips. Let Pc^{rem} are the new point cloud after preprocessing and the palm point position can be calculated using the equation (3),

$$\hat{p_{mid}} = \frac{\sum_{i=0}^{n} p_i^{rem}}{n} \quad \textbf{(3)}$$

From the palm's middle point $\hat{Pc_{mid}}$ detected from the equation (3), palm candidate point can be removed by applying the virtual sphere region in the point cloud (see Figure 5(a)). Any points intersect with the virtual sphere are removed using γ threshold. However, in order to keep the meaningful points when the finger curved, δ threshold applied to the virtual sphere.

The equation to get the virtual sphere region is defined as follow:

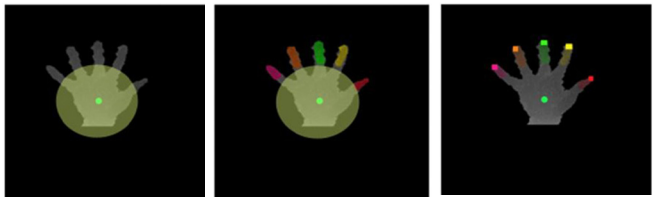
$$Pc^{V} = \begin{cases} 0, if\ V(Pc^{rem}) = true \\ 1, \quad otherwise \end{cases} \quad \textbf{(4)}$$

$$V(Pc^{rem}) = \sqrt{(pc_c^{rem} \cdot x - \hat{pc_{mid}} \cdot x)^2 + (pc_c^{rem} \cdot y - \hat{pc_{mid}} \cdot y)^2} < r \ and \ pc_c^{rem} \cdot z - \hat{pc_{mid}} \cdot z < \delta \quad \textbf{(5)}$$

We employ 2D Euclidean distance to check whether the points are within the circle and then check the depth difference between point clouds and the center of hand palm using equation 5. Note that we didn't use the absolute value on the subtraction so that the pixels that are further from camera than center point of hand will be removed (In our system, we get larger z value if the point is near to camera, it's negative value because we mirrored our scene from Real Sense). Using this method, we get a new point cloud Pc^{V}. Moving further K-distance clustering applied to the separated pixels or called the finger regions, and there should be at least K (e.g. five) clusters after the previous step, as can see in Figure 5(b). For each point in finger regions, the 3D Euclidean distance between any two points is calculated, and if two points are close enough, they are considered as the same cluster. Finally, given the candidate points of each cluster, we calculate the score by using equation (6) and the fingertips F_{hand} can select using the k-maximum score of the candidate point. The result of this step is shown in Figure 5(c).

$$dbf(Pc^{cl}) = \sqrt{(p_i^{cl} \cdot x - \hat{p_{mid}} \cdot x)^2 + (p_i^{cl} \cdot y - \hat{p_{mid}} \cdot y)^2} + ((p_i^{cl} \cdot y - \hat{p_{mid}} \cdot y)^{\infty} \times 2) \quad \textbf{(6)}$$

$$F_{hand} = Max(dbf(Pc^{cl})), \text{ where } F_{hand} = \{f_1, f_2, \ldots, f_5\} \quad \textbf{(7)}$$



(a) (b) (c)

Figure 5. Finger detection steps

3.4 Finger Tracking

The clustering process in finger detection spends a long time process. To optimize the time of finger extraction, tracking the finger position in each frame is necessary. Thus, we propose a method for tracking the finger position in 3D space, called the "*cube region search algorithm*". Given the F_{hand} from the detection algorithm, for each finger f_i in F_{hand} we generate 27 grid cubes region search (*SC*) around the previous extracted finger position for each finger available as seen in Figure 6. For each *SC* in sc_{ijk} we find the maximum *dbf* score using the previously mentioned equation. Given Pc^V and sc_{ijk} area, the new finger could be extracted by equation (8).

$$\hat{F_{hand}} = Max(dbf(sc_{ijk}(Pc^V))) \quad \textbf{(8)}$$

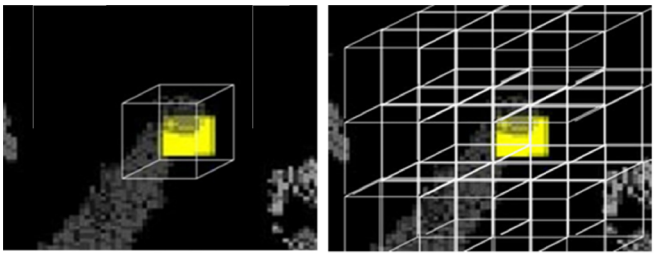


Figure 6. 3D Cube search finger tracking

The system uses the previously explained finger detection method to overcome the problem of fast finger moving and noisy point cloud. After tracking, the system will predict the hand direction by calculating the available angle and decide the direction by angle region check.

As mentioned before, this decided direction is used for the wrist removal method. When tracking the hand, we assume the direction is "UP" at the beginning and, if the tracking is failed, the system will reset the direction to the initial state and users need to adjust their hand accordingly. Detected and tracked fingers from the previous detection methods, then presented as features that represent the hand.

3.5 Finger Normalization

By the equation (8), we could extract the finger position and use it as one of the features for the FSM-FT system. However, in some cases such as: when the finger moving too fast, the noisy point cloud and finger occlusion, the proposed method not able to extract the finger properly. Thus, in the *"Hand Sample Data Collections"* section, we ask humans to fix the finger label and the missing finger position manually. This step is crucial for the process to build the FSM trajectory control model during the next step because of the actual label and finger position of the hand action that will be produced by the end of the system. Therefore, normalization of the finger data is needed since the hand position and size aren't always in the same manner, every time it recorded.

There are three steps for hand normalization. First, subtracting each finger with its palm center point. This step used to bring each hand in the origin position in the axis (0, 0, and 0) concerning its palm center point. Second, find the scale factor of each hand, which calculated by dividing the statistical measure of the size of the hand with its number of a finger. The third step is to normalize the size of the hand by dividing each finger with its scale factor. This three-step are represented in the equations below.

Where $palm\ point = (x', y', z')$

$$(x_i, y_i, z_i)' = (x_i - x' - y_i - y' - z_i - z') \quad \textbf{(9)}$$

$$s = \sqrt{\frac{\sum_{i=1}^{n}(x_i, y_i, z_i)'^2}{n}} \quad \textbf{(10)}$$

$$(x_i, y_i, z_i)'' = (\frac{x_i}{s}, \frac{y_i}{s}, \frac{z_i}{s}) \quad \textbf{(11)}$$

Given the finger sample data normalized, the FSM trajectory control model is ready to generate. A detailed explanation will be presented in the next section.

3.6 FSM-FT Builder

There are several steps that need to perform in order to transform the sequence of sample action into the FSM model, that represent the finger trajectory of the action itself. As we know, FSM consists of several members. A finite of state, a finite non-empty input or relationship between the state, and a series of transition functions. Thus, the first step is to generate the finite of state. In fact, the whole sample in one set of actions can be represented as several numbers of finite states in FSM. Those states are the cluster of the finger in hand that has the same pose and, the famous K-means clustering can work well to extract such states. However, since we do not know the exact number of the states that the sample could produce and how many may differ from the other actions, K-means clustering cannot be used.

As an alternative, the simple nearest neighbor clustering with some distance threshold is performed.

To cluster the finger, first, we extract features to represent the normalize finger. The normalize finger method is not rotation invariant and therefore, we chose the angle as the feature to cluster the finger sample. Moving further, we use spherical coordinate space on the angle feature extracted with the reason that this method is claimed to be rotation invariant. Spherical coordinate has 3 parameters: radial distance, polar angle, and azimuthal angle (r,θ,φ). In this case, we only use the angle parameters, which are polar angle and azimuth angle as features. i=1…5 (fingers).

$$mag_i = \sqrt{x_i^2 + y_i^2 + z_i^2}$$
$$\gamma_i = \cos^{-1}(\frac{z_i}{mag_i}), \theta = \gamma_i, \varphi = \tan^{-1}(\frac{y}{x}) \quad (12)$$
$$An_s = [\theta_1, \varphi_1, ..., \theta_5, \varphi_5]$$

Given the angle feature vectors of each hand, the distances of each cluster are calculated as follow

$$D_s = \sqrt{\Sigma(An_f - An_{f+1})^2}$$
$$C_n = D_s < T_c \quad (13)$$

By experimenting with the sample data, threshold scalar (Tc) was calculated. If the distance D_s is below the threshold, then it labels as the same cluster C_n and calculates the finger centroid as the representation of the finger in the state, which is called the "state hand".

3.7 State Connections

The next step is to generate the transition function. However, as shown in Figure 7, each state is not connected to each other. To connect those states, the relationships between states are needed. There are two ways to form the relationship between states. The first is using the distances between the state's hand. The distance can be calculated from the angle feature of the new hand state (finger representation of each state) of one cluster to the other and find the minimum distance to connect it. The second way is to find the minimum frame difference (MFD) between states. Each cluster is already connected in the time sequence of the sample data. Each member in each cluster belongs to a different time frame and can be connected by using the MFD algorithm. Thus, in this system, we used both of these features (distance and minimum frame difference) to feed into the classification tree. This classification tree produces three classes as the next state, not next state, and merge state class. To train the classification tree, several FSM sets are created. We collected the features from each state in each FSM and marked the ground truth class and, then feed the training sample into the classification tree. Below equations are the feature extraction steps of the state hand.

$$MFD = fr_{dif} = \min(fr_s - fr_{s+1}) \le 1 \quad (14)$$

$$Angle\ dis\tan ce = D_s = \sqrt{\Sigma(An_f - An_{f+1})^2} \quad (15)$$

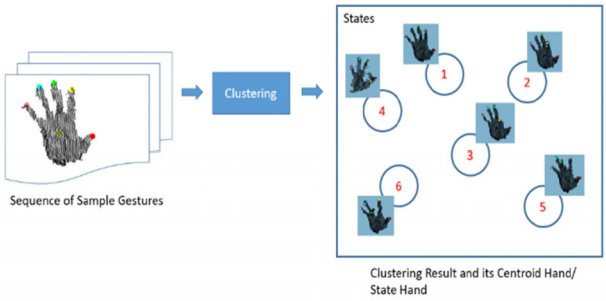


Figure 7. The result of state hand creation from finger clustering method

Given the state number, the relationship between states, and the state hand, the state transition function could be extracted directly. The key part of the FSM system is the state transition. It indicates the moving direction of the states and limits the movement of the states.

Figure 8 is the simple sample result of FSM-Builder's result. We use this simple action movement of the index and middle finger curved to the palm to explain how the FSM-FT Builder system works.

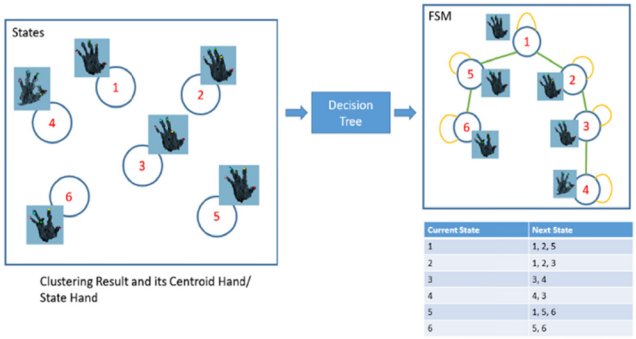


Figure 8. The final result of FSM-Builder

3.8 FSM-FT Runner

From the FSM-Builder algorithm, the FSM model representation has been generated complete with its transition function. To make the FSM model run or work, the second important algorithm, the FSM-Runner is proposed. This algorithm is used to find the initial state and predict the next state of the FSM model that could extract the representation of the correct finger position produced by the current state given the new input data by following the state transition function.

The generate FSM model created by FSM-Builder has several parameters.

$FSM = \{S, V, q_0, q_x, F\}$, where,

$S = \{s_1, s_2, ..., s_i, s_x\}$ are the states

$V = \{v_1, v_2, s_3, ..., v_n\}$ (i.e., hand and finger feature vector)

$q_0 : s_1$ (i.e., initial sate) and $q_x : s_x$ (i.e., exit state)

$F = S \times V \rightarrow S$ (e.g., $F(s_i, v_n) \rightarrow s_{i+1}$), next state function

Let FSM-Runner be the FSM control of FSM generated by FSM-Builder. FSM-Runner takes the current state s_i and current camera input V^{all} as the input to produce the next state s_{i+1}. Where V^{all} is the combination of hand features V_{ad}, V_{hog}, V_{hu}. In detail, V_{ad} is the collection of vector finger angle An_s. While the finger position in 3D F_{hand}. V_{hog} is the Histogram of Gradient feature of the image I_{rgb} and V_{hu}, is the Hu moment feature of the image I_{rgb}. The next state s_{i+1} then will produce the new finger F^*_{hand} that represent in the corresponding state. The combination of features could be 2 or more features. We did some experiments to check the best combination of features and the results have discussed under Section 4 and Figure 9 representthe FSM system.

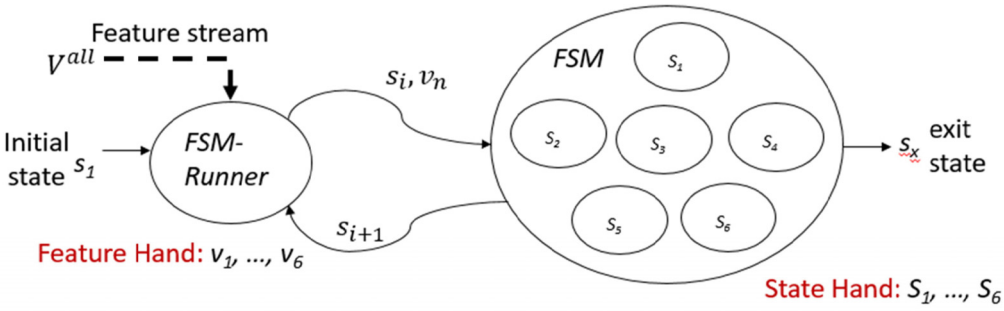


Figure 9. The scheme work of the FSM model with FSM-Runner algorithm

The speed of the finger may be one of the difficulties of selecting the correct state. Thus to solve the problem, we built the jump mechanism system that in the case of the confidence of the system to predict the correct state is lower than the threshold. And then the next state prediction could be selected without a restriction of the state transition table (using a similar method as finding the initial state).

In this work, we implemented two kinds of strategies to predict the next state using the SVM model. The first strategy called the Big Tree model. It works by training all the data among all categories, which in this case is the hand states, in one complete SVM model. This model is used to decide the initial state and also the "big jump" mechanism system. The complete strategy is in Figure 10 and by using this strategy, the system is not allowed to restrict the Transition function. Thus it allows the system to produce all of the available hand poses in the FSM model.

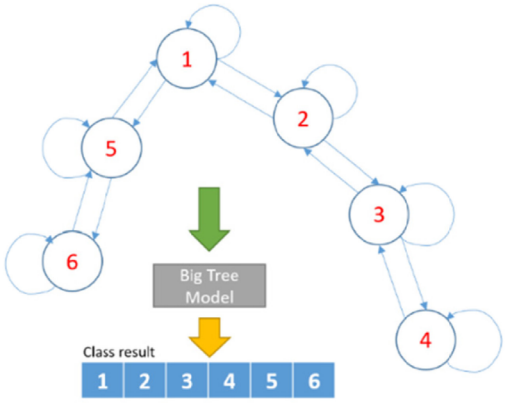


Figure 10. Example of FSM-Runner first strategy

The second strategy is called the Small tree model. Instead of using one complete model of SVM classification, we divided the model into several parts with the number of categories is one of the states and, its' neighbor that connected with the corresponding state. The depth of the neighbor connection is 1 to 2 depth of connection depends on how big the FSM model is. From the largest tree of FSM, the longest depth can be selected. In our sample works, we used only 1 depth of neighbor connection. This strategy is the main strategy that we apply to select the correct state according to the transition function. The SVM model that we trained produced the probability of the prediction in each trained state during the current state. Figure 11 is the second strategy of tree model with 2 depth of connection.

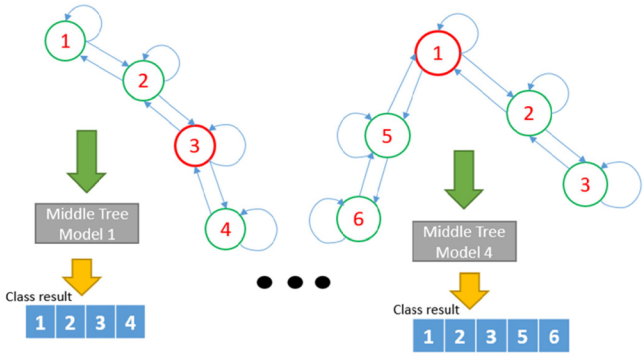


Figure 11. Example of FSM-Runner second strategy

The feature vector extracted from the input then insert into the second SVM strategy and let the two models that have been trained before to predict the next state from the current state and its input. Then, the last step for the FSM-FT Runner system is the pooling step.

This pooling will take the two probability results from three models. If both model's result is similar, then the system directly takes the result as the main result. When all the results are different, the next thing to do is fit the finger value from the two Hand state of the two models and compare its minimum distance to the point cloud. The Hand that has a minimum distance with the point cloud and highest probability result will be selected.

4 Experimental Result and Discussion

To test the robustness of our method, we conduct several experiments. Since the unavailability of any public dataset that closes to our works, we designed simple finite-repeating gestures and generated its FSM model. We used this gesture or action to test the system. Figure 12 is a simple design of finite-repeating gestures called counting action. We record 4 samples of actions, which contain 300 frames for each action. We use one of the samples to generate the FSM model and the rest will be used for training and testing in the FSM-Runner step. From 300 frames sequence of counting action, we able to extract 33 hand states and their relationship. This FSM used as the base to test our FSM-Runner along with the rest of the sample as mentioned before.

Figure 12. Sample finite-repeating set of gestures that uses to test the proposed system

4.1 First Experiment Result

The first experiment is used to investigate the best possible combination of the feature. We have three kinds of features extracted from 3d and 2d space that can be combined to enhance the result. There are 3 main features that we extract from the hand sample: 3D finger position from AFD (Appearance Finger Detection), HOG (Histogram of Oriented Gradient) and Hu moment from 2D hand images. To test the combination and also the robustness of our method, we used the first strategy, the big tree model that we mentioned before. Apart from that, 90% and 10% of the sample data have used for training and testing, respectively.

The results of the first experiment have shown in Table 1 and it shows that the combination of AFD + HOG has the best result among the other methods. However, there is no significant difference in the recognition rate between the other combinations.

Table 1. Results of the first experimental for selecting a combination of features

Name of Feature	Recognition rate
Only AFD	69.1%
HOG	68.4%
Hu Moment	63.3%
HOG + Hu	71.8%
AFD + HOG	75.0%
AFD + Hu	71.13%
AFD + HOG + Hu	74.2%

The result of the Big SVM tree only reach 75% accuracy, because since we want to keep the smoothness of the finger movement, it ends up with many categories and classes, thus it lowers the accuracy of detecting the class. It means that it's hard to distinguish the difference of the class hand created by the FSM-Builder when the smoothness of the gesture becomes the main concentration (lower threshold of clustering in FSM Building).

4.2 Second Experiment Result

The second experiment is to test the Small Tree of SVM. By using the same FSM model and sample set from the first experiment, we trained and tested the Small Tree of SVM using the top combination features, resulting from the previous experiment which is AFD + HOG. Figure 13 illustrated the results of the recognition rate after we tested all of the SVM models. The average result of cross-validation gets an average of 83.01% recognition rate while the testing gets an average of 73% recognition rate.

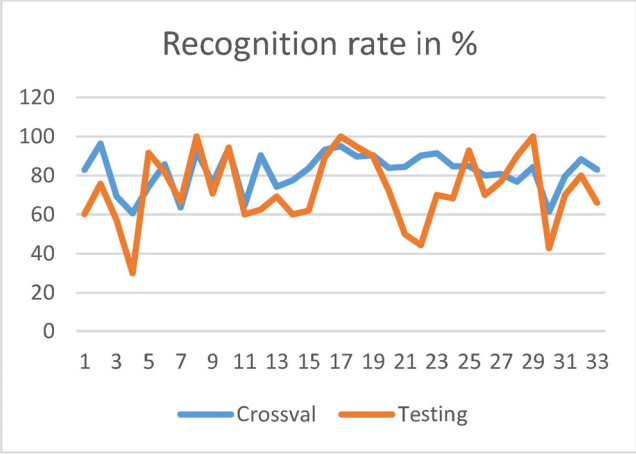


Figure 13. Result of the second experiment using Small Tree SVM method of FSM-Runner algorithm

Once more, in some parts of class or state, the recognition rate falls to a shallowvalue. We believe this

happens since we try to keep the smoothness of the hand gesture, while the feature was not able to distinguish it correctly. Therefore, it is essential to select more additional hand features to overcome this issue. Another reason is that, because of the difficulty of finding training data for each specific class/state. It ends up creating augmented synthetic data that have not variated enough comparing to the real data produced by a human. Moreover, smooth the finger movement, the difficulty of creating the training data and increase the size of classes are the main drawbacks or limitations of this approach that we need to be solved in the future.

4.3 Comparison with Existing Method

Previous work such as [15-17] able to extract the finger position from the depth data similar to our approach but they are mosly fail due to self occlusion such as finger bending. Do to the help of tracking on FSM model, whenever the data missing by some occlusion, the finger position still could be extracted by finding the neighbor hand state in the current time. Thus, our approach never have a missing finger or false labeling. Jittering finger also reduced since our finger position result is a fixed. Work on [21] also use the same approach that training RGB image of finger using SVM as the classifier to detect the finger position. But those approach are only work in the current hand position and can't get the real world finger position. The limitation of our work is that for now is its only work on repeating-like motion or action. By the nature of simple FSM model and Machine learning approach our proposed method is fast comparing any Model based approach such as in [4, 5, 19]. Our approach could run in Real-Time speed on the low end machine specification. Thus, it is possible to extend our method into embedded system.

5 Conclusion and Future Works

This work proposed a novel finger pose tracking based FSM algorithm. The proposed method consists of two main algorithms —first, the FSM-Builder system, which created to extract the FSM model from the finite-repeating action sequence. The system is clustering the similar hand poses into one and represent it as a state. Each state consists of a representation of the correct finger position and the relationship with other states. This relationship then used for the second part of the FSM-FT system called the FSM-Runner system. Given the FSM model from FSM-Builder, the Runner system helps to guide the input hand images and point cloud into the correct state of the hand hence to produce the right finger position.

Our first experiment is to select the best combination of features that we proposed and the result shows that the AFD + HOG method has achieved a significant performance compared to the other combinations. The second experiment is to test the robustness of the small SVM tree of the FSM-FT Runner system by using the best combination of features. The result shows that it gained an average of 83% recognition rate of the system. However, in some states, the result does not have a high accuracy rate because of the feature that we selected was not enough to discriminate for each class. Another reason is that it does not have enough variation in the training data due to the difficulty of collecting some specific classes training data.

As our future directions, we plan to improve our research problem by selecting more features in the 3D point cloud side using a 3D point cloud descriptor such as a histogram of 3D facet [20], instead WFD because of its missing data. Besides, we hope to apply optical flow on the 2D side of data as an additional feature. It also will increase the size of the dataset to improve the system as well. Another promising idea is to apply deep learning techniques, convolutional neural networks as the FSM-FT Runner to extract the 2D features.

Acknowledgments

We acknowledge the support of the Ministry of Science and Technology (MOST), Taiwan, under the grant number MOST 109-2634-F-008-008.

References

[1] D. J. Sturman, D. Zeltzer, A Survey of Glove-based Input, *IEEE Computer graphics and Applications*, Vol. 14, No. 1, pp. 30-39, January, 1994.

[2] R. Y. Wang, J. Popovic, Real-time Hand-tracking with a Color Glove, *ACM Transactions on Graphics (TOG)*, New Orleans, Louisiana, USA, 2009, pp. 1-8.

[3] Y. Iwai, K. Watanabe, Y. Yagi, M. Yachida, Gesture Recognition by Using Colored Gloves, *IEEE International Conference on Systems, Man and Cybernetics. Information Intelligence and Systems* (Cat. No. 96CH35929), Beijing, China, 1996, pp. 76-81.

[4] I. Oikonomidis, N. Kyriazis, A. A. Argyros, Efficient Model-based 3D Tracking of Hand Articulations Using Kinect, *British Machine Vision Conference*, Dundee, UK, 2011, pp. 101.1-101.11.

[5] C. Qian, X. Sun, Y. Wei, X. Tang, J. Sun, Real-time and Robust Hand Tracking from Depth, *IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, Ohio, 2014, pp. 1106-1113.

[6] Y. Zhou, G. Jiang, Y. Lin, A Novel Finger and Hand Pose Estimation Technique for Real-time Hand Gesture Recognition, *Pattern Recognition*, Vol. 49, pp. 102-114, January, 2016.

[7] J. L. Raheja, A. Chaudhary, K. Singal, Tracking of Fingertips and Centers of Palm Using Kinect, *IEEE Third International Conference on Computational Intelligence, Modelling &*

Simulation, Langkawi, Malaysia, 2011, pp. 248-252.

[8] M. Vanco, I. Minarik, G. Rozinaj, Evaluation of Static Hand Gesture Algorithms, *Systems, Signals and Image Processing (IWSSIP)*, Dubrovnik, Croatia, 2014, pp. 83-86.

[9] S. Manitsaris, A. Tsagaris, K. Dimitropoulos, A. Manitsaris, Finger Musical Gesture Recognition in 3D Space Without any Tangible Instrument for Performing Arts, *International Journal of Arts and Technology*, Vol. 8, No. 1, pp. 11-29, January, 2015.

[10] S. S. Rautaray, A. Agrawal, Vision Based Hand Gesture Recognition for Human Computer Interaction: A Survey, *Artificial Intelligence Review*, Vol. 43, No 1, pp. 1-54, January, 2015.

[11] N.Ç. Kılıboz, U. Güdükbay, A Hand Gesture Recognition Technique for Human-computer Interaction, *Journal of Visual Communication and Image Representation,* Vol. 28, pp. 97-104, April, 2015.

[12] P. Hong, M. Turk, T. S. Huang, Gesture Modeling and Recognition Using Finite State Machines, *Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, Grenoble, France, 2000, pp. 410-415.

[13] R. Verma, A. Dev, Vision Based Hand Gesture Recognition Using Finite State Machines and Fuzzy Logic, *International Conference on Ultra Modern Telecommunications & Workshops*, St. Petersburg, Russia, 2009, pp. 1-6.

[14] H.-S. Yeo, B.-G. Lee, H. Lim, Hand Tracking and Gesture Recognition System for Human-computer Interaction Using Low-cost Hardware, *Multimedia Tools and Applications*, Vol. 74, No. 8, pp. 2687-2715, April, 2015.

[15] Z. Ren, J. Meng, J. Yuan, Z. Zhang, Robust Hand Gesture Recognition with Kinect Sensor, *19th ACM International Conference on Multimedia*, Scottsdale, Arizona, USA, 2011, pp. 759-760.

[16] Z. Ren, J. Yuan, J. Meng, Z. Zhang, Robust Part-based hand Gesture Recognition Using Kinect Sensor, *IEEE Transactions on Multimedia*, Vol. 15, No.5, pp. 1110-1120, August, 2013.

[17] Y. Li, Hand Gesture Recognition Using Kinect, *IEEE International Conference on Computer Science and Automation Engineering*, Beijing, China, 2012, pp. 196-199.

[18] N. L. Hakim, S.-W. Sun, M.-H. Hsu, T. K. Shih, S.-J. Wu, Virtual Guitar: Using Real-time Finger Tracking for Musical Instruments, *International Journal of Computational Science and Engineering*, Vol. 18, No. 4, pp. 438-450, April, 2019.

[19] S. Melax, L. Keselman, S. Orsten, Dynamics Based 3D Skeletal Hand Tracking, *Graphics Interface 2013*, Regina, Saskatchewan, Canada, 2013, pp. 63-70.

[20] C. Zhang, X. Yang, Y. Tian, Histogram of 3D Facets: A Characteristic Descriptor for Hand Gesture Recognition, *10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, Shanghai, China, 2013, pp. 1-8.

[21] A. Sophian, D. Aini, Fingertip Detection Using Histogram of Gradients and Support Vector Machine, *Indonesian Journal of Electrical Engineering and Computer Science*, Vol. 8, No. 2, pp. 482-486, November, 2017.

[22] B. K. Chakraborty, D. Sarma, M. K. Bhuyan, K. F. MacDorman, Review of Constraints on Vision-based Gesture Recognition for Human-computer Interaction, *IET Computer Vision*, Vol. 12, No. 1, pp. 3-15, February, 2018.

[23] F. Guo, Z. He, S. Zhang, X. Zhao, Estimation of 3D Human Hand Poses with Structured Pose Prior, *IET Computer Vision*, Vol. 13, No. 8, pp. 683-690, December, 2019.

Biograpies

Noorkholis Luthfil Hakim is currently a Ph.D. candidate in Department of Computer Science and Information Engineering in National Central University, Taiwan. He received his master degree in National Central University, Taiwan as well. His research interests includes, Computer vision, Image Processing, Deep learning, Human Computer Interaction and Interactive system.

Timothy K. Shih is a Distinguished Professor at the National Central University, Taiwan. He is a Fellow of the Institution of Engineering and Technology (IET). His research awards, including IIAS research award from Germany, HSSS award from Greece, Brandon Hall award from USA, the 2015 Google MOOC Focused Research Award.

Lin Hui received the Ph.D. degree in Computer Science, Tamkang University, Taiwan, in 2006. She is an associate professor in the Department of Innovation Information and Technology at Tamkang University, Taiwan. Her current research interests include Operation Research, Data mining, Deep Learning, and Multimedia Applications.