Unsupervised gesture recognition system for learning manipulative actions in virtual basketball

Divesh Lala[1] Yasser Mohammad[1 2] Toyoaki Nishida[1]

[1] Graduate School of Informatics, Kyoto University, Japan
[2] Electrical Engineering Department, Assiut University, Egypt

Abstract: For natural human-agent interaction, the recognition of important gestures to facilitate communication is necessary. Recent advances in technology have made it possible to use the human body as a means for this interaction. In this paper, we combine techniques to create a process flow which can discover and extract a body gesture, create a low-dimensional model, and then use it to recognize body gestures provided in real-time. We show that this technique can be applied to a virtual basketball game, an environment which is highly dynamic and where the variation of gestures is high but the number of training samples is low. We demonstrate this technique to the extraction and recognition of a passing gesture.

1 Introduction

Interactions between humans and virtual agents can be achieved through a number of modalities. The most common modality is arguably mouse and keyboard, which can be seen in agent interactions during video games. However, recently there has been an increase in the range of interaction modalities, including commercial applications. These modalities focus on mapping the physical world movements of the human into the virtual world, where it is acted upon by agents.

Of course, there needs to be a streamlined process for supervised or unsupervised learning of gestures and their recognition. One constraint is that we should not predefine the gestures. Especially in systems with a high manipulative/symbolic gesture ratio, there is no way to accurately define every possible type of gesture. Due to this, we include an extra step in the process of the learning of gestures, which is that of gesture *discovery*.

If we assume that the gestures should not be predefined, then the only way to gather information on gestures is to have users perform them. Of course, this means that gestures should be able to be discovered in the time series data. For example, if we know that a throwing action is contained in some time-series data, we must have some means of both finding this information and extracting it so that useful samples can be gained. There are a number of existing measures for discovering gestures or motifs, such as [16] and [8]. In this paper, we make use of these algorithms so that appropriate gestures may be extracted.

Following on from the condition of undefined gestures, we take the constraints further and argue that gestures should also be natural in their form. Compare two situations, one in which a user is told to perform gestures in isolation and one in which the user performs gestures in the context of a task. We argue that the latter situation is more preferable for generating samples, due to the contextual environment and also the amount of variation which can be displayed. In this case, the *naturalness* is higher in the latter task.

The contribution of this paper is a process flow, beginning from a time series of raw multi-dimensional data and ending with a set of gestures which can be recognized robustly. We introduce several constraints on the system during this process:

- Training samples must be extracted from time series data - nothing is known about the initial location of the gestures inside the time series
- The form of the gestures are varied
- The number of training samples is relatively low
- Gesture must be recognized as they occur, as opposed to when they end

While the techniques used are not unique to our system (apart from the recognition algorithm), the combination of them as an entire process is novel. We introduce this process flow so that other systems with similar constraints may be constructed in a straightforward manner.

Section 2 describes the virtual environment, that of virtual basketball. Section 3 discusses how gestures should be recognized in the environment, and the specific task environment is to be further clarified in Section 4. The actual framework is given in Section 5 followed by future work and the conclusion of the paper.

2 Virtual Basketball

The setting which will be used for the implementation of gesture extraction and recognition is a virtual basketball game. Essentially, the goal is to produce a game which functions as real basketball in

terms of body movements. No peripherals such as keyboard or mouse should be used, and tracking of the body is done using a Kinect camera. A more detailed overview of the system is described in [10]. Additionally, because the user should be able to navigate throughout the environment, a pressure pad sensor is used to detect the user walking and their direction. The algorithm for this is based on that in [11], and will not be described in detail here.

Virtual basketball is an ideal platform for gesture recognition due to the large number of gestures that are used. While some are obvious, such as shooting and passing, others can have a variety of meanings. These meanings may be explicit, such as calling or looking for a pass. On the other hand, they can be more abstract, such as a simple pointing gesture. Of course, contextual meaning and recognition are intertwined and support each other and this fact should not be lost in future prototypes. The next section will also differentiate between differing types of movements and choose those which are suitable for this work.

In the virtual world, the dynamism of virtual basketball becomes an issue as we would ideally like to recognize variations of a gesture, even if the movements differ greatly. For environments in which fast-paced action occurs, these variations will increase as the user focus is not so much on producing a 'correct' movement, but one which does a 'good enough' job. Taking into account the complexities involved in gestures such as passing and shooting, gesture recognition in basketball is not a trivial task.

3 Interaction Recognition

There are several issues to consider before deciding on the method of gesture extraction and recognition. Firstly, the *type* of gestures to be recognized should be decided upon. Also, established techniques should be analyzed so that an appropriate one can be adapted to virtual basketball. We discuss these issues below.

3.1 Gestures to be recognized

One issue in the field of interaction is the recognition of gestures which are important to the task. In the case of basketball, the obvious body gestures which should be recognized are physical actions such as dribbling, passing and shooting. On the other hand, there are other types of movements which are more analogous to signals towards other actors in the virtual space. We differentiate between these two types of body movements: object-manipulation acts and communicative acts. The latter category is what is traditionally thought of as being gesture, in particular what Ekman and Friesen describe as regulators [6].

We only consider the former category, as this is necessary for the implementation of playing basketball using the body. Without dribbling, passing and shooting, the game becomes meaningless. In particular, we use the passing movement as an example in the rest of the paper. Communicative acts are, as the name suggests, used for communicative purposes. While they enhance the system and facilitate human-agent communication, they are not strictly necessary for a representation of basketball.

There is another powerful argument for only considering object-manipulation movements for recognition. When observing real-life communication between humans, various non-verbal interactions take place. However, these interactions are not standardized in any way. For example, when requesting a pass from a team-mate in basketball, there is not one particular symbolic action which encapsulates this. The actions are dependent on the player.

The actual range of communicative acts which will be employed by the user is *unknown*, while object-manipulation acts can be predicted. Additionally, common features of object-manipulation acts can be identified while communicative acts may have no common features. Above all, by recognizing passing, dribbling and shooting, a basic version of virtual basketball can be implemented. In fact, the recognition of these actions are *necessary* for the development of further prototypes. Communicative acts can be seen as the next level of implementation.

3.2 Recognition requirements and techniques

The recognition of gestures is a wide area of research, the details of which won't be explained in this paper. In order to decide on the best recognition technique, some requirements must be defined and adhered to, taking the above discussions into account.

We refer to the previously stated constraints on the system given in Section 1. From these, we obtain requirements for selecting an appropriate gesture recognition technique. We can take the following constraints:

- The form of the gestures are varied
- The number of training samples is relatively low
- Gesture must be recognized as they occur, as opposed to when they end

which we describe as generalization, training sample size, and temporal progression respectively.

Next, we compare some existing techniques for recognition and see how they handle these requirements. We focus on three techniques: state machines, hidden Markov models (HMMs) and Gaussian Mixture Regression (GMR). It will be shown that the latter of these techniques is ideal for our task.

In general, state machines enable us to define any gesture as long as we can specify the position of the user for each separate state. It is also possible to include time constraints inside each state. This technique is useful for simple symbolic gestures which are well defined. The FUBI framework is a typical example of a state machine system which is fairly robust

Table 1: Comparison of gesture recognition techniques according to constraints

Method	Generalization	Training sample size	Temporal progression
State machine	Possible but difficult	None - gestures predefined	Easily implemented
HMM	Robust depending on model	Many needed for complex gestures	Possible but non-trivial
GMR	Can generalize in feature space	Few required for model	Easily implemented

and can recognize a range of gestures such as crouching and swiping [9]. It is also trivial to include temporal information for object manipulation. Complex gestures with a wide range of implementations, such as passing, can feasibly be recognized in state machine systems. However, the variations of gesture for a new user cannot be predicted and so generalization is difficult.

HMMs are a well known probabilistic model used in the field of gesture recognition. While they have been proven to be very powerful, one drawback is that they require a number of training samples to be effective. Naturally, the amount of samples needed increases with the number of dimensions and Gaussians used. In the case of a passing gesture using Kinect, we use three vectors for each arm with 3-dimensional coordinates for each, giving a total of 18 dimensions. Even without specifying a number of Gaussians to use, this model already requires a fairly large number of training samples. Additionally, time progressions in HMMs are not a trivial task, although solutions do exist [2].

The final method, and the one which we propose to implement is Gaussian Mixture Regression. We take an approach similar to the one described in [4], where a robot learned the gesture given to it by a human, with a comparatively small number of training samples. In order to implement this technique effectively, dimension reduction is performed through principal component analysis (PCA), which includes the time components. Gaussians are then applied to the feature space in a representation of the general trajectory of the motion. For robustness, the same gesture can be repeated, though a large number of repetitions is not required (only 4-7 were used in the work). The end result is a mean trajectory over time which can be used for gesture comparison. In the original work, this technique was used to also generate gesture for a robot, but this step is omitted here.

This method allows for real-time analysis of gesture. All that is required is that the streamed data be reduced to that gesture's feature space and compared with the trajectory of the model. We have seen that few training samples are needed for this approach. Generalization is made possible as the model accounts for variation in the training samples. Time progression of a gesture may also be calculated, as time points are embedded in the model and can be referred to as fixed points where object manipulation occurs, or subjected to the detection of change points.

From the above discussions, GMR appears to be suitable for this task and it is proposed that it will be used in the real-time recognition of gestures in the basketball game. Table 1 summarizes this discussion on the comparison of techniques.

4 The Model Training Environment

While it would be ideal to learn gestures during the playing of the game, there is no way to do this given that we have no prior information about the displayed gestures. The alternative approach is to set up a training environment where a gesture can be used. The advantage is that this training environment is a subversion of the real game, so that the context of the gestures can be preserved. It does not make sense to inform the user to do a passing gesture multiple times in front of a screen. Their behavior inside and outside the environment may differ. For this purpose, we created different training environments for the user so that they could execute gestures within the basketball context. Screen shots of each of these environments can be seen in Figure 1.

Naturally, the environment requires feedback. In particular, if the user is passing or shooting in the virtual environment, they will expect their actions to produce a movement of the ball. While it is impossible for the system to determine this in the training phase, there is another method, the Wizard-of-Oz (WoZ), which has been widely implemented in agent research [18][3]. In this environment, the determination of the ball being thrown is made by a hidden human user. This hidden user is able to see both the target user and the environment so that when an appropriate gesture is made, the correct feedback can be given to the user (in this case a ball movement).

Next we describe three training environments that can be used for extracting one or more gestures. The objective is to force the user to use a gesture in the experiment several times. The first is the dribbling action, which is simply the user bouncing the ball on the ground with one hand. In basketball, dribbling occurs with navigation, so we create a navigation scenario where the user is simply instructed to virtually dribble the ball while using the pressure pad sensor to make their way around defined locations in the court.

In the shooting scenario, the objective is similar. In this case, once the user has made it to the target location, they must then shoot the ball towards the goal. Shooting without being at the target location is not permitted. Because there needs to be an objective for the user, the session will be completed once 10 successful goals have been scored.

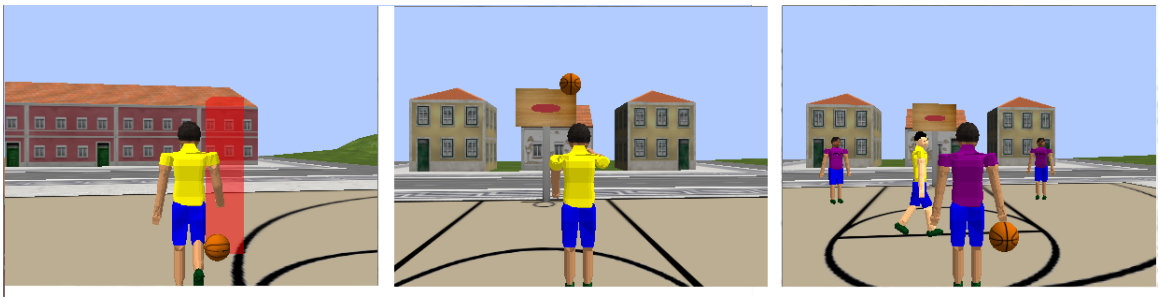

Figure 1: Screen shots of the virtual training environments to be used to gather gesture data. From left to right: dribbling, shooting and passing

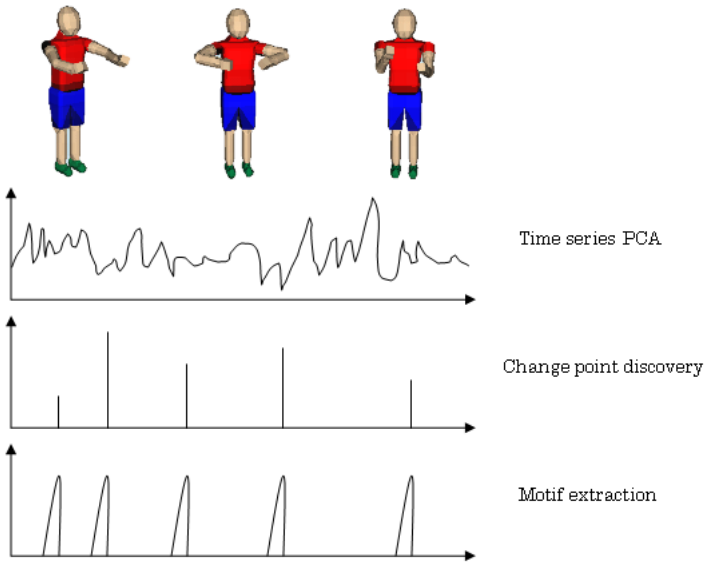

Figure 2: Process of gesture discovery and extraction. User behavior is captured and transformed to a time series PCA. Algorithms to find the change points are implemented, and these locations are used as the basis to discover the common motifs or gestures which the user executed.

Passing is arguably the most complex of the training sessions. In this scenario, the user and two agent teammates are competing against one agent opponent. The objective is to successfully pass the ball between the team of the user and two agents, while the opponent agent attempts to block the pass of the ball by standing in the way. The user and teammates are stationary, only rotating their body to pass, while the opponent agent may move. This was done to promote faster and more spontaneous behavior. The session ends once 20 consecutive passes have been made.

The final output of each training session is raw time-series data of the user's actions inside the environment. As long as the sessions contain some gesture, it is sufficient. There is no need for the data to be modified, as the process of gesture extraction will occur in the next step.

5 Process of Recognizing a Pass

In this section, we provide the details of the final system in the identification of passing by a user. The first is the extraction of the gesture in its raw form from the training environment. While this process is used for recognizing a pass, any gesture can undergo the same steps. A novel recognition algorithm is also provided.

5.1 Gathering of raw data

The user undergoes the training phase in the virtual environment and we record the motion capture data for the entire session from Kinect. In this case, the data is represented in the form of vectors, namely the bones connecting the shoulder, elbow and wrist. This data is then kept in its raw form. Essentially, we now have M time series points consisting of D dimensions. In the case of six vectors (three for each arm), the three-dimensional coordinates total to 18 dimensions.

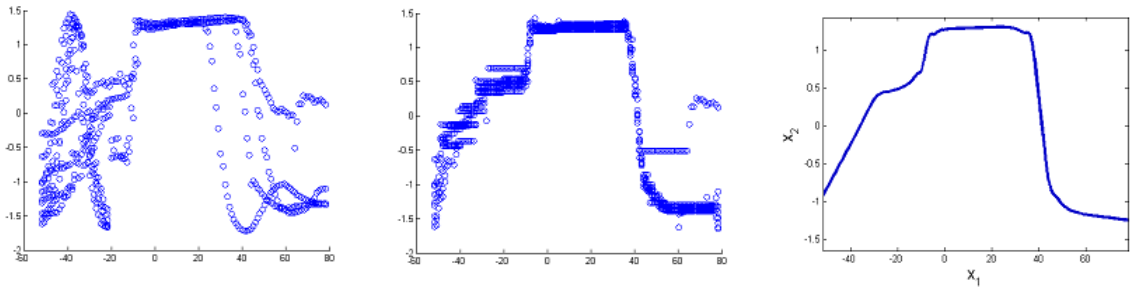

Figure 3: Construction of the gesture model using GMR. The input of data is the set of all points used in the gesture, reduced to two dimensions by PCA. After temporal alignment of the data, a regression curve can then be constructed

5.2 Gesture discovery

We must now attempt to identify the passing gestures contained within this raw time series data. Furthermore, there is no prior information of the data structure which represents a pass. To resolve this issue, an unsupervised learning technique must be used to discover these gestures. Previous work by Mohammad [12] showed that a robot could be made to learn human gestures with no previous knowledge, simply by watching them. We use this work as a basis for gesture discovery. The work in [14] also gives encouragement that gestures can be extracted from training data and this section is based on much of its findings. A summary of the steps involved is provided in Figure 2.

The first step is to discover where in the data stream a change occurs, so that there is evidence of a gesture occurring which is differentiable from other behavior. This is commonly called the change point discovery (CPD) problem. To reduce dimensionality to just one dimension, a special type of time series PCA is applied using the function in [15]. Once this has been done, CPD can be performed on the one-dimensional data.

Two main algorithms used to address this are Singular Spectrum Transform (SST) [7] and Robust SST [13], an update of the algorithm which is more resistant to noise. The output of this is a set of change scores which can be used to find change points, which are identified using Granger causality. In our passing example, there should be a change point (represented by a time point t) at every instance of a pass from the user.

Once change points have been isolated, the final step is to find the specific motifs which correspond to the passing gestures. There are several useful algorithms for this task including the Distance-Graph Constrained Motif Discovery algorithm [12], MK [17] and MK+ [14]. The output is a series of motifs, represented by B_{start}, B_{end}, which are the start and end time points of the discovered gestures. We are able to then go back to the original data and only consider the points contained in *all* common motifs. we consider N $D+1$-dimensional points from G gestures, with the total number of points T in each individual gesture g also being considered.

This process is independent of D, meaning that it is useful for a low number of discovered gestures. The motifs themselves must be verified so that they actually correspond to the target gesture. This can be achieved by recovering the raw data stream and analyzing each motif. Additionally, the nature of the training session means that there are likely few gestures which can be misidentified as passing. The probability that a discovered motif corresponds to a pass is high.

5.3 Model creation

Model creation is achieved through the GMR algorithm described above. We now give further details on this process, referring to work by Calinon in [4].

The first step is to transform all N points in the G extracted gestures to the feature space via PCA. Each associated time point t is also incorporated into the model, giving an N by $D+1$ input matrix. With the resulting feature space model, the first C principal components sufficient enough for a model can be used, though in many cases, $C = 2$. The C by $D+1$ transformation matrix A as well as the associated PCA column means are also kept, as they will be used during the recognition stage.

At this point, we now have a set of N gesture-related points in C dimensions which have been transformed by PCA. Each of these points is associated with a gesture g at a specific time point t. Note that the total number of gesture samples G can be relatively low. We also temporally align each g so that they are of a fixed length. This is achieved using the warp path found through dynamic time warping (DTW). The objective now is to create a trajectory that model all these points for use in recognition. This is where GMR becomes useful.

Firstly, a Gaussian Mixture Model (GMM) is constructed using K Gaussians through an expectation-maximization (EM) algorithm to produce a maximum likelihood estimation. This provides an iterative estimate of the prior π, mean μ and covariance Σ. That

is:

$$p(k) = \mu(k)$$
$$p(n|k) = \mathcal{N}(n, \mu k, \Sigma_k)$$

Another approach given is to use Bernoulli Mixture Modeling (BMM), with the parameters reflecting the prototype:

$$p(k) = \mu(k)$$
$$p(n|k) = \text{Bern}(n, p(k))$$

The selection of K can be based on some model selection criterion, or through direct visual analysis of the best for of the data. Once the mixture model is formulated, GMR is used to discover a regression model with associated variances to be used for comparison. Each component k is represented by a mean and covariance matrix through separating the temporal n_t, and spatial n_s values. The expected spatial value $\hat{n}_{s,k}$ and estimated covariance $\hat{\Sigma}_{s,k}$, conditioned on t for every k Gaussian is derived. Responsibilities of each Gaussian k for each time step are given through:

$$\beta_k = \frac{p(n_t|k)}{\sum_{k=1}^{K} p(n_t|k)}$$

Putting this together, the end result is a regression model where the conditional expectation and covariance of the spatial values at each time value t can be found:

$$\hat{n}_s = \sum_{k=1}^{K} \beta_k \hat{n}_{s,k}$$
$$\Sigma n_s = \sum_{k=1}^{K} \beta_k^2 \hat{\Sigma}_{s,k}$$

By integrating time into the model, we can retain a mean trajectory and variance for a particular gesture, which takes into account differing gesture styles. Each t will have an associated value. Figure 3 provides a visual summary of this process. This model was formed from a total of six samples taken from two people.

5.4 Online recognition

Given the PCA transformation matrix A and a mean trajectory model, real-time recognition of a gesture should now be feasible. Suppose that in real-time we have a stream of data consisting of T time points and D dimensions. Firstly, the data is transformed to the feature space through subtraction of the column means and then multiplication with A, which now associates each individual time point t with a point x', y' in feature space (or x'_t, y'_t).

Our first step in recognition is to determine *what* we want to recognize. From the collected samples, we identified three states: setup, release and post-pass. Of these, we are only interested in the first two. Our trajectory model allows us to visualize these states and we split these into sub-trajectories which we then use for the purposes of recognition. Each sub-trajectory model is represented by all GMR points between a start point s and end point e, where $s >= 0$, $e <= T$ and $s < e$. We define a point in the sub-trajectory model as mx_t, my_t, where $s <= t <= e$.

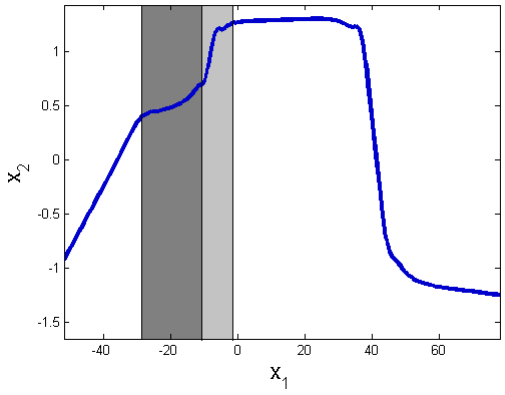

Figure 4: A passing trajectory model with two sub-trajectories. The dark and light areas indicate the setup and release sub-trajectory states respectively

Given two sub-trajectory models (setup and release), our goal is to recognize if the model corresponds to either of them. Firstly, we extract the most recent $e - s$ data points and determine the Mahalanobis distance between x'_0, y'_0 in the data stream and mx_s, my_s in the sub-trajectory model. If this is satisfied, we then use DTW to calculate the distance between all points in the data stream and the sub-trajectory model. It is possible that more than one sub-trajectory model is inside the distance threshold. In this case, we classify the gesture with the smallest amount of distance according to DTW.

A visual representation of the two sub trajectories is given in Figure 4, while pseudocode for algorithm recognition is provided in Algorithm 1.

5.5 Preliminary evaluation

We conducted a preliminary evaluation of the recognition system. For this, we observed users utilizing the system and using gestures to pass the ball. The two users from whom training samples were gathered achieved a recognition rate of approximately 80%. Two untrained users also participated and achieved a lower recognition rate of approximately 65%. It should be noted that the untrained users were given no instruction on how to perform a pass. At first, executing a pass was unsuccessful, however this improved over a time as they tested the best method with which to throw a pass. We believe that this result indicates that the training time needed to perform the gesture correctly is fairly low.

Algorithm 1 Algorithm to recognize gestures in GMR model

```
{load whole model}
GMR ← entire model found through GMR

{define sub-trajectories through start and end points in GMR}
Setup ← GMRModel(25, 30)
Release ← GMRModel(35, 45)
Gestures ← add(Setup, Release)


{online data collection}
data ← Kinect data stream
threshold ← 5

{find all gestures where the start point is close to start point of the data}
candidates ← empty set
for all g in Gestures do
  if distance(g(x_0, y_0), data(x_0, y_0)) < threshold then
    add to candidates
  end if
end for

{check all candidates - choose the one with minimum DTW distance}
for all c in candidates do
  DTWValue ← DTW Distance(data, sub-trajectory)
  if lowest DTW value then
    c is recognized gesture
  end if
end for
```

Additionally, contextual information was included as a means to classify the gesture. We found there were edge cases between the setup and release phases, which would produce false positives in certain poses. These poses could be identified during the game and were dealt with so they produced no feedback.

6 Issues and Future Work

We have described the implementation of a procedure to extract natural gestures from users in the virtual environment and then using them to form a model, based on a limited number of training samples. There are some issues which must be considered.

User behavior can be affected by their cognitive load. In the environment, there are two major sensors - the Kinect body recognition system and the pressure sensor used for navigation. If a user's attention is driven towards manipulating these sensors, the naturalness of their behavior could be affected along with the gestures they produce. We have tried to curb this effect by using non-wearable sensors to allow freedom of movement as well as preserving a user's natural motion.

In this work, we can easily determine the appropriate sub-trajectories through GMR model visualization. However, there may be instances where this is not so apparent. In this case, we can only speculate about what are ideal start and end points of the trajectories. There are several other parameters which must be tweaked in order to produce a robust system, including the threshold distance for comparison of GMR points. In our preliminary evaluation, we found edge cases which discriminated towards the release sub-trajectory. While the poses causing these edge cases could be identified with passing, we anticipate this will become more complex with the recognition of shooting gestures.

Our future work involves analyzing communicative gestures using the same process. By doing this, we hope that there are some underlying features associated with a communicative act, *regardless of the gesture performed*. As an example, the features of a gesture executed by users to request a pass might be smooth arm movements with a large spatial range. Although the actual gestures may differ considerably, the model should be able to capture these features and apply it to *any* movements of that type.

Additionally, contextual information should also be considered, particularly in the case of gestures with no fixed meaning. In this work, we only consider manipulation tasks because the context of the action is fairly constrained. Therefore, only using body movements as dimensions is sufficient. On the other hand, communicative acts require some agent knowledge of the context of the situation. If a user makes a movement directed at the ball handler with nobody marking them, it is likely that they are requesting the ball. Contextual knowledge such as this potentially allows agents to exhibit some form of higher level reasoning.

7 Conclusion

This work has proposed a process of extracting natural gesture in a virtual basketball environment and then using these to produce a gesture model which can be used for real-time recognition. The recognition system requires relatively few training samples and is independent of the number of dimensions, making it ideal for situations where samples are not plentiful. The environment is unique in that it is a dynamic game, with an assumption that gestures will vary both according to the user. We are currently implementing this technique with ball-manipulation tasks to set the foundation for a playable game. If this is successful, the next step is to use the same process to discover features of communication from the user towards intelligent agents. If these features can be identified by the agents, there is potential to create powerful agents which can infer the meaning of a gesture from both its execution and context.

References

[1] Bailly, G., Raidt, S., Elisei, F.: Gaze, conversational agents and face-to-face communication, *Speech Communication*, Vol. 52, No. 6, pp. 598–612 (2010)

[2] Bevilacqua, F., Zamborlin, B., Sypniewski, A., Schnell, N., Guedy, F., Rasamimanana, N.: Continuous Realtime Gesture Following and Recognition, *Lecture Notes in Computer Science*, Vol. 5934, pp. 73–84 (2010)

[3] Brown, E., Barrett, N.: A Wizard-of-Oz platform for embodied conversational agents, *Computer Animation and Virtual Worlds*, Vol. 17, pp. 249–257 (2006)

[4] Calinon, S., Guenter, F., Billard, A.: On Learning, Representing and Generalizing a Task in a Humanoid Robot, *IEEE Transactions on Systems, Man and Cybernetics, Part B, Special issue on robot learning by observation, demonstration and imitation*, Vol. 37, No. 2, pp. 286–298 (2007)

[5] Corradini, A., Cohen, P.: On the Relationships Among Speech, Gestures, and Object Manipulation in Virtual Environments: Initial Evidence, *Advances in Natural Multimodal Dialogue Systems: Text, Speech and Language Technology*, Vol. 30, pp. 97–112 (2005)

[6] Ekman, P., Friesen,W.V.: The Repertoire of Nonverbal Behavior: Categories, Origins, Usage, and Coding, *Semiotica*, Vol. 1, No. 1, pp. 49–98 (1969)

[7] Ide, T., Inoue, K.: Knowledge discovery from heterogeneous dynamic systems using change-point correlations *Proc. SIAM Intl. Conf. Data Mining*, (2005)

[8] Kim, D, Song, J., Kim, D.: Simultaneous gesture segmentation and recognition based on forward spotting accumulative HMMs, *Journal of Pattern Recognition*, Vol. 40, No. 11, pp. 3012–3026 (2007)

[9] Kistler, F., Endrass, B., Damian, I., Dang, C.T., Andre, E.: Natural interaction with culturally adaptive virtual characters, *Journal on Multimodal User Interfaces*, Vol. 6, No. 1, pp. 39–47 (2012)

[10] Lala, D., Nishida, T.: Joint activity theory as a framework for natural body expression in autonomous agents, *Proceedings of the 1st International Workshop on Multimodal Learning Analytics*, pp. 2:1–2:8 (2012)

[11] Lala, D.: *VISIE: A Spatially Immersive Environment for Capturing and Analyzing Body Expression in Virtual Worlds*, Master's Thesis, Kyoto University (2012)

[12] Mohammad Y., Nishida, T., Okada, S.: Unsupervised simultaneous learning of gestures, actions and their associations for human-robot interaction, *Proceedings of the 2009 IEEE/RSJ international conference on n robots and systems*, pp. 2537–2544 (2009)

[13] Mohammad Y., Nishida, T.: Robust Singular Spectrum Transform, *Proceedings of the 22nd International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems: Next-Generation Applied Intelligence*, pp. 123–132 (2009)

[14] Mohammad Y., Nishida, T.: Unsupervised discovery of basic human actions from activity recording datasets, *IEEE/SICE SII 2012*, pp. 402–409 (2012)

[15] Mohammad Y., Nishida, T.: CPMD: A MATLAB Toolbox for Change Point and Constraint Motif Discovery, 25th International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems (IEA/AIE) 2012, (2012)

[16] Morency, L-P., Quattoni, A., Darrell, T.: Latent-Dynamic Discriminative Models for Continuous Gesture Recognition, *CPVR 2007*, pp. 1–8 (2007)

[17] Mueen, A., Keogh, E., Zhu, Q., Cash, S.: Exact discovery of time series motifs, *Proc. of 2009 SIAM, 2009*, (2009)

[18] Xu, Y., Ueda, K., Komatsu, T., Okadome T., Hattori, T., Sumi, Y., Nishida, T.: WOZ Experiments for Understanding Mutual Adaptation, *AI & Society*, Vol. 3, No. 2, pp. 201–212 (2009)