



A Hierarchical Model for Continuous Gesture Recognition Using Kinect

Jensen, Søren Kejser; Moesgaard, Christoffer; Nielsen, Christoffer Samuel ; Viesmose, Sine Lyhne

Published in:
Twelfth Scandinavian Conference on Artificial Intelligence

DOI (link to publication from Publisher):
[10.3233/978-1-61499-330-8-145](https://doi.org/10.3233/978-1-61499-330-8-145)

Publication date:
2013

Document Version
Early version, also known as pre-print

[Link to publication from Aalborg University](#)

Citation for published version (APA):
Jensen, S. K., Moesgaard, C., Nielsen, C. S., & Viesmose, S. L. (2013). A Hierarchical Model for Continuous Gesture Recognition Using Kinect. In M. Jaeger, T. D. Nielsen, & P. Viappiani (Eds.), *Twelfth Scandinavian Conference on Artificial Intelligence* (Vol. 257, pp. 145-154). IOS Press. <https://doi.org/10.3233/978-1-61499-330-8-145>

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal -

Take down policy

If you believe that this document breaches copyright please contact us at vbn@aub.aau.dk providing details, and we will remove access to the work immediately and investigate your claim.

A Hierarchical Model for Continuous Gesture Recognition Using Kinect

Søren Kejser JENSEN^a, Christoffer MOESGAARD^a Christoffer Samuel NIELSEN^a
Sine Lyhne VIESMOSE^a

^a*Department Of Computer Science Aalborg University*

Abstract. Human gesture recognition is an area, which has been studied thoroughly in recent years, and close to 100% recognition rates in restricted environments have been achieved, often either with single separated gestures in the input stream, or with computationally intensive systems. The results are unfortunately not as striking, when it comes to a continuous stream of gestures. In this paper we introduce a hierarchical system for gesture recognition for use in a gaming setting, with a continuous stream of data. Layer 1 is based on Nearest Neighbor Search and layer 2 uses Hidden Markov Models. The system uses features that are computed from Microsoft Kinect skeletons. We propose a new set of features, the relative angles of the limbs from Kinect's axes to use in NNS. The new features show a 10 percent point increase in precision when compared with features from previously published results. We also propose a way of attributing recognised gestures with a force attribute, for use in gaming. The recognition rate in layer 1 is 68.2%, with an even higher rate for simple gestures. Layer 2 reduces the noise and has a average recognition rate of 85.1%. When some simple constraints are added we reach a precision of 90.5% with a recall of 91.4%.

Keywords. Human gesture recognition, Hierarchical models, Nearest Neighbour Search, Hidden Markov Models, Real-time, Microsoft Kinect, Machine learning applications.

1. Introduction

Much of the research done in the area of gesture recognition focuses on the tracking of human skeletons and joints. Since the release of Kinect in 2010 and its complementary SDK in 2011, human tracking is no longer a challenging endeavour. The challenge is now to use this information and devise methods which can be easily adapted and extended with new gestures and work in an environment with a continuous feed of data.

An in-depth overview of previous work in gesture recognition is beyond the scope of this paper, but we list the most relevant past methods here. A thorough review has been performed by Aggerwal and Ryoo [1]. Layered hidden Markov models, presented by Oliver et al. [2], uses Hidden Markov Models (HMMs) on the first level to recognise atomic gestures and HMMs on the second level to recognise the high-level gestures using the recognised atomic gestures as input. A problem for this approach is that the gestures need to be sequential and are first recognised after the end of the gesture. Recognition of high-level gestures consisting of concurrent atomic gestures has also shown to be difficult. The types of features to use in classification of gestures have also been an active

area of research. Campbell et al. have shown a large difference in the precision, when different features are used. They also show that a combination of positional features and features indicating velocity, performs superiorly to positional features alone [3]. The NNS method proposed by Lai et al. [4] performs equally well using an Euclidean and a more complex but computationally simple distance metric using the Frobenius norm.

We propose a system, which uses Nearest Neighbour Search (NNS) and HMMs to recognize gestures on a continuous stream of data in real-time with no latency. We also propose a technique to determine the force of the gesture. Microsoft's recently released new Kinect for Xbox One has a similar feature, but so far no work has been published on the matter, and to our knowledge no other research has been done in the area of measuring the force with which a gesture has been performed. The approach is generalised across different gestures and is therefore extensible. To our knowledge, not much research has been done regarding recognising a continuous stream of gestures in a real-time environment, so that is a major focus of our system. Detecting gestures in a continuous stream is significantly different and provides additional challenges, compared to detecting a gesture in a bounded window of time. Lai et al. [4] presents a method for continuous gesture recognition, where they have restricted the gestures, which shows correct classification rates close to 100%. Their gestures must be at least one second in length and there must be one second between each gesture where no gestures are performed. They classify one gesture for the collection of sliding windows that represents a gesture. In this paper we present a solution which overcomes these restrictions and recognizes the gestures before they are finished. Some noise is to be expected since some sliding windows will not contain a clear part of a gesture. The end result is a solution, which is able to interface with a game and the decisions about the solution reflects that.

The recognition of gestures is split into two parts, the lower and upper body. This makes it possible to identify two concurrent gestures as long as they appear in different body parts. An example of a possible combination is running and punching, while punching and waving is not, without making a separate gesture for the combination. As an attempt to maximise precision with the NNS, experiments have been made with a type of feature that to our knowledge has not been used for gesture recognition before; the angle between a limb, and one of the three axes in the internal Kinect coordinate system.

2. Data Representation

The data representations in our system are shown in Figure 1. Raw Kinect data are converted into positions of skeleton joints by the Kinect SDK, and are stored in a sliding window. Feature vectors are computed from the joint positions. Layer 1 of the system uses these feature vectors as the reference set for NNS and returns atomic gestures. Layer 2 combines atomic gestures into whole gestures using HMMs and returns gestures attributed with force.

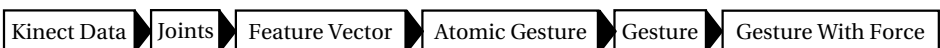


Figure 1. Data transformations performed in the system.

2.1. Kinect Data

Raw data from Kinect consists of RGB-video, depth video and sound. Since only the coordinates of the skeleton joints are used, the coordinates calculated by the SDK are stored in a CSV format for training. The skeleton joints can be seen in Microsoft's documentation [5]. The Kinect SDK performs noise reduction and jitter removal from the data, before we store it. It uses double exponential smoothing, and filters the input for large spikes based on a threshold [6]. Evaluation of the precision and robustness of the Kinect's skeletonization algorithm has been done by Obdržálek et al. [7], and shows that positional errors made by the algorithm typically is about five centimeters.

2.2. Features

Using the Cartesian coordinates directly as a feature vector for a NNS would result in a low precision, as these coordinates correspond to the player's position relative to the position of the Kinect. Instead a number of different features are calculated from the coordinates:

Joint angle: The angle between two limbs connected by a joint. The angles are computed for the wrist, elbow and shoulder for each arm, and the hip, knee and ankle for each leg. The use of this type of feature is inspired by work in Sheng [8].

Spine distance: The distance from a certain joint to the spine joint. Inspired by work in Lai et al. [4], the distance is normalized in regards to the distance between a player's *shoulder center* and *spine* joint to recognize players of different heights. It is computed for all joints, except the spine and shoulder center joints.

Velocity: Two different types of velocities are calculated, joint angle velocity and spine distance velocity. They are defined as the difference between the first and last frame of the sliding window divided by the number of frames in the sliding window. Calculating the velocity is based on work done by Campbell et al. [3]. The velocities are computed for the same set of joints as their corresponding feature.

XYZ-angle: The angle between a certain limb, and either the x, y or z-axis. This feature is an attempt to determine if the player's rotation in relation to the internal coordinate system, can be used to effectively classify gestures.

The range of every feature is normalized to the range [0,1], after outliers that lie outside three standard deviations are removed. The system is capable of computing 84 different features. The total dimensionality of the feature vectors created is dependent on the size of the sliding window. Distance and angle based features are computed for each frame, while velocity based features require a start and end position and are therefore only computed once per sliding window.

2.3. Gestures

Two types of gestures were chosen to test the system, continuous gestures and non-continuous gestures. Continuous gestures are gestures which are performed continuously over time. These gestures are Stand, Walk, Run, Sprint, Pause and Exit. The non-continuous gestures chosen are Punch, Kick, Jump, Turn Left, Turn Right, and Turn Around. Most gestures are self explanatory. Pause is performed by holding both arms vertically in parallel, and Exit is performed by holding both arms in a cross. Turns to

both left and right are performed by turning the body to the corresponding side and back again. Turn Around consists of a complete rotation, either to the left or to the right.

Non-continuous gestures are divided into atomic gestures. An atomic gesture is the start or end part of a gesture, except for Turn Around which consists of a turn start, a turn back, and an opposite turn end. Atomic gestures are needed to separate overlapping gestures such as Turn Left and Turn Around. When expanding the system with new gestures, existing atomic gestures can also be reused as part of new gestures.

In order to add another dimension to gesture detection in games, force is computed for the gestures where it is natural, such as Punch and Kick. The force is measured as the distance between the z-coordinate of the joint nearest the Kinect in the last sliding window, and the z-coordinate of joint when it was farthest from the Kinect, divided by the time between them measured in frames. This measure of force can also be used on other types of gestures where one of the other axes are used.

Force is also computed for continuous movement gestures using another method. Each type of movement gesture has a static force value. When a movement gesture is recognised, the average value of the last x gestures is computed, where x is the number of former gestures used, to determine an approximated velocity. The value of x should be kept so low that that result is affected by a change in movement as fast as possible while still being resistant against outliers in the stream of gestures detected.

3. System Description

The system consists of three major components organised as shown in Figure 2.

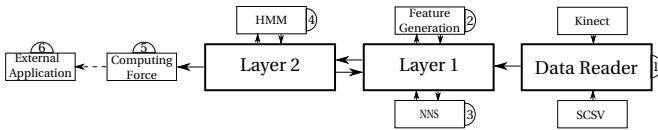


Figure 2. The overall structure of our system. Numbers indicate data flow at runtime.

Data reader: The reader ensures that the input data is uniformly presented to the rest of the library despite the data's origin, which is either a CSV file or Kinect. It uses the Kinect SDK's built in smoothing parameters to clean the data when Kinect data is used. The parameters recommended by Microsoft for games are used, although without prediction of a joint's location in future frames, so only actual recordings are used. Using these settings should keep latency to a minimum while still reducing jitter in the data produced [6].

Layer 1: Layer 1 receives frames from layer 2 and maintains an internal buffer in the form of a sliding window. The frames in the sliding window are then converted into a feature vector, which is compared to a reference set using NNS. The result of the search is then returned to layer 2 in the form of atomic gestures for both the upper and lower part of the body.

Layer 2: Layer 2 receives input from the data reader and redirects it to layer 1. Layer 2 is also responsible for receiving recognized atomic gestures from layer 1. To reduce the noise in the output from layer 1 we use HMMs. Computation of force is performed before sending the complete recognized gesture with force to the user of the library. Training of the HMMs is also handled by a class in layer 2.

3.1. Feature Selection

Feature selection is performed to select a subset of the features that maximizes precision. The number of possible subsets of a set of features is too large a set to do an exhaustive search. Therefore we have used the simple and fast Sequential Forward Selection (SFS) method proposed by Whitney [9], and the extension Sequential Forward Floating Selection (SFFS) proposed later by Pudil et al., which has been shown to produce close to optimal solutions [10]. In SFS the best single feature is selected first, then the best pair that includes the single feature and so forth. As such it guarantees to terminate after n steps, where n is defined by $n = \sum_{i=1}^D D - i + 1$, where D is a set of features. To determine the best feature in each step, k -fold cross-validation is used. In our experiments a k between 10 and 40 has been used. SFFS starts out as SFS but after each step it can also remove a feature, if doing so does not decrease precision. SFFS in that way prunes features that were selected earlier but have become irrelevant due to feature redundancy. While there is no guaranteed number of steps to termination for SFFS, it has been shown by Pudil et al. to have much faster execution time than finding an optimal subset using the often used branch and bound algorithm for feature selection [10]. Both methods are implemented and SFS has been used for tuning parameters due to its shorter running time, while SFFS has been used for finding the final features used in the system.

3.2. Nearest Neighbour Search

NNS is well suited for classification problems with a smaller amount of data, since the running time increases with the amount of data. In our case, data generation needs to be done manually so the resulting size of the data set is manageable for NNS. The NNS algorithm used is based on binary space partition trees instantiated to a k -d tree [11], which could be a bottleneck if a reference set with high dimensionality is used. In such a case an algorithm based on locality-sensitive hashing (LSH) can be more suitable.

Evaluation of Euclidean distance and the Manhattan distance showed that Euclidean distance provided a slight increase in both precision and performance compared to Manhattan distance, therefore Euclidean distance is used as our distance metric.

The reference set for the search is separated into a lower and upper body, and is represented as feature vectors. K -nearest-neighbours are computed for each body part. If one neighbour is requested, the atomic gesture indicated by the neighbour is returned. When multiple neighbours are requested the most represented type is returned. Ties are handled by random selection as this has been shown to provide the best results [12].

3.3. Constraints

Checks to ensure that the gestures detected by the system are valid are also performed in layer 2. The checks are as following: To recognise a Turn gesture, both the upper and lower part of the body needs to be turning. To recognise a gesture that consists of both a start and an end atomic gesture, both gestures needs to occur. The same gesture needs to be recognised three frames in a row. In an effort to recognise a gesture only once when it is performed, a time span is required between gestures. The length of the time span is a tuning parameter. Another consideration is whether this constraint should apply per gesture or for the whole body part so one gesture cannot be sent directly after another, even if the gesture is different. This is also a tuning parameter. If it is applied per gesture it is possible to use different values for different gestures.

3.4. Hidden Markov Models

Each HMM is trained using examples of gestures where each observation is the output of the NNS given an example. The NNS classification was done using the best features and tuning parameters found for the NNS using feature selection and parameter tuning. Training is done using the Baum-Welch algorithm [13], to find a local maximum for the model parameters. A simple HMM with only one hidden state and a uniform distribution of emission probabilities is used. Models with more states were tried but showed no increase in precision. The HMMs are used to connect the atomic gestures and reduce the noise. In layer 2 a sliding window of atomic gestures is used to classify the gesture. An HMM is constructed for each gesture, and the Viterbi algorithm [14] is run with a sliding window of atomic gestures as input. The gesture corresponding to the HMM with the highest score computed by the algorithm, is chosen as the classified gesture for the sliding window. The HMM for each body part that represents standing also represents doing no gesture and gestures that are not used in the system. Therefore it also acts as a HMM that catches noise and classifies it as no gesture.

4. Evaluation

For the purposes of testing, the system has been implemented to recognise the gestures described in Section 2.3, performed in natural form. The system runs in real-time on an average computer¹ with no latency. An attempt has been made to condense and edit the reference set using Hart's Condensing [15] and Wilson's editing [16], which resulted in a lower precision when testing. The system runs real-time without condensing and editing, so condensing and editing of the reference set has been omitted.

For NNS, a reference set of feature vectors is generated from a tagged set of data. The process of separating the recorded data into gestures is performed manually and by multiple persons, so some deviations in how a gesture is tagged are inevitable. In order to compensate for inconsistent tagging, manual filtering of the reference set is performed afterwards in order to remove files that had been tagged incorrectly.

Non-continuous gestures, such as Punch and Kick, are separated into fitting atomic gestures. Continuous gestures such as Run and Stand are kept as one. The dataset used for reference set consists of gestures from three different persons. It consists of 1280 atomic examples. Another data set from the same persons is used to test performance of layer 1 and tune layer 2. A data set from an, to the system, unknown person consisting of 393 atomic examples is used to test performance of both layer 1 and layer 2.

4.1. Parameter tuning

Parameters used for evaluation of the system are chosen through non-exhaustive grid search [17]. Seven parameters can be tuned in the system, three used by NNS and four used by layer 2. The parameters for NNS were tuned by performing feature selection, using SFS, and choosing the parameters with the highest precision. Feature selection was performed on the reference set described in Section 4. The parameters for layer 2 were tuned afterwards by measuring the accuracy of the output. The result of tuning i.e. the values used for the final evaluation are shown in Table 1.

¹Intel® Core™ 2 Quad Q9400, 8 GB RAM

Table 1. Value of tuning parameters used for testing.

Layer 1 Parameters	Value	Layer 2 Parameters	Value
Length of HMMs	10	Frames between gestures with force	Average length of training files
Number of neighbours	1		
Number of sliding windows	3	If frames between force gestures are per gesture or for all gestures	It is a tradeoff
Size of sliding windows	10		
Movement buffer	20		

The gestures are of very different length, and a fixed number of frames between gestures with force proved not to be successful. Instead the values are set to the average number of frames the given gestures consist of in the training data. Whether this time constraint should apply for all new gestures on the same body part or only new gestures of the same gesture type is a tradeoff between precision and recall. The movement buffer is a direct tradeoff between how fast the system should react to movement, and how resistant it should be to short movement changes. As no value supports every application, the value of this parameter is left for the user to decide.

4.2. Evaluation of layer 1

To determine the efficiency of the proposed new type of feature, XYZ-angles, a number of tests have been performed. The tests consisted of running SFS and SFFS with two sets of features: A set of all 84 feature types, and a set of 60 feature types, consisting of the full set excluding the XYZ-angles. The tests were done with the reference set described in Section 4, containing atomic gestures for the upper and the lower body respectively.

The results of the experiments with SFS and SFFS on the upper and lower body can be seen in Figure 3. The SFFS runs were stopped when the precision started to decline. The most notable observation is that the precision is generally 10 percent points

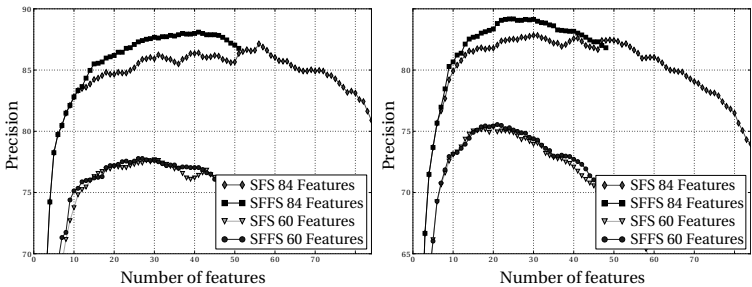


Figure 3. The results of SFS and SFFS on the gestures of the upper body (left) and the lower body (right). Created using matplotlib [18]

better when performing SFS with a feature set including XYZ-angles, than without. This difference is further illustrated in the measurements for SFFS. A set of 24 features was chosen for the final system.

To measure the efficiency of our solution to detect atomic gestures, we have created confusion matrices for the classification rates for each gesture. The results are based on the recorded data from an individual not in the reference set. The tendencies are the same as the confusion matrices of layer 2 evaluation seen in Table 2 and they are therefore not shown. The average correct-classification rate (CCR) is 69.3% for the lower body

and 66.8% for the upper body. Run, Walk and Sprint generally suffers lower precision than the average CCR, but it can be observed that the majority of the misclassifications lie within the set of these three gestures. This can be attributed to the fact that there is a certain inconsistency in the performance of these three gestures between the participants. This irregularity is to be expected given the similarity of the gestures. Furthermore the start and end of turns are more often confused than the other atomic gestures. This is due to the fact that a person doing the gesture tends to stay in the outermost position of the gesture for several sliding windows, unlike the behaviour during e.g. Jump.

4.3. Evaluation of layer 2

Table 2. Confusion matrix with HMM classification rates of gestures from the lower and upper body part in percent. Avg. CCR = 83.9% for the lower and Avg. CCR = 86.6% for the upper body part.

	Kick	Jump	TurnLeft	TurnRight	TurnAround	Stand	Walk	Run	Sprint		Punch	TurnLeft	TurnRight	TurnAround	Pause	Exit	Stand
Kick	88	1	3	0	1	0	0	2	5								
Jump	2	88	0	0	0	0	0	0	10	Punch	81	4	0	0	0	0	14
TurnLeft	0	1	97	0	0	2	0	0	0	TurnLeft	1	92	0	3	0	1	3
TurnRight	1	0	0	97	1	0	0	0	0	TurnRight	0	0	97	1	0	0	1
TurnAround	0	0	21	22	55	0	1	1	0	TurnAround	0	23	21	53	0	0	4
Stand	0	0	0	0	0	99	1	0	0	Pause	0	0	0	0	100	0	0
Walk	0	0	0	1	0	0	91	5	4	Exit	0	0	0	0	2	98	0
Run	0	0	0	0	0	0	39	43	18	Stand	0	11	3	1	0	0	85
Sprint	1	1	0	1	0	0	0	0	97								

To measure the effect of using HMMs, additional confusion matrices have been created, showing the classification rates produced by the HMMs. The confusion matrices are based on a data set with 197 examples. The matrices can be seen in Table 2. The average CCR has increased considerably to 83.9% for the lower body and 86.6% for the upper body. The remaining gestures with a low CCR are mainly TurnAround and Run. TurnAround can be explained by the fact that it consists of a TurnLeft or TurnRight in the end and beginning, and therefore gets misclassified as those. The Run gesture is often confused with either Sprint or Walk and it can simply be explained by a change in speed.

Table 3. Table with recall and precision of gestures with force with constraints applied.

	Required time span between gestures restricts only the gesture		Required time between gestures span restricts all gestures	
	Recall	Precision	Recall	Precision
Jump	76.9%	90.1%	76.9%	81.5%
Kick	100.0%	100.0%	100.0%	100.0%
Punch	76.0%	100.0%	76.0%	90.5%
TurnL	100.0%	70.8%	100.0%	95.7%
TurnR	95.7%	81.5%	95.7%	84.6%
TurnA	100.0%	100.0%	80.0%	100.0%

To evaluate the final results, the classification precision and recall have been measured. The results can be seen in Table 3. Depending on the time span constraint, the average precision and recall are respectively 91.4% and 90.5%, and 88.1% and 95.2%. The

continuous gestures are excluded in this test because they are recognised continuously, and therefore the precision in Table 2 can be used. Punch is the gesture which the system has the hardest time recognising. Some of the punches are too fast and when it already is the shortest gesture, it results in too few sliding windows. The problem could have been reduced if the tagging of the data had been less narrow and included more time before and after the gesture. It is less of a problem when using the system on continuous data.

The force component was evaluated using new recordings of punches and kicks, featuring a set of weak gestures and strong gestures. Force for the gestures was recorded and averages were computed. The results show an increase in the force computed of 89.1% for punches and 84.0% for kicks, thus verifying that the system is capable of computing force corresponding to the effort used by the tracked person.

5. Conclusion

We have presented a hierarchical system for gesture recognition based on Nearest Neighbour Search (NNS) and Hidden Markov Models (HMMs). The system performs gesture recognition on a continuous stream of data in real-time and approximates the force with which a gesture is performed. The system recognises gestures based on a stream of atomic gestures for the lower and the upper body, and calculates the force of the gestures.

Tests show that an average correct classification rate of 69.3% and 66.8% can be achieved for the lower and upper body respectively by the NNS alone. Further evaluation of the system shows that the noise reduction provided by the HMMs in layer 2 increased the CCR to 83.9% and 86.6% for the lower and upper body. After the constraints are used the precision is 90.5% and recall 91.4%. Comparison of these results with the work done by Lai et al. [4] shows that a much lower classification rate is achieved using only NNS due to classifying gestures continuously and with no latency, but a comparable classification rate can be achieved using HMMs and constraints while still performing recognition in real-time.

Use of a new type of feature for NNS, based on the angle between a certain limb of the recognized skeleton and one of the three axes of the internal Kinect coordinate system, shows an increase in precision of up to 10 percent points, compared with features from previously published results. To improve the precision, and also make our system more versatile, recordings of a more diverse set of people, without as much practice performing the gestures could be used. The current data is recorded from only three individuals, who all were knowledgeable about the inner workings of the system. The performed gestures were well known and performed extensively with less deviation as a result. Performing condensing and editing of the reference set showed no increase in precision when used in conjunction with NNS. Similar algorithms might be used to exclude parts of the reference set which causes misclassifications by the HMM. Another way to improve the data used is to make our system actively perform adjustments to the recorded data. It could be done through user calibration, or automatically on runtime, and could help to reduce the bias towards peculiarities not found in the reference set.

Future work in normalising features to different intervals based on their effectiveness, thus effectively changing the distance metric could be conducted in order to determine, if precision can be improved by relying more on best performing features. Experimenting with validation of gestures in layer 2 of the system shows promising results and

further development could increase precision and reduce noise even more. Some gestures such as Sprint creates movement in both upper and lower body, using information from both parts as input for the HMMs should allow for better accuracy.

Devising exact, quantitative tests to determine the performance of the system on a continuous stream of data is relatively difficult. Responsiveness is based on the opinion of the individual using the system, but during evaluation the system has been used actively with no noticeable delay, maintaining the precision detailed in Section 4. A relevant test is how it performs in an actual gaming situation, and how responsive and precise the system feels when using it. To determine this, a usability test of the system is needed. Both precision and performance of the system should be evaluated in this context.

Acknowledgments: We would like to thank Bo Thiesson for supervising the project and providing valuable feedback.

References

- [1] J. K. Aggarwal and M. S. Ryoo. Human Activity Analysis: A Review. *ACM Computing Surveys (CSUR)*, 43(3):16, 2011.
- [2] N. Oliver, E. Horvitz, and A. Garg. Layered Representations for Human Activity Recognition. In *Multimodal Interfaces, 2002. Proceedings. Fourth IEEE International Conference on*, pages 3–8. IEEE.
- [3] L. W. Campbell, D. A. Becker, A. Azarbayejani, A. F. Bobick, and A. Pentland. Invariant Features for 3-D Gesture Recognition. *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, pages 157–162, 1996.
- [4] K. Lai, J. Konrad, and P. Ishwar. A Gesture-Driven Computer Interface Using Kinect. In *Image Analysis and Interpretation (SSIAI), 2012 IEEE Southwest Symposium on*, pages 185–188, 2012.
- [5] Microsoft. JointType Enumeration. <http://msdn.microsoft.com/en-us/library/microsoft.kinect.jointtype.aspx>, 2013.
- [6] Microsoft. Joint filtering. <http://msdn.microsoft.com/en-us/library/jj131024.aspx>, 2013.
- [7] Š. Obdržálek, G. Kurillo, F. Ofli, R. Bajcsy, E. Seto, H. Jimison, and M. Pavel. Accuracy and Robustness of Kinect Pose Estimation in the Context of Coaching of Elderly Population. *Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE*, 2012:1188–93, 2012.
- [8] J. Sheng. A Study of AdaBoost In 3D Gesture Recognition. *Department of Computer Science, University of Toronto*, 2003.
- [9] A. W. Whitney. A Direct Method of Nonparametric Measurement Selection. *Computers, IEEE Transactions on*, 100(9):1100–1103, 1971.
- [10] P. Pudil, J. Novovičová, and J. Kittler. Floating Search Methods in Feature Selection. *Pattern Recognition Letters*, 15(11):1119–1125, 1994.
- [11] FASTLab. MLPACK Documentation. <http://www.mlpack.org/doxygen.php>, 2013.
- [12] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer, 2 edition, 2009.
- [13] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains. *The Annals of Mathematical Statistics*, 41(1): 164–171, 1970.
- [14] G. D. Forney Jr. The Viterbi Algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.
- [15] P. E. Hart. The Condensed Nearest Neighbour Rule. *IEEE Transactions on Information theory*, 1968.
- [16] D. L. Wilson. Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. *Systems, Man and Cybernetics, IEEE Transactions on*, (3):408–421, 1972.
- [17] J. Bergstra and Y. Bengio. Random Search for Hyper-Parameter Optimization. *J. Mach. Learn. Res.*, 13:281–305, March 2012. ISSN 1532-4435.
- [18] J. D. Hunter. Matplotlib: A 2D graphics environment. *Computing In Science & Engineering*, 9(3): 90–95, 2007.