<table>
<thead>
<tr>
<th><strong>Dissertation Title:</strong></th>
<th>Gesture Recognition from Depth Data, for Human-Robot interaction</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Surname:</strong></td>
<td>Guy</td>
</tr>
<tr>
<td><strong>First Name:</strong></td>
<td>Ben</td>
</tr>
<tr>
<td><strong>Student Number:</strong></td>
<td>15057674</td>
</tr>
<tr>
<td><strong>Supervisor:</strong></td>
<td>Prof Fabio Cuzzolin</td>
</tr>
<tr>
<td><strong>Dissertation Module:</strong></td>
<td>P00998</td>
</tr>
<tr>
<td><strong>Course Title:</strong></td>
<td>MSc in Computer Vision</td>
</tr>
<tr>
<td><strong>Date Submitted:</strong></td>
<td>30/9/16</td>
</tr>
</tbody>
</table>
**Statement of originality**

Except for those parts in which it is explicitly stated to the contrary, this project is my own work. It has not been submitted for any degree at this or any other academic or professional institution.

.............................................................. ..............................................................
Signature of Author                                      Date

**Regulations Governing the Deposit and Use of Master of Science Dissertations in the Department of Computing and Communication Technologies, Oxford Brookes University.**

1. The "top" copies of dissertation final reports submitted in fulfilment of Master of Science course requirements shall normally be kept by the Department.

2. The author shall sign a declaration agreeing that, at the supervisor’s discretion, the dissertation may be submitted in electronic form to any plagiarism checking service or tool.

3. The author shall sign a declaration agreeing that the dissertation be available for reading and copying in any form at the discretion of either the dissertation supervisor or in their absence the Head of Postgraduate Programmes, in accordance with 5 below.

4. The project supervisor shall safeguard the interests of the author by requiring persons who consult the dissertation to sign a declaration acknowledging the author’s copyright.

5. Permission for anyone other than the author to reproduce in any form or photocopy any part of the dissertation must be obtained from the project supervisor, or in their absence the head of Postgraduate Programmes, who will give his/her permission for such reproduction only to the extent which he/she considers to be fair and reasonable.

I agree that this dissertation may be submitted in electronic form to any plagiarism checking service or tool at the discretion of my project supervisor in accordance with regulation 2 above.

I agree that this dissertation may be available for reading and photocopying at the discretion of my project supervisor or the Head of Postgraduate Programmes in accordance with regulation 5 above.

.............................................................. ..............................................................
Signature of Author                                      Date
Gesture Recognition using depth cameras, for Human-Robot interaction

P00998

Ben Guy
15057674

Prof Fabio Cuzzolin

Submitted:
30/09/16
Acknowledgements

I would like to thank Prof Fabio Cuzzolin, and PhD student Gurkirt Singh, for their invaluable guidance throughout the course of my dissertation. Without their help and expertise the results of this project would have been much more difficult to achieve.

I would also like to thank Vangos Pterneas [18] for his excellent Kinect 2.0 tutorials, which helped to kick start my project, and Cesar Souza [19] for his online guides on using the Accord machine learning toolbox for C#.

Abstract

The main aim of this project is to “Produce a system which utilises depth information [9,10] from a Kinect 2.0 sensor, in order to recognise human gestures in real time, for the purpose of Human-Robot interaction” [16].

The method that I used to achieve this was to firstly normalize the skeletal data provided by the Kinect by translating and scaling it, I then created a 20 dimensional feature vector of 3D distances between selected skeletal body joints, and 2D angles between selected bones in the body (angles between a pair of vectors). This meant that the created feature representation was invariant to the absolute position, scale and angle of a person with respect to the Kinect camera.

I then created a forward HMM, with 7 hidden states and custom transition and probability matrices, to represent each gesture class. The HMM’s were then trained on a set of training gesture sequences from the newly available NTU RGB+D dataset, by applying the Baum-Welch algorithm to the gesture sequences, in order to learn the transitional probabilities of each HMM.

In order to recognise a gesture in an online manner, the skeletal positions of a person were captured while the person was moving (determined by the hands moving by more than a set amount for several consecutive frames) and then normalized. The normalized skeletal positions were then transformed into distance and angular features, evaluated by each learnt gesture model in turn, and the model with the highest likelihood for that particular sequence was reported onscreen.

The offline recognition process was similar to the online one detailed above, but skeletal data was read from testing files, rather than the Kinect camera, and the most likely gesture model for a sequence was compared to the ground truth for that sequence, and saved to a file.

This methodology achieved a good average recognition result of > 78% across approximately 3000 testing gesture sequences, while also proving to be highly invariant to many changes in environment which usually have negative impact on gesture recognition systems. E.g. changes in camera position, background clutter, lighting changes, and of person to camera and the person’s body shape/size.

The main conclusions are that the literature review that I performed was relevant and in depth, my initial aim has been successfully met; my system can learn, model and
recognise gestures in real time (and offline), and achieves a good classification accuracy overall. My system is also highly invariant to environmental changes and has been implemented to be easily extensible in the future.

The main recommendations for future work are the utilization of depth data in order to better define gestures which are ambiguous when just using features derived from the skeletal data, as in actions such as rock, paper and scissors, and to employ some kind of dynamic algorithm to calculate the optimal number of states for a set of gesture sequences. (Possibly a special implementation of the K means algorithm)

Table of Contents

Acknowledgements ........................................................................................................................... 2
Abstract ............................................................................................................................................... 2
Introduction and Rationale .................................................................................................................. 5
  Subject area and relation to course/career interests ..................................................................... 5
  Practical applications and Originality ............................................................................................ 5
Aim/Hypothesis .................................................................................................................................... 5
Brief description of activity .................................................................................................................. 6
Objectives ............................................................................................................................................... 6
Key Issues ............................................................................................................................................. 6
Research ................................................................................................................................................ 7
  Introduction ........................................................................................................................................ 7
  Deep Dynamic Neural Networks for Multimodal Gesture Segmentation and Recognition .......................................................................................................................................................... 7
  R3DG Features – Relative Geometry based Skeletal representations for Human action Recognition ................................................................................................................................................. 9
  Static and Dynamic hand Gesture Recognition in Depth Data Using Dynamic Time Warping ........................................................................................................................................................................ 11
  A Real-time Dynamic Hand Gesture Recognition System Using Kinect Sensor .............. 12
  Data Fusion-based Real-Time Hand Gesture Recognition with Kinect V2 .................. 13
  Evaluation of relevant software and hardware ............................................................................. 14
  Relevant professional issues ............................................................................................................. 15
  Conclusion ......................................................................................................................................... 16
Methodology ........................................................................................................................................ 16
  Introduction ..................................................................................................................................... 16
  My previous methodology ............................................................................................................... 17
Introduction and Rationale

Subject area and relation to course/career interests
The topic area that I have chosen to study for this dissertation module was that of Computer Science, or, more specifically, computer vision and machine learning. I chose to study this area for my dissertation as it has a very strong relation to both my undergraduate and postgraduate degrees, and so I already have a good understanding of the topic areas. A further reason why I chose to study this topic area is that I have had a lifelong interest in the areas of computing and machine learning. In addition to my interest in machine learning, I have a further interest in robotics, to which the topics studied in this dissertation module are highly applicable.

I feel that machine learning, computer vision, and robotics are going to be some of the key areas of technological progression in the coming future, and throughout my working career. And I hope that I can have an impact on the progression of technology in one or more of these areas throughout my lifetime.

Practical applications and Originality
The area where this project will have most impact is that of Human-Computer or Human-Robot interaction, as it will provide a more natural user interface (NUI) between man and machine, as opposed to a traditional graphical user interface (GUI) as widely used between man and machine today. Other practical applications of gesture recognition technology include: marker less motion capture systems, security and identification systems, driverless cars etc.

I consider this work to be original as, at the time of starting the dissertation, I could not find any published work which utilized the new Kinect 2.0 sensor. In addition to this, at the time of writing this report, there appears to be no published methodologies which utilize the new Kinect 2.0, along with a sparse skeletal representation of features and HMM’s for temporal modelling, and being used in a real time system.

This work can also be considered as original work as it is both trained and evaluated on a subset of the large scale Kinect 2.0 dataset which has been made publicly available just several months ago. This dataset is known as the NTU RGB+D dataset and contains recorded skeletal data with 80 different viewpoints, 40 subjects of varied ages, 60 action classes, and a total of 56880 individual samples [17].

A further way in which this work can be considered original is that it has proven to be highly invariant to environmental changes, such as lighting, background clutter, camera position and view, different body shapes etc. Which is a highly important aspect of a system designed to be used for Human-Robot interaction in real time, and in many different environments.

Aim/Hypothesis
The main aim of this project is to “Produce a system which utilises depth information [9,10] from a Kinect 2.0 sensor, in order to recognise human gestures in real time, for the purpose of Human-Robot interaction” [16].

**Brief description of activity**
This aim will be realised by producing a software system which can ‘learn’ a range of human gestures, and be able to recognise when a person is performing them, in a real time manner. The system will utilize the new Kinect 2.0 sensor, as well as some of the most widely used and most accurate approaches discovered, as a result of my literature review.

The main activities I must achieve for this project to be a success, as outlined in my project proposal; are to perform an in depth literature review of current gesture recognition techniques, to test out and evaluate some of these techniques, and then to pick the most relevant technique(s) and use them to create a software system with the ability to recognise different human gestures in real time.

**Objectives**
The main objectives of my project, as taken from my project proposal were to:
1. Complete a literature review of gesture recognition techniques, for use in real time systems.
2. Read more widely around the topic of gesture recognition by researching Human-Robot interaction
3. Identify and collect a suitable dataset, which can be used for the training of the system
4. Develop a suitable range of features to use for distinguishing between gestures
5. Develop a suitable model which can determine the start and end of a gesture
6. Test the features and temporal model in an offline format, to evaluate gesture recognition rate
7. Ensure that real time segmentation of gesture sequences is working correctly
8. Ensure that the features and gesture segmentation can be used for gesture recognition in real time, using a Kinect 2.0 sensor
9. Critically evaluate the completed system, methods used, and compare to the work of others

**Key Issues**
One key issue that I had to take into account when creating my system is that I had to ensure that the features I used to describe a gesture were as invariant to environmental changes as they could be. This included making sure that the features would be robust at overcoming the problems encountered by different lighting conditions, different scales of peoples’ bodies, different camera viewpoints etc. This meant that the best solution for me was to use features derived from depth data (more specifically, 3D skeletal data), as depth data is inherently robust to lighting changes, object colour etc, and skeletal data can be made further invariant to camera angles and the position of the actor within the frame.

Another key issue that my system had to overcome was that it had to run in a real time manner, this meant that I would have to ensure that my feature creation
process, as well as my gesture model implementation and evaluation techniques, would process quickly enough to be used for recognising gestures in real time.

An additional key issue that I had to ensure my system could cope with was that it had to automatically determine the start and the end of each gesture taking place, as in a real time Human-Robot interaction environment, gestures would not be segmented a priori. The overcoming of this issue also ensured that the system could be used for gesture recognition in a continuous manner.

A final key issue that I had to make sure that my system could overcome was that it had to be able to recognise gestures with a good level of accuracy. This included both in the online (real time) and offline manners of operation.

Research

Introduction
While my literature review covered a very large set of data, from a very diverse range of sources, I have provided a detailed overview of only the papers which I deem most relevant to my dissertation (either in terms of technique used or recognition accuracy achieved), referencing other, related sources when necessary. Full references to all sources used can be found in the “References and Bibliography” section.

Deep Dynamic Neural Networks for Multimodal Gesture Segmentation and Recognition
This source was [7] discovered through a search of the IEEE database for ‘Gesture Recognition’ and was chosen to be studied in more depth as it was a very modern paper (from the start of this year), it was published in a very good computer vision journal, and achieved good results.

I have chosen to base part of my project upon this source [7] for three main reasons. Firstly, the dataset and approach used by the authors of the paper is very similar to the gesture dataset that I captured for use in this project, and the initial approach that I had in mind as a result of my first literature review. Secondly, the paper was published at the start of this year (2016), and was published in the best IEEE journal in existence for Computer vision topics. (I was told this by my supervisor who is an expert in the field of computer vision). And thirdly, the approach used by the authors of this paper achieved a good recognition accuracy of ~82%, which is highly important for a real time system to function properly.

The approach taken by the authors of this paper comprises of three main parts, namely, a deep belief neural network which generalizes a set of skeletal distances and angles into a lower dimensional skeletal feature set, a deep convolution network which takes as input an RGB and a depth image around the hand of the person performing the gesture and produces a lower dimensional output feature vector. And then a HMM which is trained with the fused output of these two neural networks, and used to model a single gesture.

For the first part of the authors’ approach, the $N_j = 11$ upper body joints only are used and from them, their 3D pair wise positions [6] are computed, along with their
temporal derivatives. These data are used as an input feature vector to the deep belief neural network [8], this neural network generalizes the high dimensional feature vector into a lower dimensional one, which can be used to learn the transition and emission probabilities of the HMM [24,26] for each gesture class. In order for this particular neural network to model the continuous skeletal feature vector input, it consists of a stack of Gaussian-Bernoulli restricted Boltzmann machines [25], with interconnections between node layers, but no inter-layer node connections.

In the second part of the approach a 3D convolution neural network is used, consisting of multiple fully connected layers of artificial neurons, which take multiple, contiguous images as input, process the images through several convolution layers, and output a 1 dimensional feature vector. This deep convolution neural network is based on that of DeepMind, a Google company which was first to use a deep neural network to learn high level features from raw input data. The images which the authors use as input to this deep convolution neural network consist of sequences of four contiguous RGB and Depth images of the area around the hand [9][10] performing the gesture [13]. The images are produced by locating the highest hand in the frame relative to the person performing the action, cropping out a pre-specified area around the hand, and then filtering and normalizing the cropped image data.

The third element of the approach, the hidden Markov Model, is used for simultaneous gesture segmentation and recognition [1], which is achieved by defining a state transition diagram (HMM) for each gesture, similar to the conditional random fields as used in [11] and [4]. The feature vectors produced by both the deep belief network, and the deep convolution network are fused together, and used as input to the HMM to ‘learn’ the transition and emission probabilities of the HMM. The fusion process [12] takes place in a hidden layer of the convolution neural network. In order for a gesture to be properly segmented, an ergodic state is present in the combined state transition diagrams, which acts as a state for the resting position of a person performing the gesture. In order to capture the time variability of a gesture sequence, the HMM can either remain in a state at time t, move to the state at time t+1, or move to a state at time t+2. And to account for the variability in people performing a gesture sequence, a gesture can be started in one of its first three states, and can be completed in one of its last three states. The trained hidden Markov Models are then be used to classify a new gesture sequence, by the means of transitioning from the ergodic state, through all of the defined gesture models, back to the ergodic state, and outputting the model name with the highest likelihood for a given input sequence. See figure 1 below for an example of the HMM layout.
The above approach was evaluated using the ChaLearn [13] dataset and achieved an average recognition result of ~82%. This approach was highly relatable to what I wanted to achieve as I wanted to be able to efficiently model the skeleton and its dynamics in a way that could be evaluated with low latency, namely by using a sparse skeletal feature vector and modelling its temporal evolution using HMM's [7,15,16]. This approach also utilized both depth and RGB data in order to better describe the pose of the human hand while a gesture was taking place, however the authors did not mention anything about these techniques being used in real time.

R3DG Features – Relative Geometry based Skeletal representations for Human action Recognition
This source was found while searching Google for skeletal representations used for human action recognition.

I have chosen to look more deeply into this publication [5], as it presents a skeletal based approach of gesture recognition, using features modelled from explicit rotations and translations of the human body, which are highly invariant to many environmental factors (e.g. scale, viewpoint, lighting). The two main reasons for me analysing this approach in more depth are that the skeletal features used are highly invariant to environmental factors, and that the author (Raviteja Vemulapalli) undertakes a lot of work in the field of actions/gesture recognition, and consistently achieves very good results. I feel that these two factors are a large part of what I want to achieve with my own work.

The approach taken by the authors of this paper consists of three main parts, these parts are: The normalization and modelling of skeletal data to make the system robust to different body shapes and scales, the mathematical representation of the bones of the skeleton, into a gesture curve. And lastly, the classification of these curves using SVM's [3][15].
For the initial part of the author’s approach, they seek to handle the scale variations in skeletal data by resizing all of the skeletons to a fixed or reference size. They achieve this by ‘stretching’ the lengths of the bones in the skeletal data so that they match the length of those in a reference skeleton. While the translations between these body parts vary with this scale normalization process, the angles between the joints do not change, and so they ensure to use only joint rotations in order to produce features which are invariant to the performer’s position within the frame.

In the second part of their approach the authors of this paper adopt a mathematical method to modelling the temporal evolution of the body parts over time. This mathematical method of modelling the body parts over time is derived from the fact that, for any two rigid body parts, there exists a rotation and a translation that will map one of them onto the other. These rigid body part transformations are used between all pairs of body parts to model the human skeleton. The author’s brand this mathematical approach R3DG features; and it produces a skeletal representation which resides in a curved manifold, much like the representation in some earlier work by the author on lie groups [6]. See figure 2 for example of a lie group.

For the third and final stage of their approach they attempt to classify the gestures, as represented by the curved manifolds outlined above, using a combination of DTW [1,5,9], FTP and an SVM [1,3] as the classification mechanism. This is achieved by calculating a nominal curve of each gesture and warping every curve of the gesture to the nominal curve, in order to handle variations and misalignments in the temporal data. The warped curves are then represented using a low frequency FTP representation, which they state was shown to be robust to noise and temporal misalignment in previous work. Finally, classification is performed on the FTP represented data, using a one-vs-all SVM.

The above R3DG approach to gesture recognition was evaluated using a combination of five action datasets (MSRAction3D [4,20], UTKinect-Action [22], Florence3D [21], MSRPairs and G3D), upon which the technique achieved an
average accuracy across all datasets of ~93%. This paper was also highly relatable to my project as it modelled skeletal dynamics in a highly environmentally invariant way, by representing the translations and rotations between each pair of bones in order to map one to the other. In addition to this the paper achieved a very high recognition accuracy using skeletal data alone.

**Static and Dynamic hand Gesture Recognition in Depth Data Using Dynamic Time Warping**

This source [9] was also discovered through a search of the IEEE database of articles, while looking for relevant papers published in the last year or so. It was picked because this paper focuses specifically on hand gestures, and so I thought that it may give me more detailed information of how I can use the hands to differentiate between gestures.

The main reason for me studying this paper in depth as part of my project literature review is that it is highly focussed on Gesture recognition from depth data, and shows a real-time capable algorithm for segmenting the hand contours from the depth data provided by the Kinect sensor. In addition to this, the results that the authors achieve by using their approach are very good indeed, with many gestures being recognised with 100% accuracy.

This paper presents a new approach to scan a depth image, in order to extract the hand contour from within the image, without the need of scanning each pixel individually. These hand contours are then used for training a gesture model, and dynamic time warping is used to match a real time test sequence to the previously learnt models [3,5].

The initial step of this approach is to segment the hand, much like in [10] by recuperating the depth information coming from the Kinect sensor, for each pixel within the RGB image. A fast algorithm is then used to identify each point of the closed contour forming the hand, and the palm centre is defined as being the centre point of the largest circular contour. The fingers are then found by employing the K curvature algorithm. Figure 3 shows some examples of the extracted hand contours.

![Figure 3](image)

In the next step of this approach the hand information (contour, palm centre and fingertip data) is stored in a list. This list is populated throughout the time of the gesture, as new hand data is added to it. Upon a gesture sequence completing, this list of hand data is compared to reference lists of hand gesture data through the use of the DTW [3,5] algorithm. And hence the captured sequence is classified by
comparing it to the reference gesture sequences and selecting the most closely matching sequence as the gesture class.

The main contribution of this work is that it details a time efficient method for the segmentation of the palms of the hands, and the location of fingers, in order to recognise a newly presented gesture sequence by comparing it to the pre-taught models. The results that the authors of this paper have achieved average at ~94%, with many gestures being recognised with 100% accuracy. In addition to this, the approach takes approximately 40 Ms to analyse a frame, meaning that 25 frames can be processed every second using their method. The results are also very similar to the paper on Arabic sign language recognition [10]; however this approach is much faster to compute. Many gesture recognition approaches are now attempting to fuse this kind of hand data with skeletal data, in order to more accurately represent gestures [11].

A major way in which this relates to my work is that it provides a very good foundation for gesture recognition based on the shape of the hands. This, coupled with a skeletal based recognition system as shown in [5] or [7], could have allowed automatic gesture recognition and segmentation with very high accuracy in my project. Another aspect of this work which related well to my project was the low latency of their approach (40ms per frame), which could have resulted in me implementing this technique in a real time manner.

A Real-time Dynamic Hand Gesture Recognition System Using Kinect Sensor

This paper [15] was also discovered after searching the IEEE database for modern and relevant publications in the field of gesture recognition with Kinect. It was published just last year in an IEEE conference on Robotics and Biometrics.

I chose to cover this paper in more depth as it provided a technique based on the new Kinect 2.0 [14], the technique worked in real time, and it achieved an average recognition rate of ~95% across a dataset of the Arabic alphabet, and the digits 0 to 9.

The main methodology implemented by the authors of this paper is to use the Kinect 2.0 sensor, in order to capture 3D motion trajectories [2,16] of the hand performing the gesture, from the precise skeleton tracking system made available through the Kinect SDK. This approach comprises of three main stages, namely, Gesture spotting, feature extraction, and then an SVM [3,13] based classifier. Figure 4 shows one such motion trajectory of the hand, for the character ‘h’.

The way in which the authors determine the start and the end of a gesture in the first part of their method is to make use of the function that Kinect provides on the state of the hands (i.e. open, closed, or victory sign). They use the fact that the hand has opened in order to
start capturing the 3D motion trajectory [15] of it, and use the hand closed as a signal to stop the motion trajectory capture process.

In the second, data processing stage, the authors first discard the first and last 10% of the captured 3D motion trajectory, as they state that they are meaningless points, simply caused by noise and hand shaking at the boundaries of the gesture. They then calculate the arctan value of the motion orientation between any two consecutive points within the motion trajectory, as they claim that orientation is the most important feature to consider in a dynamic hand gesture. In order to improve the final recognition rate, they next apply an angular filter to the data (the formula can be seen in [15]) and bin the angular values into 20 degree sections, thereby creating a discrete vector which can be used as input to the SVM classification system.

The final stage of their methodology includes the actual gesture recognition by classifying the discretized angular values detailed above. Each of these captured feature vectors are compared with each gesture model in the database. In their experiments they use both HMM’s [7] and SVM’s respectively as the recognition algorithms, in order to compare the effectiveness of each recognition algorithm.

The dataset they use to test their methodology consists of the hand gestures for the numbers from 0 to 9, and the letters from A to H. They achieve an average recognition result of ~82% using the HMM classifier, and an average recognition result of ~95% using the SVM [3] classifier. As well as this, they noted that the HMM model took much longer to evaluate using their orientation binning approach than the SVM did.

A major way in which this relates to my work is that it focuses on the motion trajectory of the hand during a gesture, and this is a similar approach to the approach that I would like to undertake in my work for gesture feature representation. I feel that almost all of the information encoded in a gesture can be described by just a few key points of the body, for example, many hand gestures (such as waving) can be completely defined by the trajectory of the hand and elbow alone. A further way in which this relates to my work is that the classification techniques that the authors use are based on both HMM’s and SVM’s, and these are the two classification methods that I had been considering for my project from the very start, mostly because of their recent usage in many gesture/action recognition tasks, and the high recognition rate that they achieve.

Data Fusion-based Real-Time Hand Gesture Recognition with Kinect V2
This paper [14] was discovered just as I was writing up the draft of my final report, upon looking for some of the latest publications in the field of gesture recognition.

I chose to look into the paper in more depth because, even though it technically does not produce any results, it proposed a method very similar to the method that I proposed, and it made use of the new Kinect 2.0 sensor [15]. Also it was extremely up to date, being only published to IEEE during August of this year, and promised to provide a gesture recognition algorithm which could easily operate in a real time environment.
The main contribution of the paper is that it proposes a model which the authors claim will accurately and quickly recognise hand gestures of different digits by fusing skeletal information with depth information [7]. The authors also claim that their technique can easily deal with various kinds of challenges that are commonly faced in gesture recognition, such as rotation, scale changes, different lighting, cluttered backgrounds etc. Figure 5 shows an outline of their proposed framework.

The authors achieve this by firstly obtaining both the depth data and skeletal data using the Kinect 2.0 sensor, then obtaining the hand mask region by segmenting the hands [9] from the depth data, using the skeletal data. Afterwards, the mask contour is extracted and shape descriptors in the connected regions are computed. Finally, according to the geometry distribution of the points of the hand, hand gestures are recognized.

While the authors of this paper do not actually report any results in the form of gesture recognition accuracy, the results they do report show the average recognition time for the approach with regards to each of the hand digits from 0 to 5, with the average time for actually recognising one of the hand digits being around 61ms. This means that the paper does conclude that their approach can be used with the new Kinect 2.0, in a real time system [9,15], although it would not be fast enough to process every frame provided by the Kinect in real time.

This paper is fairly relevant to my project as it uses the new Kinect 2.0 sensor, along with skeletal features and a segmentation of the depth map around the hand, in order to produce a technique which can process multiple data frames in real time.

**Evaluation of relevant software and hardware**

One key observation, in terms of hardware, which can be drawn from the literature review, is that the approaches which utilize the new Kinect 2.0 appear to have a much higher recognition rate while using just skeletal features alone [14,15]. This is because the new Kinect 2.0 can track 25 skeletal joints, as opposed to just 20 from Kinect 1.0 [10], and the Kinect 2.0 SDK includes faster and more accurate algorithms for calculating joint positions, and hence is much more reliable. In relation to my project, this clearly showed me the benefits of obtaining and using the new Kinect 2.0 sensor over the older Kinect 1.0, in order to achieve better recognition results from my system.

A further key observation, in terms of hardware, is that many of the approaches which include the creation of complex features as in [7,12] would require a very powerful PC to run in real time (or even in near real time) and in some cases require
a GPU to run effectively. [7] To relate to my project, this meant that I had to be careful not to choose a methodology which was based upon computationally expensive feature creation, and hence I chose to focus mainly on a sparse skeletal representation of a gesture.

A key observation, in terms of software, is that most of the authors seek to integrate existing frameworks within their methodology where possible. For example, a lot of the methodologies mentioned in the literature review use OpenCV’s algorithms as a key part of their approaches (such as in [14]), they do this because OpenCV contains a large set of optimised computer vision algorithms, and therefore allows rapid prototyping and development of the author’s methodologies. For my project, this observation indicated that it would be beneficial to me to integrate a set of existing machine learning algorithms into my methodology, in order to allow rapid development and prototyping.

Another key observation, in terms of software, is that many approaches are based on either a variation of C, or Python. This is mainly because both languages have a wide range of packages available for them, which allow the easy processing/displaying of data and also implement many key machine learning algorithms. I concluded from this observation that my project would be based on the C# language. This was because it was easily interfaced to the Kinect sensor and Kinect SDK, and I was confident that I could find a suitable machine learning package as an add on module for it.

**Relevant professional issues**

The first professional issue that I encountered while progressing through this project was that I simply may not have the expertise required to complete the project, or that I may lack understanding in a certain sub-topic area. However, this issue was easily overcome by; performing additional research into the sub-topic area, consulting my project supervisor, or talking to the computer vision PhD students regarding the issue.

One professional issue that I had to consider throughout the course of this project was that I had to ensure not to directly copy anyone’s work without fully referencing it. This included, not copying code from any source without providing a reference to it, not directly using anyone else’s ideas and methodology without providing a full reference to their published work, and also not completely copying the methodology from one single publication (rather using a combination of discovered methodologies for originality purposes).

A further professional issue I took into account while completing my project was that I ensured that I followed best practices while writing the code for my project. This included using functions/methods where possible in order to promote code reusability, creating different classes to organise and maintain different sections of code, and commenting code to make it easily understandable when viewed by a third party. By following these criteria I have ensured that my code is understandable, reusable and compliant with the British Computing Society standards.
Conclusion

I felt that one of the main conclusions that could be drawn as a result of this literature review is that I had to ensure that the representation I use for defining my gestures was as invariant to environmental changes as possible. This included ensuring that the feature vectors produced were not affected by the changes in size and shape of different people, lighting changes, or cluttered backgrounds for example. This meant that the approach that I took was highly focussed on defining a gesture using the skeletal data [5,6,9], as skeletal data is highly invariant to some of the usual problems in gesture recognition detailed above, and it is calculated from depth data, which is a key part of my project. I also felt that I had to take into account some of the methods that the authors used throughout their work for normalizing the skeletal data, to make it further invariant to environmental changes, for example, taking a certain part of the skeleton (SpineBase) and using it is the origin (forming a local coordinate system as in [6]), from which all other skeletal points are referenced.

Another conclusion that I drew from the literature review was that the features that I produced, while being highly invariant to environmental changes, could not be overly complicated if they were to work in a real time system [9,15]. This meant that instead of using an approach based on something such as deep neural networks [7,8,12], which may have taken a long time to evaluate if they were overly complex, I would have to choose some kind of sparse skeletal representation, which would fully represent the gesture taking place, but would not require a large amount of processing power.

A further conclusion I noted as a result of my literature review was that many of the latest approaches to gesture recognition make usage of either HMM’s [1,7,15,16] or SVM’s [1,3,13,15]. Several approaches detailed in my literature review make use of SVM’s because of their ability to be evaluated extremely quickly, while providing excellent results, and other approaches make use of HMM’s because of their inherent ability to model data temporally. This conclusion resulted in me choosing to use HMM’s as my main classifier, as they could easily model some sparse skeletal data representation temporally, thereby eliminating the need to create more computationally complex, temporal features.

A final conclusion which can be drawn from my literature review is that, no matter which methodology the authors present, they have to make some assumptions in order to get their recognition system to work. Whether that is that the user has to directly face the camera [9,10], gestures are segmented by means of the person returning to a resting state [7], or that the highest hand is the one which is currently performing the gesture etc. This conclusion meant that I had to make some assumptions and integrate these into my project in order for it to be a success. For example, I wanted to continuously segment a stream of gestures, so rather than assuming a person would return to a resting position between gestures, I assumed that they would remain motionless for at least a few contiguous frames.

Methodology

Introduction

In this section, I have provided an overview of the first methodology that I was going to implement in order to complete this project (as mentioned in my project proposal,
presentation and interim report); however this initial methodology had to change before completing this final report. I have explained the initial methodology and why I felt that it had to be changed before completing my project.

I then provide a detailed explanation of the process of my current/final methodology (with a methodology diagram – Figure 8), broken down into four main sections which immediately precede the overview of my previous methodology. The sections following on from the details of my methodology explain key project management aspects and issues, and include the main organizational aspects of the project, such as my Gantt and Pert charts (see appendices).

**My previous methodology**

Below is an extract taken from my Interim report, which gives an overview of the methodology which I initially intended to use throughout my project; for additional details see the project proposal or interim report.

To outline the main stages of the project, I am going to capture RGBD data on several subjects, for several actions in an un-segmented manner. I am then going to transform the collected data into a set of spatial-temporal features which will represent each of the action classes. Next I will test the system offline to ensure that it is classifying gestures correctly, and then test it online, to ensure that it is accurate and robust enough to be used in real time for human-robot interaction.

The initial step of my implementation involves normalizing the 3D skeletal data and creating a set of skeletal features from it to use as input to the first of my neural networks (the deep belief network). The skeletal features will be created in a similar fashion to those of [5] and [7], in order to properly encode the temporal aspect of the skeletal features, while maintaining invariance to background [2], body shape/size and distance to the Kinect sensor.

The second step of my implementation involves using the skeletal data to locate and segment a small area of the depth (possibly the RGB image as well if needed) data around the hand that is currently performing the gesture [7,9] and [10]. This will then be used as input into my Deep convolution neural network, which will pick out features in this data, and generalize them into a one dimensional feature vector, while also fusing this data with the skeletal features within a hidden layer of nodes.

For the third step of my implementation, I will use the output from the two neural networks (the 1D vector of fused modality features) in order to train a HMM for each gesture, by learning its transition and emission probabilities. I have chosen to use a HMM, as it has been proven to work very well for classifying temporally related sequences, such as in speech recognition [8] and more recently, gesture recognition [7,13].

**Changes made and why they were necessary**

One change that I made to my methodology was to not use the raw depth data stream from the Kinect at all, but to instead use only the skeletal data stream derived from the depth data. The first reason for doing this is that the depth data from the training gesture sequences I had captured was not quite of a high enough resolution to actually distinguish between different hand shapes of the person performing a
gesture. I.E the subject was a fair distance from the Kinect sensor (>2m) during the data capture process, and therefore the depth information provided for each hand was of very low resolution, and when segmented and scaled, made the hand appear to be webbed. This meant that the depth data captured around the hand would add very little discriminative information to the gesture recognition process, and may actually make the gesture recognition less accurate overall. Figure 6 shows one of the confusing examples of depth data, between the gestures thumb up (right) and scissors (left) – which clearly look similar at the low resolution of depth data available.

An additional reason that I chose to change my methodology was that the acquisition, scaling, normalization and CNN processing associated with the depth data took a fair amount of processing power and a lot of time. Therefore I felt that it was best avoided if possible in a system which was primarily designed to operate in a real time manner, such as the one detailed in my report.

Another change that I made to the methodology, albeit a very minor one, was to not represent the space between gestures as an ‘ergodic state’ HMM, but instead to segment gestures based on the overall hand movement of a person. The reason for this was that I required my system to be able to successfully segment and recognize a continuous stream of gestures, without the subject having to return to a resting position (ergodic state) between each gesture sequence.

My new/final methodology

Below I have presented the details of my new approach, from skeletal normalization, distance and angular feature creation, to modelling the gestures, evaluating a previously unseen gesture sequence, and finally, evaluating the success rate of my methodology.

Normalization and Feature Creation

For the first step of my methodology I created the feature vectors that would represent each gesture. I found from my literature review that the best way to achieve this would be to base my approach mainly on skeletal data \([6,9]\) provided by the Kinect SDK, as this skeletal data was derived from depth data, and hence was robust to changes in environmental factors such as lighting and background clutter, and could be further normalized \([5,9]\) to deal with different scales and positions of peoples’ bodies.

In order to make the skeletal data robust to different people, with different body shapes and body positioning, I translated every point of the skeleton at each time frame, to a Kinect oriented coordinate system. This involved calculating the 3D
translation needed to take the base of the spine (SpineBase in Kinect SDK) to the origin of the Kinect coordinate system, and then applying this translation to every joint of the skeletal data frame. This process ensured that my system was invariant to the absolute location of a person within the frame, and that it would also be partly robust to a change in the Kinect camera position.

After that, I scaled each joint [5] within the newly created coordinate system so that it matched a predefined template of a human skeleton. This made the newly transformed skeletal data robust to people of different heights and builds, and also to the position (in terms of depth) from the origin of Kinect at which they stand while performing a gesture.

In the end, I chose to translate and scale the entire body frame rather than just the key joints required because this meant that my work would be easily extensible to gestures which are more complex, and hence require more body joints in order to be defined properly. The process of extending the body joints used is then as simple as creating distance and angular features for the extra joints, and re-training the HMM gesture models.

Finally, I selected the body joints which portrayed the most information about a gesture, namely the 3D positions of the hands, elbows, knees and feet, and used these key 3D locations in order to create distance and angular features from them. The distance features were created by simply selecting two 3D points and calculating the Euclidean distance between them, and the angular features were created by first defining two, 3D vectors, and then trigonometry was applied to find the angle between them. By defining the features in this way, it meant that the distances were in 3D relative to the body and hence invariant to changes in camera position and the position of a person within the frame, and it also meant that the calculated angles were relative to the parts of the body about which they were calculated, and as such were invariant to changes caused by people not directly facing the camera. An example of distance (the lighter coloured lines) and angular (the lighter coloured arcs) features used can be seen in figure 7.

A total of 16 features were used in order to train the models for the live demo (10 distance, 6 angular), and a total of 20 features (13 distance, 7 angular) were used in order to train the models for testing with the NTU dataset. I chose to only use some of the key body points to create distance and angular features for my demo, because by using this technique, I would not have to use a larger amount of processing power and therefore the system could be made more real time capable, furthermore much of the work presented by my literature review states that gestures are mostly defined by only a few key body points. However, I had to use more features for the training of the NTU evaluation models because this large dataset contained actions which involved, not only arm and leg movements, but also head movements, as well as some gestures which were extremely similar, and therefore required the usage of more data in order to distinguish between them.
Gesture Modelling

In the second step of my methodology, I implemented the models which could be taught using the distance and angle feature vector sequences that I had created, and then used to evaluate a gesture that was currently taking place. I discovered from my literature review that two of the most modern and accurate approaches to achieving this would be to use either an SVM \([1,3,13,15]\) or a HMM \([1,15,16]\) to model each gesture.

Ultimately, I chose to use a HMM with 7 states to model each individual gesture \([7]\), as I knew that HMM’s could easily model the temporal evolution of data as well as the spatial aspect of the data, whereas my training data would have to be modified further by the creation of temporal features (more computationally expensive) in order to be successfully used with SVM’s. I also chose to explicitly specify the initial transition and emission data for the training of the HMM’s as I wanted to restrict the way in which a gesture sequence was allowed to progress through the model. For example, the transition matrix that I chose to create allowed a gesture to start in one of the first three states of the gesture model, and to transition through the model, either at a slower speed by remaining in the same state, or to progress to the next state, and finally to finish in either of the last two states of the gesture model. This type of model, with non-zero values down its diagonal, is referred to as a forward HMM, and an example of the forward transition matrix and initial state probabilities I chose for every model is as follows.

```java
// Create the transition matrix and initial state probabilities
double[,] transitionMatrix = new double[,]
{
    { 0.45, 0.4, 0.15, 0, 0, 0, 0 },
    { 0, 0.45, 0.4, 0.15, 0, 0, 0 },
    { 0, 0, 0.45, 0.4, 0.15, 0, 0 },
    { 0, 0, 0, 0.45, 0.4, 0.15, 0 },
    { 0, 0, 0, 0, 0.45, 0.4, 0.15 },
    { 0, 0, 0, 0, 0, 0.45, 0.55 },
    { 0, 0, 0, 0, 0, 0.2, 0.8 },
};
```
double[] initialStateProbs = new double[] { 0.5, 0.3, 0.2, 0, 0, 0, 0 };

This meant that I would have a HMM which represented each individual gesture, and left me with one final problem to overcome. This problem was how to model a state which represented the human body when not performing a gesture, in other words, how to split up a stream of continuous gestures. One solution I found to this in the literature review was to explicitly model an ‘Ergodic state’, that is a state where the persons hands are by their side and no gesture is being performed. This initially seemed like a good solution; however it would not allow gestures to be continually segmented and recognised, as the person would have to return their arms to the resting state between each gesture.

Therefore, I chose to adopt an approach that segmented gestures based on when a person’s hands and arms were not moving for several contiguous frames. I chose to do this because one common point which many of the papers that I studied portrayed was that, although people would not necessarily return to a resting position between performing gestures, they would usually have a moment of stillness between them, and hence this could be used to split up continuous gesture sequences.

The training processes

My choice of sparse distance and angular features, combined with HMM’s to model each gesture, meant that the model training process involved gathering together the set of distance/angular feature sequences that had been captured for each individual gesture and applying the Baum-Welch algorithm to them, in order to ‘learn’ the transmission and emission probabilities of each of the Gesture HMM’s. The HMM’s were trained using a maximum number of iterations of 10,000 and a maximum tolerance of 0.0001, but in reality convergence was achieved long before the maximum number of iterations was met. The HMM models were implemented using C# and the Accord machine learning framework, and I created a program in C# to iterate through the training sequences, sort them all by type of gesture, and then to iteratively train a selection of HMM’s to model the gestures.

This resulted in my system including a set of HMM’s – one to model each gesture [7], which could then be used in the recognition process, by comparing a newly captured sequence to them and evaluating the likelihood produced by the new sequence, for each model.

The gesture models were trained using a dataset, only made available a few months ago. This new dataset is known as the NTU RGB+D dataset for large scale human activity analysis. I chose to train my system on this dataset because it is a newly available skeletal dataset, captured with the new Kinect 2.0 sensor, and allowed me to provide quantifiable results in terms of the recognition accuracy of both my online and offline systems. The dataset also contained actions from 20 subjects, 60 camera viewpoints, and 60 actions – see reference [17] for further details of the dataset.

The online recognition processes
The online recognition process was realised by the creation of a C# application which managed the input from the Kinect sensor, showed the Kinect data feeds to the user, translated and scaled the skeletal data, created distance and angular data from the normalized skeletal data, and then evaluated the captured sequences using the trained HMM models.

The way in which a gesture was captured in real time was that it was based on the movement of the two hands. For example, when the hands started moving by more than a certain amount, for several contiguous frames, then the sequence capture process would begin. From this point onwards, every skeletal frame would be captured, translated, scaled and had distance and angular features created from it. It was then added to a list containing the previous frames of the captured gesture sequence. When the hands became still again for several contiguous frames, the gesture capture process would end, and the list of captured sequences would be passed to a class that I created to manage and evaluate the HMM models. (I had to ensure that I was not basing the segmentation on one or two motionless frames only, as some gestures, such as waving, contain short periods of no motion as part of their definition.)

The HMM class then checked the sequence, and if it consisted of a certain number of frames or less (<20 frames) it would be rejected, as an extremely short sequence of frames was highly unlikely to contain a gesture. The sequence would then be evaluated by every gesture HMM within the class, and the gesture model which had the highest likelihood for the input sequence would be reported to the user, in the form of a message at the top corner of the screen. Note, gestures which evaluated to a very low likelihood score for every model were also rejected.

The offline recognition processes

The offline recognition process was realised by the creation of a further piece of software, which acted very similarly to the software as detailed above for online recognition, except that it did not contain any code to access the Kinect camera, but instead code to load NTU skeletal data files.

The way in which a gesture was inputted to the offline recognition program was through the CSV file reader feature of the Accord IO framework. This allowed me to iterate through a folder of testing gesture sequence skeletal files, load each one in, and to have it evaluated by all of the HMM gesture models in turn. In contrast to the online recognition process, the HMM gesture model with the highest likelihood for the particular testing sequence was found but stored in a list.

After all of the testing files in the folder had been evaluated, the truth data was drawn from the collection of file names themselves (as each gesture sequence in the NTU dataset consisted of a single file, which was labelled according to the gesture being performed), and then compared against the corresponding output of the HMM evaluation process. The overall score was then calculated as a percentage of recognition accuracy across all of the tested gestures, and recorded, with the truths and output labels, into a file so that it could be easily looked at again at a later date.
Real-time adding of a gesture

One extra part that I added to my initial system design during development was the ability to capture a gesture sequence in an online manner, using the Kinect sensor and a person performing the gesture, and then adding and training a HMM to model the gesture.

This process involved clicking the ‘add gesture’ button, which started a 10 second countdown in order for the person to get into place for the gesture performance, and to allow the Kinect SDK to locate the person’s skeleton from the depth image. The person performing the gesture would then have a 3 second interval in which to perform their new gesture sequence, and this process would iterate a total of five times, in order to capture variations in the gesture sequence. The 5 captured sequences were then passed to the Baum-Welch algorithm to train a new HMM to represent the gesture.

I felt that this was a useful feature to add to a system that was primarily focused on Human-Robot interaction, and would allow my system to be easily used in a wide range of environments, and with different robots and machines. An example of this could be teaching a custom designed robot a bespoke gesture, in order to better interact with your home environment.

Methodology diagram

FIGURE 8
Testing/Evaluation techniques

Below are the details and results of the testing of my completed system. The system testing involved two main parts, online testing to demonstrate that the system could accurately recognize gestures, in a real time manner of operation, and offline testing to provide quantifiable results.

Online Testing Details

For the online part of my system testing, I first added a selection of pre-trained (using the NTU dataset) gesture models to my main C# program, (This is the program which takes its input directly from the Kinect sensor) and then laid out a few test cases for testing the accuracy of the recognition in real time, using the Kinect sensor. The pre-trained models were trained on 6 action classes, using 158 examples of each gesture class, performed by 40 different people, from 17 different Kinect viewpoints, with 100% of the data being used for training. The test cases consisted of a sequence of different gestures, with different speeds, and from slightly different camera angles/positions.

Next I started the application, stood in front of the sensor, and performed a sequence of ten gestures as laid out in the test cases, in a continuous manner. I ensured to follow my laid out test cases correctly, by performing the gestures at the different speeds specified, and by altering my body position/the camera angle when necessary.

I also got someone to record the results for me, as I was performing these gestures, and they were evaluated and displayed onscreen. This left me with a list of truths for the performed gestures, as well as the output from my program. This data could then be used for comparison sake, and to evaluate the overall accuracy of the system. It is important to point out that the HMM gesture models were not trained on gestures performed by me at any point, they were trained on examples from the NTU gesture dataset [17], and hence this is a completely fair test. I repeated these tests a total of 10 times, (giving a total of 100 tests) and calculated the average recognition accuracy, as well as the accuracy per class. The results that I achieved are detailed in the ‘results’ section.

Offline Testing Details

For the offline part of my testing, I decided to train the system on a larger selection of actions from the NTU dataset [17], in order to make my results comparable to those of some of the papers I have studied as part of my literature review. This meant that I had to select a number of gestures which were close to the number of gestures contained within other evaluation datasets, such as the 3DActionPairs (10 subjects, 12 classes) or UTKinect Action3D dataset (10 subjects, 10 gestures). It also meant that I had to use gestures which were similar to those present in many of the widely used datasets for gesture recognition, for example waving, shaking someone’s hand, reading, answering the phone etc. The selected gestures are as follows, and I feel that they accurately represent the majority of the gestures commonly included in other publicly available gesture datasets.
Selected Gestures

1. Waving Hand
2. Clapping
3. Shaking someone’s hand
4. Pointing at something
5. Sitting down
6. Answering the phone
7. Picking an object up
8. Reading a book/magazine
9. The ‘Cheer up’ gesture
10. Hopping on one leg
11. Kicking something/someone
12. Putting an object into your pocket
13. Standing up from a seated position

To run the offline testing, I first selected the action classes needed, and split them up randomly into a training set of 70% of the files, and a testing set of the remaining 30% of the files. I used MatLab to do this, by creating a random permutation of the files in order to select a random testing set, and remove the selected files from the training data. This gave me two folders of data, one folder which contained the training files, and one which contained the testing files. (Before the data was randomly split into training and testing subsets, it consisted of, 12 gesture classes, 158 examples of each gesture class, performed by 40 different people and captured from 17 different Kinect viewpoints)

It was then as simple as running my other C# program, which was created to perform offline testing on data such as that outlined above. The program loaded up the training files, split them into gesture classes, and then iteratively trained a set of HMM models which could be used later for classification of the testing files. The program then iterated through the testing files, evaluated them with respect to each gesture model and recorded the classification output. Finally it compared the output with the truth data obtained from the file names, gave classification accuracy as a percentage, and wrote all of this output data to a text file.

This process was repeated a total number of six times, meaning that approximately (6 * 513 = 3078) individual gesture sequence tests were run, and a good variation of subsets of the training and testing data were used. Figure 9 shows the completed system being evaluated by me in real time, and the correct gesture classification being output to the screen.
Project specification/ Project management

The main ways in which I organized my project were to create and follow both a Gantt chart and a Pert chart, and to keep archives of each software release of my project on Google Drive, along with a description of the changes made during each release. By organizing my work using these two charts I was able to ensure that the project would be completed in the timeframe allowed, and therefore I would minimise the risk associated with not completing the project in the allotted timeframe.

The Gantt chart also allowed me to focus on small parts of the project at any one time, ensuring I did not get overloaded with work and get confused about what to do to in order to progress, and the Pert chart showed in which order the smaller tasks needed to be completed, in order to achieve my overall goal.

The regular archives of code on Google Drive also ensured that I had backups of my work at various stages throughout the project, and meant that I could restore my project back to any previous time interval should something go wrong, hence mitigating the circumstances which would have arisen through loss of work.

Functional and non-functional requirements
Below are both the functional and non-functional requirements of my system, which are more or less the same as outlined at the start of his project:

<table>
<thead>
<tr>
<th>Functional</th>
<th>Non-functional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Develop suitable features to represent a gesture</td>
<td>Complete a literature review of relevant gesture recognition techniques</td>
</tr>
<tr>
<td>Develop a suitable algorithm which can be used to determine the start and end of a gesture</td>
<td>Read more widely around the topic of gesture recognition by looking at papers on Human-Robot interaction</td>
</tr>
<tr>
<td>Choose an appropriate mechanism to model each gesture</td>
<td>Provide test results to demonstrate that the system is actually achieving what I designed it to do</td>
</tr>
<tr>
<td>Ensure that the model can effectively recognise and categorise new gesture sequences</td>
<td>Critically evaluate the completed system, and discuss any further amendments that could be made to it.</td>
</tr>
<tr>
<td>Ensure that the complete system can run in a real time manner</td>
<td></td>
</tr>
</tbody>
</table>

**FIGURE 10**

**Hardware and software requirements**

Below are the hardware and software requirements of my system, which have evolved slightly since starting the project, and testing out some new tools and techniques:

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>XBOX Kinect 2.0 sensor</td>
<td>Kinect 2.0 SDK</td>
</tr>
<tr>
<td>Kinect for windows adapter</td>
<td>Visual Studio for coding in C#</td>
</tr>
<tr>
<td>A fairly modern, windows based PC (Does not require a GPU)</td>
<td>MATLAB for rapid prototyping of initial ideas and techniques outlined in research papers studied</td>
</tr>
<tr>
<td>A high bandwidth USB 3.0 PCI card, for transferring the large volume of data output by the Kinect 2.0 sensor</td>
<td>The Accord machine Learning framework</td>
</tr>
<tr>
<td></td>
<td>The Accord IO framework for easily loading data from files</td>
</tr>
</tbody>
</table>

**FIGURE 11**

**Issues and risk analysis**

One of the main risks associated with this project was not completing the project in the allotted timeframe. These included completing the code to automatically segment and recognise a gesture, as well as completing the documentation for the project and software produced. This risk had a small chance of occurring, but the effects of this risk occurring were nullified by carefully following the steps as outlined in my Gantt/Pert charts, and ensuring that each task was completed before the deadline.

Another potential risk was that the hardware (Kinect Sensor or PC) may fail while I was working on the coding for my project, this risk was not very likely to occur,
however if it had occurred then I would have been fairly quickly able to repair or to purchase new equipment and therefore mitigate the long term implications of this risk.

A further risk was illness or the inability to work on my project, and therefore not completing the project in the allotted timeframe. This risk was also not very likely to occur, however it would have been very hard to rectify should it have happened, without getting an extension on my project deadline.

**Legal, Social and Ethical issues**

One potential social issue related to this project is that it could potentially help to change the way that Humans and machines interact in the future, as it provides a more user friendly, natural user interface (NUI). For example, if this project is considered a good success, then it may become easier for people to adopt this methodology in order for people to communicate with machines in a much more natural way than they do currently.

One Legal/Ethical issue associated with my project could be a result of obtaining and using of the captured Kinect gesture data from different subjects. However, this was not an issue throughout my project as I was only using skeletal data within my system, which could not be used to identify an individual, and therefore complies with the data protection act and personal confidentiality.

One potential legal issue that could have arisen as a result of this project could be that of plagiarism. For example, if I were to copy parts of someone else’s work without fully referencing it, or if I were to copy an entire project that belonged to someone else.

Apart from the Legal, Social and Ethical issues as outlined above, and the professional issues as detailed earlier in this report, I see no more issues arising from my system, as it is not dependant on confidential data, and is not considered socially unethical.

**Conclusion**

To conclude the methodology section, I have detailed my entire system design, based on the findings from my literature review, and justified my choices and decisions throughout. I have also detailed the hardware and software requirements of the system, the functional and non-function requirements of the system, and any relevant issues I had encountered while the system was being developed.

**Results and findings**

**Introduction**

In this section, I present the results achieved from performing both real time and offline testing of my project, along with tables and screenshots where appropriate, to justify the achievements of my gesture recognition system.
Online testing results

The graph below (Figure 12) shows the recognition rate of each of my ten gesture sequence tests, as a percentage of the gestures correctly classified for that test. The 11th bar (to the far right of the graph) shows the average recognition rate across all one hundred of the individual online gesture tests. The average classification accuracy in these examples turns out to be 87%, with accuracy for any individual set of 10 gestures ranging from 70 to 100%. I feel that an average recognition result of 87% is very good for an online gesture recognition system such as this, and it demonstrates that I have met my initial goal of implementing a real time gesture recognition process, which is based on depth (skeletal) data.

In addition to this, the system was able to achieve this high average accuracy while never being trained to recognise the specific actor performing the tests, while the actor was performing the tests purposefully at different speeds, and while changing the horizontal position and angle of the camera throughout the tests. This shows that my methodology is highly invariant to body size, shape, angle of view, position of camera etc, which is very important for a real time system to work effectively, in a range of different environments. I have also included a chart of the average accuracy per gesture, which can be seen below (Figure 13).
The above chart shows that certain gestures were classified correctly 100% of the time, this is probably because these types of gestures are very well defined, and are not overly similar to any other gesture in the testing set. However, some gestures were misclassified some of the time, such as checking the time on your watch, this is probably because these gestures are slightly ambiguous in the fact that they are very similar to other gestures in the test set. (Checking your watch could be considered as similar to making a phone call, as the hand and arm are in approximately the same 3D space, and perform approximately the same movement to get there).

**Offline testing results**

Figure 14 shows the average recognition rate of each of my six tests. Each of the individual tests comprised of 2051 individual gesture sequences, which were randomly split (differently for each test) 70/30 for training and testing respectively. This means that each test represents the average recognition rate across 158 individual gesture sequences. The bar to the far right represents the average recognition rate across all of the individual tests, which comes to just over 78%.
The offline testing results also show good average gesture recognition accuracy across the entirety of the tests. I did however state initially that I would like to achieve an average gesture recognition accuracy rate of about 85%, but I am happy with the average recognition accuracy result of >78% that I did achieve, as my feature representation has proven to have been extremely robust to many environmental changes, which would otherwise negatively impact the recognition process.

I have also included a confusion matrix (Figure 15) for all ~3000 of the offline gesture classification tests, the confusion matrix can be seen following this.

The confusion matrix shows that, while actions are generally classified correctly as a whole, there are clearly certain confusing cases, where the system misclassifies some of the gestures. A clear example of where this happens most is between the

FIGURE 14

FIGURE 15
pointing and waving classes, with about 20% of the gestures being misclassified in either direction.

**Comparison to other work**

In order to properly compare my work to the approaches that I have studied as part of my literature review, I am now going to provide two tables which summarise the recognition accuracies of the work and the datasets used for their training/testing phases.

**Details of commonly used gesture recognition datasets**

<table>
<thead>
<tr>
<th>Dataset Name</th>
<th>Number of Classes</th>
<th>Number of Samples</th>
<th>Number of Subjects</th>
<th>Camera Viewpoints</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSR-Action3D</td>
<td>20</td>
<td>567</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>3D Action Pairs</td>
<td>12</td>
<td>360</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>Florence 3D actions dataset</td>
<td>9</td>
<td>215</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>UTKinect-Action3D Dataset</td>
<td>10</td>
<td>20</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>Multiview 3D Event</td>
<td>8</td>
<td>3815</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td><strong>My Dataset</strong></td>
<td><strong>13</strong></td>
<td><strong>2051</strong></td>
<td><strong>40</strong></td>
<td><strong>17</strong></td>
</tr>
</tbody>
</table>

**FIGURE 16**

**Details of methodologies and results as found in my literature review**

<table>
<thead>
<tr>
<th>Method used</th>
<th>Average Result</th>
<th>Dataset Used</th>
<th>Number of Classes</th>
<th>Number of Subjects</th>
<th>Camera Viewpoints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deep Neural Networks [7]</td>
<td>82.83%</td>
<td>ChaLearn LAP</td>
<td>20</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Relative Geometry [5]</td>
<td>84.74%</td>
<td>MSRAction3D</td>
<td>20</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>Hand Contours DTW [9]</td>
<td>92.2%</td>
<td>Custom – hand gestures</td>
<td>8</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Joint Trajectories SVM/HMM [15]</td>
<td>89.14%</td>
<td>Custom – hand digits/letters</td>
<td>18</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><strong>My Methodology</strong></td>
<td><strong>78.18%</strong></td>
<td><strong>NTU RGB+D subset</strong></td>
<td><strong>13</strong></td>
<td><strong>40</strong></td>
<td><strong>17</strong></td>
</tr>
</tbody>
</table>

**FIGURE 17**
When comparing results of my approach to some of the results from other work that I have studied, I can conclude that, while my overall accuracy rating is not quite as high as some of the other methodologies, my methodology can produce a good accuracy rating, while being highly invariant to different people, different gesture styles, and different camera viewpoints. This can be seen by comparing my results and the training/testing datasets that I used to that of other peoples. For example, the NTU subset which I used for my testing and training consisted of >2000 individual samples, from 17 different camera viewpoints, recorded from 40 different people, and some actions also included 2 actors (which meant that the action was being seen from a side on viewpoint). This demonstrates that my system is highly robust to many different environmental changes. See the table in the previous section for comparisons of common datasets to mine.

An additional conclusion which can be drawn as a result of my testing is that methods which also utilize the raw depth information around the hands, in addition to the skeletal data, tend to achieve a higher accuracy of classification. This is because the raw depth information surrounding the hand [7], or the hand contour as in [9] can be used to create additional features to describe the gesture, and therefore help to discriminate one gesture from the other. As a result of this, in any future work that I undertake on gesture recognition, I will try to integrate the depth data into my work, in order to improve my classification accuracy overall. It is important to note however, that most of the methodologies which achieve a high classification result from using raw depth data cannot be used in real time, nor are they generally tested on diverse datasets such as the NTU one that I used, but instead use specific hand digit/letters datasets [9,15].

One other comparison I can make with some of the work studied is that I have adopted an approach based on skeletal data, as it is naturally invariant to many environmental factors, and can be further normalized to increase its robustness [5,6,9]. I have found that the usage of normalized skeletal data has indeed allowed my system to recognize gestures quickly and accurately, as can be seen in the results section above.

One advantage of my work over some of the others which use skeletal features is that my work makes use of the new Kinect 2.0, which provides more precise skeletal joint locations, and more of them. A further advantage of my work is that because all of the body joints are normalized to a set location and body template, rather than just the upper body joints, my work is easily extensible to leg/foot gestures, by including these skeletal points in the HMM training sequences. One possible issue with the approach of normalizing all of the skeletal data is that it requires more computation than it would if just using the upper body joints, however in practice; this has not been an issue while testing in real time on my PC.

To compare to another piece of work as detailed in [7], I have implemented a very similar approach to modelling the gestures. I.E I have adopted the use of an ensemble of HMM classifiers, one for each gesture learnt; in order to model the gesture progression through time. The small part of their gesture modelling methodology that I altered was to not have a HMM to represent an ergodic, or resting, state. The reason for choosing not to do this was because I wanted to create
a system which could continuously segment a stream of gestures, without the need for the person performing the gestures to have to return to a resting position between each gesture instance. An advantage of my system therefore over the work presented in [7] is that my system is much more robust to people's individual styles of performing gestures, as well as highly environmentally invariant. However one possible limitation of my system is that if a person were to not stay still for a few frames, between each gesture sequence, then my system may recognise multiple gestures as one gesture sequence and hence not be able to classify the gestures as intended. This however is very odd, as it is a natural process for people to stay temporarily motionless for a few frames between performing different gesture sequences, unless they are intentionally trying to break the system.

A further comparison that I would like to point out is that I did initially intend to use the depth data around the hand region, as well as the skeletal points from the Kinect, and fuse them together into a feature vector in order to better define some of my gestures [7,9,10]. However, in the middle of my project, I concluded that the gesture example sequences that I already captured for training my system included depth data that was of too low a resolution to be properly used. This meant that often hands appeared to be webbed, and hence this data would not be sufficient in describing some of the gestures that I had recorded. In particular, rock, paper and scissors gestures. This leads to a small downfall in my system at the moment, in which it would struggle to recognise actions which have the same motion trajectory, but different hand shape – such as rock, paper, and scissors actions. Although in reality, there are very few gestures which are like this.

One more comparison I can make with all of the other papers is that, to the best of my knowledge, they do not incorporate a facility to capture and learn a new gesture sequence in real time like my system does. I feel that this gives my system an advantage over the others as it is important to be able to add a new gesture to a system designed specifically for Human-Robot interaction, and would greatly aid the usage of this system in a home environment, with a range of different robots.

**Conclusion**

To conclude my testing, and relate it to my functional requirements as outlined previously, I have achieved the following goals:

1. Developed suitable features to represent a gesture
2. Develop a suitable algorithm which can be used to segment a gesture sequence
3. Choose an appropriate mechanism to model each gesture
4. Ensure that the model can effectively recognise and categorize new gesture sequences
5. Ensure that the complete system can run in a real time manner

My online testing results show that all of the above criteria have been successfully met, as the combination of the features that I chose, the gesture models that I chose, and the gesture segmentation algorithm that I used combine to produce a gesture recognition system that can be used in real time. Gestures are recognized as intended, as can be seen from my testing results above.
Both my online testing and my offline testing also demonstrate that my gesture recognition methodology is highly invariant to camera angles, background clutter, lighting, scale, absolute body position, and angle of user relative to camera (Presuming the Kinect does not suffer from occlusion and fail to pick up the skeleton). This high invariance to change in surroundings, combined with the good recognition accuracy I achieve, and the fact that my system can easily operate in real time show that overall my project has been a good success.

The main advantages/contributions of this project are that the system utilizes the new Kinect 2.0, it can successfully model gestures using the sparse feature representation that I chose, it is highly invariant to environmental changes, and it is not computationally expensive – so can easily run in real time.

Another advantage (which was not initially intended for implementation) is that my system allows you to record a new gesture in real time, which should be very useful for a human-robot interaction environment, where bespoke gestures need to be learnt.

**Discussion**

**Introduction**

In this section I have related my results and findings to my main project aim, identified and discussed key project issues, critically reflected upon my work, and considered any relevant legal, social, ethical and environmental issues related to my project.

**Relation of results and findings to project aim**

The main aim of this project was to “Produce a system which utilises depth information from a Kinect 2.0 sensor, in order to recognise human gestures in real time, for the purpose of Human-Robot interaction”.

I can conclude from my results and findings that this aim was successfully met, as I have produced a system which utilises the skeletal data from the Kinect SDK, which is directly derived from the depth data. My results also show that this system is able to learn, model, and ultimately recognise (with a good accuracy of >78% across 13 gestures, 17 viewpoints, and 40 subjects) a range of human gestures, based on the distance and angular skeletal features that I have used. The system is real time capable, and is highly invariant to changes in the environment around the Kinect sensor (Figure 17), as both my test results and the live demo show. And finally, the system has been optimised for Human-Robot interaction by running in real time, and also by allowing the adding of a new gesture model in real time, for use in bespoke human-robot interaction environments.

**Key issues encountered**

The key issues encountered, as detailed in the ‘key issues’ section of my report can be summarised as follows:
1. Ensure created features were invariant to environmental changes
2. Ensure that the system could operate in real time
3. Ensure that gestures could be automatically segmented
4. Ensure that a good recognition accuracy was achieved – both in offline and real time testing

A further key issue, which has not been detailed in the 'key issues' section, is that I found out that the depth data that I had captured for training my models was not of good enough quality to use in the learning process. This had a large impact on my work as it meant that I had to change key parts of my original methodology half way through the project, but was completely necessary and ultimately ended in a good working system.

Critical reflection of work

To critically reflect on the work that I have achieved, I can state that overall the project has been a success, as the initial aim of the project has been met, by applying some of the relevant methods and techniques found as a result of my literature review, and provides a good recognition accuracy comparable to other state of the art approaches.

One aspect of my project that I think went very well was my literature review, as I performed an in depth review of all modern approaches to the problem of gesture recognition, using only respectable sources such as the IEEE database. I also managed to find a good range of techniques applicable to my problem; meaning that I could select the one(s) which I felt suited it best. In addition to this, many of the sources found were published in some of the top computer vision journals in existence, and all achieved very good results.

Something that I think I could have done better with my project is that I should have started trialling some of the gesture recognition techniques that I found a bit earlier than I did. By doing this, I would have realized sooner that the depth data that I had captured for training my gesture models was insufficient for its intended purpose, thereby giving me more time to find and implement an alternate technique. While this could have been a major issue for my project overall, in reality I realised that the depth data was insufficient for its purpose early enough to allow me to change the direction of my methodology and still implement the system in time. One crucial factor which aided the success of this change of methodology was that I intentionally left some ‘spare time’ in my Gantt chart, just in case an issue like this were to arise.

Another aspect of the project which I think went well was the consideration of any profession issues throughout the coding and writing of my dissertation. I feel that professional issues were taken into account at all stages of the project as I did not directly copy anyone else’s code or methodology, and I fully referenced any work which my work was based on. As well as this I ensured that I adopted good coding practice by creating classes for code organization, methods for code reusability and commenting code so that it is understandable by a third party.

A further aspect of my project which I feel went well was that my completed system includes a skeletal normalization procedure which used both the upper and lower
skeletal data produced by the Kinect system. I see this as an advantage because, while a lot of the other methodologies studied only use the upper body joints to define gestures, my system is perfectly capable of modelling gestures produced below the waist, such as kicking a ball, or sitting down; as my testing results show.

In addition to the above, I feel my project was a good success because it produces good results, when tested on a larger than normal dataset which included many different actors and viewpoints. My gesture recognition system can therefore be considered highly invariant to changes in the surroundings, and still produces good results in both a real time and an offline environment.

**Overall Conclusion**

In this section I have also highlighted and discussed any key issues encountered throughout my project and critically reflected on the work that I have produced, comparing to other authors’ work when necessary, and then presented a final paragraph to summarise my project.

The main application area of my system is in an environment, such as a home or a school, where people need to interact with a Robot or machine in a more natural way. With possible other applications of this system including security applications, marker less motion capture systems and possibly human recognition for driverless cars.

The main possible shortcoming of this project is that I did not get to utilise the depth data around the area of the hand as I initially intended, due to the fact that the data was of a low resolution and therefore not suitable in aiding the gesture recognition system. This is an issue that I would like to look into and overcome in the future by capturing gesture data of a higher quality. Although, the skeletal based recognition method that I have implemented has been a good success, I feel that adding features derived from the raw depth data of the hand, would further improve the classification accuracy.

In terms of personal professional development, I can conclude that this project has really helped me to understand many computer vision and machine learning topics in more detail; it has helped me to perform a much better literature review and critical evaluation of relevant material, and has greatly developed my coding skills with C# and the Kinect SDK. More specifically, it has helped me to understand the mathematics and theory behind Markov models, and why they are good at modelling sequences or temporal data, has helped me to be able to interpret the results achieved by others and relate them to my own, and has helped mw to critically evaluate both my and others’ methods and results.

Another way, in which this project has helped me to progress on a professional level, is that it has allowed me to more easily see which tools and techniques are best suited to the problem at hand. This is probably as a result of my in depth literature review, and I feel that I now have a much better understanding of how to tackle any professional problems that I may encounter throughout my career.
A final way, in which this project has helped me to progress on a profession level, is that it forced me to change some of my methodology part way through the project, in order to make sure that I achieved good results from my system. This is as a result of me discovering that the depth data I initially captured for use throughout this project was insufficient for the purposes that I required it for, and has helped me to be able to effectively change my methodology part way through the project. This is a skill which will be invaluable throughout my professional career, as project requirements and customer specifications change throughout any projects that I am undertaking.

As a final conclusion, I think that the literature review that I performed was relevant and in depth, my initial aim has been successfully met; my system can learn, model and recognise gestures in real time, and achieves good classification accuracy while being very robust. My system is also highly invariant to environmental changes and has been implemented to be easily extensible in the future.

**Recommendations**

One possible recommendation for future work is the utilization of depth data in order to better define gestures which are ambiguous when just using the skeletal positions or a skeletal points trajectory, as in actions such as rock, paper and scissors. This would give my system the ability to recognise a larger range of gestures, as well as gestures which are defined by the shape of the hands (such as sign language).

Another possible recommendation for future work is that I would like to adapt my system so that it can recognise group gestures, such as people shaking hands or people playing a game. This would make my system more applicable to a larger range of real world situations, and would greatly aid its deployment in a Human-Robot interaction environment.

A further recommendation that I think would be very good to implement into the next gesture recognition system that I produce would be to create some kind of adaptive algorithm for deciding the number of HMM states for defining a gesture, which means that the HMM would not have to have a constant number of states for every gesture. I feel that this would be beneficial as it would allow more discriminative power between gestures which are generally of a longer or shorter length, and would prevent a HMM from modelling almost the same data twice in a short gesture sequence. I think that this could probably be implemented by a form of K-means clustering algorithm, to cluster a gesture sequence into an optimal number of states.

One recommendation which would make the gesture recognition system more useful in the future would be altering it slightly so that it could recognise multiple gestures being performed, by multiple people. I feel that it will not actually be that difficult to extend my work in this way, as my sparse representation of skeletal features, and gesture model evaluation process are not highly computationally expensive at the present time.

A final recommendation for future work would be the modification of my system so that it can be used in an outside environment. Technically, it could be used outside at the moment; however the depth sensing capabilities of the Kinect sensor are very
erratic in an outdoor environment currently. I feel that the adaptation of my system to an outside/mobile environment would make it much easier to apply to almost any Human-Robot interaction scenario.

References and bibliography

Online Journals


Other online resources


[23] List of RGBD Datasets - http://www0.cs.ucl.ac.uk/staff/M.Firman/RGBDdatasets/


Appendices

Gantt chart, Pert Chart and Online code changes record
<table>
<thead>
<tr>
<th>Folder Name</th>
<th>Date Archived</th>
<th>Changes Made</th>
</tr>
</thead>
</table>
| 1.0 KinectGestureRecognition | 10/06/2016    | Initial archive of code  
Created user interface  
Created Kinect live data viewer classes - color, body, bodyindex  
Created ‘stop’ method to clean up sensor data and unused class references |
| 1.1 KinectGestureRecognition | 13/06/2016    | Added code to playback existing capture from Kinect Studio (XEF file)  
Added code to capture new data in Kinect Studio (XEF file) format  
Created ‘autostop’ method, to clean up after video playback/capture had finished |
| 1.2 KinectGestureRecognition | 17/06/2016    | Added code to calculate change in angular momentum of body joints  
Added code to calculate a total angular body movement to use for segmentation |
| 2.0 KinectGestureRecognition | 18/06/2016    | Created a class which implements a circular queue, for use with moving average  
Added code to create/use the circular queue class for real time moving average calculation  
Offline/Real time views now able to segment a gesture |
| 2.1 KinectGestureRecognition | 18/06/2016    | Added ‘Resting position’ to further aid gesture segmentation |
| 2.2 KinectGestureRecognition | 19/06/2016    | Removed unused viewbox  
Added two viewboxes for showing segmented hands in color space  
Altered ‘ViewLiveFeed’ and ‘RecordXEF’ to now show depth and body index while recording |
| 2.3 KinectGestureRecognition | 20/06/2016    | Created method for cropping a sub writeablebitmap from a larger writeablebitmap  
Mapped the skeletal data to color space coordinates  
Used this mapping to segregate the area around each hand into a new writeablebitmap image |
<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.4</td>
<td>24/6/2016</td>
<td>Combined ‘gesture segmenter’ and ‘hand segmenter’ classes into a single class, so that recognition can now be performed on the segmented gestures and hands. Both color video and skeletal view can now be seen, while gesture segmentation and hand segmentation algorithms are running. Fixed the live view to display correct feeds.</td>
</tr>
<tr>
<td>2.5</td>
<td>28/6/2016</td>
<td>Added check to make sure right and left hand cropping areas do not exceed color image dimensions. Added code to show if a gesture is in progress from right or left hand independently - two gestures could be recognised at once now. Code amended so hand crop size can be adjusted by changing one variable. Renamed ‘gestureRecognizer’ to ‘gestureRecognizerColor’ and created ‘gestureRecognizerDepth’ based on the depth feed.</td>
</tr>
<tr>
<td>2.6</td>
<td>28/6/2016</td>
<td>‘gestureRecognizerDepth’ calculates when a hand is not moving, but a gesture is taking place - these points of the gesture are used as the key frames defining the gesture (the gesture nuclei).</td>
</tr>
<tr>
<td>2.7</td>
<td>31/07/2016</td>
<td>‘gestureRecognizerDepth’ modified to convert body points to depth space, and save converted points to CSV file for easy training.</td>
</tr>
<tr>
<td>2.8</td>
<td>04/08/2016</td>
<td>‘gestureRecognizerDepth’ modified to save raw skeletal points to CSV file for easy training.</td>
</tr>
</tbody>
</table>
| 2.9     | 19/08/2016 | ‘gestureRecognizerDepth’ modified to crop the highest hand image (the active hand performing the gesture). Crop size determined by the inverse of distance to the camera (Z value of hand), image then scaled to a set size for neural net processing (60 * 60 pix). Image also normalized according to min and max value in image, in order to utilize the full byte range (0-255). Added function to segment player from depth image, so that unnecessary information (floor plane and
<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.0</td>
<td>19/08/2016</td>
<td>Added bone length normalization to incoming skeletal data - translated joints so SpineMid is at 0,0,0 and scaled bone lengths to a set of reference bone lengths. Added pre-trained SVM (18,000 normalized skeletal frames) to 'gestureRecognizerDepth' to determine when a gesture is taking place.</td>
</tr>
<tr>
<td>3.1</td>
<td>22/08/2016</td>
<td>Modified previously implemented circular queue to hold a range of skeletal positions at previous timesteps. Wrote code to add each newly received skeletal frame to circular queue. Started implementing a method in 'circularqueue' to convert skeletal frames to byte array.</td>
</tr>
<tr>
<td>3.2</td>
<td>23/08/2016</td>
<td>Wrote code to further normalize the skeletal data (by scaling by the min and max of X,Y,Z independently). Also wrote code to turn normalized skeletal data into the same format as an image, and add it to the top of the hand depth image (this acts as the temporal element of the gesture).</td>
</tr>
<tr>
<td>4.0</td>
<td>07/09/2016</td>
<td>Initial upload of new approach. Modified the GUI, added 'gestureadder' class and 'gesturerecognizer' class.</td>
</tr>
<tr>
<td>4.1</td>
<td>08/09/2016</td>
<td>Initial attempt at skeletal normalization by stretching bone lengths to a template.</td>
</tr>
<tr>
<td>4.2</td>
<td>08/09/2016</td>
<td>Implemented a much more accurate and less processor intensive skeleton normalization procedure.</td>
</tr>
<tr>
<td>4.3</td>
<td>09/09/2016</td>
<td>Implemented process to capture a new gesture - to capture 3 sequences of the same gesture. A new HMM is then trained for the gesture and added to the models list. Models class has been created to store and evaluate the gesture model HMM's.</td>
</tr>
<tr>
<td>4.4</td>
<td>12/09/2016</td>
<td>Bugs need to be sorted out but appears to be first completely working archive. Added to 'gesturerecognizer' code to detect when hands are moving, stores the movement sequence and evaluates.</td>
</tr>
</tbody>
</table>
the sequence on the recorded gesture HMM's. Sequences too short to be considered a gesture are ignored.

**BUG***Occasionally skeleton will not be picked up at all when recording a gesture***

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.5</td>
<td>21/09/2016</td>
<td>Modified ‘gesturerecognizer’ to load models on startup. Also models evaluate on movement and return most likely class name. Detected class name now prints to screen, and gesture names are imported from gesture models.</td>
</tr>
<tr>
<td>5.0</td>
<td>24/09/2016</td>
<td>‘GestureRecognizer’ now working as a complete system, ready for demo. Includes six pre-trained gesture models for demo evaluation.</td>
</tr>
<tr>
<td>5.1</td>
<td>27/09/2016</td>
<td>‘GestureRecognizer’ modified to work with distances and angles, can also work at many different camera heights, and viewpoints of ~\pm 40 degrees</td>
</tr>
</tbody>
</table>