A System for a Hand Gesture-Manipulated Virtual Reality Environment

Andrew Clark
UKZN/CSIR Meraka Centre for Artificial Intelligence Research (CAIR)
University of KwaZulu-Natal, Westville Campus, Durban, South Africa
andrewclark.1905@gmail.com

Deshendran Moodley
CSIR Meraka Centre for Artificial Intelligence Research (CAIR)
University of Cape Town, Cape Town, South Africa
deshen@cs.uct.ac.za

ABSTRACT

Extensive research has been done using machine learning techniques for hand gesture recognition (HGR) using camera-based devices; such as the Leap Motion Controller (LMC). However, limited research has investigated machine learning techniques for HGR in virtual reality applications (VR). This paper reports on the design, implementation, and evaluation of a static HGR system for VR applications using the LMC. The gesture recognition system incorporated a lightweight feature vector of five normalized tip-to-palm distances and a k-nearest neighbour (kNN) classifier. The system was evaluated in terms of response time, accuracy and usability using a case-study VR stellar data visualization application created in the Unreal Engine 4. An average gesture classification time of 0.057ms with an accuracy of 82.5% was achieved on four distinct gestures, which is comparable with previous results from Sign Language recognition systems. This shows the potential of HGR machine learning techniques applied to VR, which were previously applied to non-VR scenarios such as Sign Language recognition.

CCS Concepts

•**Human-centered computing** → **Gestural input;**
•**Computing methodologies** → *Virtual reality;*

Keywords

Virtual Reality; Hand Gesture Recognition; Machine Learning; Leap Motion Controller

1. INTRODUCTION

Virtual reality devices for the PC, such as the HTC Vive and Oculus Rift, both employ the use of hand-held remote-based devices to interact in a virtual world. These devices accurately track positional hand data, but partly rely on button presses instead of gestures for VR interaction. Future VR systems are expected to evolve towards natural hand gestures as the primary input, thus there is a need to explore different methods of capturing hand data. One such method is a camera-based method, where image recognition techniques and classification methods are combined to recognize a hand gesture. Camera-based methods are able to track the posture of individual fingers while not encumbering users' hands, thus possibly replacing cumbersome hand-held remotes.

Figure 1: The Oculus Rift DK2 with the LMC and its mount.

The LMC and Microsoft Kinect are two popular depth cameras used in research for camera-based HGR. The Leap Motion Controller (LMC) is a lightweight and affordable commercial stereo infrared camera that specializes in tracking a user's hands with a high degree of accuracy. The LMC has its own infrared LEDs that illuminate the hand, meaning that it can work under any lighting conditions and with any skin colour. The device also comes with a VR mount, allowing it to be attached to the front of the Oculus Rift, as seen in figures 1 and 2.

The Microsoft Kinect is a depth camera that specializes in tracking a user's body, but not individual fingers. Users that wish to perform HGR using the Kinect have to either access the raw image data provided or only use the provided positions of the hands, whereas the LMC provides a level of abstraction by giving users a skeleton model of the hands it tracks.

Hand gestures can be either static or dynamic [6]. Static gestures are simply the posture of the hand at a single point in time while dynamic gestures can have the posture and position of the hand change over time. Classification of these gestures usually involve deciding which data points or fea-

SAICSIT '16, September 26-28, 2016, Johannesburg, South Africa
 ISBN 978-1-4503-4805-8/16/09…$15.00
DOI: http://dx.doi.org/10.1145/2987491.2987511

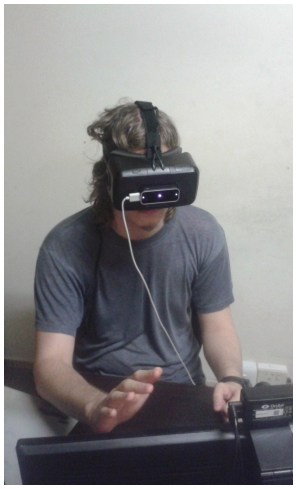

Figure 2: A user of the system wearing the Oculus Rift with the LMC mounted on it.

tures accurately describe a gesture, followed by training a machine learning system to recognize the gesture using these features. The trained system should then be able to extract the necessary features from the testing data and classify the test gesture accordingly.

A review of current literature has revealed that many researchers have used the LMC to classify Sign Language gestures, but very few have applied it to VR. Those that used the LMC in VR did not create a system to classify any arbitrary static gesture made using machine learning algorithms. This system was created to explore the efficacy of machine learning techniques for HGR in VR applications.

The objective of this system is to develop a static HGR system that can be used in a VR case-study environment that users can interact with using gestures only. The system needs to be flexible enough to recognize any static gesture, and thus must incorporate a machine learning algorithm. The VR world is displayed on the Oculus Rift DK2 and hand data is provided by the LMC, which is mounted onto the Oculus Rift. The system is evaluated with regards to recognition accuracy, recognition latency and extendability. The latency and accuracy metrics are important factors in 3D gestural interaction [20], and the latency metric shall also provide insight into the impact the system has on the performance of the VR application. A high recognition latency could cause the framerate of the application to stutter, which in turn may cause users to experience cybersickness [21].

In section 2, this paper will outline the state of the art in HGR research, as well as identify the limitations in existing systems. This is followed by section 3, an overview of the architecture of the static HGR system for VR applications and the evaluation of this system in a case study stellar data visualization application. In section 4, the results obtained from the application incorporating the system are stated, followed by a discussion in section 5, and ended in the conclusion in section 6.

2. RELATED WORK

This section discusses previous work done in the field of static HGR. The review will start off with a discussion of popular devices used for static HGR in literature. This is followed by how static HGR has been applied in research, and the remaining challenges in the more relevant applications will be discussed. Finally, the popular machine learning algorithms for static HGR will be discussed and compared, and specific features used in literature will be mentioned.

2.1 Devices

A variety of input devices have been used in HGR research, the more popular amongst those include gloves and depth cameras. Popular depth camera-based devices for HGR in research include the Microsoft Kinect and the LMC, while glove-based research usually utilizes the CyberGlove [33, 34]. Glove-based devices, such as the CyberGlove, are cumbersome and not as easily available as cameras are, but do not suffer from occlusion as cameras do. This subsection briefly describes the Oculus Rift head-mounted display, then goes on to describe the use of the LMC and Kinect in research, before substantiating the choice of using the LMC over the Kinect.

The Oculus Rift is a head-mounted display (HMD) that displays a 3D application's graphics stereoscopically with a wide field of view. The device is able to track changes in users' head position and orientation, causing the application to update its camera's position and orientation accordingly to make the user feel as though they are physically looking around.

The Kinect API does not provide data pertaining to individual fingers, but rather only the positions of the hands and other parts of the body. Thus, the use of the Kinect in HGR research usually involves processing the captured depth image for gesture classification, or simply using the calculated position of the hands. Kim et al. used the Kinect to manipulate virtual objects using the positions of the hands from the API [19], while Messaci et al. used the Nimble SDK to detect hand joint positions from the Kinect depth image to interact with virtual objects [24]. Trigueros et al. compared various machine learning techniques for classification of gestures using the Kinect, where a highest average accuracy of 96.99% was achieved using a neural network [31]. Various researchers have used the Kinect for Sign Language recognition, achieving accuracies of over 80%. [2, 23, 30]

The LMC specializes in hand detection. The Leap driver calculates features such as palm and joint positions and makes these features available through its API. The LMC has been shown to have sub-millimeter accuracy [14], and Weichert et al. state that the Kinect is not able to achieve such an accuracy [32]. The LMC has been applied to Sign Language recognition [7, 11, 23, 25] with success rates comparable to the Kinect. Other research using the device include VR and 3D application interaction, which is discussed in the next subsection.

The LMC has been deemed more suitable for this research than the Kinect, due to its compatability with VR using a HMD mount and the higher level of abstraction provided by its API.

2.2 Applications of HGR in Literature

HGR has been applied to a wide array of fields, such as in the control for the training simulation of a self-propelled gun [34], Sign Language recognition [7, 11, 23, 25, 28], navigation through a VR world [18], control of a 3D visualization application [29], the treatment of amblyopia [4], data visu-

alization [9], and the playing of virtual instruments [15] and 3D games [27].
Multiple researchers have investigated the use of HGR in a VR application: Donalek et al. used the LMC placed on a table and the Oculus Rift for data visualization [9], while Beattie et al. had a similar setup as Donalek et al., but applied to Computer Aided Design programs [3]. Khattak et al. presented a framework for immersive augmented VR interaction with virtual objects with the LMC placed on a table [17], and Khundam mounted the LMC on the Oculus for gesture-controlled movement in a VR world [18]. All of these researchers used the Oculus and LMC for interaction in a VR application through HGR, however no information is given about recognition mechanisms over and above the data received directly from the LMC API. It is assumed that these systems did not incorporate machine learning algorithms for HGR.
Researchers have also investigated the use of HGR in 3D non-VR applications. Investigations include the control of a virtual scene camera using the LMC [13], the control of a 3D molecular graphic system using the LMC and Kinect as separate inputs [29] and virtual 3D object manipulation [16, 19, 24]. As with VR HGR research, these papers do not mention the use of machine learning for arbitrary static gesture detection. However, there does exist machine learning-based HGR research for 3D applications when it comes to raw image processing, where Dardas and Alhaj implemented a support vector machine to recognize static gestures captured with a webcam [8].
Another one of the predominant applications of HGR in literature is Sign Language recognition. Unlike VR interaction gestures, the static gestures used in Sign Languages are often complex and numerous, thus requiring the use of machine learning algorithms to classify arbitrary static gestures. Machine learning approaches to classify Sign Language static gestures have generally achieved over 80% accuracy rates. Results such as these show the potential of using machine learning algorithms to classify arbitrary static gestures to be used in VR environments.

2.3 Algorithms and Feature Sets

Researchers in the field of Sign Language recognition often employ machine learning algorithms for static gesture recognition, and are met with usually favourable results. This subsection encompasses the algorithms and features they used, and motivates the choice for the feature extraction and classification algorithms.
In terms of the Kinect, researchers have often utilized the Support Vector Machine (SVM), k-nearest neighbour (kNN) algorithm [10], Artificial Neural Networks (ANN), the Naïve Bayes classifier, and Linear Discriminant Analysis (LDA) [2, 23, 30, 31]. When it comes to classification using the LMC, Elons et al. achieved an 88% recognition rate using an ANN [11], while Chuan et al. achieved a 72.78% and 79.83% using the kNN algorithm and SVM respectively [7]. An accuracy of 91.28% was observed when Marin et al. used both the Kinect and LMC with an SVM as a classifier [23]. Mohandes et al. attempted to limit the impact of occlusion by using two LMCs for Sign Language recognition, and achieved a success rate of 97.7% using LDA for classification [25].
From the above, it is clear that the ANN, kNN algorithm, SVM and LDA are all popular machine learning techniques for static HGR. Of these techniques, the kNN algorithm is quick to implement thanks to the value of k being the only parameter. The algorithm also runs quickly with a small enough dataset without much of an accuracy tradeoff. Note that the accuracy rates found in Sign Language recognition may not apply in the same manner to HGR in VR, primarily due to the freedom of choosing one's own gesture set that separates the feature vectors well. Furthermore, when the LMC is mounted onto the Oculus Rift, it is expected that more inaccuracies will occur since the LMC is no longer stationary and it views the back of the hand, causing curled fingers to be obscured.
In terms of features, it is important to choose a feature set that both describes a gesture well, but is at the same time hand size invariant and lightweight enough to reduce classification time. One of the features used by Marin et al. involved the distance from the palm to each of the fingertips divided by the length of the middle finger [23]. A set of 5 float values corresponding to these normalized distances should be able to describe a gesture well enough, and is lightweight as well as hand orientation and size independent. This feature set is described in detail in section 3.1.1.

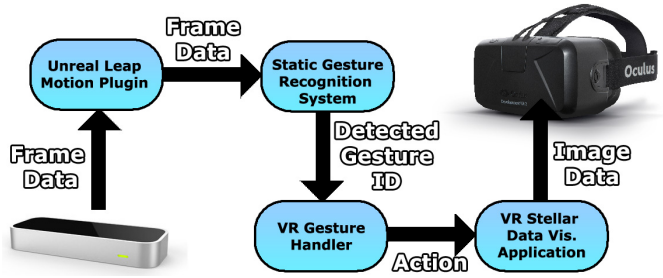

Figure 3: An abstracted view of the architecture.

3. PROPOSED SYSTEM AND APPLICATION

The architecture of the system and application follows the illustration seen in figure 3. The LMC collects and data and processes it into frame data consisting of positions and orientations of fingers and palms amongst other information. This data is sent to the Leap plugin in the Unreal Engine. The static gesture recognition system, implemented in the Unreal Engine, takes the frame data provided through the plugin and applies machine learning algorithms for HGR to the data to classify which gesture is being made. The classified gesture ID is then passed to the VR gesture handler, which interprets the gesture to an action in the VR stellar data visualization application according to the current state of the application.
The case-study application for the static HGR system is a stellar data visualization application developed in the Unreal Engine 4 game engine. This was done to demonstrate the use of this system in a real-world scenario, which may affect accuracy and latency measures. Game engines, such as the Unreal Engine 4 and Unity, have replaced VR-specific rendering engines due to their flexibility and wide array of tools for creating interactive applications quickly and easily [1]. The application places users in space, surrounded by virtual stars. The user of the system is seated in front of the PC monitor while wearing the Oculus Rift with the LMC mounted in front of it. By using gestures only, they are able to view more information about a particular star,

revolve all the stars around themselves and toggle the way stars are represented spatially.

3.1 Static Gesture Recognition System

This system applies the kNN algorithm [10] to the data received from the Leap plugin in the Unreal Engine 4 to detect the gesture currently being made. Data received from the plugin is first processed into a lightweight feature vector before being classified into a gesture by the kNN algorithm. Figure 4 illustrates the static recognition process. The feature extraction and normalization as well as the gesture classification processes are described in the subsections below.

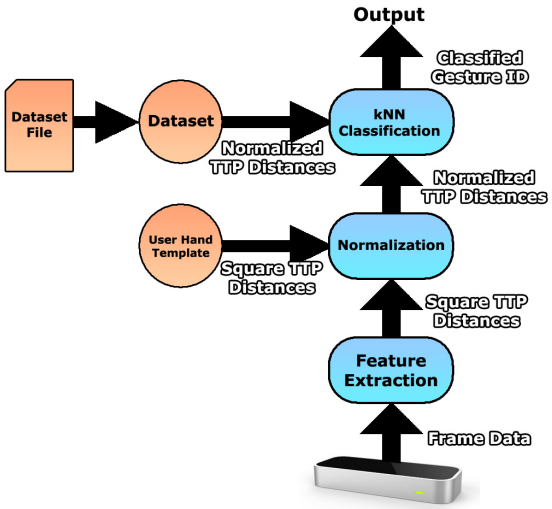

Figure 4: The workflow of the static gesture recognition system.

3.1.1 Feature Extraction

Machine learning algorithms usually require certain features to be extracted from the input data to be classified. The feature set used in this system was inspired by the research of Marin et al. [23]. One of the features used in their research was a set of 5 normalized distances from the tips of each finger to the center of the palm. These distances will be referred to as tip-to-palm (TTP) distances. The TTP distances in the research of Marin et al. were normalized by dividing each distance by the maximum TTP distance of the middle finger, thus constraining the values between 0 and 1. The maximum middle finger distance is known as a scaling factor, and is recorded at the start of the system.

In this study, the feature vector shall contain five normalized TTP distances corresponding to each finger, where the normalization process divides the TTP distance of each finger by the maximum distance of the same finger, instead of the middle finger. The feature vector will thus describe the proportion by which each finger is extended. During an individual user calibration step, the user is prompted to hold their outstretched hand in front of the LMC. The length of each outstretched finger is measured, and the system records the TTP distance of each finger from this display. This measurement vector is the user hand template, and every gesture made thereafter will have its TTP distances divided by the template. The user hand template is specific to a certain user, and is recorded once for each session in the VR application. The normalized TTP distance of finger i in feature vector $\vec{v}$ is determined as follows:

$$v_i = \frac{1}{t_i} \| \vec{f_i} - \vec{p} \| \tag{1}$$

The vectors $\vec{f_i}$ and $\vec{p}$ refer to the positions of the tip of finger i and the center of the palm with respect to the LMC respectively, both of which are supplied through the Leap API. The scalar t_i refers to the user hand template's TTP distance of finger i, which has been stored since the start of the system.

The use of these 5-dimensional feature vectors make the gestures orientation-independent and are able to accurately describe the hand posture. While more features could be added to improve accuracy, keeping the feature vectors as lightweight as possible will assist in speeding up the entire classification process.

Calculating the magnitude of a vector requires a square root operation, which can be computationally expensive, especially when performed during every update step of the application. To alleviate this issue, the square of the magnitude of the TTP distance is used to approximate the square root. This also means that the user hand template will need to have its magnitudes squared, thus resulting in the normalized distances no longer being scaled linearly. However, by avoiding a square root operation, the system's performance is improved. The following equation describes the amended feature construction process:

$$v_i = \frac{1}{T_i} \left((f_{ix} - p_x)^2 + (f_{iy} - p_y)^2 + (f_{iz} - p_z)^2 \right) \tag{2}$$

Where T_i is the square TTP distance of finger i in the user hand template. This square value is stored in memory to be used in normalization. Figure 5 illustrates the normalization process of feature extraction.

3.1.2 Classification Algorithm

The kNN algorithm [10] is used to classify gestures due to its low complexity, high accuracy, and its speed with a small dataset. A suitably low value of $k = 3$ was chosen, due to the small dataset size and simplicity of the gestures in the application. The algorithm is also quick to implement, meaning that more time could be allocated to creating a realistic case-study application.

The kNN algorithm never rejects an input for being too different from the dataset entries, and always returns a classified input. Thus, gestures that are not in the dataset will be classified incorrectly, instead of being rejected completely. When it comes to VR it is not appropriate for the application to respond incorrectly to a gesture that should have been classified as unknown. To this end, the kNN algorithm was adapted to reject certain input gestures should they be too different from the training gestures. This was implemented through a simple threshold function, where if the Manhattan distance to the test gesture's k nearest neighbours all exceed the threshold (found empirically), then the test gesture is classified as unknown and the system does not respond to it.

In order for gesture types in the dataset to persist between sessions, a dataset file is stored on disk. Whenever the ap-

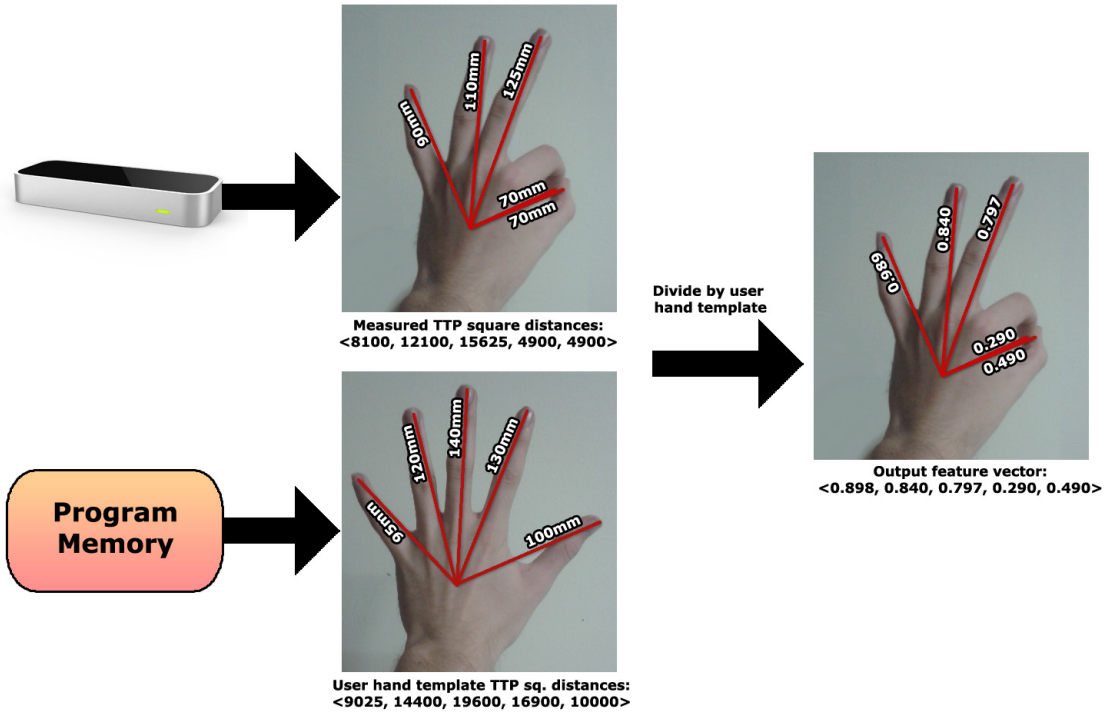

Figure 5: The normalization process of TTP distances. Note that the LMC does not provide the distances illustrated on the left, but rather square distances are directly calculated from the palm and fingertip vectors provided (See equation 2).

plication starts up, this file is read into working memory for the kNN algorithm to use. The dataset in working memory can be written back to the file on request if a new gesture type is introduced. Figure 4 illustrates the flow of data in the static gesture recognition system.

3.2 VR Gesture Handler

The gesture handler is a component embedded in the application that receives a classified gesture and performs the appropriate action on the VR environment. While the static gesture recognition system and the stellar VR application could both exist as stand-alone systems, this component links the two into a cohesive VR HGR application. Four distinct static gestures were selected for control for the case-study application. These gestures are seen in figure 6. These gestures were chosen as they are easily distinguishable from each other when using the feature set mentioned in section 3.1.1. Some actions require a series of static gestures, while others require a single static gesture.

Below are the actions that users can perform in the case-study application:

1. **Revolve Stars:** Since the user is seated, it is difficult to turn around and interact with stars placed behind. To combat this problem, the user can use the point gesture to revolve the stars until a desired star is in their field of view. Pointing to the left revolves them to the left, and vise-versa when pointing right. Determining whether the user was pointing left or right was done by computing the dot product of the vector from the user's palm to their index finger with their camera's right vector. A value above zero indicates they are pointing right. The stars only revolve while the pointing gesture is being made, and immediately stops when the user stops pointing.

Figure 6: The recognizable static gestures in the system. From left to right are the Fist, Point, Open Hand and OK gestures.

2. **Toggle Arrangement of Stars:** To demonstrate the use of data visualization, a mechanism was introduced to rearrange the stars in such a way that their spatial positions mean something else entirely. Whenever the OK gesture is made, the stars toggle between spatial representation and HR representation. HR representation refers to a Hertzsprung-Russell diagram, which is a 2-dimensional plot where a high y-value indicates high luminosity, and a low x-value indicates high temperature. Figure 7 illustrates an HR diagram. From the diagram, it is easy to separate stars that are of different sizes, such as dwarf stars from supergiants.

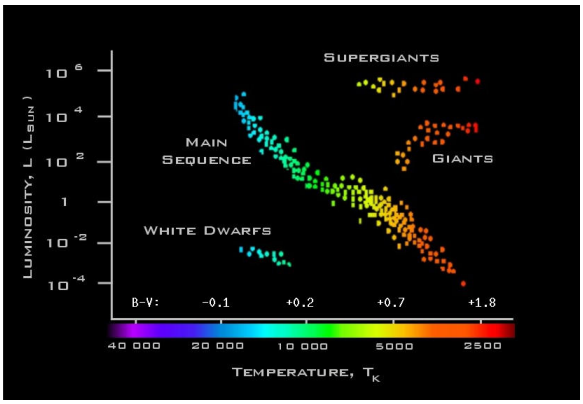

Figure 7: An HR diagram.[1]

3. **View Star Details:** The main action the user can perform is one in which they view more details about a star. While simply looking at a star only displays its name, performing this action will allow the user to view its temperature and other attributes. This action was broken up into the following steps:

   - The user keeps the open hand gesture while reaching out past a certain threshold towards the star.
   - The user makes the fist gesture while the hand still reaches.
   - The user brings their fist in close behind the threshold.
   - The user opens their hand again.

   Together, this series of static gestures form a grasping motion, where the user reaches out, grabs a star and pulls it close. Upon releasing, an enlarged version of the star is shown along with its detailed attributes. To stop viewing these extra details, the user does the following:

   - The user performs the open hand gesture behind the threshold distance.
   - The user then moves their open hand past the threshold distance. This completes a "pushing away" motion.

   To control this series of static gestures, a finite state machine was implemented to divide each step into a state. Figure 8 depicts the structure of this state machine. The user starts in the idle state and transitions to the star details state upon "pulling" a star in. The "close" and "far" suffixes indicate whether the hand is behind or past the threshold value.

When it comes to performing these actions, there is a need for the system to know when a gesture detect operation has to be performed without the user manually requesting for one. Since actions such as revolving the stars and viewing star details require constant checking whether the hand

[1]From www.le.ac.uk/ph/faulkes/web/images/hrcolour.jpg

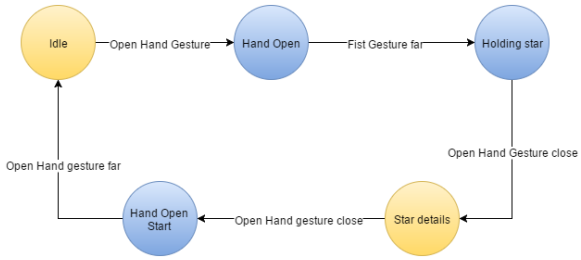

Figure 8: The state machine that controls the pulling in and pushing away gestures.

is still in the correct pose, it was decided that a gesture detect has to occur at regular intervals. More specifically, every update of the game loop in the Unreal Engine 4 calls a gesture detect operation on the user's closest hand, and appropriate actions are performed depending on what state the application is in and what gesture was detected. Every classified gesture is handled by both the state machine for the View Star Details action and the static gesture handler, that deals with the Revolve Stars and Toggle Arrangement actions. Figure 9 depicts this process within the full architecture of the system.

3.3 VR Stellar Data Visualization Application

The user is in a fixed position and given a first person view of the world. They are represented by an avatar, where the avatar's hands mimic the data that the LMC receives, creating the illusion that the user is present in this world. These virtual hands and the algorithms controlling them are provided by a LMC plugin for the Unreal Engine 4. Figure 10 shows the user's perspective and avatar in the world.
One of the many uses of VR technology is for advanced data visualization, and the case-study application depicts a simplified version of such a case. Each entry of data in the application is 7-dimensional, and represents the data of a single star. Each star or data entry has:

- x, y and z co-ordinate (position).
- Name
- Temperature
- Luminosity
- Radius

All stars are represented as glowing spheres at their respective positions. Hotter stars are blue while colder stars are red. More luminous stars glow brighter and larger stars are represented as larger spheres. A star is automatically selected whenever a user looks at it. A selected star displays its name on a 3D user interface, indicating to the user that they may perform a manipulation task on it.

4. TESTING AND RESULTS

The system and application were tested with respect to latency, accuracy, extendability, ease of use and comfort. Each of the gestures seen in Figure 6 are made twenty times each by the researcher, with some variation between each, and are placed into the kNN dataset. All tests took place

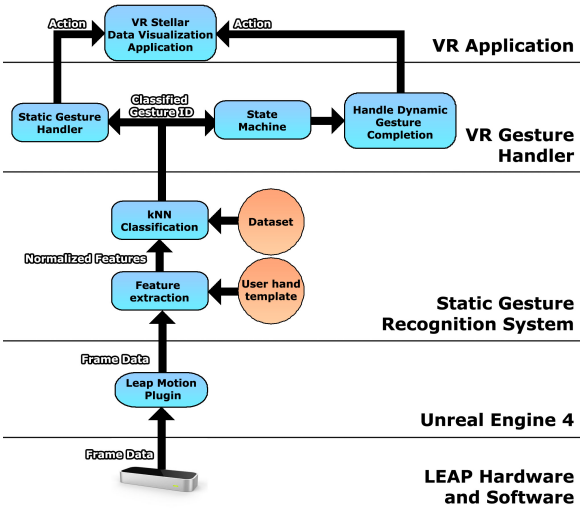

Figure 9: Detailed architecture of the system.

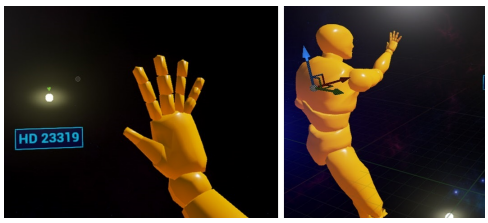

Figure 10: The user avatar. Left: The world from the user's perspective when their hand reaches out. Right: The same user action but from outside the avatar.

on a Windows 8.1 PC with an Intel Core i5-3570K 3.4GHz CPU, an AMD Radeon HD 7970 GHz Edition GPU and 16GB of DDR3 RAM.

4.1 Latency Test

The latency for the classification of a static gesture is the time from when classification is requested to the time the classified gesture is returned. It is assumed the classification for a static gesture is the same regardless of hand pose, since the kNN algorithm has to run through every dataset entry regardless. The fist gesture was made and classified 1000 times over for this test. This test was performed in the VR application to properly measure latencies in a practical scenario. The maximum and minimum measured latencies were 0.227ms and 0.037ms respectively, while the average latency was 0.057ms.

4.2 Accuracy Test

Each of the 4 gestures seen in figure 6 were made ten times each with a rest period in between. The testing user would raise their hand in the correct pose in the VR application, regardless of whether their avatar's hand was in the correct pose, before the gesture is classified. The user then lowers their hand before raising it again for the next classification. The fist, point, open hand and OK gestures were correctly classified 100%, 60%, 80% and 90% of the time respectively, resulting in an average accuracy of 82.5% across all gesture types. The incorrect classifications were all classified as the "unknown" gesture, which performs no action in the application.

Table 1: Number of Attempts with no Practice

Participant	Task 1	Task 2	Task 3	Task 4	Task 5
1	1	1	3	2	1
2	3	1	7	6	2
3	2	2	1	3	1

Table 2: Number of Attempts after Practice

Paricipant	Task 1	Task 2	Task 3	Task 4	Task 5
1	2	1	2	2	2
2	2	1	3	5	1
3	1	1	2	3	1

4.3 Extendability Test

To identify how well the small dataset performs when a new gesture type is added, the "L" gesture was introduced. This is made by extending the thumb and index fingers only in the shape of an L. Five of the L gestures were added to the dataset, and the gesture was tested ten times over to measure classification accuracy, where a 70% accuracy was achieved.

4.4 Preliminary User Study

Three new users, two men and a woman of varying hand sizes, were included in the study to measure the ease of use and comfort of the system. They were given the following tasks:

- Revolve the stars to the left, then right.
- Perform the Toggle Arrangement of Stars action to view the HR representation.
- Perform the View Star Details action on the hottest star.
- Stop viewing the star details by pushing that star away.
- Perform the Toggle Arrangement of Stars action to return to spatial representation.

The number of attempts taken to perform each task was recorded. Afterwards, users may practice the tasks in their own time for a while before going onto a second round of the same tasks. Tables 1 and 2 show the recorded number of attempts from the first and second round respectively. The users were also asked a series of questions relating to comfort and ease of use. These questions are as follows:

- Did you feel the gesture recognition occurred in real time or was there a noticeable delay?
- Do you feel the gestures were easy enough to perform to complete each task? If not, do you feel you could perform better with practice?
- Did you feel motion sick or claustrophobic at all during the test?

Below are the findings of this preliminary study:

1. Participants showed a general decrease, albeit with some slight increases, in the number of attempts required to get a task right after having at most 2 minutes of practice.

2. Plasters on the fingers as well as rings and bracelets have a negative impact on the LMC's accuracy. After removing these objects, an improvement was observed.

3. None of the participants observed noticeable delays in gesture recognition, and did not feel any symptoms of cybersickness or claustrophobia.

5. DISCUSSION

Since little research has been done in the field of applying the LMC for HGR in VR, the results obtained are compared to HGR for Sign Language instead. Chuan et al. [7] achieved a 72.78% accuracy using the LMC and kNN classifier [10] for American Sign Language recognition. While the results obtained in the case-study application show an average accuracy of 82.5%, it is important to note that American Sign Language contains multiple gestures that are similar, and could cause incorrect classifications. At the same time, the gesture set used in this VR application are very distinct, and are easily separable using the TTP feature set. The point gesture had the lowest accuracy of 60%, which occurred whenever the LMC represented the hand incorrectly. This is likely due to the LMC software struggling to ascertain which finger is being raised, since there are no other fingers raised to provide a point of reference.

It is important to note that good classification techniques applied to Sign Language may not be applied just as effectively to VR. While the general goal in both cases is to maximize accuracy, there is also an increased need for resource-light classifiers in VR. Additionally, gesture recognition in VR using the LMC involves the camera being mounted on the front of the HMD, thus allowing the camera to only view the back of the user's hand. This causes issues when the fingers are bent down and are obscured by the palm. Sign Language recognition on the other hand usually has the camera facing the palm-side of the hand, which removes the issue of palm occlusion. Finally, in Sign Language recognition, a fixed set of signs are available for researchers to recognize, and they must adapt their algorithms to suit the signs in the language. However, HGR in VR allows more flexibility in the sense that one may adapt the gestures to suit the algorithms and feature sets. By doing this, one can dramatically increase the accuracy.

If one were to attempt to create a HGR system to classify a pre-defined set of gestures instead, the system presented in this paper may not be well-suited. The feature set used in this research is relatively simple and may not be able to separate more complex gestures. For example, the feature set will easily distinguish an extended finger from a curled finger, but won't be able to distinguish different hand orientations, such as a thumbs-up versus a thumbs-down gesture. A more generic HGR system is expected to use more complex feature sets that separate the chosen gestures well.

The average latency of 0.057ms from a request for a gesture detect to the completion of the classification shows that the system is fast enough to run in a VR application. To put this in perspective, it takes more than a 100ms delay for users to notice a delay in computer interaction [5]. The minimal impact on performance of the classification process allows for gesture detection to occur every application update step without dropping the framerate below the recommended value of 75 frames per second [12], which would cause user discomfort. Furthermore, users of the system in a preliminary user study all stated that they felt no noticeable delay during their interactions.

The speed of the kNN algorithm used for classification can be attributed to the lightweight feature vectors and small size of the dataset, which consisted of 80 entries. This algorithm has to iterate over each entry in the dataset, and therefore there could be a drop in classification speed with very large datasets. However, adding a small amount of entries should not affect classification speed drastically. The extendability test demonstrated that just by entering the "L" gesture five times over, the system was still able to classify the new gesture correctly most of the time. Machine learning algorithms used in literature that do not have their classification speed dependent on the dataset size, such as an ANN or SVM, can be explored for VR HGR.

The feature vectors, while being lightweight, are unable to represent hand orientation. While this may be good in some cases where orientation independence is important, other cases where gestures are orientation dependent will have issues. One such case is differentiating the thumbs-up and thumbs-down gesture; they can both be described by the same feature vector used in this investigation, even though they should be classified as different gesture types.

While these results show a high accuracy, it is uncertain whether cameras will replace remote controls as the means for VR interaction. A remote-based approach is always accurate as it does not suffer from occlusion, however cameras do not encumber the user with hand-held devices and allow for a freer range of movement. It is possible that a combination of multiple cameras (such as a dual-LMC setup [25]) or other devices would improve the accuracy to the point that they become more viable than remote-based devices. However, the cost and lack of mobility when using multiple cameras may prove to be prohibitive.

6. CONCLUSION AND FUTURE WORK

This paper describes the creation of a system for machine learning-based HGR in a VR world, as there is a need to analyze how well the kNN classifier, which is widely used for static HGR in non-VR scenarios, performs in VR applications. The Oculus Rift DK2 was used to render the case-study VR world built using the Unreal Engine 4, while the Oculus-mounted LMC was used to detect the user's hands. The system was applied to a 3D stellar data visualization application controlled entirely by hand gestures. The system needed to be able to learn and classify any static gesture made in real-time as well as classify two predefined application-specific dynamic gestures in real-time. The system was found to be extendable enough to accurately recognize any newly learned static gesture, and the static gesture recognition algorithm took 0.057ms on average to run. This demonstrates the fact that the system is lightweight enough to allow the program to run at a high framerate, which is an important consideration to take when developing for VR. A preliminary user study was taken out on three new users, all of whom observed no latency or discomfort. Static gestures used in the application had an 82.5% accu-

racy rate, with a minimum of 60% accuracy with the point gesture and a maximum of 100% accuracy with the fist gesture. Gestures with low accuracy ratings primarily suffered from occlusion issues. The sample size of three users in the preliminary study cannot prove that the system is effective for all users. It does show some potential of the system, but further testing will be required. Leap Motion have released a beta software update for the LMC called Orion, which is built specifically for hand detection in VR [22]. Using Orion in future work may reduce occlusion complications and improve accuracy. Other means of improving accuracy could include augmenting the LMC input with other data, such as that captured by a webcam, the MYO [26], or using an additional LMC placed on a different axis. The use of ANNs and SVMs could be analyzed with respect to classification speed and accuracy in VR. The low computation time also suggests that more complex features could be used for a higher recognition accuracy with an acceptable latency.

7. ACKNOWLEDGMENT

The author would like to thank the National Research Foundation (NRF) of South Africa and the UKZN/CSIR Meraka Centre for Artificial Intelligence Research for their financial assistance.

8. APPLICATION DEMONSTRATION

A short demonstration of the application can be seen at https://www.youtube.com/watch?v=gA7nc1TTM2k.

9. REFERENCES

[1] N. Al-Najdawi. Introduction to visualization using game engines. In *AHRC Methods Network Workshop*, 2007.

[2] S. Aliyu, M. Mohandes, M. Deriche, and S. Badran. Arabic sign language recognition using the Microsoft Kinect. In *2016 13th International Multi-Conference on Systems, Signals Devices (SSD)*, pages 301–306, Mar. 2016.

[3] N. Beattie, B. Horan, and S. McKenzie. Taking the LEAP with the Oculus HMD and CAD - Plucking at thin Air? *Procedia Technology*, 20:149–154, Jan. 2015.

[4] J. Blaha and M. Gupta. Diplopia: A virtual reality game designed to help amblyopics. In *Virtual Reality (VR), 2014 iEEE*, pages 163–164, Mar. 2014.

[5] S. K. Card, A. Newell, and T. P. Moran. *The Psychology of Human-Computer Interaction*. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 1983.

[6] L. Chen, F. Wang, H. Deng, and K. Ji. A Survey on Hand Gesture Recognition. In *2013 International Conference on Computer Sciences and Applications (CSA)*, pages 313–316, Dec. 2013.

[7] C.-H. Chuan, E. Regina, and C. Guardino. American Sign Language Recognition Using Leap Motion Sensor. In *2014 13th International Conference on Machine Learning and Applications (ICMLA)*, pages 541–544, Dec. 2014.

[8] N. H. Dardas and M. Alhaj. Hand Gesture Interaction with a 3d Virtual Environment. In *ResearchGate*, Sept. 2011.

[9] C. Donalek, S. Djorgovski, A. Cioc, A. Wang, J. Zhang, E. Lawler, S. Yeh, A. Mahabal, M. Graham, A. Drake, S. Davidoff, J. Norris, and G. Longo. Immersive and collaborative data visualization using virtual reality platforms. In *2014 IEEE International Conference on Big Data (Big Data)*, pages 609–614, Oct. 2014.

[10] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification (2nd Edition)*. Wiley-Interscience, 2000.

[11] A. Elons, M. Ahmed, H. Shedid, and M. Tolba. Arabic sign language recognition using leap motion sensor. In *2014 9th International Conference on Computer Engineering Systems (ICCES)*, pages 368–373, Dec. 2014.

[12] Epic Games. Oculus rift - epic wiki. https://wiki.unrealengine.com/Oculus_Rift#Simulation_Sickness. Accessed: 2016-06-17.

[13] B. Fanini. A 3d Interface to Explore and Manipulate multi-scale Virtual Scenes using the Leap Motion Controller. In *ACHI 2014 : The Seventh International Conference on Advances in Computer-Human Interactions*, 2014.

[14] J. Guna, G. Jakus, M. Pogačnik, S. Tomažič, and J. Sodnik. An Analysis of the Precision and Reliability of the Leap Motion Sensor and Its Suitability for Static and Dynamic Tracking. *Sensors*, 14(2):3702–3720, Feb. 2014.

[15] M. H. Hsu, T. Shih, and J. S. Chiang. Real-Time Finger Tracking for Virtual Instruments. In *2014 7th International Conference on Ubi-Media Computing and Workshops (UMEDIA)*, pages 133–138, July 2014.

[16] S. Kerefeyn and S. Maleshkov. Manipulation of virtual objects through a LeapMotion optical sensor. *ResearchGate*, 12(5):52–57, Oct. 2015.

[17] S. Khattak, B. Cowan, I. Chepurna, and A. Hogue. A real-time reconstructed 3d environment augmented with virtual objects rendered with correct occlusion. In *2014 IEEE Games Media Entertainment (GEM)*, pages 1–8, Oct. 2014.

[18] C. Khundam. First person movement control with palm normal and hand gesture interaction in virtual reality. In *2015 12th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pages 325–330, July 2015.

[19] J.-O. Kim, M. Kim, and K.-H. Yoo. Real-Time Hand Gesture-Based Interaction with Objects in 3d Virtual Environments. *International Journal of Multimedia and Ubiquitous Engineering*, 8(6):339–348, Nov. 2013.

[20] J. J. LaViola. 3d Gestural Interaction: The State of the Field. *ISRN Artificial Intelligence*, 2013:1–18, 2013.

[21] J. J. LaViola, Jr. A Discussion of Cybersickness in Virtual Environments. *SIGCHI Bull.*, 32(1):47–56, Jan. 2000.

[22] Leap Motion. Leap motion - orion beta webpage. https://developer.leapmotion.com/orion. Accessed: 2016-05-24.

[23] G. Marin, F. Dominio, and P. Zanuttigh. Hand gesture recognition with leap motion and kinect devices. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 1565–1569, Oct. 2014.

[24] A. Messaci, N. Zenati, A. Bellarbi, and M. Belhocine. 3d interaction techniques using gestures recognition in virtual environment. In *2015 4th International*

Conference on Electrical Engineering (ICEE), pages 1–5, Dec. 2015.

[25] M. Mohandes, S. Aliyu, and M. Deriche. Prototype Arabic Sign language recognition using multi-sensor data fusion of two leap motion controllers. In *2015 12th International Multi-Conference on Systems, Signals Devices (SSD)*, pages 1–6, Mar. 2015.

[26] Myo. Myo Gesture Control Armband. https://www.myo.com/. Accessed: 2016-08-17.

[27] S. S. Rautaray and A. Agrawal. Interaction with virtual game through hand gesture recognition. In *2011 International Conference on Multimedia, Signal Processing and Communication Technologies (IMPACT)*, pages 244–247, Dec. 2011.

[28] J. Rekha and S. Majumder. Hand Gesture Recognition for Sign Language: A New Hybrid Approach. In *ResearchGate*, volume 1, Jan. 2011.

[29] K. Sabir, C. Stolte, B. Tabor, and S. O'Donoghue. The Molecular Control Toolkit: Controlling 3d molecular graphics via gesture and voice. In *2013 IEEE Symposium on Biological Data Visualization (BioVis)*, pages 49–56, Oct. 2013.

[30] V. Tiwari, V. Anand, A. G. Keskar, and V. R. Satpute. Sign language recognition through kinect based depth images and neural network. In *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 194–198, Aug. 2015.

[31] P. Trigueiros, F. Ribeiro, and L. Reis. A comparison of machine learning algorithms applied to hand gesture recognition. In *2012 7th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–6, June 2012.

[32] F. Weichert, D. Bachmann, B. Rudak, and D. Fisseler. Analysis of the Accuracy and Robustness of the Leap Motion Controller. *Sensors*, 13(5):6380–6393, May 2013.

[33] J. Weissmann and R. Salomon. Gesture recognition for virtual reality applications using data gloves and neural networks. In *International Joint Conference on Neural Networks, 1999. IJCNN '99*, volume 3, pages 2043–2046 vol.3, 1999.

[34] D. Xu. A Neural Network Approach for Hand Gesture Recognition in Virtual Reality Driving Training System of SPG. In *18th International Conference on Pattern Recognition, 2006. ICPR 2006*, volume 3, pages 519–522, 2006.