ISSN 2464-4617 (print) WSCG 2016 - 24th Conference on Computer Graphics, Visualization and Computer Vision 2016
ISSN 2464-4625 (CD-ROM)

Enabling Gesture Interaction with 3D Point Cloud

Harrison Cook
School of Computing, Engineering and Mathematics, Western Sydney University
harrison.cook42@gmail.com

Quang Vinh Nguyen
MARCS Institute and School of Computing, Engineering and Mathematics, Western Sydney University
q.nguyen@westernsydney.edu.au

Simeon Simoff
MARCS Institute and School of Computing, Engineering and Mathematics, Western Sydney University
s.simoff@westernsydney.edu.au

Mao Lin Huang
School of Software, Faculty of Engineering & IT, University of Technology, Sydney
Mao.Huang@uts.edu.au

ABSTRACT

This paper presents a novel 3D point cloud gesture recognition system, based on an existing low-cost, accurate and easy to implement 2D point cloud gesture recognition system called $P. Our work improves recognition rates and lowers algorithmic complexity. We develop new 3D gestures, such as the GUN gesture and the SHAKE gesture, while also developing 3D poses like the L pose, OK pose, ROCK pose and PEACE pose for the LeapMotion Device. We demonstrate proposed gesture and pose methods on various 3D environments including a Monsoon mini-game, a cave painting interaction and a target practice scene. The average recognition rates for 3D gestures and poses were compared against the 2D, 3D and 3D+ recognition systems. The results indicate that most gestures in the proposed system were improved in comparison to the existing ones.

Keywords

User Interaction, Gesture Recognition, Finger Interaction, Leap Motion, 3D Point Cloud.

1. INTRODUCTION

Gesture recognition refers to determining when a gesture has occurred and to the general process of determining when a gesture has started and stopped [Yin14]. Natural gestures can be grouped into manipulative and communicative gestures. Manipulative gestures are about moving or interacting with objects, such as pressing a button or rotating an object around, while communicative gestures have the intent of conveying information to others. Communicative gestures can be an interpretation or movement via a symbol or via an act. A gesture via a symbol is often conveyed with a static hand pose and a gesture via an act is determined by the movement of the hand itself.

While natural gestures work in the real world, determining when the user is making a gesture requires our computer implementation to take on a more structured approach of flow and form gestures. A flow gesture could be a continuous gesture, where it progresses over a period of time or series of moments of time. The form gesture can be defined by determining whether the gesture follows a distinct path (such as scrolling a webpage based on position of words) or if it is based on a pose. This research uses the flow and form gestures as a guide to develop our own discrete and continuous gestures to allow the user a range of possibilities.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Hand gesture interaction should be simple enough to understand, while distinct enough so that the user understands which gesture has occurred. For example, uWave is an interactive application using gestures for various mobile devices that have a very small custom gesture recognition system to allow maximum space on the device [Liu09]. We expand this idea for maximum recognition rates while using only a small dataset.

This paper presents gesture recognition techniques that aim to allow better recognition of 2D and 3D gestures by extrapolating gesture sets using similar classifiers. We develop new gesture recognition algorithms that provide a seamless and effective natural interaction at various distances and screen resolutions. The contribution of our paper is as follows.

- New gesture recognition algorithms for fingertip interactions. The gesture pattern recognitions are identified in a simpler and faster way than the available techniques that use databases to store gesture templates for matching, such as [Ren11].
- An expansion of a 2D gesture recognition system [Vat12] into 3D, based on utilising LeapMotion device as input for the gesture recognition system.
- An evaluation of the developed system in terms of user experience and its effectiveness.

2. RELATED WORK

Kinect devices have been useful for medical practices for doctors and staff to interact with patient information without the requirement of the typical mouse and keyboard and the more inaccurate voice recognition [Ebe13]. Using third party libraries such as OpenNI [Pri16] and Libfreenect [Ope16] they were able to implement finger gestures into the OsiriX system [Ebe13]. This was achieved by taking the Kinect depth image, collecting the closest objects to the sensor then approximating the hand and fingers based on this blob. Once the hand and fingers have been recognised from the blob, each point around the hand is recorded and over time is checked with a database of other hands already recorded. However, the use of the Kinect V1 in [Ebe13] restricts the range of the interaction due to the limitations of the depth camera on these devices. The other major limitation of their system is the lack of explanation of the gestures to the user.

Another major work develops a new Kinect hand recognition system using the 'golden' energy function [Sha15]. It renders all possible poses of the hand and selects the pose, whose corresponding rendering best matches the input image while accounting for the prior probability of poses. The technique uses regions of interest (RoI) to determine the approximate hand position from the depth image stream and a learned pixel-wise classifier [Sho13]. When used with 3Gear [3Ge16] and LeapMotion device, this technique achieves a better percentage of 3D hand detection than those described previously. This system requires extremely large FingerPaint datasets of 3.2GB of storage space [Sha15] that could make portability and memory management an issue. High memory latency and a high end GPU are required when accessing the dataset in the memory in real time. In addition, the inability to work with both hands and the lack of a gesture recognition system for fingers could limit the utilisation.

Tang [Tan11] introduced a method for identifying grasping and pointing gestures in which a person's hand model was estimated based on a skeletal tracker. This is limited in terms of tracking resolution and inability to track at a close distance. Chen and Wong introduced an interactive sand art drawing using Kinect [Che14]. Four key gestures were detected by the system, including sing-point finger for single point erosion, single splicing and leaking, V-shape 2 fingers for pinch spilling and pinch erosion. Lee and Tanaka used databases for fingertips and hand shapes to enable gesture cognition with any finger in the palm as well as two fingers (thumb and index finger) [Lee13]. The interaction techniques were applied to sample applications for finger painting and mouse controlling. Song et al proposed an algorithm for gesture recognition algorithm based on depth information from a Kinect device [Son13]. Cook et al also introduced a real-time finger-gesture interaction system using Kinect v2 that identifies finger gestures at close range [Coo15]. However, vision-based systems are often limited in terms of accuracy, occlusion and inconsistency when changing the environment, such as dark light and rough background.

3. GESTURE RECOGNITION

Our gesture recognition system was developed based on gestures as point clouds approach [Vat12], to address the problems identified in Section 2. The Point-Cloud Recogniser ($P) is a gesture recognition system designed for quick and simple recognition of two dimensional gestures from a user's input. $P works by collecting a bunch of pre-determined points already created in the system, i.e. the database of gestures that the system can recognise. When ready to, it takes the users current points as an input to determine the closest matching set of user points to data points using a nearest neighbour classifier.

We expanded the recognition system from 2D to 3D that improved recognition accuracy and performance. The enhancement includes additional classifiers to the dataset to improve comparisons and collections of points by segmenting cloud stroke points based on certain parts of the hand and fingertips.

Gesture Recognition Systems

Gesture recognition systems that use $P include single stroke ($1) and multi stroke ($N) systems. A single stroke system can handle only one stroke on a canvas for comparisons. A multi stroke system can handle more than one stroke. $P uses the $1s algorithm for nearest neighbour matching and gesture recognition, while also using $Ns algorithm for allowing multiple strokes to be recognised. The benefit of the dollar family gestures is how users can add their own variations to the data set to be used for future recognition [Wob07].

The uni-stroke recogniser ($1) developed by [Wob07] is the first gesture recognition system developed in the $ family. Its implementation is similar to the $P system, which uses the nearest neighbour and Euclidean scoring systems. $1 is an extension of an existing recognition system called SHARK^2 [Kri04]. It performs extremely well with recognising gestures even when given a low amount of training samples to compare user input.

The multi-stroke recogniser ($N) is a solution to $1's problem of not being able to use multiple strokes to determine a gesture [Ant10]. $N was designed so that the input of strokes would be stroke and direction invariant. This was achieved by recording all distinct directional behaviours of the gesture to compare the user data. $N has an immensely increased complexity by loading all of these permutations. This issue could

be improved by removing multiple stroke types and directional changes when comparing single stroke gestures [Ant10]. Protractor uses a closed-form template-matching method eliminating the simplicity and complexity of the Golden Section Search algorithm [Ant12].

We adopted multi-stroke recognition system that could handle multiple start and stop points. The system needs to be fast without the need to remember any permutations in the strokes as the stroke order is not important for our gesture recognition. This rules out $N and $N-Protractors complex algorithms leaving $P. $P does have one drawback with its implementation, where the point clouds are rotation variant. This means that each gesture is not rotated when comparing point clouds. To overcome this problem, when we developed the database we added gestures of different angles to the training set to compensate for this drawback.

$P in a 3D world space ($P3D)

Extending $P into a 3D implementation was done by following the same algorithms in 3D space, such as Protractor [Kra11]. We utilise the practicality, simplicity and usability of the $P algorithm, extending it to operate with three-dimensional gestures.

We take the existing method for gathering and assigning points with each gesture created and change the creation method depending on the data. Because the protractor recogniser takes the Euclidean distance between points in 2D to determine the distance, they take the Z position of the set of points when applicable. The result means that the distances between points when classifying gestures will be slightly more between 2D and 3D gestures because of the added axis.

The 3D centroid formula determines the centre of the 3D data points which is defined by equation 1. We used a 32-point sampling resolution (N) [Kra11] to define the insignificant change in accuracy. Translating the data requires the 3D centroid position to be equal to the subtraction of each of the data points with the centroid. The new translated data points will make the centroid be at position (0, 0, 0). This translating makes it possible to centre the data for recognition.

$$Cx=\sum_{i=0}^{N-1} x_i \Big/ N,\ Cy=\sum_{i=0}^{N-1} y_i \Big/ N,\ Cz=\sum_{i=0}^{N-1} z_i \Big/ N \qquad (1)$$

Where Centroid $C = (Cx, Cy, Cz)$ and N is the sampling resolution.

Resampling the data points is the most crucial of the pre-processing techniques. It requires multiple calculations to convert the data from any number of points to the sampling resolution of 32 that we want for 1:1 conversions between user data and gesture data. We extended the 2-dimensional method to create the linear interpolation (LERP) point as equation 2, where a new 3D point (P) was calculated by LERP point (∂), first point (f) and current point (p).

$$\alpha = \frac{(I-D)}{D} \quad where\ D\ != 0$$
$$\partial = \begin{cases} \partial & if \quad \alpha > 0.0 \ \ and \ \ \alpha < 1.0 \\ 1.0 & if \quad \alpha \geq 1.0 \\ 0.0 & if \quad \alpha \leq 0.0 \end{cases}$$
$$P.x = (1.0 - \partial) * f.x + \partial * p.x$$
$$P.y = (1.0 - \partial) * f.y + \partial * p.y$$
$$P.z = (1.0 - \partial) * f.z + \partial * p.z \qquad (2)$$

$P3D+

While $P3D generally provides good gesture recognition in 3D, its effectiveness normally depends on datasets and user inputs. We added to the $P system a set of classifications and ordering the LeapMotion data so that each finger and palm position was its own stroke and followed its previous position. Our goal is to improve the 3D recognition system by eliminating the need to search all the gestures within the database. We also improved pre-processing gestures by assigning stroke IDs to point clouds to remove inaccuracies. The enhancements are as follows.

Removing Redundant Point Clouds

We limited the complexity of the gesture recognition by only selecting gestures that reach a certain classification. These classifications would remove the ambiguity of searching through all possible gestures and instead focus on the range. We developed three classifiers in the $P3D system. The first classifier determines whether the recorded gesture was in 3D or 2D. This classifier only worked when there was a 2D gesture being recognised. The second classifier is for identifying whether the gesture is a pose or not. A pose gesture features the input to contain very little movement to no movement whereas a normal gesture requires much more movement over a series of time. The third classifier determines which left or right hand that the gesture is used with. This implementation aims to remove ambiguity and noise when comparing gestures from the left hand that could increase recognition when using the right hand. For example, a left PEACE pose matches well with a right OK pose and vice versa.

Assigning and Ordering Stroke IDs to the Hand

$P was designed for a 2D drawing scenario that was inputted one stroke at a time and it did not take the stroke ID into consideration. While this is normally fine when the user draws the gesture one stroke at a time, it could be a problematic when recording

different strokes currently. This is because the system thinks the entire gesture is made of single strokes.

$P does not take into consideration the stroke order, stroke direction and stroke permutations as in $N/$N-Protractor solutions [Ant10, Kra11]. It implements the stroke system for pre-processing, resampling and calculating the path length for the gesture. We assigned the palm and the fingers with their own stroke IDs. The IDs of *Palm*, *Thumb*, *Index finger*, *Middle finger*, *Ring finger*, and *Little finger* were numbered 0 to 5 respectively. The order of stroke indexes of the hand and fingers was used to improve the matching rate between these point clouds. This improvement provides a closer approximation on how to interpret the data where inaccuracies can be reduced to produce a closer match to determine the correct gesture.

4. INTERACTION WITH LEAPMOTION – A CASE STUDY

The LeapMotion is a controllable device that tracks hand and finger movements using LEDs and infrared cameras. While the Kinect can track skeletons from long distances with its sensors, the LeapMotion is dedicated to tracking hands and fingertips with a shorter range and more accurate than the Kinect.

Capturing the 3D Point Clouds

In order for the LeapMotion to be recognised by the point cloud gesture recognition system, we created the training set that was used in the classifications of the gestures. The chosen data from the LeapMotion system to determining these gestures were the stabilised palm position and the stabilised tip position of all the fingertips. Other points such as the wrist and distal bone positions were not considered for the training set because these details were not used for the recognition system.

Environment Building

We used Unity 3D game engine to develop our interaction environment. Unity provides an excellent framework for developing tools and applications within its integrated development environment (IDE). Unity's IDE allows for 2D or 3D applications with a variety of different tools that will improve a user's experience with the system.

It is crucial for first time users to learn about the system interactively. We developed a tutorial system to illustrate all the different elements of our environment in such a way that is easy to follow and understand. The system also allows the user to revisit old lessons when required. Once completing the tutorial, we provide a showcase scene where users can collaborate and manipulate objects within the scene using these new abilities learnt.

Monsoon Minigame

We developed a simple minigame to help the users be familiar with the interaction using the LeapMotion. The game revolves around the user trying to catch as many boxes, barrels, orbs and traffic cones as they can. The objects are spawned above the user and trickle down every second. It is up to the user whether they want to hold these objects or to play around with them (See Figure 1). This minigame serves as an initial learning task for beginners and a challenge for the experts who want the highest score.

Figure 1. Mosoon Minigame monsoon state.

Tutorial System

The tutorial system was developed to help the users understand the systems. They included 2D drawing, 2D gesture recognition, moving the camera, 3D pose recognition and 3D gesture recognition. Each of these systems is also a state within the tutorial system that the user can cycle through. Each state has its own visual, written and interactive way of showing the user the current state.

2D Drawing

This state trains the user how to draw 2D objects on a canvas. The drawing system requires the use of the LeapMotion and a set of criteria needs to be met to begin drawing (see Figure 2). To begin with, the only hand that the drawing system recognises is the primary hand selected by the user in the first tutorial state. The index finger is chosen for drawing. When the user wants to stop drawing, they can either move all their fingers back from the sensor, or open their thumb out without the need of moving. If a user makes a mistake and wishes to undo the previous stroke on the canvas, they can do so by swiping right.

2D Gesture Recognition

The 2D gesture recognition scenes goal is to help the user with recognising 2D shapes. While this process could have been combined with the drawing state, our experiences show that learning to draw first with the added gesture recognition was too steep a learning curve for first time users. Once users are

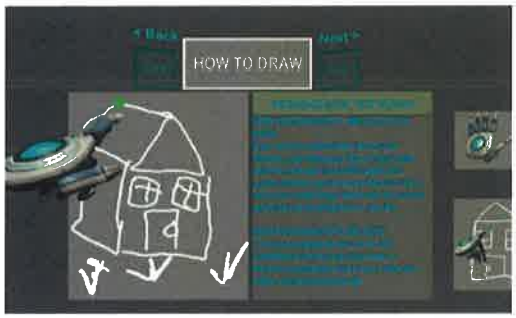

Figure 2. Tutorial state default template.

familiar with the drawn shapes, they can move their hand by positioning all the fingers forward on the LeapMotion device. The system then classifies the gesture using $P and returns the best scoring point cloud for each gesture. The implemented gestures in this tutorial stage are *Circle*, *Rectangle*, *Triangle* and *Cross*. The user can create shapes and then grab, move or throw them away.

Moving the Camera

This tutorial follows the main principle where users can move around the play area with their secondary hand by closing it into a fist. This scene is used as a breaker from the 3D gesture recognition.

3D Pose Recognition

The 3D pose recognition scene teaches users how to hold their secondary hand when performing a pose. To perform a pose, the user positions their hand so that it forms a symbolic gesture and hold that pose until it is recognised. The Pose recognition system can use 4 different poses including the L Pose, the ROCK Pose, the OK Pose and the PEACE Pose (See Figure 3).

Poses are recognised within the gesture recognition system through the use of velocity within the

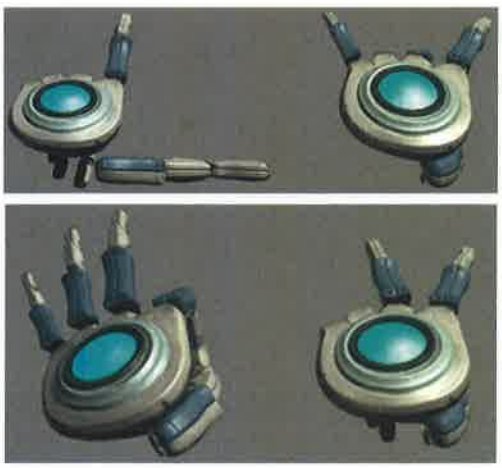

Figure 3. L Pose, ROCK Pose, OK Pose and PEACE Pose respectively.

LeapMotion device. We use this data to find the minimum x, y or z value from the hands and fingers velocity data. We then check if the hand has stopped moving with a velocity of under a minimum value. Once this occurs, the gesture recognition system records the position information from the fingers and palm for that frame. This process repeats until either the hand is moving quicker than the minimum velocity and our data points need to be reset, or the user holds their hand over the minimum number of frames required, 75 frames in our implementation.

Making users wait while performing a pose could be an issue if there is no hint or indication. To overcome this, a pose gesture indicator was presented showing the progression of a pose gesture until the pose recognition system can determine the pose.

3D Gesture Recognition

The 3D gesture recognition follows the 3D pose recognition state where we want to teach the user how to do gestures in the environment. To perform gestures, the user uses their secondary hand and makes a quick movement. As opposed to poses that require little to no movement, gestures requires a sudden movement that when slowed down is recognised as a gesture. We implemented two sample gesture recognitions in our system including SHAKE and GUN.

To recognise a gesture, the system gathers the same velocity data from the palm and fingers as with the pose recognition system but determines the maximum velocity from the x, y and z values instead. It then waits until the velocity from the palm or fingers is greater than the maximum velocity of 500 millimetres and then it starts recording the gesture. Once this happens, each frame is recorded until the hand or fingers velocity has fallen under 500 millimetres or the number of gesture frames is over our sampling resolution of 32 points. A gesture indicator was also created to assist the interaction.

Practice Systems

We developed two practice systems to complement the lessons taught in the LeapMotion tutorial environment. The first system is a target practice scene that uses 2D drawings to create objects for the user to fling around and 3D poses for altering those shapes properties. This scene also uses 3D gestures for users to interact with the environment as opposed to just the shapes. The second system is a *Cave Painting* system that emulates how users could use the 2D recognition system in a creative and fun way with an image matching game.

Target Practice System

The target practice system is the showcase scene for our LeapMotion environment. It contains a number of gestures in both 2D and 3D and offers an open

Figure 4. Target Practice system with multiple objects in play.

sandbox where the user can use all these different systems together to interact with the environment. The interactable objects within the play area are cube walls and targets. Targets generate points based on how close an object gets to the centre from the users throw and a cube wall acts as a breakable barrier which users can break open by using multiple objects at once (see Figure 4).

The types of 3D Poses that our system can recognise are: L Pose this turns the gravity off for all created objects, OK Pose changes the colours of all newly created objects, ROCK Pose triples the current objects created and the PEACE Pose makes all the objects heavier/lighter. The 3D gestures that the Target Practice scene can also recognise are SHAKE Gesture that shakes the camera around and the GUN Gesture that creates a bullet launched from the fingertips that explode on impact.

Cave Painting

The Cave Paintings goal is to closely match one of the 5 popular animals from the Australian outback (see Figure 5). The Bat and Turtle contain the fewest shapes; the Emu and Lizard are more complicated due to their curves and the Kangaroo is the most difficult one with both curves and difficult shapes. Users are required to trace the animal drawing using their finger as close as they can. The system gives the users feedback on how close they were to the drawing as well as acknowledge their performance. These messages do not have negative comments written on the drawing to provide an enthusiastic approach when trying to draw the animal.

5. EVALUATION

We compared the results of three different point cloud gesture recognition systems ($P, $P3D and $P3D+). Using the training set defined and created by using the program, we evaluated the performance (time complexity and the average recognition percentages) of the original 2D algorithm ($P), the added 3D gesture recognition algorithm ($P3D) and our improved 3D recognition algorithm ($P3D+). The gesture set consists of 15 gestures ranging from iconic aboriginal animals, 2D shapes, 3D poses and

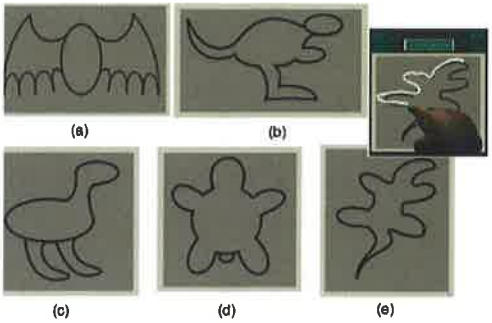

Figure 5. The 2D drawing gestures recognised in the Cave Painting scene. (a) Bat, (b) Kangaroo, (c) Emu, (d) Turtle, (e) Lizard.

3D gestures. The results in these experiments were evaluated based two sample gesture databases.

The first database consists of all 15 gestures from the gesture set, from which each pose per hand has 4 angle variations that were split into normal, rotated forward, rotated left/right (based on what hand was used) and rotated back. Each gesture on each left- and right-hand has five variations. Each of the 2D animal gesture contains three similar variations while each 2D shape also has five variations in average per shape. Overall, this database totals to 28 poses and gestures recorded (300kb) and 35 animal and shape drawings (600kb).

The second database expands the 3D gesture set by doubling the number of variations per hand to 10 each. The 3D poses also receive an increased number of variations per hand, from five to six. We also exclude all 2D gestures to focus on 3D recognition. To create unbiased results, the classification of the results uses the 3D poses and gestures from one database as the gesture/pose input to compare point clouds and determine the minimum distance by using another training set.

We initially used the scoring equation defined in [Wob07, Ant10]. However, this scoring method produces scores within the 95%-99% margin with very few scores ranging outside this segment. Our implementation requires the scores to range based on our observed minimum distances for the correct and incorrect gestures. To overcome this limitation, the scores were calculated using a polynomial formula to extrapolate the data explained above into a curve to map the distance data to the following percentage values, particularly *0.5 = 99%, 0.9 = 90%, 1.1 = 85%, 1.3 = 80%, 1.5 = 75%, 3.5 = 20%, and 4 = 0%.*

We do not consider the average recognition rate for $P on 2D gestures as the recognition average has already been conducted by [Ant12]. We test $Ps ability to recognise 3D gestures versus $P3D recognition system using both databases.

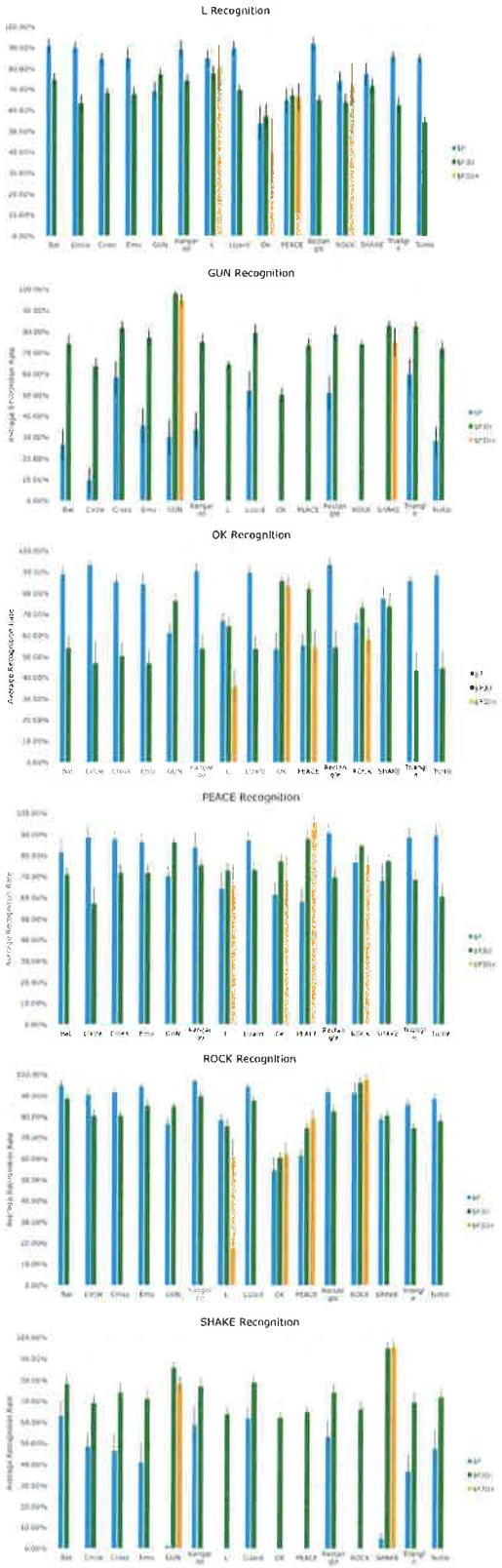

Figure 6. Gesture recognition averages using $P, $P3D and $P3D+ on small dataset with L, Ok, Gun, Peace, Rock and Shake Respectively.

First Dataset

From the results on the first dataset (see Figure 6), we can see a clear indication that $P could not determine any differences between 3D and 2D data. It is a surprise that the recognition rate in the L pose, $P proved to be more efficient when compared to $P3D (84% Range CI = [81, 87] to 77% CI = [74, 80]). However, the recognition rates for other gestures were higher in $P3D in comparison to $P particularly the L pose. There is no clear indication from the noise of the other gestures like Bat 90% CI = [87, 93], Circle 89% CI = [87, 91], Kangaroo 89% CI = [85, 93], Lizard 89% CI = [86, 92], Rectangle 91% CI = [88, 94] that the L pose was the most recognised gesture. All other poses and gestures have significantly decreased average recognition rates for the correct gesture with high recognition rates for the wrong gestures. With the ROCK pose, $P does have a high average recognition rate of 91% CI = [86, 96], but is still beaten by $P3Ds average of 95% CI = [93, 97]. This indicates our 3D implementation works better than the traditional 2D system.

Second Dataset

From the second dataset (see Figure 7) with a larger 3D database, $P performs miserably when determining the GUN gesture with an average recognition rate (< 50%) as well as an extremely low average recognition rate for SHAKE gesture (< 5% recognition mark). $P3Ds performs much better above the 95% recognition average, for example the SHAKE Left Hand (96% CI = [94, 98]), Right Hand (98% CI = [97, 99]) and GUN Left Hand (98% CI = [97, 99]) and GUN Right Hand (100%). When presented with a larger L pose database, the $P recognition average actually performs better than $P3D with the Left Hand scoring an excellent 90% (CI = [89, 91]) versus $P3Ds 85% (CI = [83, 87]) recognition average and the Right Hand achieving similar results with a 83% (CI = [76, 90]) average compared to $P3Ds 78% (CI = [72, 84]). This means that while the $P system performs worse on all gestures it has the ability to perform well or better than $P3D on pose recognition.

Comparing $P3D to $P3D+

With the improvements made to $P3D+, we evaluate whether our new $P3D+ performed better than the original $P3D. These evaluations follow the above comparisons, and T-value tests are used to determine if the improvements are statistically significant (alpha level of .05 for all statistical tests).

First Dataset

Figure 6 shows the GUN gesture has a decreased recognition rate from $P3Ds 98% (CI = [97, 99]) to $P3D+s 95% (CI = [92, 98]) but there is no statistical significant difference between these two systems

$t(38)=1.71$, $p=0.09$. This result means that while \$P3D+ recognition average is lower than \$P3D, there is not enough evidence to class both sets of data as different. With the PEACE Pose, we found that there was a significant difference in recognition averages for \$P3D 88% (CI = [84, 92]) and \$P3D+ 96% (CI = [93, 99]) $t(22)=3.01$, $p<0.01$. This result suggests that the improvements made to increase the recognition for the PEACE pose have increased the average enough to be statistically significant.

Second Dataset

Figure 7 also indicates that our hypothesis of the best-case scenario is apparent in the GUN (Left/Right) hand. For the Left Handed GUN gesture, the \$P3D Right Handed GUN recognition average is almost the same as the Left Handed average (99% CI = [98, 100]) while the \$P3D+ system has a lower percentage of 98% (CI = [96, 100]) - $t(8) = 0.59$, $p = 0.57$. This means that the \$P3D+ system is missing the best recognisable gun gesture and that is lowering its average. With the Right Handed L pose we can also determine that the recognition average is not significant between these systems while \$P3D+ has a significantly increased recognition rate when compared to \$P3D (92% CI = [82, 100]) and 78% CI = [72, 84]) respectfully - $t(8) = 1.00$, $p = 0.35$. For the Left Handed L pose the difference is extremely noticeable. For the \$P3D system, the average recognition rate is around 85% CI = [82, 88]). The \$P3D+ system achieves a higher recognition rate of 98% (CI = [95, 100]) with a significant difference between the 3D and 3D+ systems $t(6) = 5.97$, $p < 0.01$. This shows that when the user performs an L pose, the \$P3D+ system is going to recognise that pose easier.

Discussion

When differentiating \$P with \$P3D, the assumption was that \$P3D would outperform \$P. With the majority of gestures, \$P3D recognises the 3D point cloud with excellent gesture recognition average while \$P has high averages. The L Pose interestingly with the small database and the L and Rock Poses in the large database, the \$P gesture recognition system perform better than the 3D system.

When comparing \$P3D and \$P3D+, the improved gesture recognition system hypothetically yields slightly lower results based on the limitations of the training data gathered for testing. Our experiment shows that with some gestures, it actually increased the recognition rate by a small margin of 4.55%. The Left Hand L pose was the best result that achieved an increased recognition average of 13.17%. While this improvement increased the recognition average, we can claim that the confidence interval for the \$P3D+ gesture recognition average falls way out of the more condensed \$P3D versions interval as \$P3D factors in all gestures and gets the best-case scenario for each pose/gesture. However, the GUN gesture and the OK Pose did follow our hypothesis of having a smaller recognition rate compared to the traditional 3D gesture recognition. This is because of the small difference between left and right hand GUN gestures.

User Experiences

We demonstrated the system to several people during various events hosted by the Western Sydney University. While no formal questionnaires were given, we have documented the difficulties and differences between users who have used the system for the first time and some who have had experience with this device before. The users were a variety of ages ranging from primary school children, high school students, university undergraduates and some academic researchers. The users mostly inciated that they never used the LeapMotion system before.

From those who have no prior experience with the LeapMotion it was clearly that they found it difficult to first locate the LeapMotion system and use it properly. The main problem was the simplicity in the LeapMotions design causing users to not believe that the system could perform 3D hand recognition. Another issue was users who would immediately try to move their hand as close to the sensor as possible. While the LeapMotion can detect hands from a short distance, if the hands are too close to the IR cameras they could not distinguish the hands properly. Once instructed to move their hands upward, users understood the distance required for recognition and rarely brought their hands too close to the sensor again. Users were presented with either the default LotsOfBlocks demo or the Monsoon minigame developed for the first time users.

For the users who were presented with the Monsoon minigame, comments were made about what they were supposed to do while the minigame was playing. As the premise for the scene was to catch as many objects as possible, users found that the LeapMotion could not track their hands very well when they were cupped together. Some users decided to ignore grabbing the objects but rather tried to fling the objects around and as far as they could to see who could get the furthest.

The overall consensus with most individuals who used these demos was overwhelmingly positive. While some had seen the LeapMotion technology before or were not enthusiastic with the devices capabilities, the majority enjoyed using the tracking system.

6. CONCLUSIONS

In this paper, we have presented gesture recognition techniques that allowed recognition of both 2D and 3D gestures using point cloud gesture recognition. These techniques extrapolate gesture sets using

similar classifiers to recognise when to start looking for a gesture. We implemented tools that collected gesture data for recognition and processing. We also developed new 2D and 3D interactive environments that acted as a visual representation of the data.

We enhanced a 2D point cloud gesture recognition system into a 3D gesture recognition system that supports the data points from the LeapMotions hand, palm and fingers. We implemented four poses and two gestures to demonstrate the effectiveness of the new 3D gesture recognitions. We used a small database system for our gesture set in comparison to large database in existing systems.

Our experimental results showed that the 3D system outperformed the traditional 2D system in most cases as well as some small improvements on the $P3D+ in comparison to the original $P3D.

7. REFERENCES

[3Ge16] 3GearSystems. (Feb 2016). *Nimble VR.* http://nimblevr.com/

[Ant10] Anthony, L. and Wobbrock, J. O. A lightweight multistroke recognizer for user interface prototypes. In Proc. Graphics Interface 2010, Ottawa, Ontario, Canada, 2010.

[Ant12] Anthony, L. and Wobbrock, J. O. $N-protractor: a fast and accurate multistroke recognizer. In Proc. Graphics Interface 2012, Toronto, Ontario, Canada, 2012.

[Che14] Chen, K.-M. and Wong, S. K. Interactive Sand Art Drawing Using Kinect. In Proc. 7th International Symposium on Visual Information Communication & Interaction, pp. 78-87, 2014.

[Coo15] Cook, H., Nguyen, Q.V., Simoff, S., Trescak, T. and Preston, D. A Close-Range Gesture Interaction with Kinect. In Proc. IEEE International Symposium on Big Data Visual Analytics, Hobart, Australia, pp. 1-8, 2015.

[Ebe13] Ebert, L. C., Hatch, G., Thali, M. J., and Ross, S. Invisible touch—Control of a DICOM viewer with finger gestures using the Kinect depth camera. Journal of Forensic Radiology and Imaging, vol. 1, pp. 10-14, 2013.

[Kra11] Kratz, S. and Rohs, M. Protractor3D: A Closed-Form Solution to Rotation-Invariant 3D Gestures. *Intelligent user interfaces,* pp. 371, 2011.

[Kri04] Kristensson, P.-O. and Zhai, S. SHARK2: A Large Vocabulary Shorthand Writing System for Pen-based Computers. pp. 43, 2004.

[Lee13] Lee, U and Tanaka, J. Finger identification and hand gesture recognition techniques for natural user interface. *In Proc. Asia Pacific Conference on Computer Human Interaction,* pp. 274-279, 2013.

[Liu09] Liu, J., Zhong, L., Wickramasuriya, J. and Vasudevan, V. uWave: Accelerometer-based personalized gesture recognition and its applications. Pervasive and Mobile Computing, vol. 5, pp. 657-675, 2009.

[Pri16] PrimeSense and Apple. (Feb 2016). *OpenNI.* https://github.com/OpenNI/OpenNI

[Ope16] OpenKinect. (Feb 2015). *Libfreenect.* https://github.com/OpenKinect/libfreenect.

[Ren11] Ren, Z. Meng, J., Yuan, J. and Zhang, Z. Robust hand gesture recognition with kinect sensor. In Proc. ACM International Conference on Multimedia, pp. 759-760, 2011.

[Sha15] Sharp, T., Wei, Y., Freedman, D., Kohli, P., Krupka, E., Fitzgibbon, A. et al. Accurate, Robust, and Flexible Real-time Hand Tracking. *Computer Human Interaction,* pp. 3633-3642, 2015.

[Sho13] Shotton, J., Sharp, T., Kipman, A., Fitzgibbon, A., Finocchio, M., Blake, A. et al. Real-time human pose recognition in parts from single depth images. *Communications of the ACM,* vol. 56, p. 116, 2013.

[Son13] Song, L., Hu, R., Xiao, Y., Gong, L. Real-Time 3D Hand Gesture Recognition from Depth Image. In Proc. the 2nd International Conference On Systems Engineering and Modeling (ICSEM-13), pp. 1134-1137, 2013.

[Tan11] Tang, M. Recognizing hand gestures with Microsoft's kinect. Department of Electrical Engineering, Stanford University, CA, USA, Technical Report, 2011.

[Vat12] Vatavu, R.-D., Anthony, L. and Wobbrock, J. O. Gestures as Point Clouds: A $P Recognizer for User Interface Prototypes. In Proc. International Conference on Multimodal Interaction, p. 273, 2012.

[Wob07] Wobbrock, J.O., Wilson, A.D. and Li, Y. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. *User Interface Software & Technology,* pp. 159, 2007.

[Yin14] Yin, Y. Real-time continuous gesture recognition for natural multimodal interaction. PhD thesis, Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 2014.

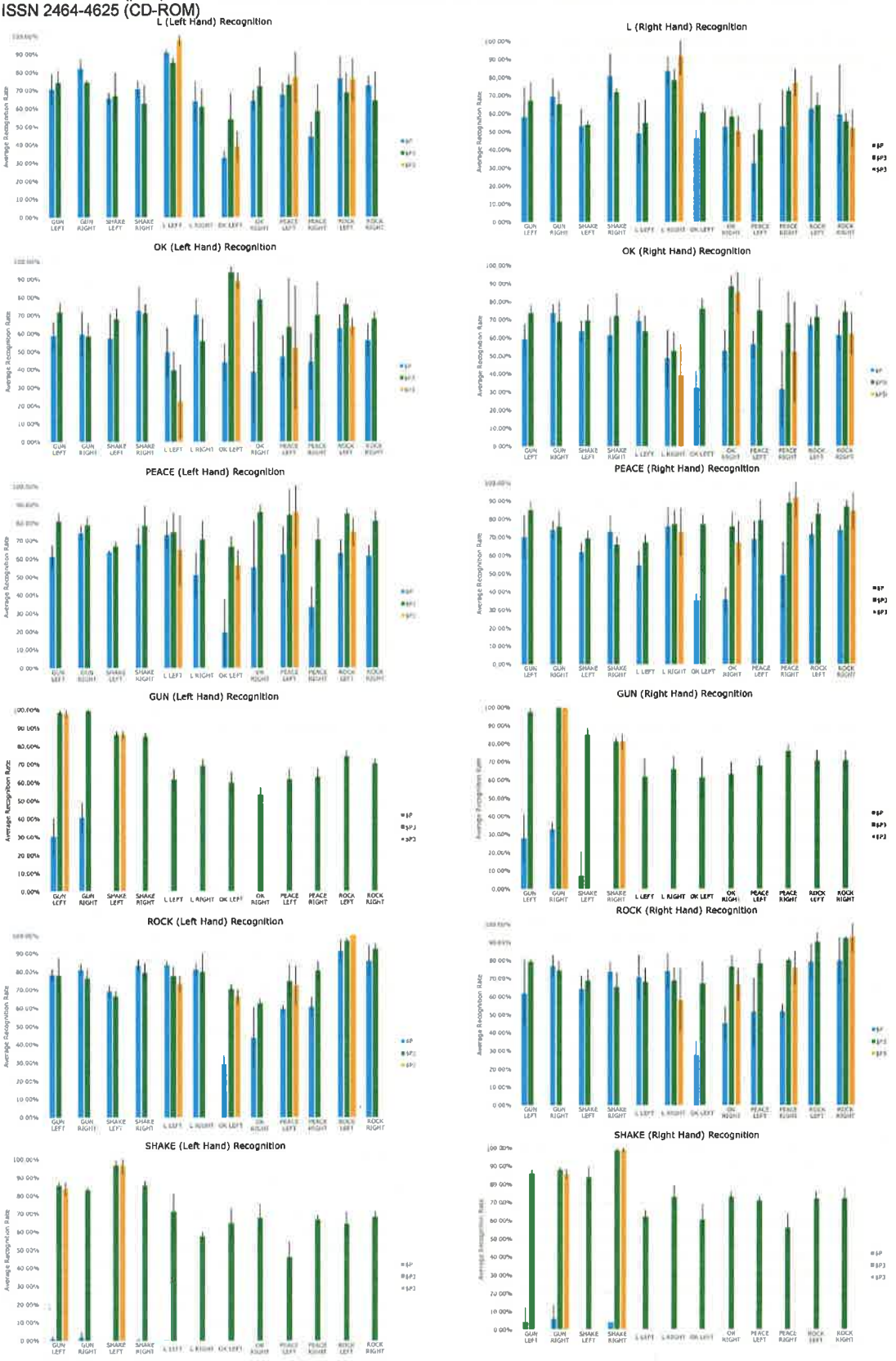

Figure 7. Gesture recognition averages on a large dataset with L, Ok, Gun, Peace, Rock and Shake on both left and right hands respectively.