A system for pose analysis and selection in virtual reality environments

Andrew Clark
Anban W. Pillay
andrewclark.1905@gmail.com
pillayw4@ukzn.ac.za
School of Mathematics, Statistics and Computer Science,
University of KwaZulu-Natal
Centre for Artificial Intelligence Research, Council for Scientific and Industrial Research

Deshendran Moodley
deshen@cs.uct.ac.za
Department of Computer Science,
University of Cape Town
Centre for Artificial Intelligence Research, Council for Scientific and Industrial Research

ABSTRACT

Depth cameras provide a natural and intuitive user interaction mechanism in virtual reality environments by using hand gestures as the primary user input. However, building robust VR systems that use depth cameras are challenging. Gesture recognition accuracy is affected by occlusion, variation in hand orientation and misclassification of similar hand gestures. This research explores the limits of the Leap Motion depth camera for static hand pose recognition in virtual reality applications. We propose a system for analysing static hand poses and for systematically identifying a pose set that can achieve a near-perfect recognition accuracy. The system consists of a hand pose taxonomy, a pose notation, a machine learning classifier and an algorithm to identify a reliable pose set that can achieve near perfect accuracy levels. We used this system to construct a benchmark hand pose data set containing 2550 static hand pose instances, and show how the algorithm can be used to systematically derive a set of poses that can produce an accuracy of 99% using a Support Vector Machine classifier.

CCS CONCEPTS

• **Computing methodologies → Supervised learning**; • **Human-centered computing → User interface design**.

KEYWORDS

gesture recognition, virtual reality, gesture interaction

ACM Reference Format:
Andrew Clark, Anban W. Pillay, and Deshendran Moodley. 2020. A system for pose analysis and selection in virtual reality environments. In *Conference of the South African Institute of Computer Scientists and Information Technologists 2020 (SAICSIT '20), September 14–16, 2020, Cape Town, South Africa.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3410886.3410909

1 INTRODUCTION

Virtual Reality (VR) systems simulate three-dimensional environments to provide an immersive experience that is achieved through a VR head-mounted display, which displays a stereoscopic view of the simulated world. Users typically interact with VR environments using hand-held controllers, cameras, or gloves. However, hand-held and wearable devices detracts from the immersive experience. A more natural and intuitive means of interaction may be provided by hand gestures and hand poses. Depth cameras, such as the Leap Motion Controller (LMC) and Microsoft Kinect have proven to be effective input devices for vision-based hand pose recognition, as they make the task of separating foreground from background considerably easier than RGB cameras. Depth cameras, such as the LMC, that capture infrared are especially useful as they are invariant to skin colour and visual lighting conditions.

The Leap Motion Controller (LMC) is a lightweight and affordable stereoscopic infrared camera that specializes in tracking a user's hands with sub-millimeter accuracy [8]. The device consists of three infrared LEDs, that make it invariant to lighting conditions and skin colour, and two infrared cameras that provide a stereoscopic view of the user's hands, allowing it to create a depth map. Due to its small weight, the device can be mounted onto virtual reality head mounted displays that renders 3D environments. However, gesture recognition systems using the LMC typically do not produce the level of recognition accuracy (> 99%) required for many real world VR applications. Some of the causes of misclassification are occlusion, the effect of hand orientation and confusion between similar gestures.

This work explores the limits of the Leap Motion depth camera for building an effective VR application. The paper describes and evaluates a system for selecting a set of static hand poses with near-perfect recognition accuracy.

The rest of the paper is organised as follows: Section 2 provides a literature review, section 3 describes the pose analysis system, section 4 presents a discussion and section 5 concludes.

2 LITERATURE REVIEW

2.1 Hand Poses for VR

The Leap Motion Camera (LMC) has been extensively applied to the field of Sign Language recognition [5, 17–19, 30]. Other studies have used the LMC for 3D virtual scene and object manipulation [7, 13], television remote control [29], 3D painting [26], and medical rehabilitation [1, 11, 25]. Several studies have used the LMC for gestural input and a VR head-mounted display for visual output. These include data visualization and interaction through hand poses [6], robotic arm remote operation [27], 3D model manipulation and visualization [2, 14, 22], 3D virtual navigation [15], and medical rehabilitation [3]. Hand pose recognition by the LMC was used in a VR Computer Aided Design application [2]. The two-fingered pinch pose was used to pick up a stationary component of a virtual mechanical device, and users could freely move this component around provided the pinch pose is held. Hand poses were also used to control movement in VR [15].

Publicly available datasets of hand poses include the *Innsbruck Multi-View Hand Gesture* dataset [23] (captured by the Kinect), the *ChaLearn* dataset [9] (captured by the Kinect), and the dataset by Molina et al. [20] (captured by a time-of-flight camera). Datasets captured by an LMC have also been created, such as the *LeapMotion-Gesture3D Dataset* and *Handicraft-Gesture Dataset* both by Lu et al. [16], however these datasets consist of dynamic gestures. None of the above datasets contain gestures that were made in VR environments.

2.2 Gesture Taxonomies

Gesture taxonomies that categorize gestures are useful in deriving and describing hand poses. One way to categorize gestures is by the style of gesture [12]. For example, gestures can be classified as being either acts or symbols, where sign languages often employ symbolic gestures, while acts are context-sensitive [24]. Vafaei argues that previous taxonomies, such as the one by Karam and Schraefel [12], are too broad, and do not capture specific dimensions, such as the physical form of the hand [28]. Vafaei proposed a taxonomy by adjusting and combining dimensions used in the taxonomies of Wobbrock et al. [31] and Ruiz et al. [21]. The categories defined in the taxonomy include: Nature, Form, Binding, Temporal, Context, Dimensionality, Complexity, Body Part, Handedness, Hand Shape, and Range of Motion. Since many recent studies in hand gesture recognition involve user-elicitation, Choi et al. set the focus of their study on developing a taxonomy that allows researchers to notate these gestures systematically [4].

2.3 Classification of Hand Poses

Widely used techniques for recognizing hand poses utilizing different input devices and feature sets include the Support Vector Machine, k-Nearest Neighbour algorithm, and Artificial Neural Networks. The Leap Motion Controller, mounted onto the Oculus Rift to capture hand poses for a virtual reality application [6], used the k-Nearest Neighbour algorithm as a classifier and achieved a recognition rate of 82.5% four distinct poses with a value of $k = 3$. The LMC was used to detect the American Sign Language alphabetical hand poses [5]. The k-Nearest Neighbour algorithm achieved a 72.78% accuracy, while the the Support Vector Machine achieved an accuracy of 79.83%. Features used by both of these classifiers consist of the pinch and grab strength, both of which are provided by the Leap Motion API, and a set of derived features.

3 POSE ANALYSIS SYSTEM

The pose analysis system consists of the three key components:

(1) A pose taxonomy, pose notation and a pose data set
(2) A machine learning system for pose recognition.
(3) An algorithm to construct a reliable pose set.

The taxonomy, with the structured notation, facilitated the construction of a pose data set. This ensured that the pose data set had a good coverage of different poses. Various machine learning techniques were evaluated for automated recognition of poses. The results of the pose recognition experiments made it clear that the machine learning algorithms were not capable of providing high enough recognition accuracies for use in VR systems when using all poses in the pose data set.. An algorithm was therefore developed to systematically derive a pose set that would guarantee recognition accuracies suitable for use in VR systems.

3.1 A pose taxonomy, notation and data set

In order facilitate the acquisition and analyse of poses, we developed a pose taxonomy and a pose notation and used this to construct a pose data set.

3.1.1 Pose taxonomy and notation. The taxonomy was based on Choi et al.'s comprehensive hand gesture taxonomy [4], which includes both hand gestures and static hand poses. Since this research is limited to static poses, only the *Hand Shape* and *Hand Orientation* parameters were included and the gesture type was set to one hand. Dynamic gestures, hand location and arm shape were discarded. Two other modifications were made to Choi's taxonomy. The *finger inter relation* parameter of the hand shape is ambiguous when a finger crosses in front of another. This depends on whether the hand is facing forwards or backwards, since the same finger would now instead be behind the other. This is resolved by defining the *Cross (F_i in front of F_j)* as finger i crossing over finger j on the palm side. The left and right hands are mirror-images of one another, which would cause problems with the *Hand orientation* values of *Left* and *Right*. To resolve this we renamed *Left* and *Right* to *Inwards* and *Outwards*, where *Inwards* is left for the right hand and right for the left hand, and *Outwards* is the opposite. The modified taxonomy is shown in Figure 1.

As per Choi et al.'s notation, a single hand shape (HS) is represented by five finger poses, then four finger inter-relations between the thumb and each of the fingers, then three inter-relations between adjacent non-thumb fingers. The format is as follows: $HS = f_1 f_2 f_3 f_4 f_5; f_{12} f_{13} f_{14} f_{15} - f_{23} f_{34} f_{45}$ where f_i represents the finger pose of finger i, and f_{ij} represents the finger inter-relation between fingers i and j. Finger 1 is the thumb, and finger 5 is the pinky. For example, a fist pose with the thumb pointing up would be represented by $16666; 3222 - 333$. Hand orientation (HO) is represented in the following format: $HO = PO; FFO$ where PO and FFO are Palm Orientation and Fist-Face Orientation respectively. In the case where the palm faces forward and the fists point up, the Hand Orientation would be denoted as $5; 1$.

3.1.2 The pose data set . A total of 2550 poses were captured from 25 [1] participants. The dataset consists of 102 pose captures taken for 29 different hand poses for each participant. A VR environment

[1]All participants were Computer Science students at the University of KwaZulu-Natal who gave informed and voluntary consent for the capture and use of the data in the study.

Hand Shape (HS)	**Finger Pose (FP)**	Pointing (Up)	1
		Pointing (Forward)	2
		Pointing (Side)	3
		Neutral	4
		Bend	5
		Close	6
	Finger Inter-Relation (FIR)	Neutral	1
		Separate	2
		Group	3
		Cross (F_i on palm-side of F_j)	4
		Cross (F_j on palm-side of F_i)	5
		Touch	6
		Loop	7
Hand Orientation (HO)	**Palm Orientation (PO)**	Top	1
		Bottom	2
		Outwards	3
		Inwards	4
		Forward	5
		Backward	6
	Fist-Face Orientation (FFO)	Top	1
		Bottom	2
		Outwards	3
		Inwards	4
		Forward	5
		Backward	6

Figure 1: A modified version of Choi et al.'s notation. The Left and Right values are replaced by Inwards and Outwards respectively, and the ambiguity in the *Cross* values has been resolved.

was implemented using the Unity game engine and used to capture the poses. The environment rendered the Leap Motion Controller's output as virtual hands, mimicking the participant's physical hands. In the event of the virtual hands displaying a significantly different output to the pose they were attempting, the participants were instructed to remove and re-introduce their hands to the scene in the same pose. If on the second try, there was still erroneous output, the pose was captured regardless.

Each participant was asked to make the hand pose displayed in front of them at any orientation of their choosing. Once the experimenter was satisfied that they were making the correct pose, the pose was captured. They were then asked to do the same pose, but in a different orientation of their choosing again. This is followed by capturing the same pose twice in the orientation displayed to them, known as the requested orientation. This results in two poses being made at any orientation, followed by two poses in the requested orientation. The requested orientation is the orientation of a particular pose that attempts to minimize the number of occluded fingers such that the LMC can detect finger positions accurately. This process was repeated for the *Fist Poses*, *Index Pointing Poses*, *Open-Palm Poses*, *Finger Touches and Loops*, and *Finger Crosses* poses. These poses will be referred to as the *Normal Poses.* For the *Thumbs-Up Poses*, hand orientation plays an important role, thus participants were not asked to choose a random orientation. For these poses, two data captures were made at the requested orientation. The requested orientation is not necessarily optimized for the LMC, but rather illustrates to the participant which thumbs-up hand orientation was required. A selection of the poses is given in Fig 2.

The pose set includes all the common pose types found in LMC-based VR applications. The pose set contains all the common poses found in VR applications and includes the Open Hand, Point, Classic

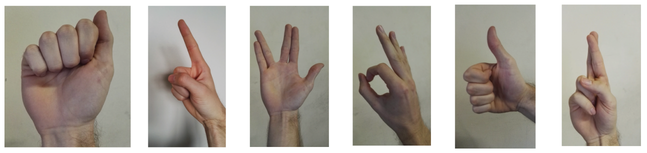


Figure 2: A selection of Hand Poses

Fist, Pinch, and Thumbs-Up poses. The pose set covers a wide array of possible poses, for each parameter in the Hand Shape and Hand Orientation categories in the pose taxonomy shown in Figure 1. It also contains several poses that are similar to one another for testing separating power in pose recognition systems.

3.2 Pose Recognition

Three machine learning classification algorithms viz. k-Nearest Neighbour, Multilayer Perceptron and Support Vector Machines, were evaluated for pose recognition.

3.2.1 Feature Engineering. A set of features used in similar research [6] was extracted from the LMC's data and fed as training data to the classifiers. Both Hand Shape and Hand Orientation features were used to fully describe a pose. The choice of these features are based on the modified version of Choi et al.'s hand pose notation as seen in Figure 1. Further details can be found in [6].

(1) *Hand Shape Feature:*
   - *Normalized tip-to-palm distances* A set of five length measurements representing a normalized distance from each of the fingertips to the centre of the palm. Each distance is normalized by dividing the tip-to-palm distance of a finger by the maximum extended length of that finger.
   - *Finger Tri-Areas* A set of four area measurements, each representing the area of the triangular space between adjacent fingers. [5]

(2) *Hand Orientation Features*
   - *Palm Normal Vector* A three dimensional normalized vector depicting the normal direction of the palm in Cartesian coordinates. This feature can be extracted directly from the Leap Motion API.
   - *Palm Direction Vector* A three dimensional normalized vector depicting the direction from the palm to the base of the fingers in Cartesian coordinates. This feature can be extracted directly from the Leap Motion API.

3.2.2 Pose Classification. The average recognition accuracy and average recognition latency (average time in milliseconds for a single pose to be recognised) were used as performance metrics for comparing the algorithms. Hyper-parameters for each algorithm were tuned manually.

The classification accuracy of the three algorithms were compared across three experiments using stratified k-fold cross validation, with $k = 25$.

The three experiments are summarised below:

(1) *Experiment 1 - Orientation-Independent Experiment:* To determine the effectiveness of a classifier in classifying hand

Table 1: Results for the Orientation-Independent Experiment.

Classifier	Average Accuracy	Average Latency	Average Training Time
k-Nearest Neighbour	66.6667%	0.7835ms	**0s**
Artificial Neural Network	63.0980%	0.5874ms	145.691s
Support Vector Machine (PUK)	**70.3922%**	31.5743ms	2.525s
Support Vector Machine (Linear)	59.098%	**0.0727ms**	2.605s

Table 2: Results for the Requested Orientation Experiment.

Classifier	Average Accuracy	Average Latency	Average Training Time
k-Nearest Neighbour	76.6087%	0.4037ms	**0s**
Artificial Neural Network	**88.1739%**	0.5845ms	76.77s
Support Vector Machine (PUK)	81.3913%	17.2373ms	0.725s
Support Vector Machine (Linear)	78.6087%	**0.0721ms**	0.609s

Table 3: Results for the Thumbs-Orientation Experiment.

Classifier	Average Accuracy	Average Latency	Average Training Time
k-Nearest Neighbour	**96.5714%**	0.6547ms	**0s**
Artificial Neural Network	92.8571%	0.5314ms	141.305s
Support Vector Machine (PUK)	96.0%	44.8328ms	2.475s
Support Vector Machine (Linear)	92.8571%	**0.1419ms**	2.177s

poses regardless of orientation. All captured data was used, with the exception that all *Thumbs-Up Poses* were grouped together.

(2) *Experiment 2 - Requested Orientation Experiment:* To determine the effectiveness of a classifier in classifying hand poses at the requested orientation. The results of this experiment can be compared to the *Orientation-Independent Experiment* to determine the effect of fixing the orientation.
Only requested orientation poses are used from the data set. Most *Thumbs-Up Poses* were not used, except for the *Thumbs-up, fist-in* pose which puts all the fingers in the LMC's view.

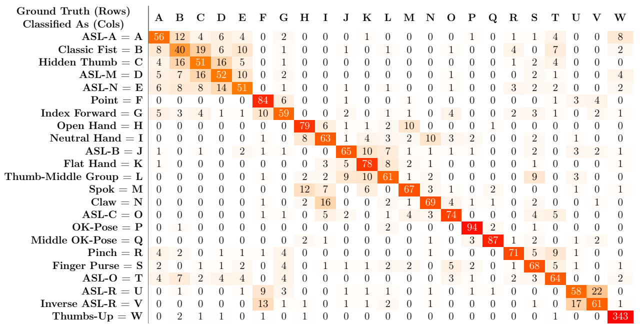
(3) *Experiment 3 - Thumbs-Orientation Experiment:* To determine the effectiveness of a classifier in classifying pose shape and orientation simultaneously. The *Thumbs-Up Poses* group differ from other pose groups by hand shape, and from one another by hand orientation. By classifying the poses in this group, a classifier would have its hand shape and orientation-determining capabilities tested to achieve a correct classification. This provides insight into the orientation-distinguishing capabilities of a classifier. All poses form part of the training set, however only the *Thumbs-Up Poses* group are tested.

The results from Experiment 1 show that when using all poses in the data set, no classifier achieved a sufficient accuracy (>99%) to be effectively used in a VR application. The SVM-PUK classifier achieved a substantially higher accuracy than the other classifiers. However, most VR applications require only a subset of these 29 unique poses. The next section provides a method for selecting such a subset.

3.3 Constructing a reliable pose set

To reduce problematic poses we designed a method to measure similarity and an algorithm that used this to reduce an input set of poses to a set that will guarantee near-perfect recognition accuracy.

3.3.1 Measuring Similarity. A simple way to illustrate which poses are often mis-classified as one another is through a confusion matrix. Fig 3 shows such a matrix, with high classification occurrences highlighted in red. The matrix represents the results from the orientation-independent SVM-PUK experiment. All rows show accuracies in percent for each unique pose, except for the Thumbs-Up pose in the last row which combines all Thumbs-Up poses and shows counts out of a total of 350 as described in Experiment 1. The other experiments showed similar patterns.



Ground Truth (Rows) Classified As (Cols)	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
ASL-A = A	56	12	4	6	4	0	2	0	0	0	1	0	0	0	0	1	0	1	1	4	0	0	8
Classic Fist = B	8	40	19	6	10	0	1	0	0	1	0	1	0	0	1	0	0	4	0	7	0	0	2
Hidden Thumb = C	4	16	51	16	5	0	1	0	0	0	0	0	0	0	0	0	0	1	2	4	0	0	0
ASL-M = D	5	7	16	52	10	0	2	0	0	0	0	0	0	0	1	0	0	0	2	1	0	0	4
ASL-N = E	6	8	8	14	51	0	1	0	0	1	0	1	0	0	1	0	0	3	2	2	0	0	2
Point = F	0	0	0	0	0	84	6	0	0	1	0	0	1	0	0	0	0	0	0	1	3	4	0
Index Forward = G	5	3	4	1	1	10	59	0	0	2	0	1	1	0	4	0	0	2	3	1	0	2	1
Open Hand = H	0	0	0	0	0	0	0	79	6	1	1	2	10	0	0	0	1	0	0	0	0	0	0
Neutral Hand = I	0	0	0	0	0	1	0	8	63	1	4	3	2	10	3	2	0	0	2	0	1	0	0
ASL-B = J	1	0	1	0	2	1	1	0	0	65	10	7	1	1	1	1	0	0	2	0	3	2	1
Flat Hand = K	1	0	0	0	0	0	0	0	3	5	78	8	2	1	0	0	0	0	1	0	0	0	1
Thumb-Middle Group = L	0	0	0	0	0	1	0	2	2	9	10	61	1	2	0	0	0	0	9	0	3	0	0
Spok = M	0	0	0	0	0	0	0	12	7	0	6	0	67	3	1	0	2	0	0	0	1	0	1
Claw = N	0	0	0	0	0	1	0	2	16	0	0	2	1	69	4	1	1	0	2	0	0	1	0
ASL-C = O	0	0	0	0	0	1	1	0	5	2	0	1	4	3	74	0	0	0	4	5	0	0	0
OK-Pose = P	0	1	0	0	0	0	0	0	0	0	0	2	0	0	0	94	2	0	1	0	0	0	0
Middle OK-Pose = Q	0	0	0	0	0	0	0	2	1	0	0	0	0	1	0	3	87	1	2	0	1	2	0
Pinch = R	4	2	0	1	1	1	4	0	0	0	0	0	0	1	0	0	0	71	5	9	1	0	0
Finger Purse = S	2	0	1	1	2	0	4	0	1	1	1	2	2	0	5	2	0	1	68	5	1	0	1
ASL-O = T	4	7	2	4	4	0	4	0	0	0	0	0	0	0	3	1	0	2	3	64	0	0	2
ASL-R = U	0	1	0	0	1	9	3	0	0	1	1	1	0	1	0	1	1	0	0	0	58	22	0
Inverse ASL-R = V	0	0	0	0	0	13	1	1	0	1	1	2	0	1	0	0	0	0	1	0	17	61	1
Thumbs-Up = W	0	2	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	343

Figure 3: Confusion Matrix

From this table, it is evident that the *Fist Poses* (Poses A through E) are all often misclassified as one another. Additionally, the *ASL-O* and *Index Forward* poses are often confused with the *Fist Poses* and vice versa. The *Thumb-Middle Group* pose had the lowest accuracy with only 61% correct classifications, where it was sometimes even classified as the *Finger Purse* and *ASL-R* poses. The *Pinch*, *Finger Purse*, and *ASL-O* poses were all sometimes misclassified as a *Fist Poses* and the *Index Forward* pose. Additionally, these three poses are often misclassified as one another, leading to a low recognition accuracy for all of them.The *ASL-R* and *Inverse ASL-R* poses (poses U and V) both have a low recognition accuracy rate. Both were misclassified as one another very often, and were regularly misclassified as the *Point* pose.

Since the notation strings are all of the same length, Hamming Distance may be used as a simple measure of similarity. Hamming Distance is the number of occurrences of differences between two strings of equal lengths, and was introduced in [10]. For example, the *ASL-R* and *Open Hand* poses have notation strings $61166; 2252-423$ and $31111; 2222-222$, giving them a distance of 6. From this example, one could predict that the Hamming Distance between two poses is low if the poses appear to be visually similar to one-another, and high if they're visually different.

When comparing the distances to the misclassification errors in Fig 3, some distances are too large for poses that are visually not that different for e.g. *ASL-M* and the *ASL-A* has a distance of 6. Also, the distance between the *Point* and *ASL-A* poses is less than the distance of 6, implying that a *Point* is considered to be more similar to the *ASL-A* pose than *ASL-M*. Flat Hand and ASL-A were never misclassified as one-another, yet have a low distance of 4. Conversely, Claw and Finger Purse have a single misclassification between them, which should never happen between two poses at maximum distance from one another with twenty-two other poses to be chosen from.

A possible solution to this problem is to create a weighted Hamming Distance that would take silhouette changes into account. Where previously any change between notation string elements increases Hamming Distance by one, weighted Hamming Distance would increase the distance according to how much the visual silhouette of the pose is changed. The change in visual silhouette is not objectively measured, but is rather used as a tool to estimate the weights to be assigned. For example, the difference between a fully curled and fully extended finger should be much larger than that of a fully curled and partially curled finger. These weightings are depicted in Tables 4 and 5. A heatmap showing similarity scores between pose pairs is given in Fig 4. Bright red cells with values close to 0 indicate high similarity while lighter coloured cells with high values indicate poses that are substantially different from one another.

Table 4: Finger Pose Distance Weightings

Finger Pose Notator	1	2	3	4	5	6
Point (Up): 1	0.0					
Point (Forward): 2	1.0	0.0				
Point (Side): 3	1.0	1.0	0.0			
Neutral: 4	0.3	0.5	0.5	0.0		
Bend: 5	1.0	0.5	1.0	0.2	0.0	
Close: 6	2.0	1.5	1.5	1.5	0.7	0.0

Table 5: Finger Inter-Relation Distance Weightings

Finger Inter-Relation Notator	1	2	3	4	5	6	7
Neutral: 1	0.0						
Separate: 2	0.2	0.0					
Group: 3	0.6	1.0	0.0				
Cross (i on palm-side of j): 4	0.6	1.0	0.2	0.0			
Cross (j on palm-side of i): 5	0.6	1.0	0.2	0.1	0.0		
Touch: 6	0.6	1.0	0.3	0.3	0.3	0.0	
Loop: 7	2.0	2.0	1.5	0.9	0.9	0.5	0.0

3.3.2 *The algorithm for creating a reliable pose set.* In order for the pose recognition system to work effectively, a set of poses must be identified which achieves a near-perfect pose recognition accuracy. We term such a pose set a reliable pose set. An algorithm for constructing a reliable pose set is given in algorithm 1. The algorithm takes a set of poses as input and iteratively identifies and discards problematic poses until the target accuracy is achieved on the remaining poses. Poses are discarded based on their weighted Hamming distances to other other poses. In general, the poses that are most frequently misclassified and that have the lowest average distance to other poses are removed. Removing these poses results in an increase in the recognition accuracy. This process is repeated until the target accuracy is reached. However, certain poses, or key poses can be tagged for inclusion in the reliable pose set. These could be poses intuitive poses or poses typically used in VR applications. These key poses are skipped during the removal process and will be included in the final pose set. For example the key pose set could consist of: Classic Fist, Point, OK-Pose, Thumbs-up and either the Open Hand or Neutral Hand, but not both. If one of the key poses is identified for removal it is skipped and the next pose that is not a key pose is selected for removal.

```
Input: Set A: All pose types
Output: Set B: Reliable pose set

factorThreshold ⟵ 0.6
set B ⟵ clone(set A)

while SVM-PUK accuracy on Set B < 99% do
    foreach pose P in set B do
        calculate P.averageHammingDistance
        calculate P.misclassificationPercent
    end
    foreach pose P in set B do
        normalize P.averageHammingDistance
         ▷ Highest average Hamming Distance becomes 1,
          lowest becomes 0
        normalize P.misclassificationPercent
         ▷ Highest misclassification % becomes 1, lowest
          becomes 0
        P.factor <- P.averageHammingDistance *
          P.misclassificationPercent
    end
    R ⟵ non-key pose in Set B with highest factor
    if R.factor >= factorThreshold then
        remove R from set B
        skip to next iteration of while loop
    end
    R ⟵ non-key pose with lowest averageHammingDistance
    remove R from set B
end

return Set B
```
Algorithm 1: Algorithm for creating a reliable pose set.

Following algorithm 1, the reliable pose set consists of three poses, i.e. the *Open Hand*, *OK-Pose*, and *Thumbs-Up* poses. The data set consisting of these three poses at arbitrary orientations achieved an accuracy of 99.45% with the SVM-PUK classifier. This algorithm could be used to produce other reliable pose sets. For example, the key poses to be kept or the target accuracy could be changed if desired. By changing the `factorThreshold`, one could alter the frequency at which the poses with a high `factor` are removed. Other parameters, like the pose input set and classification algorithm, could also be changed.

4 DISCUSSION

In order to facilitate the acquisition and analyse of poses, we developed a taxonomy and a notation for hand poses. We used the taxonomy to construct a representative pose data set consisting of 29 different static hand poses. Machine learning experiments for pose recognition on the full pose set yielded accuracies that were well below what might be expected in the VR industry. However, typical VR applications will not require the full pose set. We thus propose a method for analysing pose similarity and eliminating

Ground Truth (Rows) Classified As (Cols)	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
ASL-A = A	0.0	3.2	4.2	4.8	3.8	7.0	5.7	13.0	8.7	11.0	8.0	9.8	11.0	7.8	4.8	11.0	12.5	2.8	10.3	7.3	9.2	9.2	2.0
Classic Fist = B	3.2	0.0	1.2	3.5	2.5	4.0	2.6	14.5	10.9	10.0	11.2	10.9	12.5	8.5	5.5	11.1	11.1	4.3	10.1	5.3	8.2	8.2	3.5
Hidden Thumb = C	4.2	1.2	0.0	2.3	1.4	5.1	3.6	15.5	11.9	11.0	12.2	11.8	13.5	9.5	6.5	12.1	12.1	5.3	9.4	6.3	7.3	7.3	4.5
ASL-M = D	4.8	3.5	2.3	0.0	2.3	7.4	5.9	15.2	11.0	13.3	12.8	12.4	14.6	9.0	7.4	11.8	11.8	5.9	8.3	7.2	9.6	9.6	5.6
ASL-N = E	3.8	2.5	1.4	2.3	0.0	6.4	4.9	14.2	10.0	12.3	11.8	11.4	12.2	8.0	6.4	10.8	10.8	4.9	9.0	6.2	7.1	7.1	4.6
Point = F	7.0	4.0	5.1	7.4	6.4	0.0	2.0	10.5	8.9	8.0	11.0	9.4	10.5	6.8	5.8	12.1	7.7	5.5	11.3	7.7	6.0	6.0	5.5
Index Forward = G	5.7	2.6	3.6	5.9	4.9	2.0	0.0	12.5	9.5	10.0	11.2	9.4	12.5	7.3	6.3	10.5	9.7	3.8	9.6	6.1	8.0	8.0	6.0
Open Hand = H	13.0	14.5	15.5	15.2	14.2	10.5	12.5	0.0	2.5	4.5	5.0	5.2	2.0	5.0	8.0	4.6	4.6	12.0	12.0	12.0	8.5	8.5	11.0
Neutral Hand = I	8.7	10.9	11.9	11.0	10.0	8.9	9.5	2.5	0.0	4.7	3.9	4.3	3.3	1.8	3.0	3.3	3.5	7.9	7.9	6.8	7.7	7.7	8.5
ASL-B = J	11.0	10.0	11.0	13.3	12.3	8.0	10.0	4.5	4.7	0.0	3.0	4.6	2.5	7.7	4.7	5.5	5.5	9.5	9.5	8.7	6.2	6.2	9.5
Flat Hand = K	8.0	11.2	12.2	12.8	11.8	11.0	11.2	5.0	3.9	3.0	0.0	1.8	3.0	9.0	6.0	5.3	6.8	8.3	8.3	8.5	9.2	9.2	10.0
Thumb-Middle Group = L	9.8	10.9	11.8	12.4	11.4	9.4	9.4	5.2	4.3	4.6	1.8	0.0	4.6	9.2	7.6	5.1	5.7	9.9	8.2	8.0	10.6	10.6	11.6
Spok = M	11.0	12.5	13.5	14.6	12.2	10.5	12.5	2.0	3.3	2.5	3.0	4.6	0.0	7.0	6.0	5.4	5.4	10.0	10.0	10.0	6.7	6.7	9.0
Claw = N	7.8	8.5	9.5	9.0	8.0	6.8	7.3	5.0	1.8	7.7	9.0	9.2	7.0	0.0	3.0	5.6	5.6	7.1	9.5	7.0	7.1	7.1	6.8
ASL-C = O	4.8	5.5	6.5	7.4	6.4	5.8	6.3	8.0	3.0	4.7	6.0	7.6	6.0	3.0	0.0	6.8	6.8	4.1	6.5	4.0	6.3	6.3	3.8
OK-Pose = P	11.0	11.1	12.1	11.8	10.8	12.1	10.5	4.6	3.3	5.5	5.3	5.1	5.4	5.6	6.8	0.0	6.0	9.3	9.3	6.8	10.1	10.1	11.5
Middle OK-Pose = Q	12.5	11.1	12.1	11.8	10.8	7.7	9.7	4.6	3.5	5.5	6.8	5.7	5.4	5.6	6.8	6.0	0.0	11.0	9.3	6.8	10.1	10.1	11.5
Pinch = R	2.8	4.3	5.3	5.9	4.9	5.5	3.8	12.0	7.9	9.5	8.3	9.9	10.0	7.1	4.1	9.3	11.0	0.0	7.5	5.6	7.7	7.7	3.5
Finger Purse = S	10.3	10.1	9.4	8.3	9.0	11.3	9.6	12.0	7.9	9.5	8.3	8.2	10.0	9.5	6.5	9.3	9.3	7.5	0.0	5.5	11.0	11.0	11.0
ASL-O = T	7.3	5.3	6.3	7.2	6.2	7.7	6.1	12.0	6.8	8.7	8.5	8.0	10.0	7.0	4.0	6.8	6.8	5.6	5.5	0.0	10.3	10.3	7.8
ASL-R = U	9.2	8.2	7.3	9.6	7.1	6.0	8.0	8.5	7.7	6.2	9.2	10.6	6.7	7.1	6.3	10.1	10.1	7.7	11.0	10.3	0.0	0.1	7.7
Inverse ASL-R = V	9.2	8.2	7.3	9.6	7.1	6.0	8.0	8.5	7.7	6.2	9.2	10.6	6.7	7.1	6.3	10.1	10.1	7.7	11.0	10.3	0.1	0.0	7.7
Thumbs-Up = W	2.0	3.5	4.5	5.6	4.6	5.5	6.0	11.0	8.5	9.5	10.0	11.6	9.0	6.8	3.8	11.5	11.5	3.5	11.0	7.8	7.7	7.7	0.0

Figure 4: Heat Map showing weighted Hamming distance

similar poses which results in a selection of distinct poses that meets industry performance requirements. By eliminating poses to form a reliable set, a trade-off is made between including more poses for a given VR application or a higher recognition accuracy.

Finger occlusion has been a persistent issue throughout this research. Often, poses that have their features hidden from the camera were not detected correctly. During the data gathering process, it was often noted that the participant's virtual hands did not correctly mimic their real hands. This usually occurred whenever the pose being made included some form of finger occlusion, either due to the shape or orientation of the hand itself. Finger occlusion significantly impacts recognition performance, possibly more so than the choice in machine learning classifier. Some research has attempted to solve this problem, such as [19], where two LMCs were placed at right angles to gain different perspectives on the same hand. They will be required to keep their hands in a stationary space for tracking, whereas a single camera can be mobile by attaching it to the front of the head-mounted display. A solution needs to be found where a second camera could be used without restricting user movement.

The primary advantage of using cameras over hand-held controllers is the freedom of being able to make any arbitrary hand pose and having the environment react accordingly. In creating a VR application with hand pose controls, it is up to the creator to decide whether the freedom and convenience of a camera-based approach is preferable to the reliability of remote-based controls.

5 CONCLUSIONS

This work explored the limits of the Leap Motion Controller to capture hand pose input in a virtual reality environment. A pose taxonomy, pose notation and pose data set was developed to analyse and evaluate the use of the LMC for pose recognition. To achieve acceptable recognition accuracies for VR applications, a system was developed that systematically derived a pose set that could produce an accuracy of 99% using the SVM-PUK classifier.

This work contributes a parameterized algorithm for producing a pose set with the desired recognition accuracy. Further contributions include a benchmark pose set and dataset that can be used by other researchers to compare machine learning classifiers for VR pose recognition.

The findings of this research have shown that cameras suffer from input inaccuracies too often to replace controllers. Poor input data is generally caused by finger occlusion. However, cameras allow for a wider array of poses, provide more freedom, and are less cumbersome than hand-held controllers.

ACKNOWLEDGMENTS

This work is based on the MSc project of the first author. It has not been previously published. The first author gratefully acknowledges the financial support of the CSIR-DST Interbursary Support program.

REFERENCES

[1] M. Alimanova, S. Borambayeva, D. Kozhamzharova, N. Kurmangaiyeva, D. Ospanova, G. Tyulepberdinova, G. Gaziz, and A. Kassenkhan. 2017. Gamification of Hand Rehabilitation Process Using Virtual Reality Tools: Using Leap Motion for Hand Rehabilitation. In *2017 First IEEE International Conference on Robotic Computing (IRC)*. 336–339. https://doi.org/10.1109/IRC.2017.76

[2] Nathan Beattie, Ben Horan, and Sophie McKenzie. 2015. Taking the LEAP with the Oculus HMD and CAD - Plucking at thin Air? *Procedia Technology* 20 (Jan. 2015), 149–154. https://doi.org/10.1016/j.protcy.2015.07.025

[3] J. Blaha and M. Gupta. 2014. Diplopia: A virtual reality game designed to help amblyopics. In *Virtual Reality (VR), 2014 iEEE.* 163–164. https://doi.org/10.1109/VR.2014.6802102

[4] Eunjung Choi, Heejin Kim, and Min K. Chung. 2014. A taxonomy and notation method for three-dimensional hand gestures. *International Journal of Industrial Ergonomics* 44, 1 (Jan. 2014), 171–188. https://doi.org/10.1016/j.ergon.2013.10.011

[5] Ching-Hua Chuan, E. Regina, and C. Guardino. 2014. American Sign Language Recognition Using Leap Motion Sensor. In *2014 13th International Conference on Machine Learning and Applications (ICMLA).* 541–544. https://doi.org/10.1109/ICMLA.2014.110

[6] Andrew Clark and Deshendran Moodley. 2016. A System for a Hand Gesture-Manipulated Virtual Reality Environment. In *Proceedings of the Annual Conference of the South African Institute of Computer Scientists and Information Technologists (SAICSIT '16).* ACM, New York, NY, USA, 10:1–10:10. https://doi.org/10.1145/2987491.2987511

[7] Bruno Fanini. 2014. 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.* http://citeseerx.ist.psu.edu/viewdoc/citations;jsessionid=DBFED2BDCFA6A3B6FE84489C8DA2528E?doi=10.1.1.673.7112

[8] Jože Guna, Grega Jakus, Matevž Pogačnik, Sašo Tomažič, and Jaka Sodnik. 2014. An Analysis of the Precision and Reliability of the Leap Motion Sensor and Its Suitability for Static and Dynamic Tracking. *Sensors* 14, 2 (Feb. 2014), 3702–3720. https://doi.org/10.3390/s140203702

[9] Isabelle Guyon, Vassilis Athitsos, Pat Jangyodsuk, and Hugo Jair Escalante. 2014. The ChaLearn Gesture Dataset (CGD 2011). *Mach. Vision Appl.* 25, 8 (Nov. 2014), 1929–1951. https://doi.org/10.1007/s00138-014-0596-3

[10] R. W. Hamming. 1950. Error detecting and error correcting codes. *The Bell System Technical Journal* 29, 2 (April 1950), 147–160. https://doi.org/10.1002/j.1538-7305.1950.tb00463.x

[11] D. E. Holmes, D. K. Charles, P. J. Morrow, S. McClean, and S. M. McDonough. 2016. Using Fitt's Law to Model Arm Motion Tracked in 3D by a Leap Motion Controller for Virtual Reality Upper Arm Stroke Rehabilitation. In *2016 IEEE 29th International Symposium on Computer-Based Medical Systems (CBMS).* 335–336. https://doi.org/10.1109/CBMS.2016.41

[12] Maria Karam and M. C. Schraefel. 2005. *A Taxonomy of Gestures in Human Computer Interactions.* Master's thesis. North Dakota State University.

[13] Stoyan Kerefeyn and Stoyan Maleshkov. 2015. Manipulation of virtual objects through a LeapMotion optical sensor. *ResearchGate* 12, 5 (Oct. 2015), 52–57. https://www.researchgate.net/publication/283643278_Manipulation_of_virtual_objects_through_a_LeapMotion_optical_sensor

[14] S. Khattak, B. Cowan, I. Chepurna, and A. Hogue. 2014. A real-time reconstructed 3D environment augmented with virtual objects rendered with correct occlusion. In *2014 IEEE Games Media Entertainment (GEM).* 1–8. https://doi.org/10.1109/GEM.2014.7048102

[15] C. Khundam. 2015. 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).* 325–330. https://doi.org/10.1109/JCSSE.2015.7219818

[16] W. Lu, Z. Tong, and J. Chu. 2016. Dynamic Hand Gesture Recognition With Leap Motion Controller. *IEEE Signal Processing Letters* 23, 9 (Sept. 2016), 1188–1192. https://doi.org/10.1109/LSP.2016.2590470

[17] Rajesh B. Mapari and Govind Kharat. 2016. American Static Signs Recognition Using Leap Motion Sensor. In *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies (ICTCS '16).* ACM, New York, NY, USA, 67:1–67:5. https://doi.org/10.1145/2905055.2905125

[18] G. Marin, F. Dominio, and P. Zanuttigh. 2014. Hand gesture recognition with leap motion and kinect devices. In *2014 IEEE International Conference on Image Processing (ICIP).* 1565–1569. https://doi.org/10.1109/ICIP.2014.7025313

[19] M. Mohandes, S. Aliyu, and M. Deriche. 2015. 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).* 1–6. https://doi.org/10.1109/SSD.2015.7348113

[20] Javier Molina, JosÃ© A. Pajuelo, Marcos Escudero-ViÃ±olo, JesÃºs BescÃ³s, and JosÃ© M. MartÃnez. 2014. A natural and synthetic corpus for benchmarking of hand gesture recognition systems. *Machine Vision and Applications* 25, 4 (May 2014), 943–954. https://doi.org/10.1007/s00138-013-0576-z

[21] Jaime Ruiz, Yang Li, and Edward Lank. 2011. User-defined Motion Gestures for Mobile Interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '11).* ACM, New York, NY, USA, 197–206. https://doi.org/10.1145/1978942.1978971

[22] K. Sabir, C. Stolte, B. Tabor, and S.I. O'Donoghue. 2013. The Molecular Control Toolkit: Controlling 3D molecular graphics via gesture and voice. In *2013 IEEE Symposium on Biological Data Visualization (BioVis).* 49–56. https://doi.org/10.1109/BioVis.2013.6664346

[23] D. Shukla, Ã– Erkent, and J. Piater. 2016. A multi-view hand gesture RGB-D dataset for human-robot interaction scenarios. In *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN).* 1084–1091. https://doi.org/10.1109/ROMAN.2016.7745243

[24] Gurminder Singh, Steven K. Feiner, and Daniel Thalmann. 1994. *Virtual Reality Software & Technology: Proceedings of the VRST '94 Conference, 23-26 August 1994, Singapore.* World Scientific. Google-Books-ID: ywTGrWPf518C.

[25] M. Sourial, A. Elnaggar, and D. Reichardt. 2016. Development of a virtual coach scenario for hand therapy using LEAP motion. In *2016 Future Technologies Conference (FTC).* 1071–1078. https://doi.org/10.1109/FTC.2016.7821736

[26] Jeremy Sutton. 2013. Air Painting with Corel Painter Freestyle and the Leap Motion Controller: A Revolutionary New Way to Paint!. In *ACM SIGGRAPH 2013 Studio Talks (SIGGRAPH '13).* ACM, New York, NY, USA, 21:1–21:1. https://doi.org/10.1145/2503673.2503694

[27] Michail Theofanidis, Saif Iftekar Sayed, Alexandros Lioulemes, and Fillia Makedon. 2017. VARM: Using Virtual Reality to Program Robotic Manipulators. In *Proceedings of the 10th International Conference on PErvasive Technologies Related to Assistive Environments (PETRA '17).* ACM, New York, NY, USA, 215–221. https://doi.org/10.1145/3056540.3056541

[28] Fereydoon Vafaei. 2013. *Taxonomy of Gestures in Human Computer Interaction.* Master's thesis. North Dakota State University. http://library.ndsu.edu/tools/dspace/load/?file=/repository/bitstream/handle/10365/23110/Vafaei_Taxonomy%20of%20Gestures%20in%20Human%20Computer%20Interaction.pdf?sequence=1

[29] Radu-Daniel Vatavu and Ionut-Alexandru Zaiti. 2014. Leap Gestures for TV: Insights from an Elicitation Study. In *Proceedings of the ACM International Conference on Interactive Experiences for TV and Online Video (TVX '14).* ACM, New York, NY, USA, 131–138. https://doi.org/10.1145/2602299.2602316

[30] Q. Wang, Y. Wang, F. Liu, and W. Zeng. 2017. Hand gesture recognition of Arabic numbers using leap motion via deterministic learning. In *2017 36th Chinese Control Conference (CCC).* 10823–10828. https://doi.org/10.23919/ChiCC.2017.8029083

[31] Jacob O. Wobbrock, Meredith Ringel Morris, and Andrew D. Wilson. 2009. User-defined Gestures for Surface Computing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09).* ACM, New York, NY, USA, 1083–1092. https://doi.org/10.1145/1518701.1518866