VRGestures: Controller and Hand Gesture Datasets for Virtual Reality*

Georgios Papadopoulos[1][0009−0006−1726−3714],
Alexandros Doumanoglou[1][0000−0002−4337−1720], and
Dimitrios Zarpalas[1][0000−0002−9649−9306]

Centre for Research and Technology Hellas, Thessaloniki, Greece[1]
https://vcl.iti.gr/
giorgospap@iti.gr aldoum@iti.gr zarpalas@iti.gr

Abstract. Gesture Recognition is attracting increasingly more attention over the years and has been adopted in main applications in the real world and the Virtual one. New generation Virtual Reality (VR) headsets like the Meta Quest 2 support hand tracking very efficiently and are challenging the research community for more breakthrough discoveries in Hand Gesture Recognition. What has also been quietly improved recently are the VR controllers, which have become wireless and also more practical to use. However, when it comes to VR gesture datasets, and especially controller gesture datasets there are limited data available. Point-And-Click methods are widely accepted, which is why gestures are being neglected, combined with the shortage of available datasets. To address this gap we provide two datasets one with controller gestures and one with hand gestures, capable of recording with either controller or hand and even with both hands simultaneously. We created two VR applications to record for controllers and hands the position and the orientation and also each timestamp that we record data. Then we trained off-the-shelf time series classifiers to test our data, export metrics, and compare different subsets of our datasets between them. Hand gesture recognition is far more complicated than controller gesture recognition as we take almost thrice input and the difference is being analyzed and discussed with findings and metrics. The datasets are available online https://doi.org/10.5281/zenodo.8027807

Keywords: Hand Gestures · Controller Gestures · Virtual Reality · Machine Learning · Dataset.

1 Introduction

A technology that supports highly accurate and effective Gesture Recognition (GR) in Virtual Reality (VR) environments has been envisioned in many popular sci-fi movies, from a long time ago. So many years after those movies, it is now the time, that finally the technology has just begun to become mature enough

* Centre for Research and Technology Hellas - Information Technologies Institute

for those visions to become a reality. VR devices have rapidly evolved over the last decade and massively increased their capabilities and potential. Improved features, wireless equipment, better resolution, and lower market price are just a few of the major upgrades. New research horizons have broadened the Artificial Intelligence field, unlocking possibilities that were abandoned before due to the lack of methods and equipment. The recently released Meta's Quest 2[1], is the first standalone VR device, which supports bare hand tracking, wireless controllers, and wireless head-mounted display (HMD). As of now, hand tracking has been a milestone, that has not been conquered entirely. Machine learning models were unable to handle big datasets with many features and for that reason, deep learning methods were used to address the hand gesture recognition research field. VR controllers have also experienced revolutionary changes, cabled HMDs are outdated and are being replaced by wireless ones. What is common between controllers and hands is the Human-Computer-Interaction, which is achieved with Point and Click. This method is widely accepted by the users resulting in gestures being neglected on many occasions even where they could be extremely meaningful. We grounded on these flaws to contribute with two datasets to fill some gaps in the bibliography and provide two solemn tools for researchers to work with in future gesture recognition systems. Introducing gestures in a VR application consists of disproportionate sizes in terms of usability and effort required and that, precisely, is the main reason for gestures to be neglected, in combination with the shortage of available datasets, regarding controller and hand gestures. Collecting a dataset is a high effort time-consuming task, with large ambiguities even between samples of the same gesture class(Fig.1). However, without this effort, one can not train a machine-learning algorithm to identify dynamic movements as gestures. After all, gesture recognition whether with controllers or bare hands ends up being a time series classification problem and we will research whether the state-of-the-art time series classifiers with machine learning can achieve high accuracy or deep learning methods are truly required. Our contribution concerns the creation and publication of two completely different datasets, the former with controller gestures and the latter with hand static and dynamic gestures, supporting both left or right controller and hand. Additionally, we created a benchmark with off-the-shelf time-series algorithms from Sktime [13, 14] and will showcase a large diversity of subsets and metrics. We will discuss the findings and share some thoughts on future work, what else we could try to get different results, and what conclusions we finally concluded.

[1] meta.com/quest/products/quest-2

Fig. 1. Subjects while recording controller and hand gestures with the Meta Quest 2.

2 Related Work

The literature should be divided into two parts, one for controller gestures and one for hand gesture recognition. While the former seems more user-friendly, the latter draws more attention due to its potential. To begin with, not so long ago the standard way to record hand gestures in real time was either with a camera[12] or a depth sensor[11]. There are also approaches to classifying gestures outside VR[6] from video and image recordings[15, 16]. Rather similar research interest has been shown by[8], focusing on user experience and what is preferable either controllers or hands. A static Hand Gesture research work[20] using a publicly open hand dataset with grayscale images of hand poses demonstrates high accuracy in short-time recognition. Meanwhile, combining gestures with deep learning[2, 5, 29] in real-time has recently been a trend [3, 9]. In this work[19] 10 static gestures can be recognized by using a camera as a tracking device. Gesture classifiers have been researched in other fields, including neuroscience[21, 28] with prosthetic devices like armbands and sensors to create a human-machine interaction for higher gesture recognition[18, 26]. What should be mentioned here is that having gestures with both controllers or hands is not common. Attention is being drawn to one-handed gestures, especially with the right hand. A similar work[23] demonstrates an application of one-handed dynamic gestures to perform tasks in VR or even Augmented Reality (AR) and showcases why this kind of dataset needs to exist and improve. Another recent work[1] with gestures using the same equipment as us. In the aforementioned paper, a new game-based application with gesture interaction was created and the users had to perform gestures to proceed to the next stages of the game. Skeleton-based data[17] is being used to analyze patterns and train new convolutional neural networks[10]. These data have been widely used for action recognition[24] and extended to other research areas like human-robot-collaboration[25]. Finally, datasets are rare to find and differ vastly in gesture classes, design, and recording methods. One of the most recent papers[4] extends the NVIDIA[7] dataset to provide a new dataset with video recordings.

3 Gap Analysis and Motivation

The existing literature provides valuable insights into gesture recognition, although there are gaps that need to be addressed. We built upon existing work while also contributing with some new perspectives. As of now, there is limited availability in VR gesture datasets especially regarding controller gestures. To the best of our knowledge, these datasets are unique regarding the open access availability they provide, together with details on which hand was used. With respect to other existing datasets, we introduced a diversity allowing the user to draw a gesture with either hand or controller, concerning the user's preferred hand and we imported not only static gestures but also dynamic ones. Our datasets take into consideration the case of both controllers or hands being used simultaneously to draw a gesture. We are recording detailed information on hand position and orientation through time, as one unified object but also as a union of 24 joints with respect to the wrist. Our participants are relative to computer science, the age variance is 25-40 years old and previous VR experience with hand tracking is little to none. Our motivation arises from the fact that already existing datasets are short in terms of information, mostly by ignoring some details that could be meaningful and essential to others and are limited to one-handed gestures and only with a specific hand. We take a novel approach that combines all the aforementioned work, reinforcing the existing literature with a VR controller dataset and publishing a VR hand dataset with respect to both hands and the HMD. We argue that the aforementioned datasets provide unique dynamic two-handed VR gestures and also natural hand and controller movements. For the hand dataset, gestures are common actions resembling grabbing or pushing with either hand, and for the controller dataset, gestures are easily memorable as they resemble symbols that are being used daily. Taking into account the HMD relocation is crucial for VR applications as it is also pretty elegant to have access to each recording. New subsets can be created with great flexibility as one can choose to create subsets only from a specific hand or with limited classes or even a combination of both. It should be mentioned that despite having information on which controller or hand was used, the sampling methods were completely anonymous and could not be matched to the subjects participating in the experiment.

4 Applications

To record the two datasets we created two different applications one for recording controller gestures and one for hand gestures. The controller recording application is tracking both the controllers' position and rotation and each recording timestamp. It is built using the Steam VR SDK[2]. In Fig.2 there are examples of gestures being recorded. The user can observe what he records from these white spawning spheres. The hand gesture recording application is built on a different

[2] store.steampowered.com/app/250820/SteamVR

Fig. 2. Screenshots from the VR controller gesture recording application, showing how gestures were recorded. On top are gestures drawn with the left controller and examples with the right controller in the bottom row. All gestures could be drawn with either controller.

principle. We used the Voice SDK[3] deployed by meta so that the user activates the recording with his speech. There was a hand resembling figure on the left of the user's VR view which was performing on repeat the gesture the user should draw, as presented in Fig.3 and Fig.4. The user could start the process by saying a predefined word, then he would get optical feedback and an indication that recording is active. While in recording mode, the user performs one gesture and says a second predefined word to stop the recording and save the gesture. In our application, there is visual feedback on when the subject was recorded, a hand animation on one side to describe the gesture that should be drawn, and a newly created hand animation after the gesture was stored to observe what was recorded.

With the Meta Quest Developer Hub[4] application, we can monitor the user in both applications, guide him if necessary and make sure the recordings were performed under the same conditions for all the subjects. After each recording, the data were evaluated and reviewed concerning their quality, we retained some gestures where the HMD lost tracking momentarily, since this may occur in real-time applications as well. In an attempt to preserve the authenticity of the dataset, we asked the participants to perform each repetition slightly differently each time and only removed recordings that did not resemble the target gesture.

[3] assetstore.unity.com/packages/tools/integration/oculus-integration-82022
[4] developer.oculus.com/downloads/package/oculus-developer-hub-win

Fig. 3. The circled game object resembles a hand performing on repeat a gesture the user should draw. This figure portrays the RightInfinity gesture.

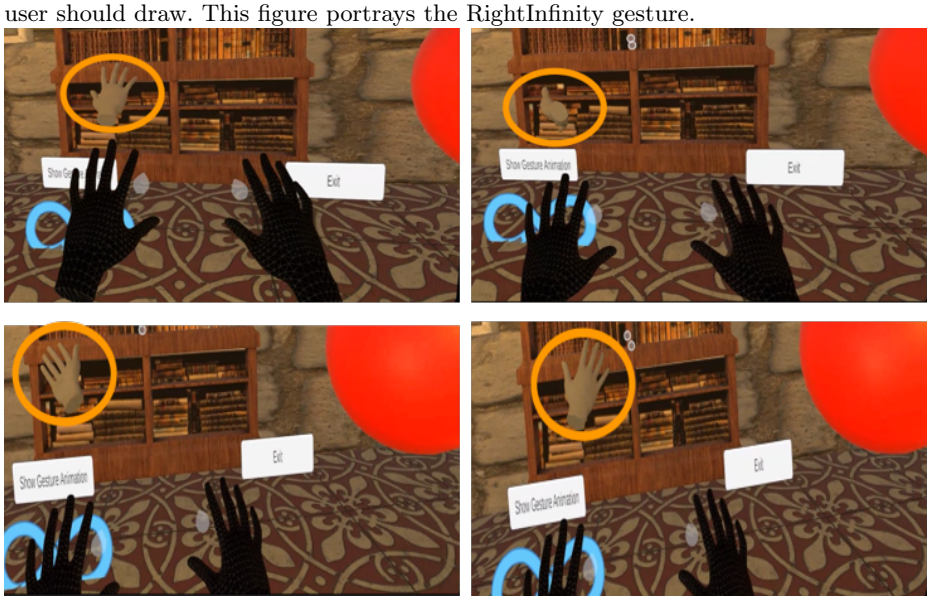
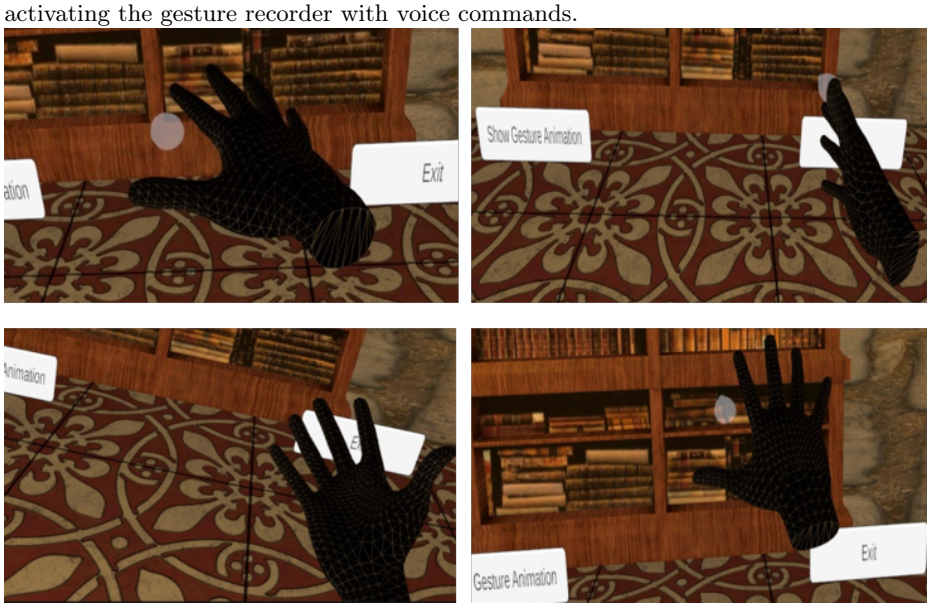


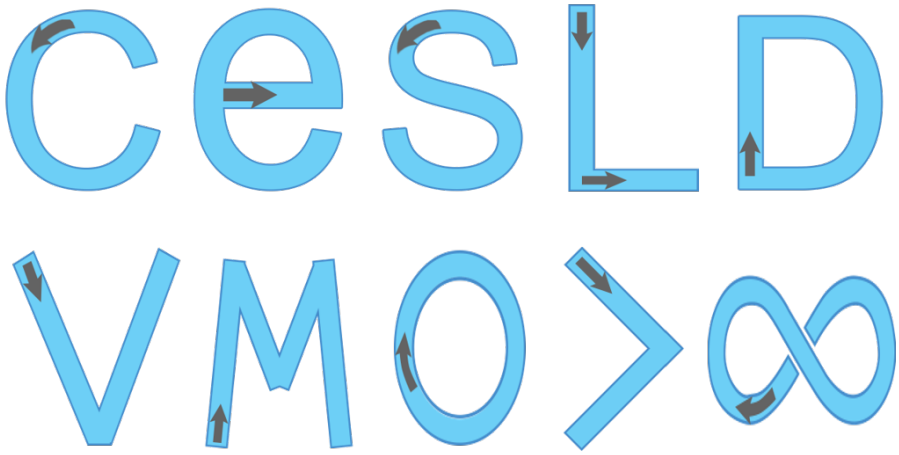
Fig. 4. After observing the motion of Fig.3 the user attempts to replicate the gesture, activating the gesture recorder with voice commands.



5 Datasets

Both datasets were recorded using the left-handed coordinate system of Unity3D. The VR device used was the Meta Quest 2. Drawing Controller gestures with the right and left hand are completely identical, while hand gestures with the left hand are mirroring the right hand gestures, to make them more approachable and easy to perform naturally and realistically. All gestures are stored in .bin files. To ensure that everything was recorded properly and labeled as it should, we created animation methods to observe and remove wrong recordings, which may occur due to tracking issues from the HMD or a misunderstanding of the task.

Fig. 5. Controller Gestures Vocabulary. The arrows are indicating the starting point and the route that should be followed.



To create our controller gesture dataset we recorded 16 subjects, 12 of which were right-handed and 4 were left-handed. Each participant was asked to draw some sample controller gestures with his preferred hand for 10 minutes. We collected samples from 15 different classes, 12 of them were English letters and 3 were symbols, with an average of 570 samples per class for right-handed gestures and 120 for left-handed. Our gesture names are L, O, I, U, V, e, S, D, Z, C, N, M, >, <, INFINITY. These symbols were chosen because there is no repetition in patterns when designed, to be easily memorable and useful in actions with a starting letter the same as the gesture name. These gestures were created and applied to the H2020 EC project INFINITY. They were tested under real-time conditions by end-users who did not participate in the collection of the dataset and were successfully integrated into different environments. Some of

these gestures are similar, to evaluate the performance in ambiguous cases and others differ to assure the generality of the dataset. To draw a gesture, the user must press and hold the trigger button of the controller and perform the desired gesture which resembles either letters of the English alphabet or common symbols. Upon release, we write down on the .bin file the subject tag, the gesture class, which hand was used, and discard the other. Each time, a dictionary is created with numpy arrays containing the position of the controller and HMD(X, Y, Z), the orientation of both of them(Qx, Qy, Qz, Qw), and each timestamp that was recorded. The subjects had been told to draw each gesture in a similar pattern and yet with many tiny differences in gesture shape, size, and controller orientation. Hand Recordings were trickier. Since there is no trigger button, somehow we should record the hand motion only for the duration of the gesture. As said before, we used voice commands to start/stop recording the gestures and even erase them if they did not respond to the desired one. We split the hand into 24 joints as placed by Oculus SDK in Unity3D[5] and recorded each joint for each hand. For one-handed gestures, we emptied the numpy array of the non-dominant hand. Similarly to the controller dataset, we record the position and orientation of each joint, of each hand as a whole object, and of the HMD. Finally, we recorded an array of timestamps for each recording frame, to access the duration of each gesture. For our Hand Gesture dataset, we collected samples from 34 subjects, 25 were right-handed and 9 were left-handed. Hand Gestures were either dynamic or static. 11 different gestures were recorded for either hand and two additional gestures with both hands drawing at the same time, which are included in the dataset but were not used in the benchmark. We collected an average of 307 samples per gesture with the right hand, 226 samples with the left, and 266 from both-handed gestures. Our 11 one-handed gesture names are: Back, Click, Close, Grab, Home, Infinity, OpenPalm, Point, Push, Rotate, and Scroll each with a prefix in front of the name, for example, the actual name is RightBack, LeftClick, etc. The sampling procedures were monitored and the samples have been cleared of wrong labeling or different design pattern. This way ensures that each sample in each class is a correctly drawn gesture and yet not quite identical to the rest of the samples in the same class. It must be emphasized that the datasets were designed consistently for the recognizer to achieve maximum accuracy. All participants were informed of the procedures and signed an informed consent form before the experiment.

6 Benchmarking

Sktime provides state-of-the-art transformers and classifiers for time-series classification and forecasting in pandas data frames. It is an open-source python library that provides a unified API, compatible with scikit-learn, in terms of methods fit and transforms to train time-series data. The selected algorithms offer a variety of different options and although they manage time-series data as input, the inner processing varies vastly and could be manipulated to produce

[5] docs.unity3d.com/Manual/index.html

a ton of different results based on the selected hyperparameters. Our data format takes as input a pandas dataframe with features and samples as columns and rows respectively but with different dimensions in the two datasets. For the controller dataset, the features are 3 positions(X Y Z) and 1 timestamp, which equals 4. For each cell, which is called a panda series, a new one-dimensional numpy array is being stored, the length of which depends on the duration of the gesture, in other words, how many timestamps were recorded. The hand dataset is more complex. Each feature consists of 3 positions(X Y Z), 24 more positions for each X, Y, Z calculating each joint position and 1 timestamp, which equals 76, and, similarly, with the controller dataset, each panda series is calculated for every timestamp of each sample. What is important to mention here is that each sample has an equal length of X, Y, Z and timestamps but the length may vary between samples. For instance, a sample contains 80 timestamps recorded, which means that it also contains 80 X, Y, Z matching each one of these timestamps, and that is one row of the pandas dataframe while another sample/row consists of 150 recorded timestamps and equally X, Y, Z. Unfortunately, this is not functional for our classifiers, for it is a necessity the pandas series to have the same length in the whole dataframe. To resolve this issue we applied a preprocess interpolation step, TSInterpolator from sktime, to our benchmark so that each sample duration is fixed to 180 timestamps. This essentially means that a sample with fewer than 180 timestamps will be stretched and another sample with 200 timestamps will be compressed so that all the samples of the dataframe have the same length of 180 for each panda series. To achieve translation invariance, we perform a pre-processing step before interpolation. In particular, before providing the input feature time series to the classifiers, we position the center of the feature space at the location of the first element in the time series, by subtracting from the X,Y,Z controller/hand/joint coordinates of frame t, the X,Y,Z controller/hand/joint coordinates of frame 0. This method allows us to calculate only the movement of the controller/hand/joint without distractions, like a possible head movement while recording, that could insert noise and weird angles. For simplicity reasons, we did not include orientation recordings as features to our benchmark although orientation has been recorded to both datasets as quaternions(Qx, Qy, Qz, Qw) for each recorded timestamp of controllers and hands and also for each of the 24 hand joints.

7 Experiments

We split our data to 70-30% train-test and we performed 5-fold cross-validation to sweep our data with the nevergrad optimizer and find the best hyperparameters from a selected range for our classifiers on the training set then we applied these hyperparameters to the test set to compare our classifiers. To implement the Sktime off-the-shelf algorithms for experiment tracking of our results we used mlflow[6], an open source platform, to access our experiments, be able to reproduce them and find the best hyperparameters for each algorithm and

[6] mlflow.org

concerning the evaluation metrics, which could be used to export results and findings. Moreover, to parse our data and be able to manipulate our code with minimal changes we used another Python framework, Hydra[27] which was combined with nevergrad[22] to sweep and find the best optimization parameters in a rapid and elegant approach. Our sweeper optimizer was set to NGOpt, which offers several settings to work within each run. A challenging task to increase the accuracy was not only to find the best algorithm for the complete datasets but also to divide them into subsets, explore the different metrics of each one of them and try to justify why this is the case on each subset. Defining a subset with gestures without many similarities between them is the best choice, but what is critical is to compare classifiers on gesture vocabularies containing gestures that are hard to distinguish. For the controller dataset, we tested our classifiers at first with 5 controller gestures(L, e, S, >, INFINITY) and then we added 5 more gestures(O, V, D, C, M). We wanted to test the accuracy of our classifier on a small dataset compared to one with a bigger gesture vocabulary, in an attempt to research their learning capacity. This metric was chosen to evaluate the performance of the gesture recognizer and not the quality of the data itself, to keep the authenticity of the dataset intact. All hyperparameters were set the same but the budget was doubled for the bigger dataset to achieve a fair comparison between the datasets. For the hand dataset, we used a subset of 5 gestures(Infinity, Scroll, Back, Grab, Point) with both the left and the right hand and we will showcase the best results for each classifier. For both datasets, we used 4 classifiers(Catch22, IndividualTDE, KNeighborsTimeSeries, RocketClassifier) that take as input different hyperparameters. The range of the hyperparameters that were used for the controller gesture subsets is presented in Table 1. With MLflow we got the best hyperparameters for each classifier. Table 2 presents the best hyperparameters as found for the small controller dataset while Table 3 contains those of the bigger one. The next step is to set these hyperparameters to the models to extract some metrics and compare the classifiers. Table 4 and Table 5 present the results of the test set on both controller datasets that we tested. The same process has been done for the hand gestures in the aforementioned subset of 5 hand gestures and we present the metrics of the right-hand gesture subset as the findings were better than the ones of the corresponding left. Table 6 presents the range of the hyperparameters that were tested, which is slightly different due to the increased complexity of the input of the hand data. Finally, Table 7 presents the best hyperparameters after the sweep, and Table 8 compares the classifier's metrics after testing on unseen data with the best hyperparameters for each of them.

8 Discussion

The findings are indicating that controller gesture recognition can be handled extremely well with the already existing classifiers being able to achieve the highest accuracy reaching out next to perfect classification on unseen data. The accuracy was almost the same whether with a small subset or two times bigger.

Table 1. Hyperparameters tested for each classifier for our controller subsets. The budget was set to 15 for the small subset and 30 for the bigger subset. The NGOpt sweeps random values in the given range of each hyperparameter for each classifier with respect to the budget.

Catch22	IndividualTDE	KNeighborsTimeSeries	RocketClassifier
n_estimators: range 20-100	Window_Size: range 5-20	n_neighbors: range 1 -20	num_kernels: range 20-200
outlier_norm: True or False	norm: True or False	weights: Uniform or Distance	max_dilations: range 1-50
	igb: True or False	algorithm: brute or ball_tree or kd_tree	n_features: range 1-3
	bigrams: True or False	distance: dtw or euclidean	
	alphabet_size: range 2-10	leaf_size: range 10-30	
	dim_threshold: range 0.7-1.0		
	max_dims: range 5-30		

Table 2. Best hyperparameters for each classifier of the 5 gesture Controller dataset

Catch22	IndividualTDE	KNeighborsTimeSeries	RocketClassifier
n_estimators: 58	window_size: 11	n_neighbors: 9	num_kernels: 142
outlier_norm: false	norm: false	weights: uniform	max_dilations_per_kernel: 46
	igb: false	algorithm: brute	n_features_per_kernel: 3
	alphabet_size: 5	distance: euclidean	
	bigrams: false	leaf_size: 20	
	dim_threshold: 0.824		
	max_dims: 14		

Table 3. Best hyperparameters for each classifier of the 10 gesture Controller dataset

Catch22	IndividualTDE	KNeighborsTimeSeries	RocketClassifier
n_estimators: 60	window_size: 15	n_neighbors: 13	num_kernels: 155
outlier_norm: true	norm: false	weights: distance	max_dilations_per_kernel: 16
	igb: false	algorithm: brute	n_features_per_kernel: 2
	alphabet_size: 5	distance: dtw	
	bigrams: true	leaf_size: 22	
	dim_threshold: 0.924		
	max_dims: 22		

Table 4. Best metrics on unseen data with the best hyperparameters for each classifier on the 5 gestures controller subset.

Classifiers	accuracy	balanced_accuracy	precision	f1 score	recall
Catch22	0.995	0.995	0.995	0.995	0.995
IndividualTDE	0.899	0.895	0.894	0.902	0.895
KNeighborsTimeSeries	0.992	0.992	0.992	0.992	0.992
RocketClassifier	0.997	0.997	0.997	0.997	0.997

Table 5. Best metrics on unseen data with the best hyperparameters for each classifier on the 10 gestures controller subset.

Classifiers	accuracy	balanced_accuracy	precision	f1 score	recall
Catch22	0.992	0.993	0.993	0.993	0.993
IndividualTDE	0.862	0.86	0.86	0.862	0.86
KNeighborsTimeSeries	0.992	0.992	0.992	0.992	0.992
RocketClassifier	0.994	0.994	0.994	0.994	0.994

Table 6. Hyperparameters tested for each classifier for our hand subset. The budget was set to 30.

Catch22	IndividualTDE	KNeighborsTimeSeries	RocketClassifier
n_estimators: range 20-100	Window_Size: range 5-20	n_neighbors: range 1 -20	num_kernels: range 5-500
outlier_norm: True or False	norm: True or False	weights: Uniform or Distance	max_dilations: range 1-50
	igb: True or False	algorithm: brute or ball_tree or kd_tree	n_features: range 1-3
	bigrams: True or False	distance: dtw or euclidean	
	alphabet_size: range 2-10	leaf_size: range 10-30	
	dim_threshold: range 0.7-1.0		
	max_dims: range 5-30		

Table 7. Best hyperparameters for each classifier of the 5 gesture Hand dataset

Catch22	IndividualTDE	KNeighborsTimeSeries	RocketClassifier
n_estimators: 100	window_size: 14	n_neighbors: 16	num_kernels: 329
outlier_norm: true	norm: false	weights: uniform	max_dilations_per_kernel: 35
	igb: true	algorithm: brute	n_features_per_kernel: 3
	alphabet_size: 7	distance: dtw	
	bigrams: true	leaf_size: 22	
	dim_threshold: 0.7		
	max_dims: 12		

Table 8. Best metrics on unseen data with the best hyperparameters for each classifier on the 5 gestures hand subset.

Classifiers	accuracy	balanced_accuracy	precision	f1 score	recall
Catch22	0.753	0.747	0.744	0.754	0.555
IndividualTDE	0.559	0.555	0.556	0.557	0.555
KNeighborsTimeSeries	0.368	0.364	0.353	0.359	0.364
RocketClassifier	0.692	0.685	0.673	0.675	0.685

On the contrary, even state-of-the-art machine learning algorithms are being confused when it comes to classifying hand gestures with many features. Deep learning algorithms are necessary to have a decent accuracy score. Particularly left-handed gestures failed to handle the ambiguity between the samples because the left-hand dataset was way smaller than the one with the right-hand gestures. Adding more hand gestures further confused the classifiers and they were not able to return worth showing results. The fact that in all the subsets used, the accuracy was almost equal to the balanced accuracy indicates that all the gestures are about equally important and the samples are distributed desirably.

9 Conclusion and Future Work

Gesture Recognition provides a shortcut to add another extra feature to VR applications. What we offer are two completely different datasets, containing all the information needed to make good use of them, which are also capable to function with either controller or hand allowing the user the choice to perform a gesture in any way that suits him better. Having also recorded the HMD position and orientation through time we are able to get out of the equation some random unintended head movements. What is an essential task in gesture recognition is to not recognize falsely the gestures. A drawing is preferred to be labeled as unknown rather than labeled incorrectly and perform the mapped action, which may lead to a domino of undesired sequential actions, causing discomfort to the user. In future research, we will focus on using these datasets to integrate them into a gesture-based application and achieve the same high results that we obtained in our experiments. What could also be researched in the future is a step between, which gives feedback to the user on the percentage of gesture recognition accuracy while drawing the gesture. It is meaningful to import an online/continuous recognition feature, to track the hands continuously and recognize when a pattern was drawn mid-air that is mapped to a predefined gesture. Another interesting topic to research would be Gesture Elicitation, giving the privilege the user to create gestures based on his preferences and map them to actions. Interesting as it may be to test our controller dataset with even smaller subsets even creating subsets by tag, to try deep learning methods with few-shot learning or one-shot learning, and try to trace the point where the accuracy of the controller gesture recognition is being reduced significantly, However, this is a challenging task to train the recognizer with limited samples of a gesture, because disambiguate issues will occur that require special treatment to make sure the precision of the recognizer is not affected deeply by the new inducted samples.

Acknowledgements We also acknowledge financial support by the H2020 EC project INFINITY under contract 883293. We thank Athanasios Ntovas for his help with the experiment setup, and the subjects recorded in our datasets for their participation.

References

1. Arendttorp, E.M.N., Rodil, K., Winschiers-Theophilus, H., Magoath, C.: Overcoming legacy bias: Re-designing gesture interactions in virtual reality with a san community in namibia. In: Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems. CHI '22, Association for Computing Machinery, New York, NY, USA (2022)
2. Bhaumik, G., Verma, M., Govil, M.C., Vipparthi, S.K.: Hyfinet: hybrid feature attention network for hand gesture recognition. Multimedia Tools and Applications **82**(4), 4863–4882 (2023)
3. D'Eusanio, A., Simoni, A., Pini, S., Borghi, G., Vezzani, R., Cucchiara, R.: A transformer-based network for dynamic hand gesture recognition. In: 2020 International Conference on 3D Vision (3DV). pp. 623–632 (2020). https://doi.org/10.1109/3DV50981.2020.00072
4. Fronteddu, G., Porcu, S., Floris, A., Atzori, L.: A dynamic hand gesture recognition dataset for human-computer interfaces. Computer Networks **205**, 108781 (2022)
5. Gnanapriya, S., Rahimunnisa, K.: A hybrid deep learning model for real time hand gestures recognition. Intelligent Automation & Soft Computing **36**(1) (2023)
6. Guo, L., Lu, Z., Yao, L.: Human-machine interaction sensing technology based on hand gesture recognition: A review. IEEE Transactions on Human-Machine Systems **51**(4), 300–309 (2021). https://doi.org/10.1109/THMS.2021.3086003
7. Gupta, P., Kautz, K., et al.: Online detection and classification of dynamic hand gestures with recurrent 3d convolutional neural networks. In: CVPR. vol. 1, p. 3 (2016)
8. Huang, Y.J., Liu, K.Y., Lee, S.S., Yeh, I.C.: Evaluation of a hybrid of hand gesture and controller inputs in virtual reality. International Journal of Human–Computer Interaction **37**(2), 169–180 (2021)
9. Jiang, S., Kang, P., Song, X., Lo, B.P., Shull, P.B.: Emerging wearable interfaces and algorithms for hand gesture recognition: A survey. IEEE Reviews in Biomedical Engineering **15**, 85–102 (2022). https://doi.org/10.1109/RBME.2021.3078190
10. Köpüklü, O., Gunduz, A., Kose, N., Rigoll, G.: Real-time hand gesture detection and classification using convolutional neural networks. In: 2019 14th IEEE international conference on automatic face & gesture recognition (FG 2019). pp. 1–8. IEEE (2019)
11. Kurakin, A., Zhang, Z., Liu, Z.: A real time system for dynamic hand gesture recognition with a depth sensor. In: 2012 Proceedings of the 20th European signal processing conference (EUSIPCO). pp. 1975–1979. IEEE (2012)
12. Lai, H.Y., Lai, H.J.: Real-time dynamic hand gesture recognition. In: 2014 International Symposium on Computer, Consumer and Control. pp. 658–661 (2014)
13. Löning, M., Bagnall, A., Ganesh, S., Kazakov, V., Lines, J., Király, F.J.: sktime: A unified interface for machine learning with time series. arXiv preprint arXiv:1909.07872 (2019)
14. Löning, M., Király, F., Bagnall, T., Middlehurst, M., Ganesh, S., Oastler, G., Lines, J., Walter, M., ViktorKaz, Mentel, L., chrisholder, Tsaprounis, L., RNKuhns, Parker, M., Owoseni, T., Rockenschaub, P., danbartl, jesellier, eenticott shell, Gilbert, C., Bulatova, G., Lovkush, Schäfer, P., Khrapov, S., Buchhorn, K., Take, K., Subramanian, S., Meyer, S.M., AidenRushbrooke, rice, B.: sktime/sktime: v0.13.4 (Sep 2022)
15. Materzynska, J., Berger, G., Bax, I., Memisevic, R.: The jester dataset: A large-scale video dataset of human gestures. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops (Oct 2019)

16. Mujahid, A., Awan, M.J., Yasin, A., Mohammed, M.A., Damaševičius, R., Maskeliūnas, R., Abdulkareem, K.H.: Real-time hand gesture recognition based on deep learning yolov3 model. Applied Sciences **11**(9) (2021). https://doi.org/10.3390/app11094164, https://www.mdpi.com/2076-3417/11/9/4164
17. Nguyen, X.S., Brun, L., Lezoray, O., Bougleux, S.: A neural network based on spd manifold learning for skeleton-based hand gesture recognition. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
18. Ntovas, A., Lazaridis, L., Papadimitriou, A., Psaltis, A., Axenopoulos, A., Daras, P.: Data-driven haptic feedback utilizing an object manipulation data-set
19. N'u nez-Fern'andez, D.: Development of a hand gesture based control interface using deep learning. In: Lossio-Ventura, J.A., Condori-Fernandez, N., Valverde-Rebaza, J.C. (eds.) Information Management and Big Data. pp. 143–150. Springer International Publishing, Cham (2020)
20. Núñez-Fernández, D.: Development of a Hand Gesture Based Control Interface Using Deep Learning. In: Lossio-Ventura, J.A., Condori-Fernandez, N., Valverde-Rebaza, J.C. (eds.) Information Management and Big Data. pp. 143–150. Springer International Publishing, Cham (2020)
21. Rahimian, E., Zabihi, S., Asif, A., Farina, D., Atashzar, S.F., Mohammadi, A.: Fs-hgr: Few-shot learning for hand gesture recognition via electromyography. IEEE Transactions on Neural Systems and Rehabilitation Engineering **29**, 1004–1015 (2021). https://doi.org/10.1109/TNSRE.2021.3077413
22. Rapin, J., Teytaud, O.: Nevergrad - A gradient-free optimization platform. https://GitHub.com/FacebookResearch/Nevergrad (2018)
23. Schäfer, A., Reis, G., Stricker, D.: Anygesture: Arbitrary one-handed gestures for augmented, virtual, and mixed reality applications. Applied Sciences **12**(4) (2022)
24. Shi, L., Zhang, Y., Cheng, J., Lu, H.: Skeleton-based action recognition with directed graph neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
25. Terreran, M., Lazzaretto, M., Ghidoni, S.: Skeleton-based action and gesture recognition for human-robot collaboration. In: Intelligent Autonomous Systems 17: Proceedings of the 17th International Conference IAS-17. pp. 29–45. Springer (2023)
26. Toro-Ossaba, A., Jaramillo-Tigreros, J., Tejada, J.C., Peña, A., López-González, A., Castanho, R.A.: Lstm recurrent neural network for hand gesture recognition using emg signals. Applied Sciences **12**(19) (2022)
27. Yadan, O.: Hydra - a framework for elegantly configuring complex applications. Github (2019), https://github.com/facebookresearch/hydra
28. Zabihi, S., Rahimian, E., Asif, A., Mohammadi, A.: Trahgr: Transformer for hand gesture recognition via electromyography (2022)
29. Zou, Y., Cheng, L.: A transfer learning model for gesture recognition based on the deep features extracted by cnn. IEEE Transactions on Artificial Intelligence **2**(5), 447–458 (2021). https://doi.org/10.1109/TAI.2021.3098253