Recognizing Head Gestures for Head-mounted Displays

Timon Ruban, Jeremy Wood
Stanford EE 267, Virtual Reality, Course Report, Instructors: Gordon Wetzstein and Robert Konrad
Spring 2016

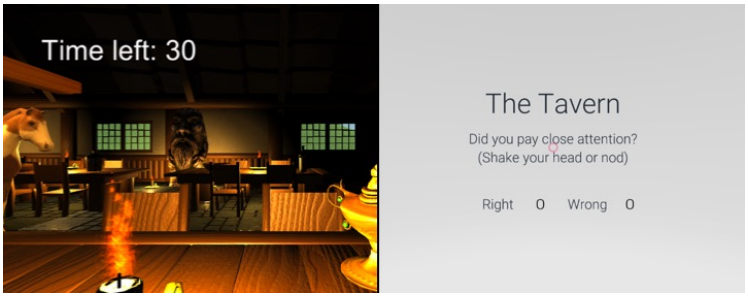


Figure 1: *Two images from different sections in our proof-of-concept demo that uses head gestures as input from the user.*

Abstract

Having intuitive user interfaces is critical for immersive virtual reality experiences. In this course project we set out to automatically recognize different head gestures and use them as input in a virtual environment. We use a k-Nearest-Neighbor classifier with Dynamic Time Warping as a distance measure, since it does not require encoding any gesture-specific knowledge into the input features and can therefore be easily extended to classify other user-defined head gestures. The inputs to our algorithm are sensor measurements from a head-mounted display and the output is a label YES, NO, or NULL (i.e. no substantive head gesture could be detected). After trading off accuracy for time needed to classify a data point and therefore using a reduced training dataset of size 198 we achieve a test accuracy of 0.8504.

Keywords: Virtual Reality, UI, Gesture Recognition, KNN, Dynamic Time Warping, Machine Learning

1 Introduction

Virtual Reality is on the rise. As building high-fidelity virtual environments becomes easier and easier finding an intuitive user interfaces for users to interact with their surroundings becomes a crucial step on the path to designing a truly immersive virtual experience. One of the most intuitive and natural reactions during interactions between humans is head gesturing. During conversation, head gestures often accompany or precede verbal communication, allowing for vocal communications to supervene on silent gestures. Not only is it used explicitly for communication, people often unconsciously perform head gesturing to indicate agreement: the act of moving our head to communicate is deeply tied to intuitive interaction. However, at present we found no packages that could perform any sort of complex classification of multiple head gestures. The best we found used only the fact that user had raised their head to indicate a nod. Yet as one of the most basic methods of communication, VR needs to support head gesturing as one of the basic inputs in interactive content.

For this project we want to explore the possibility espoused above, of using different head gestures (like nodding yes or shaking your head no) as a natural way to interact with the virtual environment. To this end, we developed a proof-of-concept wherein we create an intuitive environment that uses head gestures to create a VR-specific mechanism for immersing the user in an interactive environment.

2 Related Work

Gesture recognition has become increasingly prevalent as a research topic since the onset of body-tracking devices such as the KINECT. However, it seems there are only a few examples of open source gesture recognition systems for HMDs. The only example of existing code that we came across used hard-coded differences over a set time-interval [Bryla]. We think this leaves room for substantial improvement in the realm of HMDs.

Most work we have found on hand and head gesture recognition does not use only IMU sensor data as input. Instead, [S.J. Hwang] uses Kinect sensor data, [Frank Althoff and Walchshausl] uses infrared pictures and [Georgi et al. 2015] combines EMG data with IMU sensor data.

As far as algorithms to detect and classify a gesture are concerned: [Georgi et al. 2015] trains a simple SVM to classify different hand gestures. [S.J. Hwang] uses Dynamic Time Warping in conjunction with Nearest Neighbor Classification and Hidden Markov Models as their learning algorithms of choice. Another time-series comparison algorithm that seemed appropriate is one SpADe: ”The algorithm finds out matching segments within the entire time series, called patterns, by allowing shifting and scaling in both the temporal and amplitude dimensions. The problem of computing similarity value between time series is then transformed to the one of finding the most similar set of matching patterns.” However, over a wide set of datasets, Dynamic Time Warping performed just as well as SpADe [Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, Eamonn Keogh 2008].

3 Approach

In the following section we will outline the approach taken to go from sensor measurements to classification, explain the methods used and lie out metrics used to evaluate our algorithm.

3.1 Data Collection

Since there was no dataset with HMD-sensor measurements and corresponding head gesture labels readily available, we had to col-

lect our own. Using the Oculus Rift DK2 we had 15 different people collect 747 head gestures consisting of successive measurements of acceleration (X,Y,Z), angular velocity (X,Y,Z) and rotation quaternions (X,Y,Z,W) across variable-length time intervals.

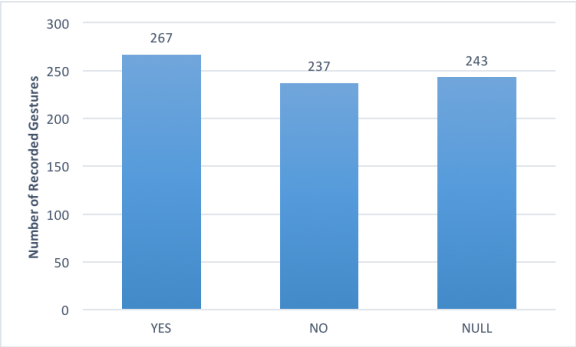


Figure 2: *Distribution of YES, NO and NULL in the collected dataset.*

We only collected these measurements if the average magnitude of the acceleration measurements within the last 30 frames was above a certain threshold (i.e. we only anticipate a head gesture if the user is moving his head fast enough). We used the same procedure during the proof-of-concept demo to decide when to collect sensor measurements that are then used as input to our trained classifier.

3.2 k-Nearest-Neighbor

We chose k-Nearest-Neighbor (KNN) as our baseline algorithm to classify the head gestures. It is a simple, but effective classification algorithm that involves no real training step and thus can load quickly. The training set is simply stored and when it comes to classifying a new data point it finds the k closest neighbors (according to some distance metric). These k closest neighbors then vote on the label to assign as a classification to the new data point. Any ties in majority vote would lead to a classification of NULL, indicating that the gesture was ambiguous and could not be classified.

By choosing k, the number of neighbors considered in making a classification decision, one can try to find the middle ground between the bias and variance of the algorithm. In addition, by changing the rules of how ties are treated and how many nearest neighbors need to agree on a label for a specific class one can make a trade-off between precision or recall (explained in subsection 3.4.2).

3.3 Dynamic Time Warping

As a distance measure for our KNN-classifier we use Dynamic Time Warping (DTW). DTW is a commonly used algorithm to compare varying time sequences. It computes the similarity ("distance") between any two time sequences by dynamically attempting to fit one time series to another by variably stretching or contracting the points in the time series. Thus DTW applied to a person shaking their head slowly and the same person shaking their head at times fast and at times slowly should output a small distance in Y-axis rotation (for a person whose axis of rotation is the Y-axis), as should both of these when compared to the same shake done very quickly. Though extremely useful for classifying time-series data of different origins, the baseline implementation of DTW requires $O(n^2)$ time. There are enhancements that reduce this time, but DTW remained the primary bottleneck in our algorithm. [Wang 2010]

3.4 Error Analysis

3.4.1 Forward Selection

We use forward selection to select which of the sensor measurements to use as features for our algorithm. Forward selection starts with no features selected and then greedily chooses the next feature that improves the validation accuracy the most until it reaches a local maximum. To get the validation accuracy we used Leave-One-Out Cross-Validation on the training set (LOOCV). We also used LOOCV to determine the best k for the KNN-classifier.

3.4.2 Precision and Recall

In order to better evaluate our performance on the validation sets across the different classifications, we used precision and recall to evaluate improvements on our algorithm. In our context, the precision of a gesture A is

$$Prec_A = \frac{\text{gestures of type A classified as A}}{\text{all gestures classified as A}}$$

and the recall is

$$Rec_A = \frac{\text{gestures of type A classified as A}}{\text{all gestures actually of type A}}$$

In order to combine these metrics in a substantive way, we employed tuned F-Scores to evaluate our performance on each gesture.

$$F(\beta, A) = (1+\beta^2)\frac{Prec_A \cdot Rec_A}{\beta^2 Prec_A + Rec_A}$$

This generalized F weights Recall β times more than precision. We hypothesized that for meaningful, substantive gestures (YES, NO) the precision matters more: a false positive–when a user does not intend to make a substantive gestures–leads to confusion and unintended interactions. However, for nonsubstantive classification (NULL), this same logic leads to an increased weighting on recall: meaningless gestures should always be classified as meaningless to avoid unintended responses. This led us to use the values $\beta_S = 0.7$ for substantive gestures and $\beta_{NS} = 1.5$ for non-substantive gestures. In order to holistically evaluate different metrics, we then took the average of the computed F-Scores as follows:

$$F_{avg} = \frac{F(\beta_S, YES) + F(\beta_S, NO) + F(\beta_{NS}, NULL)}{3}$$

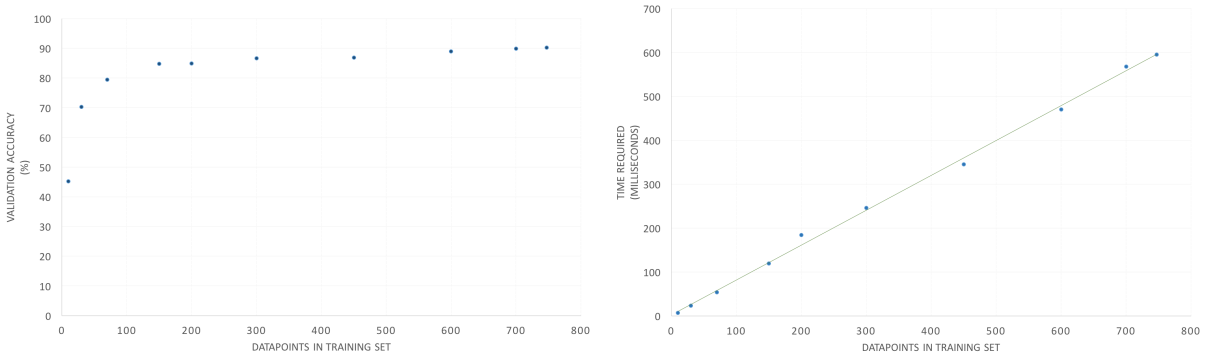
3.4.3 Test Accuracy

After selecting the best features, k and dataset size we measured the performance of our algorithm by calculating the accuracy on the remaining data points not touched during training or cross-validation.

4 Evaluation

The following section deals with the analysis, evaluation and improvement of our algorithm. We implemented it and all experiments using C# for the purpose of aiding integration with Unity during the demo phase [1].

[1] In order to implement K-Nearest-Neighbors with Dynamic Time Warping in the context of head mounted displays, we heavily modified, extended, and combined code from the following githubs: https://github.com/ahawker/KNN and https://github.com/doblak/ndtw



(a) *The learning curve for KNN with Dynamic Time Warping plateaus at training set size around 200.*

(b) *The time needed to classify one data point increases linearly with the training set size.*

Figure 3: *Comparison between validation accuracy and time needed to classify a data point dependent on training set size.*

4.1 Initial Results

Running forward selection and LOOCV on the entire training set yields a best validation accuracy of 0.9022 and the following results:

- Best $k = 5$
- Best Features:
  - Acceleration (X, Y, Z)
  - Rotation Quaternion (X, Y)

4.2 Speed Improvement

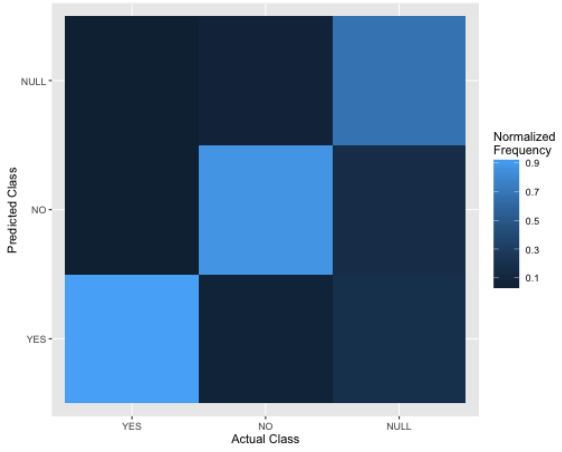
While both quantitative and qualitative results of the baseline established above are satisfactory (as far as accuracy is concerned) it suffers from an unacceptably long classification time (approx. 600ms). This leads to an lag in the interactive demo after inputting a head gesture, which disrupts immersion and forces the user to wait. This lag arises from two bottlenecks: the slow performance of DTW and the number of DTW calculations (i.e. training set data points) performed. To mitigate this problem we investigated how the validation accuracy and time needed for a classification step depend on the size of the training set (see Figure 3). Unsurprisingly, the time needed to classify a data point increases linearly with the training set size: every new data point must have the DTW-distance computed between it and every point in the training set (3b). Comparing this to the learning curve one can see that the validation accuracy plateaus quickly, leading to only marginal increases in accuracy with respect to increases in dataset size. By choosing a training set size of 198 (66 of each gesture type), around where the plateau begins, we concede less than a tenth of our accuracy but increase the speed of classification by a factor of $\sim$3x.

4.3 Precision/Recall Enhancements

Class	Precision	Recall	F-Score
YES	0.8801	0.9218	0.893
NO	0.8904	0.8074	0.861
NULL	0.8003	0.8302	0.821

Table 1: *Precision, Recall and F-Score for training set of size 198*

Using our speed-enhanced version with a condensed dataset, we performed repeated trials on training sets and observed the precision and recall (summarized in Table 1) based on the confusion matrix visualized in Figure 4. Combined, this yields an average F-Score of 0.859.



gra

Figure 4: *Normalized Confusion Matrix for LOOCV on training set of size 198.*

In order to improve performance, we attempted two enhancements on our initial algorithm. Our initial recall and precision results presented above suggested that our performance on NO was lower than our YES performance, specifically for recall. Thus in an attempt to bolster this aspect, we increased the sensitivity to classifying gestures as NO. Specifically, if $\frac{2}{5}$ths of the k-nearest-neighbors belonged to the NO class and the NO class had a majority relative to other substantive gestures (in this case, only YES) then the algorithm would classify the gesture as a NO. For K=5 and our example, this could result in a gesture with three NULL nearest neighbors and only two NO neighbors being classified as a NO. For the sake of comparison, we also examined the effect of increasing the sensitivity to YES and increasing the sensitivity to both. Results from both trials are included in Figure 5. We decided to forgo these enhancements for two reasons. Firstly, the specificity to individual gestures seemed to countervail our focus on extendability. Secondly, the gain in NO-Recall and NULL-Precision ran against our earlier stipulations that we cared more about precision for substantive gestures and recall for non-substantive gestures, producing the opposite effect.
Following from this focus on recall for non-substantive gestures and

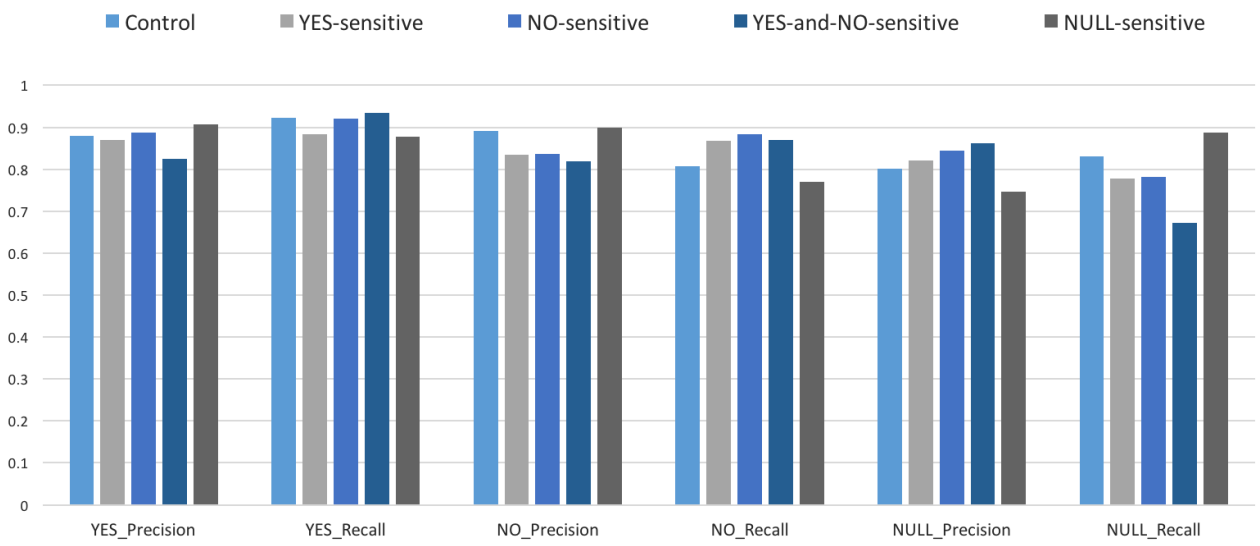


Figure 5: *Precision and Recall comparison across gestures for various enhancements*

precision for the rest, we attempted a second enhancement aimed at improving the NULL-Recall (NULL-sensitivity) to the benefit of substantive gesture precision. Under this paradigm, any apparent ambiguity would lead to a NULL classification. Specifically, if the number of NULL nearest neighbors was greater than the average nearest neighbor count of all other gestures then a NULL classification would be issued. Practically, for K=5, this results in distributions such as 3-YES, 0-NO, 2-NULL resulting in a NULL classification ($2 > 1.5$). Generally, this results in only classifying gestures as substantive if they do not encroach on the space of any ambiguity. Results from this enhancements provide the final set of columns in Figure 5. Finally, the F-Scores from our enhancements is provided in Figure 6. As can be observed, the NULL-sensitivity led to an F-Score similar to that of our control, and seemingly slightly better than it (within the range of ambiguity). However, we ultimately chose to remove this enhancement for two reasons. Firstly, the slight gain in F-Score did not seem large enough to merit such an untested change. Secondly, extending beyond two substantive classes would increasingly weight NULL-classifications to the point where, in cases where $K <$ the number of substantive gesture classes, even a single NULL neighbor would trump a majority of $K - 1$ of a substantive class. Thus we ultimately reverted to our control for its ability to adapt to greater numbers of gesture classes.

4.4 Test Accuracy

Using $k = 5$, acceleration (X, Y, Z) and rotation quaternions (X, Y), a training set of size 198, and the original voting rule for nearest neighbors we achieve a test accuracy of 0.8504.

5 Discussion

Classifying a head gesture as YES or NO seems like an inherently easy problem. It mostly depends on whether the head motion is happening in a vertical or horizontal manner. While it would have been possible to hand-craft features to capture this understanding of the problem, using our approach of KNN combined with DTW is more flexible and allows for the addition of more gestures in the future. Since this approach does not encode any knowledge about the head movements needed to perform a certain head gesture it is easily extendable to other user-defined gestures. Overall, we managed to create a classification package capable of rapidly categorizing different HMD head gestures.

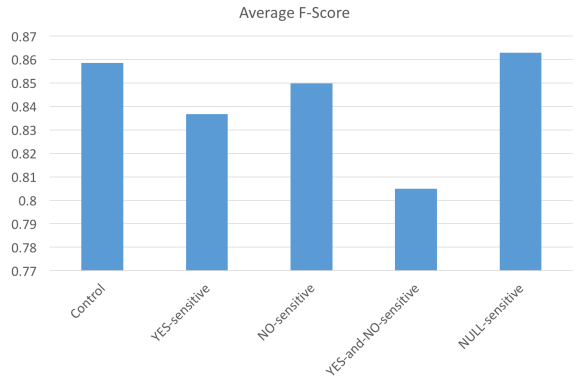


Figure 6: *The average F-Score for various enhancements, using $\beta_{NO} = \beta_{YES} = 0.7$ and $\beta_{NULL} = 1.5$ (Precision weighted for substantive gestures, recall weighted for non-substantive gestures)*

6 Demo

To show how our head gesture recognition algorithm could be used in practice we built an interactive demo using the Unity game engine (see Figure 1). The demo places the user in a dimly-lit tavern and gives him 30 seconds to memorize the scene. Afterwards the user will be asked questions about the scene (and life in general) and has to answer by shaking his head no or nodding yes. Testing the demo with passers-by at the poster session showed that this kind of user interface is intuitive and leads to a smooth experience. Only when the head shaking or nodding was performed too gently did

our algorithm have problems recognizing the head gesture. A likely explanation for this is that we had too few gentle gestures recorded during our data collection. People tend to shake their heads energetically when prompted to say YES and NO repeatedly, as opposed to when they organically produce such gestures.

7 Future Work

Going forward the main focus should be on speeding up the time it takes to classify one data point. One potential avenue is to find a faster implementation of DTW. We made use of Sakoe-Chiba bounds within our algorithm but there are other ways of improving DTW that merit further investigation. Aside from improving DTW we believe one might be able improve performance by selectively pruning seemingly distant neighbors from needing to be compared with new samples. Though the triangle inequality does not hold for DTW, a heuristic based on not calculating the DTW with neighbors that have high similarity (low DTW-distance) to neighbors which are dissimilar to the current sample might yield better speed while only sacrificing marginal accuracy. With a faster classification step one could choose a bigger training set, which leads to increased accuracy of the algorithm or allows for recognizing more than two head gestures without having to accept lower accuracy.

With more time to collect a bigger dataset one could also try using Recurrent Neural Networks (RNN), which have proven to be effective at classifying time series data [Connor et al. 1994], as an alternative machine learning approach.

References

BRYLA, K. Riftgesture. https://github.com/kbryla/rift_unity_scripts/blob/master/RiftGestures/RiftGesture.cs. Accessed: 2016-05-05.

CONNOR, J. T., MARTIN, R. D., AND ATLAS, L. E. 1994. Recurrent neural networks and robust time series prediction. *IEEE TRANSACTIONS ON NEURAL NETWORKS 5*, 240–254.

FRANK ALTHOFF, R. L., AND WALCHSHAUSL, L. Robust multimodal hand and head gesture recognition for controlling automotive infotainment systems. http://far.in.tum.de/pub/althoff2005vdi/althoff2005vdi.pdf. Accessed: 2016-05-06.

GEORGI, M., AMMA, C., AND SCHULTZ, T. 2015. Recognizing hand and finger gestures with imu based motion and emg based muscle activity sensing. In *International Conference on Bio-inspired Systems and Signal Processing*. BIOSIGNALS 2015.

HUI DING, GOCE TRAJCEVSKI, PETER SCHEUERMANN, XIAOYUE WANG, EAMONN KEOGH. 2008. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment 1*, 2, 1542–1552. http://dl.acm.org/citation.cfm?doid=1454159.1454226.

S.J. HWANG, J.P. NA, S. P. J. B. Ada-boost based gesture recognition using time interval window. http://www.atlantis-press.com/php/download_paper.php?id=22306. Accessed: 2016-05-06.

WANG, X. E. A. 2010. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 1–35.