Gesture and Action Discovery for Evaluating Virtual Environments with Semi-Supervised Segmentation of Telemetry Records

Andrea Batch*[1], Kyungjun Lee†[2], Hanuma Teja Maddali‡[2], and Niklas Elmqvist, *Senior Member, IEEE*§[1]

[1]College of Information Studies, University of Maryland, College Park
[2]Department of Computer Science, University of Maryland, College Park

Abstract

In this paper, we propose a novel pipeline for semi-supervised behavioral coding of videos of users testing a device or interface, with an eye toward human-computer interaction evaluation for virtual reality. Our system applies existing statistical techniques for time-series classification, including e-divisive change point detection and "Symbolic Aggregate approXimation" (SAX) with agglomerative hierarchical clustering, to 3D pose telemetry data. These techniques create classes of short segments of single-person video data–short actions of potential interest called "micro-gestures." A long short-term memory (LSTM) layer then learns these micro-gestures from pose features generated purely from video via a pre-trained OpenPose convolutional neural network (CNN) to predict their occurrence in unlabeled test videos. We present and discuss the results from testing our system on the single user pose videos of the CMU Panoptic Dataset.

1. Introduction

With the exception of frameworks such as GOMS [3], there are few general theoretical models for discovering user-driven actions in human-computer interaction (HCI) that are applicable to virtual reality (VR) systems while also being sufficiently sophisticated to enable formal verification. Formal models for evaluating interaction have typically focused on specific classes of actions, such as steering [31], pointing [19], object pursuit [17], arm fatigue [11, 18], or some combination of these [5]. Often, these models are intended to highlight features of interaction that could have adverse effects on user experience [11, 18, 27], or are intended to predict predefined actions even when sufficiently sophisticated for highly accurate gesture prediction and validation (e.g., KeyTime [15]).

As a result of this specialization in models, validating a new technique, innovation, or system in HCI often leaves empirical evaluation as the only available recourse. In this context, models for discovering novel user-driven actions—i.e., interactions not anticipated by the researcher—are virtually nonexistent. It is not uncommon that such empirical evaluation reduces to systematically observing and coding video recordings of users engaging with the interactive system. This is particularly true for VR systems, where the interaction to be evaluated may involve the user's whole body. However, such coding is generally costly, time-consuming, and prone to inconsistencies [14]. Furthermore, it often requires multiple coders agreeing on and calibrating a common code book. Finally, the very nature of this process injects subjective biases that make the experiment results difficult to reproduce [23].

To address this issue, we propose a semi-automated computer vision system for behavioral coding of videos that will make the process of discovering full-body user gestures more robust and scalable. Existing systems and action recognition studies tend to focus on actions that are familiar and meaningful in a larger range of contexts such as walking, running, eating, opening the fridge. These studies are able to take advantage of large amounts of publicly-available video data, but these datasets are not typically applicable to usability testing of a novel VR system. In VR, users perform specialized actions to interact with objects in the virtual environment using a custom interface. The actions may take the form of moving parts of the body in a non-generalizable way such as swinging arms diagonally, or tilting their head.

*ajulca@umd.edu
†kjlee@cs.umd.edu
‡hmaddali@umd.edu
§elm@umd.edu

These actions do not necessarily map perfectly to real-world scenarios that may be assigned semantic labels in existing video datasets.

In our scenario, researchers might be more interested in outlier actions and want to manually filter out certain actions indicative of bugs in the system in unlabeled videos. In this paper, we propose a semi-automated pipeline that segments videos into potential actions of interest (AOI) and indicates these in unlabeled videos to make full-body gesture discovery faster, more scalable, and easier to reproduce. We draw from the disciplines of computational statistics and signal processing, Our approach makes use of data that is less computationally costly to evaluate than video data—such as depth, audio, tracked marker positions, and so on—for efficient identification of AOIs, and synchronizes it with video data for validation at the end of the pipeline. We believe that our approach is well-suited to most contemporary VR devices, which rely on sensors and cameras to detect the positions of the user's controllers and head-mounted display (HMD). In a user study involving the collection of video data, this ground truth can be used for video segmentation.

Our pipeline can be used by HCI researchers who can collect video and telemetry data capturing users during sessions in which the user explores the virtual environment. After all user data has been collected, telemetry data is segmented, and then segments are clustered into micro-gesture classes, using a set of statistical methods described in Section 3. Video data is temporally labeled with gesture codes based on telemetry segmentation, and predicting these gestures becomes the training/testing target of our neural network architecture. While our results leave some room for improvement, we believe that—given the lack of a semantic ground truth—our model performs with reasonably high accuracy relative to the current state of the art in action discovery.

2. Related Work

Our system draws from prior work in behavioral coding, and action classification from video. Prior research in behavioral coding illustrates what would need to be done to approach the envisioned pipeline as in Figure 1 and work on unsupervised action detection and classification suggests a method for achieving it.

2.1. Deep Learning Models for Human Poses

Considerable prior research has explored the topic of human pose estimation using deep learning in single-person single camera [32], multi-person single camera, and multi-person multi-camera settings [2, 4]. Recent work includes top-down approaches using a 2-stage pipeline with a CNN for frame-level pose prediction followed by a matching algorithm to efficiently link the predictions to specific people [2, 4, 25, 29]. The CNN itself can use a 3D mask as in Girdhar et al. [4] to incorporate temporal data for more robust prediction. In our project, we use the pretrained OpenPose model [2] to jointly detect human body, hand, and facial keypoints (in total 135 keypoints) on single frames.

Walker et al. tried to address the video forecasting problem by taking advantage of the strengths of Variational Autoencoders (VAEs) and GANS [28]. Instead of solving this forecasting problem directly in the pixel-level space, this paper projects the problem into the human-pose space through the human-pose estimation. Our project is similar to their approach in that we also try to address the action classification problem in the human-pose space, instead of classifying actions directly from videos.

2.2. Behavioral Coding

Coding behavioral events in video is common research practise in HCI and other fields, often those related to the social sciences [24]. It is largely performed in three steps. First, a coding scheme that describes the categories of actions has to be created via a bottom-up, top-down [30], or a hybrid approach. In a bottom-up approach the themes for the actions emerge from the data itself and are agreed upon by the coders after watching and rewatching of the videos. In a top-down approach, labels emerge from the theoretical literature on human gestures. The second step would be to train some number of coders which takes an amount of time proportional to the complexity of the videos. The final step is to actually label the videos and ensure that the coders are able to label videos in a consistent way which is measured by an agreement metric such as Cohen's Kappa [9]. The codebook might be rewritten in iterations during this process.

Several existing tools have been built to support the video coding process, particularly to help with coder training and video labeling in a systematic way. For example, ANVIL, Datavyu [26], VACA [1], and VCode [7]. There have also been systems in the past that have leveraged crowdworkers instead in the codebook creation and video labeling process [14].

In our system, we implement a hybrid approach in which an unsupervised clustering mechanism groups actions in the data by a measure of similarity related to change in pose. A human in the loop can then use knowledge of theory to select potential AOI, either though expected actions or outlier detection. The action detection and label assignment process in our pipeline, however, is completely automated via an action classification model.

2.3. Unsupervised Video Summarization

Summarization models are probably closer to our objective than any other, but our target is the narrow context of HCI researchers discovering new actions based on user interactions in systems using 3-dimensional body motion and gestures, and reducing the computational cost of model training is a high priority. Mahasseni et al. [20] take what might be considered the most contemporary approach to detecting events in video for summarization by using generative adversarial networks (GANs) to detect keyframes—frames marking the end or beginning of transitions in motion—in high-resolution video. In their model, the generative network (summarizer) creates a summary of a longer video in order to trick the discriminator, and the discriminator network is trained to discriminate between the summarizer and the human-summarized video. They use the SumMe dataset, which has short, human-made summaries for a corresponding set of longer videos (1 to 6 minutes in length) [6].

The use of keyframes itself is not a new idea. In fact, as an alternative approach to detecting keyframes, the study that originated the SumMe benchmark dataset used by Mahasseni et al. [20], Gygli et al. [6] draw from video editing theory in proposing *Superframe segmentation*, a technique that cuts video into arbitrary segments and then shifts the the cuts to neighboring frames with the least motion, as part of a video summarization pipeline. Following segmentation, they evaluate numerous other features of the video—including attention, color, contrast, edge distribution, and object detection (people and landmarks)—and then calculate an "interestingness" score. The interestingness of a segment must meet a predefined threshold in order to be cut into the output of the model, which concatenates the most interesting segments of the video into a short summary.

3. Methodology

Figure 1 shows the overall pipeline for our system. The videos in our dataset have synchronized 3D pose data available that will be used in the training phase. The output of this part of the pipeline is a list of pseudo-ground-truth labels for a selection of "micro-gestures" detected in this 3D data which act as our AOI.

1. **Clustering Phase:** The synchronized 3D pose data is converted to features indicating temporal variance using e-divisive change-point detection followed by SAX edit distance matrix transformation. A hierarchical clustering method is then used to group these features into clusters that indicate similar video segments. A researcher can then be presented with a display of the identified video segment groups to check for qualitative similarity and a set of potential AOI.

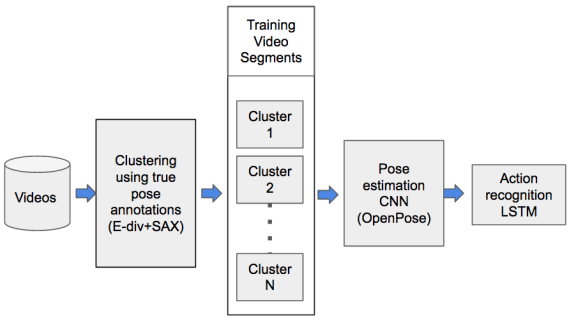

Figure 1: Our overall pipeline.

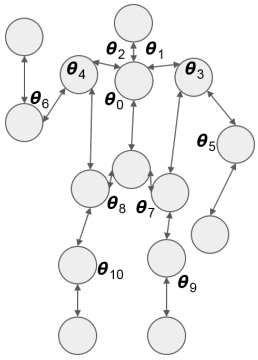

Figure 2: Subject joint angles.

2. **Training Phase:** The AOI video segments act as training data for the following LSTM network. The input frames are converted into pose feature vectors using a pre-trained CNN and the output of the LSTM is an action label for this input frame.

3. **Testing Phase:** In the testing phase we only use the unlabeled video to predict an action label for every frame. The ground truth for these is the output from the hierarchical clustering.

3.1. Statistical methods

We exploit the spatio-temporal continuity of the human body by using sensor data tracking human joint positions to temporally segment full-length video into short (less than 15 seconds) micro-gestures. Before beginning the process, we select eleven angles (θ) between 15 joints from the CMU Panoptic dataset (Figure 2).

3.1.1 Joint Angle Segmentation

Matteson and James [21] originate the e-divisive method for detecting changes in the mean of multivariate time series: Estimated temporal divergence measure for any two joints $\theta^X, \theta^Y \varepsilon\ \mathbb{R}^d$ is given by Equation 1,

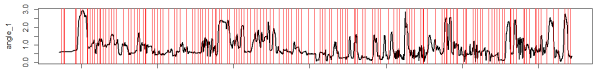

Figure 3: Example of joint angle time series segmented based on change point estimates ($\hat{\tau}_k$).

$$\widehat{\mathscr{E}}(\theta_n^X, \theta_m^Y; \alpha) = \frac{2}{mn}\sum_{i=1}^{n}\sum_{j=1}^{m}|\theta_n^X - \theta_m^Y| - \gamma(\theta_n^X; \alpha) - \gamma(\theta_m^Y; \alpha) \tag{1}$$

where α is some value between 0 and 2; following James and Matteson [10], we use $\alpha = 1$. γ is given by Equation 2.

$$\gamma(\theta_n; \alpha) = \binom{n}{2}^{-1} \sum_{1 \le i < k \le n} |\theta_i - \theta_k|^\alpha \tag{2}$$

This gives us a measure of scaled empirical divergence between joint angles, $\widehat{\mathscr{Q}}(\theta_n^X, \theta_m^Y; \alpha)$, described in Equation 3.

$$\widehat{\mathscr{Q}}(\theta_n^X, \theta_m^Y; \alpha) = \frac{mn}{m+n}\widehat{\mathscr{E}}(\theta_n^X, \theta_m^Y; \alpha) \tag{3}$$

Then the locations of change points (τ) can be estimated as shown in Equation 4.

$$(\hat{\tau}, \hat{\kappa}) = \underset{\hat{\tau}, \hat{\kappa}}{\operatorname{argmax}}\ \widehat{\mathscr{Q}}(\theta_n^X, \theta_m^Y; \alpha) \tag{4}$$

In Figure 3, these change points are shown as red vertical lines laid over the time series representation of θ_1 from Figure 2. This can be generalized to any number of variables.

3.1.2 Symbolic Representation

Once the time series representation of the human joint angles have been segmented based on change points, we transform the segments into strings to derive an edit distance matrix of the symbolic representations of each segment, reduced to a constant number of observations (or "characters"). Lin et al. [16] introduce this method of string representation for time series clustering, known as SAX, as being performed in the following steps:

1. Normalize the segment around a mean of zero.

2. Piecewise Aggregate Approximation (PAA) [13]: Convert a series x of length n to series $\bar{x}$ of N dimensions as shown in Equation 5.

$$\bar{x}_i = \frac{N}{n}\sum_{j=\frac{N}{n}(i-1)+1}^{\frac{N}{n}i} x_j \tag{5}$$

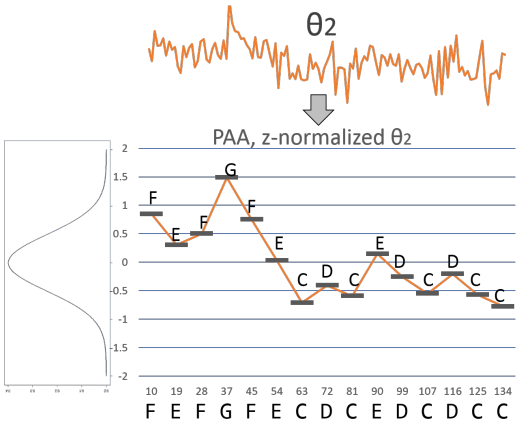

Figure 4: SAX: PAA-converted, z-normalized, character representation of θ_2 for a single segment determined by change points estimated by e-divisive.

This effectively stretches or squashes all of the micro-gesture joint angle segments to the same length.

3. Symbolic representation: Given a series segment converted to a specified length N with normal distribution around zero, assign letters to values along the segment (Figure 4).

4. Derive the edit distance matrix. We use Levenshtein edit distances: The lowest number of transformations—character insertion, substitution, or deletion—required to turn one word into another.

We modify this approach somewhat by padding our segments with their surrounding neighbors: Each micro-gesture $\mathscr{G}_t$ is grouped with the segments $\mathscr{G}_{t-1}$ and $\mathscr{G}_{t+1}$ prior to applying the first step of the above process. At the end of the clustering process described in Section 3.1.3, the micro-gesture label is only assigned to segment $\mathscr{G}_t$.

3.1.3 Semi-Supervised Clustering

We create clusters from the edit distance matrix created from the process described in Section 3.1.2 by applying a fast iterative agglomerative clustering algorithm implemented by Müllner [22]: Individual nodes are grouped together with their nearest neighbor, values are stored for the group, and the next nearest neighbors are grouped together, iterating until all nodes are in one group. For this paper, the number of clusters is determined by the frame rate and the length of the video: $k = \mathscr{T}/\text{fps}^2$, where $\mathscr{T}$ is the total number of frames in the tracking data; in this case, $\bar{k} = 166$.

Figure 5 represents the clusters used in this paper, given arbitrary labels $\mathscr{G}_{1,2,\ldots,166}$ (not shown). The human-

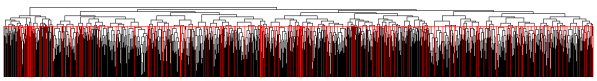

Figure 5: fastcluster method, with hierarchical tree cut at depth of $k = 166$.

in-the-loop comes into play at this stage in two ways: First, the researcher must determine k; second, the researcher must select a subset of videos to train. In application, the HCI researcher could, at this point, view samples of all micro-gestures by cluster, select the groups that constitute their AOI to be trained on, and walk away from the model.

For this paper, we simply select the ten most frequently observed micro-gestures across all videos. Prior to initiating model training, it is also worth noting that extra frames are at the end of a segment

Once the researcher chooses a subset of gesture classes, a randomly selected sample of 70% of all micro-gesture segments with a label in the set of AOI is marked for training, the remaining 30% is marked for testing, and a proportionally equivalent random selection of micro-gestures without labels in the chosen set are selected for training and testing as well. We take this approach because our experiment involves comparing models trained with a no-gesture class in the target vector (a detector) against models trained which simply does not count no-gesture frames in its target vector. This approach is described in greater detail in Section 4.

3.2. Deep Learning Network

Given the labels generated by the statistical methods discussed in Section 3.1, we train a deep learning model to recognize an action from videos. A naive approach would be to recognize an action by giving a raw video directly to the model. This approach, however, would make this action classification problem intractable, as it induces huge computations costs to solve the problem in a pixel-level space. Instead, we first project this classification problem into a human-pose-level space by getting human pose data through a CNN and then use a LSTM network architecture [8] to recognize an action given a sequence of human poses, which is similar to prior work [28]. As noted by Walker et al. [28], projecting this problem into a human-pose space would reduce the problem complexity as it reduces the computation costs in the deep learning model. Our deep learning architecture is shown in Figure 6.

3.2.1 OpenPose CNN

To obtain human pose data from an image, we utilize the OpenPose CNN model [2]. In this model, a human

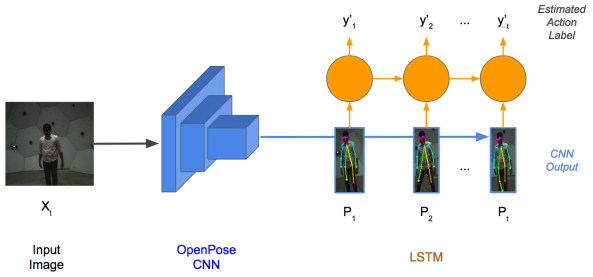

Figure 6: Our deep learning architecture: Input images are given to the pre-trained OpenPose CNN, the images and joint positions are given to a LSTM unit trained to predict the labels assigned (as described in Section 3.1), and the LSTM remembers and forgets input and output information from previous frames.

pose is estimated by first finding human joints and part affinity fields (PAFs) from an image and then combining the joints using the PAFs information. The PAFs data consists of vectors that contain the connection information between the joints. With PAFs, the CNN model can correctly estimate an appropriate edge only between relevant joints and thus generate a human pose skeleton data. This model is publicly available on GitHub [1]. As shown in Figure 6, we first pass an image frame into the OpenPose CNN model to collect its human pose data. This human pose data is then used in the following recurrent neural network, LSTM, to perform action classification.

3.2.2 LSTM

Based on human pose data generated by the OpenPose CNN model, our recurrent neural network LSTM [8] is used to estimate its action label. As the LSTM architecture is used to remember important values over arbitrary time intervals and forget other values. Since we are focusing on an action classification problem, we place the softmax layer on top of the output of the LSTM network to obtain a label inference. We force the model to learn only on the features that we believe to be useful for action classification during training–namely, a cell's input consists of output of the OpenPose CNN at a given time t, and the inferred labels from the remembered preceding periods (Figure 6). During training, the cross entropy loss function (Equation 6) is used to enforce the model to encode in the hidden states any features relevant to action estimation.

$$L = -\sum_i y_i \log(y_i') \tag{6}$$

[1] https://github.com/CMU-Perceptual-Computing-Lab/openpose

4. Experiments

For our experiments we used the CMU panoptic dataset [12] consisting of videos collected from simulated social settings in a massively multiview environment. We trained our system using VGA videos split into 5,603 segments of varying temporal length, along with the segments' associated synchronized 3D pose data for 15 joints, using a single GPU (EVGA GeForce GTX 1080 SC). The target output is a vector representing the labels generated by the combined $e-divisive \rightarrow SAX \rightarrow fastcluster$ technique described in Section 3.1.

For the deep learning model, we use the OpenPose CNN model [2] and a single-layer LSTM network [8] consisting of 256 hidden units. We compare results from three different approaches:

- A **constant detector network** that attempts to discern between no-gesture frames and frames with *any* AOI and then classify them using a constant learning rate of 0.0001.
- An **adaptive detector network** that, like the previous network, targets a ground truth that includes a no-gesture class using a learning rate starting at 0.001 that linearly decreases each epoch by 0.000018 until it reaches 0.00001.
- An **adaptive AOI classification network** without a no-gesture class that uses the same linearly-decreasing learning rate as the adaptive detector network.

We use the Adam optimizer to train the LSTM network, taking the two different approaches to learning rates noted above. As mentioned in the previous section, the cross entropy loss is used to calculate loss between the ground-truth labels and the estimated labels during the LSTM training. As the OpenPose CNN well estimates human poses from videos in the target dataset, we do not train this CNN model — the performance of the OpenPose CNN model is evaluated in the next section. The OpenPose CNN is simply utilized to generate human pose data during the training and testing phase of the LSTM network

4.1. Qualitative Results

To evaluate the performance of the OpenPose CNN model, we first tested the pose model with some of the action videos. As shown in Figure 7, the model successfully estimates human poses from the videos. Based on this observation, we decided not to train the CNN model, but simply use it to generate human pose data for the following recurrent neural network to simplify the problem;

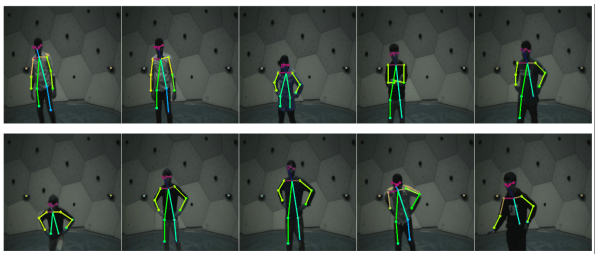

Figure 7: Skeleton overlaid onto frames by the pretrained OpenPose network within a group clustered by statistical methods described in Section 3.1.

Figure 8: Constant learning rate detector model output of predictions for gesture label shown in Figure 7. Gestures have been qualitatively subgrouped and shown in bounding colors by the authors to highlight apparent similarity, with one micro-gesture in the lower left corner being visually similar to both the green and the purple subgroups. Conversely, the center micro-gesture shows little similarity to any other micro-gesture in the class.

in other words, the action classification problem is projected into the human pose space, which size is smaller than that of the pixel space.

Our qualitative results intuitively demonstrate both a shortcoming and a nice feature of our implementation: Because of our agglomerative hierarchical edit distance clustering, gestures are grouped together into what may accurately be described as "like classes," but there is a trade-off between the amount of data available within a class and the similarity any one micro-gesture has across all of the other micro-gestures it is classed with. As such, we see what could arguably be defined as multiple classes that overlap somewhat in both our modeled input and in our test output (Figure 8).

4.2. Quantitative Results

Our results indicate that the trained model performs best with a constant learning rate of .0001, and is slightly better at detection than classification.

Table 1 shows the highest accuracy scores of all approaches used, that of the detector using a constant learning rate of 0.0001. The most surprising result of our

Table 1: Constant detector with a "no-gesture class" included. For all tables, best-performing accuracy scores are shown in bold.

Epochs	Training Accuracy	Training Loss	Test Accuracy
1	0.385990	1.969203	**0.366118**
5	0.391999	1.943364	**0.366118**
10	0.385557	1.896348	0.355147
15	0.391457	1.831509	0.339201
20	0.418742	1.796008	0.344304
25	0.427891	1.762800	0.339074

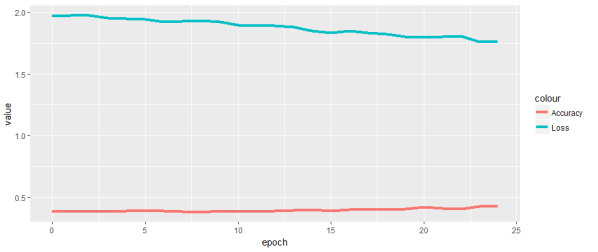

Figure 9: Training loss and accuracy for our detector with constant learning rate .0001.

Table 2: Adaptive detector with a "no-gesture class" included.

Epochs	Training Accuracy	Training Loss	Test Accuracy
5	0.505847	1.563875	0.067457
10	0.513859	1.559077	0.067571
15	0.514508	1.560636	0.087170
20	0.515808	1.558239	0.073952
25	0.517757	1.557487	**0.326231**
30	0.517757	1.557487	0.063241

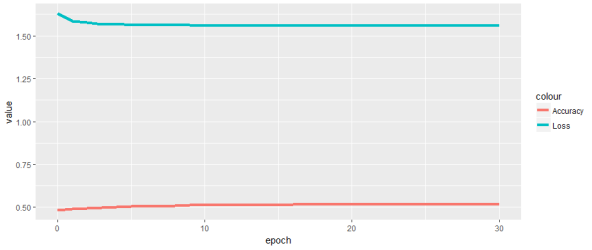

Figure 10: Training loss and accuracy for our detector with adaptive (linearly diminishing) learning rate.

Table 3: Adaptive classifier model without a "no-gesture" class.

Epochs	Training Accuracy	Training Loss	Test Accuracy
1	0.575123	0.983921	0.086918
50	0.639233	0.979427	**0.147600**
105	0.639233	0.979359	0.119579
160	0.641106	0.979615	0.106015

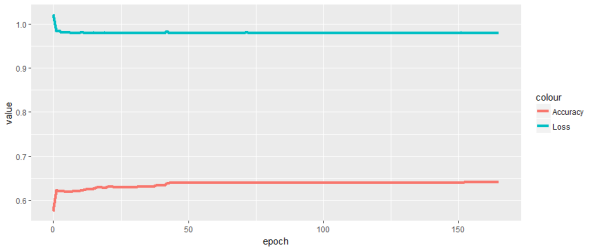

Figure 11: Training loss and accuracy for our classifier with learning rate starting at 0.001, decreasing linearly to .00001 over 500 epochs (note: only 160 epochs were trained during this study).

study is also shown in Table 1: While training accuracy continues to rise and training loss falls at each epoch (Figure 9), the test results from our first epoch are greater than or equal to those of any further training epochs.

The shrinking learning rate detector performs poorly relative to the previous approach, and quickly overfits by epoch 30, as demonstrated in Table 2 and Figure 10. A surprising result is also highlighted in Table 2: Surrounded by poorly-performing epochs at all other tested stages of training, epoch 25 shows an accuracy score comparable to those seen in Table 1. Comparable results were seen with multiple tests of the trained model stored from epoch 25 of the shrinking-rate detection model. The most plausible explanation for this jump in test accuracy is that it may simply be noise, reflecting a need to reorder the test data; evaluation of this unexpected result may be an area for future work.

For classification using a shrinking learning rate, the optimal number of epochs appears to fall in the range $[2, 105)$–probably closer to a value in the middle of the range, if the trend in test accuracy is smooth, or in the range $[2, 30)$, if our adaptive detector's results can be generalized. The classification model performed more poorly than the detector, but due to resource and time constraints, a constant classification model was not included in this study.

5. Discussion

Our results do not appear extremely successful relative to existing video action classification work using pre-existing labels. Beyond the reasonable allowances that may be made for an architecture that creates novel gesture labels as a pseudo-ground-truth, we believe that the low accuracy score is largely due to three reasons. First, hyper-parameter configurations have not been fully explored and optimized. In our deep learning model, there are various hyper-parameters–epochs, learning rates, and the hidden state size. Although finding effective hyper-

parameters for a deep learning network is notoriously difficult, it is essential to try various combinations of hyper-parameters. We found that our adaptive learning rate, which we had hoped would improve our test accuracy, had the opposite effect, and overfit the model early in the training process.

Second, the portion of the background class ("NA") in the training and testing dataset should be carefully evaluated. In early stages of our experiments, we used actions labelled as "NA," which means that the actions do not belong to any of our predefined clusters, and classify them as one of the action classes—a catch-all "no-gesture class" in our target ground truth. Our expectation was that the NA label would make training difficult for our deep learning mode due to the fact that the actions in the NA class may not have consistent features in the human pose data. To bypass the effects of having the NA class in the dataset, we spent most of our training time on a model that does not consider these actions as a class; instead, they are simply represented as a non-clustered action (no class), and the target ground truth for the associated frame is a zero vector. Our results show, however, that the detector network has significantly higher test accuracy–even the adaptive detector has at least one epoch outperforming any of the observed test accuracy scores of the classifier network. However, we trained our classifier network with an adaptive learning rate, which appears to perform poorly relative to the constant rate.

Finally, but most importantly, the statistical methods may not generate what might be considered "valid" action clusters. After segmenting the videos into action clusters generated by the statistical methods, we manually reviewed all of the actions within a certain cluster (a certain class). Throughout this review process, we observed that segments within the single cluster do not always all appear to have the same gestures; rather, they may contain dissimilar or semi-similar gesture sub-classes that are "bridged" by one or two video segments featuring gestures that are somewhat similar to two or more other sub-classes in the cluster (Figure 8). Thus, our training and testing datasets may not be valid. This implies that our deep learning model may not learn features from action videos appropriately, so our statistical methods warrant additional examination in future work.

6. Conclusion

We have presented our pipeline for discovering novel human actions from video and telemetry data using both statistical methods for time series analysis and deep learning networks for video analysis. Using the e-divisive and SAX methods, we grouped AOI into several classes. Given these pseudo action classes, we used a sample of frames from the Panoptic dataset to train a simple LSTM network. By training the recurrent neural network, we were expecting the deep learning model to detect and classify human actions from videos in which a human is an active agent.

Relative to architecture featuring existing semantic labels, our model did not perform overwhelmingly well, with the best observed test accuracy (0.366118) resulting from a model used simply to detect whether there is *any* AOI in the view that was trained for only one to five epochs. It is possible that the accuracy of our approach may be improved in future implementations by modifying hyper-parameters in the LSTM network: More epochs, larger or smaller learning rates, and so on. Furthermore, creating additional layers in the LSTM network has the potential for improving future iterations of this work; for example, a stacked LSTM network has multiple hidden LSTM layers, which allows for greater model complexity. Finally, future work should thoroughly review the statistical methods used and their outputs (action clusters) to ensure that they generate valid datasets.

In spite of somewhat limited results, we believe that this approach is relevant for HCI researchers as a means of reducing the time cost of coding user interactions. In particular, we believe this to be true for HCI research involving implementations within a VR environment. Most contemporary consumer-ready VR devices feature sensors for detecting object position and orientation. The VR experience is three-dimensional, and involves full-body gestures. Our pipeline is applicable for inferring three-dimensional points on the human body in relatively small video datasets with telemetry data, but without existing semantic labels. Video data of this nature could be collected during a user study in a typical academic research lab environment. Our methods are feasible on hardware that is a minimum system requirement for most contemporary VR HMDs: A single, yet reasonably powerful GPU. As such, in the context of HCI research, future implementations of the method we present should focus more narrowly on the specific problem space of evaluating novel VR applications.

References

[1] B. Burr. VACA: A tool for qualitative video analysis. In *Extended Abstracts of the ACM Conference on Human Factors in Computing Systems*, pp. 622–627. ACM, 2006. doi: 10.1145/1125451.1125580

[2] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. Realtime multi-person 2D pose estimation using part affinity fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1302–1310, 2017. doi: 10.1109/CVPR.2017.143

[3] S. K. Card, T. P. Moran, and A. Newell. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates Inc., 1983.

[4] R. Girdhar, G. Gkioxari, L. Torresani, M. Paluri, and D. Tran. Detect-and-Track: Efficient pose estimation in videos. *CoRR*, abs/1712.09184, 2017.

[5] D. Guinness, A. Jude, G. M. Poor, and A. Dover. Models for rested touchless gestural interaction. In *Proceedings of the ACM Symposium on Spatial User Interaction*, SUI '15, pp. 34–43. ACM, New York, NY, USA, 2015. doi: 10.1145/2788940.2788948

[6] M. Gygli, H. Grabner, H. Riemenschneider, and L. Van Gool. Creating summaries from user videos. In *Proceedings of the IEEE European Conference on Computer Vision*, pp. 505–520, Sep. 2014.

[7] J. Hagedorn, J. M. Hailpern, and K. Karahalios. VCode and VData: Illustrating a new framework for supporting the video annotation workflow. In *Proceedings of the ACM Conference on Advanced Visual Interfaces*, pp. 317–321, 2008.

[8] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[9] H. Holle and R. Rein. The modified cohen's kappa: Calculating interrater agreement for segmentation and annotation. *Understanding Body Movement: A Guide to Empirical Research on Nonverbal Behaviour*, pp. 261–275, Jan. 2013.

[10] N. A. James and D. S. Matteson. ecp: An R package for nonparametric multiple change point analysis of multivariate data. *Journal of Statistical Software*, 62(i07), 2015.

[11] S. Jang, W. Stuerzlinger, S. Ambike, and K. Ramani. Modeling cumulative arm fatigue in mid-air interaction based on perceived exertion and kinetics of arm motion. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, CHI '17, pp. 3328–3339. ACM, New York, NY, USA, 2017. doi: 10.1145/3025453.3025523

[12] H. Joo, T. Simon, X. Li, H. Liu, L. Tan, L. Gui, S. Banerjee, T. Godisart, B. C. Nabbe, I. A. Matthews, T. Kanade, S. Nobuhara, and Y. Sheikh. Panoptic Studio: A massively multiview system for social interaction capture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. doi: 10.1109/TPAMI.2017.2782743

[13] E. J. Keogh and M. J. Pazzani. Scaling up dynamic time warping for datamining applications. In *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining*, pp. 285–289, 2000.

[14] W. S. Lasecki, M. Gordon, D. Koutra, M. F. Jung, S. P. Dow, and J. P. Bigham. Glance: Rapidly coding behavioral video with the crowd. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pp. 551–562. ACM, New York, NY, USA, 2014. doi: 10.1145/2642918.2647367

[15] L. A. Leiva, D. Martín-Albo, R. Plamondon, and R.-D. Vatavu. KeyTime: Super-accurate prediction of stroke gesture production times. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pp. 239:1–239:12. ACM, New York, NY, USA, 2018. doi: 10.1145/3173574.3173813

[16] J. Lin, E. Keogh, L. Wei, and S. Lonardi. Experiencing SAX: A novel symbolic representation of time series. *Data Mining and Knowledge Discovery*, 15(2):107–144, 2007.

[17] L. Liu and R. Van Liere. Modeling object pursuit for desktop virtual reality. *IEEE Transactions on Visualization and Computer Graphics*, 18(7):1017–1026, July 2012. doi: 10.1109/TVCG.2012.31

[18] Z. Liu, D. Vogel, and J. R. Wallace. Applying the cumulative fatigue model to interaction on large, multi-touch displays. In *Proceedings of the ACM International Symposium on Pervasive Displays*, PerDis '18, pp. 1:1–1:9. ACM, New York, NY, USA, 2018. doi: 10.1145/3205873.3205890

[19] I. S. MacKenzie, A. Sellen, and W. A. S. Buxton. A comparison of input devices in element pointing and dragging tasks. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pp. 161–166. ACM, New York, NY, USA, 1991. doi: 10.1145/108844.108868

[20] B. Mahasseni, M. Lam, and S. Todorovic. Unsupervised video summarization with adversarial LSTM networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 202–211, 2017.

[21] D. S. Matteson and N. A. James. A nonparametric approach for multiple change point analysis of multivariate data. *Journal of the American Statistical Association*, 109(505):334–345, 2014.

[22] D. Müllner. fastcluster: Fast hierarchical, agglomerative clustering routines for R and Python. *Journal of Statistical Software*, 53(i09), 2013.

[23] L. K. Nelson. Computational grounded theory: A methodological framework. *Sociological Methods & Research*, 0(0):0049124117729703, 0. doi: 10.1177/0049124117729703

[24] H. T. Reis and C. M. Judd. *Handbook of Research Methods in Social and Personality Psychology*. Cambridge University Press, 2nd ed., 2014. doi: 10.1017/CBO9780511996481

[25] J. Song, L. Wang, L. Van Gool, and O. Hilliges. Thin-slicing network: A deep structured model for pose estimation in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5563–5572, July 2017. doi: 10.1109/CVPR.2017.590

[26] D. Team. Datavyu: A video coding tool. *Databrary Project, New York University*, 2014.

[27] R.-D. Vatavu, D. Vogel, G. Casiez, and L. Grisoni. Estimating the perceived difficulty of pen gestures. In *Proceedings of the International Conference on Human-computer Interaction*, INTERACT'11, pp. 89–106. Springer-Verlag, Berlin, Heidelberg, 2011.

[28] J. Walker, K. Marino, A. Gupta, and M. Hebert. The pose knows: Video forecasting by generating pose futures. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3352–3361, 2017.

[29] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4724–4732, 2016.

[30] L. Weingart, M. Olekalns, and P. Smith. Quantitative coding of negotiation behavior. *International Negotiation*, 9(3):441–456, 2004. doi: 10.1163/1571806053498805

[31] S. Zhai, J. Accot, and R. Woltjer. Human action laws in electronic virtual worlds: An empirical study of path steering performance in VR. *Presence: Teleoperators and Virtual Environments*, 13(2):113–127, 2004. doi: 10.1162/1054746041382393

[32] X. Zhou, M. Zhu, S. Leonardos, K. G. Derpanis, and K. Daniilidis. Sparseness meets deepness: 3D human pose estimation from monocular video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4966–4975, June 2016. doi: 10.1109/CVPR.2016.537