Robust Motion Recognition using Gesture Phase Annotation

Hannah VanderHoeven[0000−0003−3234−6797], Nathaniel Blanchard[0000−0002−2653−0873], and Nikhil Krishnaswamy[0000−0001−7878−7227]

Colorado State University, Fort Collins CO 80523, USA
{hannah.vanderhoeven,nathaniel.blanchard,nikhil.krishnaswamy}@colostate.edu

Abstract. Robust gesture recognition is key to multimodal language understanding as well as human-computer interaction. While vision-based approaches to gesture recognition rightly focus on detecting hand poses in a single frame of video, there is less focus on recognizing the distinct "phases" of gesture as used in real interaction between humans or between humans and computers. Following the semantics of gesture originally outlined by Kendon, and elaborated by many such as McNeill and Lascarides and Stone, we propose a method to automatically detect the preparatory, "stroke," and recovery phases of semantic gestures. This method can be used to mitigate errors in automatic motion recognition, such as when the hand pose of a gesture is formed before semantic content is intended to be communicated and in semi-automatically creating or augmenting large gesture-speech alignment corpora.

Keywords: Gesture semantics · Gesture annotation · Gesture phases.

1 Introduction

A key requirement in modeling the behavior of humans in interaction with each other and with intelligent agents is the ability to model and recognize gestures. Not only is this crucial to creating multimodal corpora [16, 17], it is also a critical component of intelligent systems that communicate multimodally with humans [11–13]. The technical solution for most modern gesture recognition systems is some form of computer vision algorithm. As these are usually large neural networks, successful vision systems depend on a sufficient volume of high-quality training data.

In order to create effective training data for machine learning tasks, as well as for robust modeling of human gestures, accurate annotation is necessary to establish correct ground truth values. This often involves identifying the key frames where the semantics of the gesture are expressed. For instance, a difficulty in automatic recognition may arise if an algorithm focuses only on individual frames, such as when a distinct hand pose is formed (e.g., a pointing gesture) before the semantic denotatum of the gesture is intended to be communicated (cf. [10]). Therefore, correct identification of key frames helps reduce noise and aids accurate identification of gestures, but manually sifting through

video data to identify when exactly a gesture of interest starts and stops can be time-consuming and labor-intensive. In this paper, we propose an approach to automatically identify the key phases of gestures to aid in semi-automatic detection and annotation of gesture semantics. Following the gesture semantics originally formulated by Kendon [8] and elaborated by McNeill [21] and Lascarides and Stone [15], among others, we focus on automatic identification of key frames that comprise the *pre-stroke*, *stroke*, and *post-stroke* phases of a gesture unfolding across multiple frames. We present our methodology, qualitative and quantitative outputs, and evaluate the utility of the entire pipeline on a gesture recognition task, showing very promising results.

2 Related Work

2.1 History of Gesture Semantics

There is a wealth of foundational and theoretical work in gesture semantics that informs our research. In this section, a small selection of key contributions in this area follow.

Adam Kendon [8] pioneered an approach that models gestures as simple schemas consisting of distinct sub-gestural *phases*. Of these, the *stroke* phase is regarded as the content-bearing phase. David McNeill [20, 21] treated gesture and speech as sharing "the same psychological structure" and "a computational stage." This position set the stage for interpretations of gesture to be modeled using similar semantic structures as verbal language. McNeill's formulation [19] extends Kendon's to include "preparation" and "retraction" phases.

According to Lascarides and Stone [14, 15], gesture is interpretable in the direct visual context if and only if the denotation of its interpretation function is also directly co-perceptible by both the gesture-making agent and said agent's interlocutor. An example of this would be the object intended to be denoted by a deictic gesture; namely, the deixis is only fully interpretable when its pointing "cone" [9] aligns with the object, while in the case of an iconic gesture (say, holding thumb and forefinger together to indicate "smallness"), the directly-perceptible visual denotatum is the gesture itself. However, in both cases, the physical act of making the gesture involves a pause or *hold* on either side of the stroke phase, along with the initial preparation and final retraction phases.

Lücking et al. [18] similarly adopt McNeill's gesture phase scheme for a study of deixis, which models a pointing cone that extends outward from the extended digit. The intersection of the cone and the objects in the environment becomes the region that contains the denotatum, but only during the stroke phase. This makes automated detection of the stroke phase critical for intelligent systems that model and interpret gesture.[1]

The work that most closely approaches ours is Gebre et al. [5], on automatic stroke phase detection. But, where the authors approach detecting stroke phases

[1] A similar formulation that models deictic precision decreasing with distance is adopted by van der Sluis and Krahmer [25].

as an end in itself, we utilize detection of holds along with sequence-based segmentation techniques to implicitly segment stroke phases from other phases by utilizing the aforementioned foundational semantic frameworks [8, 15, 19].

2.2 Real World Applications

Over time, gesture recognition has become an increasingly ubiquitous method to facilitate humans interaction with computers. Multimodal interactive agents, e.g., [13, 12, 11], can use gesture along side speech, but typically require gestures to held still for a short period for the gesture recognition algorithm to achieve the requisite confidence, which leads to lag.

Hand detection tools, such as MediaPipe, an open source library developed by Google [27], can aid in gesture recognition by returning joint locations, or *landmarks*, of detected hands on a frame-by-frame basis [27] that can be used to train custom gesture recognition models with a wide range of applications. In prior studies and HCI applications, these tools have proven to be an effective foundation on which to develop custom models that classify generally static, standalone gestures that are identifiable in a single frame. For example, using a model trained to recognize a collection of simple gestures, a user can interact with a graphical user interface hands-free [6]. Additionally, gesture recognition models have been utilized as an effective simple solution to control various robotic systems [1]. In both of these examples, a dataset containing multiple static gestures is used to develop a recognition model, allowing the user to change between various gestures and commands in real time. However, in most studies the change in motion of a hand between gestures is not important to the intended command—once the user settles on a new gesture, the shape of the hand is static and identifiable in a single frame. In this paper, we present a pipeline to recognize gestures that extend across multiple frames, thus allowing more granularity in the gestures that can be used to command a system, and allowing more intuitive movements to represent a command from the user's perspective (i.e., pinching to zoom.)

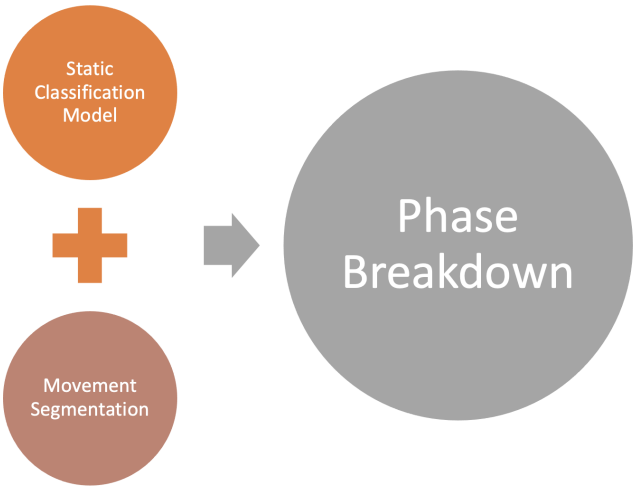
Tools like MediaPipe can reduce the inherent ambiguity present in single still images of a gesture, but a collection of frames may still contain redundant information. Thus, we propose to filter out surplus information by identifying key frames. The combined use of landmark detection and key frame identification has enabled model success for other complex (non-gestural) movements [23]. Our model, driven by effective features for robust motion detection, both aids recognition and, by smart segmentation, will cut down on time and effort that researchers spend manually annotating video data.

3 Methodology

The pipeline presented in this paper aims to automatically identify the key phases of gestures to aid in semi-automatic annotation of gesture semantics

through identifying the sub-gestural phases. Specifically, we focus on the automatic detection of "key frames," which we define as frames comprising the union of the pre-stroke, stroke, and post-stroke phases, where the most of the semantically significant movement takes place. Our pipeline consists of three stages that aid in reducing noise and distilling videos down to these key frames, and we evaluate it over a collection of complex, multi-frame gestures in two datasets. Key components of the pipeline include a classification model whose main purpose is to recognize the general static shape of complex gestures when in a hold phase (Section 3.3), a movement segmentation routine which aids in breaking down a video into segments of similar movements (Section 3.4), and a phase markup annotation which uses the classification model and the video segments in order to identify and annotate the segments and frames that are in a "hold" phase, and thus most semantically significant, or adjacent to the most semantically-significant frames. Figure 1 shows an overview of the major components and the order in which they work together. In this section we will walk step by step through each piece of the pipeline and tools used in more detail.

Fig. 1. Complex gesture annotation pipeline



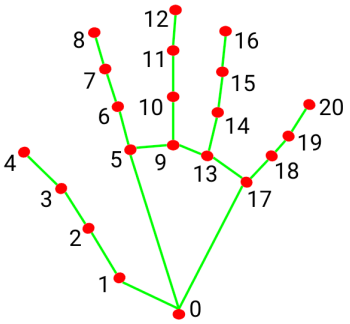
3.1 MediaPipe

Categorizing gestures inherently involves a focus on hand poses and movements. This is key to the gesture semantics approaches discussed in Section 2.1, as well as to computer vision approaches to gesture recognition [22]. Therefore, extraction of features of hand location is critical. We use MediaPipe to return joint locations, or landmarks, of detected hands on a frame-by-frame basis, which

are used to help identify key frames. MediaPipe tracks hands using 21 landmarks that consist of x, y, and z (relative depth) coordinates. It uses a two-stage pipeline: 1) the palm is detected using the full input image, and a bounding box is formed around the hand that marks its initial location; 2) more precise hand landmarks (joint positions) are located and modeled based on the location of the hand bounding box.. Fig. 2 shows the location of hand landmarks and indexes returned by MediaPipe. This process is optimized by using the hand bounding box from the previous frame to help track the bounding box in the next frame. The entire input image is only used again if tracking confidence drops below a defined threshold [27].

Google's API includes a handful of user definable parameters that can be configured when using MediaPipe. We use a few of these, including maximum number of hands, minimum detection confidence, and minimum tracking confidence. Maximum number of hands allows the user to define the number of hands expected in the frames and defaults to 2. Minimum detection confidence, which is on a scale of 0 to 1, allows the user to specify the confidence value required to consider a hand detected, higher values require clearer images but also reduce false positive values. In addition minimum tracking confidence, which is also on a scale of 0 to 1, allows the user to specify the confidence level for tracking via the previous frame's data. A higher value leads to more robust tracking but could cause latency issues if the entire input image is consistently used for tracking. Both the detection and tracking confidence default to 0.5. For annotation, we use different values, mentioned in Section 3.3.

Fig. 2. MediaPipe Hand Landmarks (reproduced from [27])



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

3.2 Datasets

Microgesture Dataset Microgestures are a category of small, subtle hand gesture requiring less gross motion [26]. The overall goal of such gestures is to reduce the fatigue of a user interacting with a system long-term. The "Microgesture dataset" is a collection of short single-gesture videos, focusing on a single

hand. 49 gestures are included in this dataset, each unique and identifiable, fitting into categories including: taps, rotations, move, snap, numbers, zoom, open/close, and slide. Using 3 Microsoft Azure Kinect cameras positioned at 3 different angles, 72 videos were collected for each gesture, spanning 10 participants [7]. Each video consists of approximately 60-80 frames with different start and end points for each gesture, thus making the data a good candidate to test our phase detection solution.

Weights Task Dataset The Weights Task dataset provides data that can be used to test our gesture phase segmentation pipeline on a more realistic scenario. This dataset is a collection of audio-visual recordings data where triads perform a task involving correctly identifying the weights of various colored blocks using a balance scale. Unknown to the participants, the weights of the blocks follow a specific pattern, and the task involves participants collaboratively uncovering this pattern. This shared collaborative task involves multimodal communication using speech, gesture, gaze, and action in context, making the identification of key gesture semantics important for automated analysis. The dataset spans 10 groups of 3, and includes videos from 3 different camera angles [2]. The videos include many potential gestures of interest, including pointing, grasping and placing blocks on a scale, which are good candidates for automatic phase segmentation and annotation. Because there are multiple hands in these videos, there are processing issues that need to be overcome to make our solution more robust, which we discuss further in Section 6.

3.3 Static Classification Model

The first step of our pipeline is a classifier that runs over individual frames to identify frames where the hand appears in the shape identifiable as the gesture of interest. In this section we outline the annotation method used to create data for the classifier and the details of the classifier itself.

Annotation Some manual annotation is required to create a static classification tool for any given complex gesture. In order to create this data, we developed an annotation script for the Microgesture and Weights Task datasets. Using this script we step frame by frame through a sample video using OpenCV. For any selected frame of the Microgesture data we save off the *phase type*, *gesture type*, *participant ID*, and normalized landmarks returned from MediaPipe. For the Weights Task dataset we include additional fields such as *hand index* and *group ID*. *Phase type* can be one of two values: "hold," which signifies a frame in which the hand shape is similar to the shape of the gesture in any of the stroke phases, and "no hold" which represents a noise shape (dissimilar from stroke) that should be ignored. *Gesture type* maps directly to the index of the gesture in the Microgesture dataset, and in the case of the Weights Task data can be user-defined to map to a predefined gesture of interest. Figures 3 and 4 show examples of annotated data from each of the datasets used in this paper.

For annotation using the Microgesture dataset, we used the default tracking and detection values for MediaPipe (cf. Section 3.1), and set the maximum number of hands to 1. For the Weights Task data both tracking and detection values were set to 0.6 with maximum hands set to 6, thus allowing us to annotate any combination of hands returned from MediaPipe.

Fig. 3. Annotation examples from Microgesture dataset

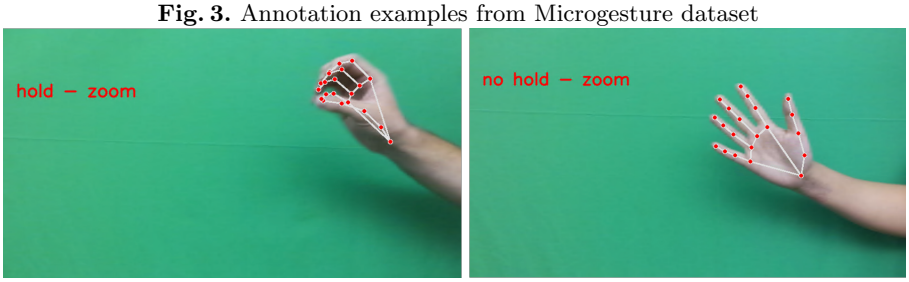
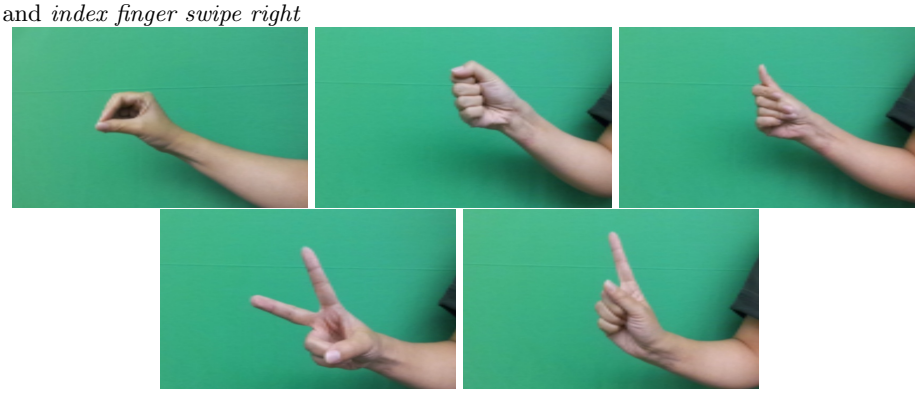


Fig. 4. Annotation examples from Weights Task dataset

Classification Model To test our pipeline we experiment on a subset of gestures from the Microgesture dataset and created a classification model for each. To generate this subset we selected one gesture from a handful of the hierarchical categories of the Microgesture dataset [7]. We selected one gesture from the following categories: *move*, *snap*, *number*, *zoom*, *open/close*. The categories were with the intent to create a diverse selection of gestures across multiple categories. From here we selected one specific gesture from each category, including *two* for number, *index finger swipe right* for move, *hand close* for open/close, *zoom out with palm* for zoom and *snap* for snap. Because each gesture comes from a different category, we expect each to be distantly identifiable using key frames. Fig. 5 shows samples of each of the 5 gestures of interest selected for evaluation.

Fig. 5. Hold phase stills (from top left): *zoom out with palm*, *hand close*, *snap*, *two*, and *index finger swipe right*



Using the annotated values for each frame in the sample data for each gesture, we create a random forest binary classifier using the scikit-learn library for each gesture of interest. A random forest classifier is made up of a collection of independent decision-tree classifiers generated from a randomly selected subset of training data. These classifiers then vote on the most probable classification for any given input [3]. Random forest classifiers have been used in other MediaPipe based classification projects (e.g., [4] and [24]). The overall purpose of this model is to identify if a hand is in a "hold" phase or not. Table 1 shows the accuracy metrics, Cohen's Kappa, and AUROC values of each of the random forest classifiers generated. The high values for all metrics on each gesture shows that we can use landmarks to identify the general shape of the hold in a single frame, however as mentioned before this is not enough to identify key frames and additional processing is required.

Gesture	Accuracy	Balanced Accuracy	Kappa	AUROC
Two	0.92	0.93	0.92	0.92
Index finger swipe right	0.91	0.87	0.89	0.89
Hand close	0.87	0.78	0.84	0.82
Zoom out with palm	0.83	0.91	0.88	0.87
Snap	0.94	0.95	0.95	0.95

Table 1. Overall accuracy, balanced accuracy, Kappa, and AUROC. Gestures of interest come from the Microgesture Dataset.

3.4 Movement Segmentation

Our pipeline's second phase involves grouping similar individual frames into like movements. For each frame, we compute a value representing the hand's

general location in the frame, representing the average (x, y) values in pixel space, using the 21 landmarks returned from MediaPipe. Fig. 6 shows an example of average hand location determined from the individual landmarks. Hereafter we define a collection of frames with a difference in average location under a defined threshold as a "segment."

In order to generate segments for a video, we first cache off information for each frame, including the video number, frame number, MediaPipe landmarks, the normalized landmark values, and the average (x, y) location of the hand. If the current frame's average location is significantly different from the last frame this marks the start of a new segment, otherwise the frame is added to the current segment. The threshold value used in our experiments is 0.8 pixels, however the most effective value for other scenarios or datasets may very and thus the value is user-definable. Figs. 7 and 8 show a visualized example of these steps on a subsection of *zoom with palm* frames.

Fig. 6. Example of average hand location, shown by the yellow dot.

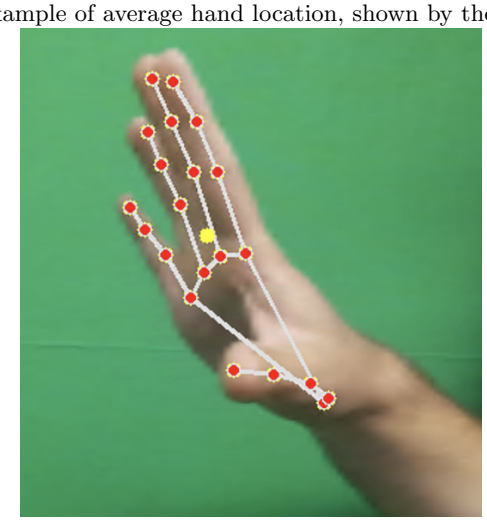


Fig. 7. Movement Segmentation: cached frame values

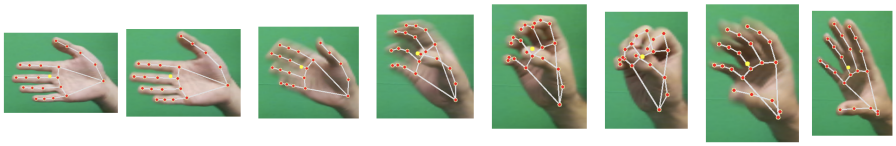
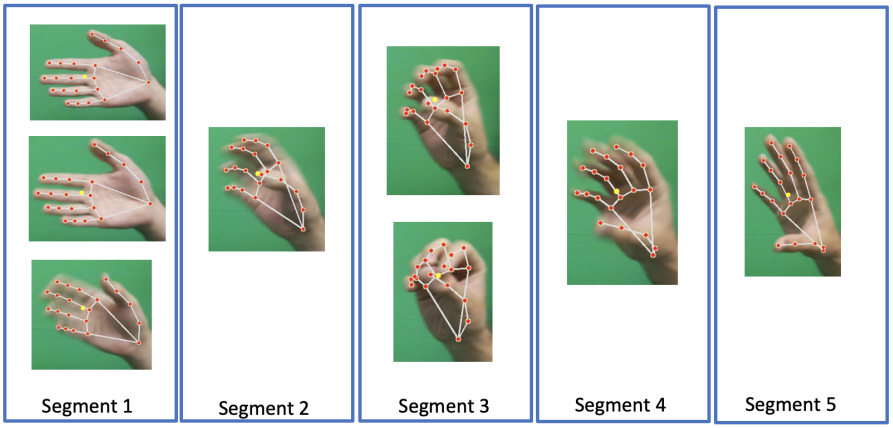


Fig. 8. Movement Segmentation: creating segments

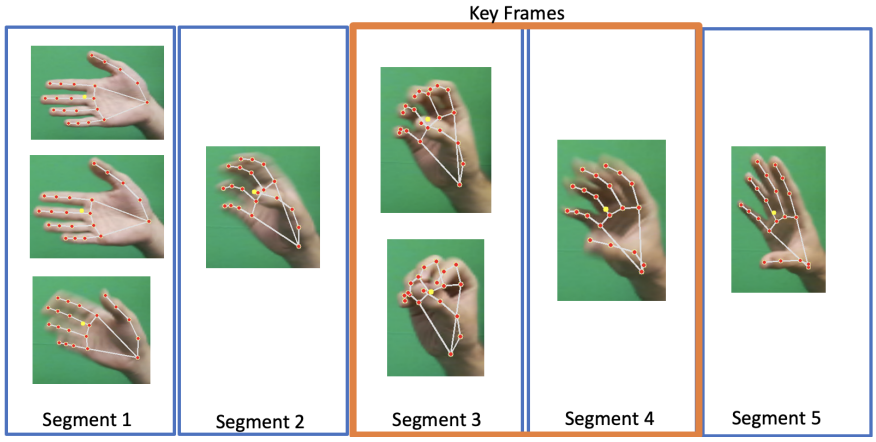


3.5 Phase Breakdown

After the video has been broken down into segments, we analyze the segments using our classification model to identify segments with a significant percentage of frames in *hold*. For each segment in the video we run our static shape classifier on each frame in the corresponding segment. If at least 80% of the frames are in *hold*, we mark that entire segment in *hold*. During this process we look for the following pattern: the first *hold* segment found marks the potential start of the key frames interval, or a "section of interest." After a hold is found, the next "no hold" segment marks the end of the section of interest. The entire detected section should span the pre-stroke, stroke and post-stroke phases. For each section of interest found in a video, we also calculate the total frames. The user can define the number of frames required to mark the section as significant or key frames and thus as a candidate for automatic annotation. Fig. 9 shows a visualized example of this grouping, continuing the example from the previous section.

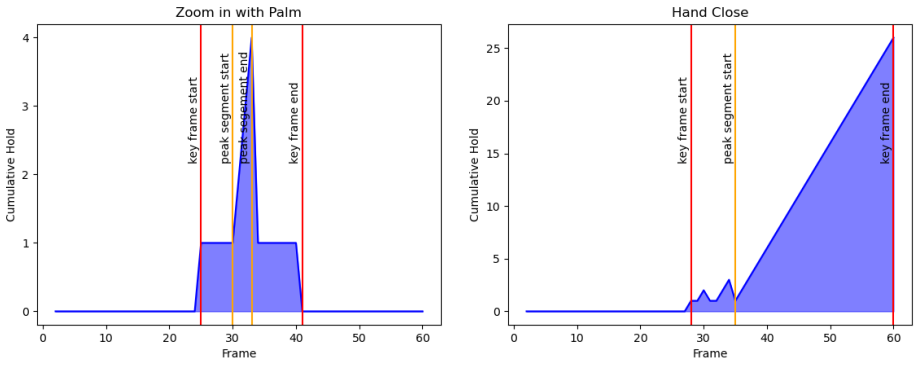
Once key frames are identified a few values are saved off that can then be utilized to help extract key frames from a video subsequently to train additional models to classify a collection of gestures in real time. These values include the first and last frame of the key frame section, and the first and last frame of the peak segment, also defined as the segment that contained the most frames. In addition, the frame count for each segment can be plotted to show the pattern of motion for any gesture of interest. It is worth noting that the pattern of peaks and valleys on this chart can be used to map the occurrence of the stroke phase, and for some gestures this could take place in or around the peak segment. Figs. 10-12 show example plots of the distributions for each of the five gestures examined. It is worth noting that for each gesture the start location and length of the key frames vary, showing the importance of dynamic key frame selection.

Fig. 9. Phase Breakdown: locating key frames



Selecting one static key frame location across all values in the dataset would lead to noise in training inputs. The "peaks" shown in these charts indicate a section of frames where motion was relatively small (that is, a high concentration of similar frames), whereas valleys show locations where change in motion is more significant.

Fig. 10. Key frame examples from *zoom out with palm* and *hand close*



4 Evaluation

Our major motivation in this paper has been to develop a way to automatically pinpoint the exact locations of key frames for any given gesture. Success in this

Fig. 11. Key frame examples from *snap* and *two*

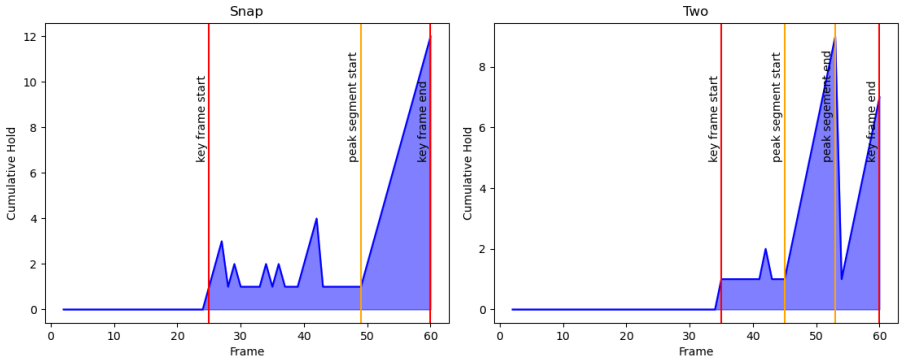
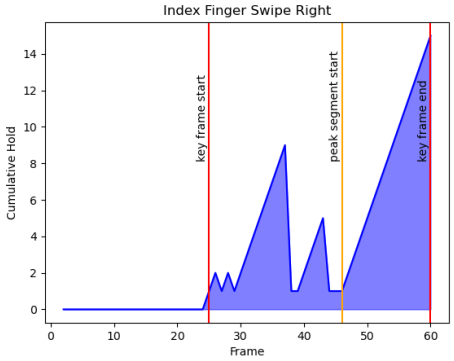


Fig. 12. Key frame examples from *index finger swipe right*



aim means that automatically extracted key frames should be more informative of a gesture to a recognition system than a naive or random baseline. We demonstrate our success on such a recognition task.

Our initial goal application was to train a classifier to recognize multi-frame gestures across the entire Microgesture dataset using MediaPipe's landmark data. In an initial attempt, where all frames in each video were gathered and fed into a simple classifier, the resulting accuracy that was equivalent to a random guess. To reduce noise and trim down the number of frames being fed into our classifier, we gathered 10 frames starting at the 20th frame for each video. This resulted in performance very similar to the first attempt. Various additional attempts were made to select a number of frames from the same static location in each video that best fit the entire dataset. Through our experiments we were able to improve accuracy slightly only to an accuracy of about 20%. This value using naive or static key frame selection serves as a baseline. We hypothesize this occurred because the true location of the key frames varied greatly for each gesture, and no single starting position would be effective across the entire dataset.

Selecting one static location across all videos leads to noise being mixed into the training data. In order to improve accuracy key frames would need to be identified on a video by video basis, and our dynamic key frame selection method serves as a possible solution.

To test our proposed solution we repeated our initial experiment on the Microgesture subset referenced in Section 3.3 (the gestures of interest include *snap*, *two*, *index finger swipe right*, *hand close*, and *zoom in with palm*). Using both the static and dynamic key frame collection methods, we trained a feedforward neural network on this subset, using 2 ReLU-activated hidden layers of 20 and 10 units respectively with a final softmax classification layer. The model was trained for 100 epochs with an Adam optimizer, a learning rate of 0.001 and batch size of 16, using a leave one out split for each of the 10 participants. In both cases we gathered 10 frames from each video.

5 Results and Discussion

Our results show an average of all 10 leave-one-out splits. Table 2 shows statistics from both classifiers. Table 3 shows associated standard deviations across all 10 splits for each figure. Our baseline values are much higher then the values in our initial experiments that spanned the entire Microgesture dataset, as for the subset selected, the static frame selection does better at selecting relevant information. However our results show that when using dynamic key frame selection, there is still a significant increase in overall performance. Figs. 13 and 14 show the performance of the classifier using both frame selection methods in the form of confusion matrices (summed over all splits). It is evident from Fig. 14 that dynamic key frame selection does significantly better over all the gestures of interest. Our experiments show that our pipeline and dynamic key frame selection is a promising solution to reduce visual noise in a dataset by focusing on segmenting gestures using semantically-informed key inflection points, thus improving the performance of models tasked with identifying complex multi-frame gestures.

Method	**Precision**	**Recall**	**F1**	**Top-1**	**Top-3**
Static	35.28	39.16	33.86	41.48	83.49
Dynamic	66.35	66.48	62.78	69.10	89.10

Table 2. Average reported metrics: precision, recall, F1, and top-k accuracy. 10 frames were gathered for both methods. Static collection started at frame 20 of each video. Dynamic started at the unique key frame location for each individual video.

Method	Precision	Recall	F1	Top-1	Top-3
Static	23.99	20.36	21.97	21.16	9.78
Dynamic	22.80	19.00	22.00	18.28	9.86

Table 3. Standard deviation of reported metrics: precision, recall, F1, and top-k accuracy.

Fig. 13. Classification results using static key frame selection

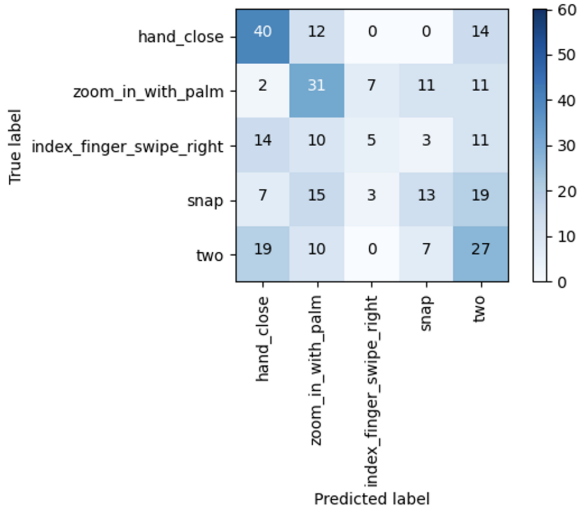


Fig. 14. Classification results using dynamic key frame selection

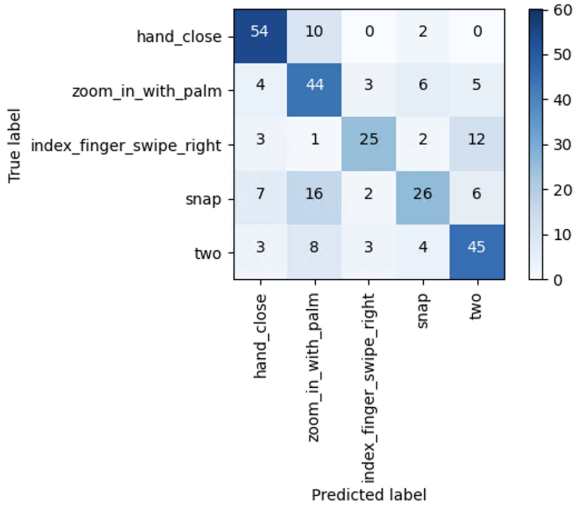
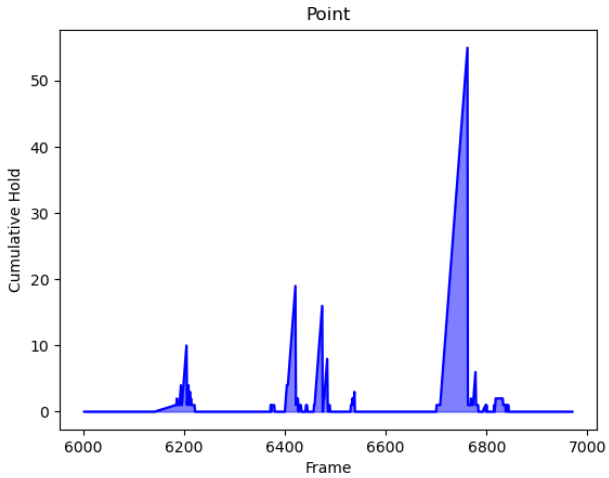


Fig. 15. Weights Task, point detection



6 Conclusion and Future Work

In this paper we presented a pipeline and methodology for dynamic key frame selection of gestures in video. Our method is informed by foundational work on the semantics of gesture used to model the behavior of humans in multimodal communication, and our key frame selection method strongly correlates to segmentation of the hold and stroke phases of a gesture. Our methodology is expected to aid in the semi-automatic annotation of gesture, e.g., for the purposes of augmenting gesture-speech corpora or correcting errors in gesture recognition. We demonstrated the utility of our approach on a gesture classification task, and demonstrated up to a 30% increase in classification accuracy against a static frame selection baseline, even when the subset of gesture chosen artificially elevated baseline performance.

While our pipeline shows promise in its ability to help identify the location of key frames that increase the performance of complex gesture classifiers, there are a few areas of potential future work. Our method relies on MediaPipe, which itself has a few limitations that need to be overcome before our pipeline is a feasible to aid in annotating video that contain multiple hands. While MediaPipe does provide the ability to track multiple hands, it does not guarantee the order of the returned hands. This means that in real-world situations with multiple hands and participants, such as in the described Weights Task dataset (Section 3.2), hands can get mixed up and lead to mislabeling in our annotations.

Our pipeline did show promise in being able to identify pointing gestures in various subsections of videos, however the videos needed to be vetted first to make sure the same hand was continuously tracked, whereas for the Microgesture dataset we configured MediaPipe to only identify one hand. Fig. 15 shows the

output of our segmentation pipeline over a subsection of one of the Weights Task videos. The figure shows that our pipeline could detect deictic pointing gestures (cf. the video stills in Fig. 4) in a group work scenario and return meaningful output, however to make our solution more robust to this type of dataset additional work needs to be done to consistently associate a hand with a participant. One potential solution for this could involve rendering the Kinect body locations on each frame, along with the MediaPipe output. That way the wrist location of the body closest to a hand could be tracked to potentially make hand tracking more consistent.

Finally, our experimental results, while demonstrating promise and also computational efficiency by relying not on one large model but a series of small ones, were presented only a subset of gestures from the Microgesture dataset. Our pipeline needs to be evaluated over a full dataset, such as the entire Microgesture dataset or a more complex scenario, such as the Weights Task dataset.

Acknowledgements

This work was partially supported by the National Science Foundation under awards CNS 2016714 and DRL 1559731 to Colorado State University. The views expressed are those of the authors and do not reflect the official policy or position of the U.S. Government. All errors and mistakes are, of course, the responsibilities of the authors.

References

1. Allena, C.D., De Leon, R.C., Wong, Y.H.: Easy hand gesture control of a ros-car using google mediapipe for surveillance use. In: HCI in Business, Government and Organizations: 9th International Conference, HCIBGO 2022, Held as Part of the 24th HCI International Conference, HCII 2022, Virtual Event, June 26–July 1, 2022, Proceedings. pp. 247–260. Springer (2022)
2. Bradford, M., Khebour, I., Blanchard, N., Krishnaswamy, N.: Automatic detection of collaborative states in small groups using multimodal features. In: International Conference on Artificial Intelligence in Education (under review)
3. Breiman, L.: Bagging predictors. Machine learning **24**, 123–140 (1996)
4. Bugarin, C.A.Q., Lopez, J.M.M., Pineda, S.G.M., Sambrano, M.F.C., Loresco, P.J.M.: Machine vision-based fall detection system using mediapipe pose with iot monitoring and alarm pp. 269–274 (2022). https://doi.org/10.1109/R10-HTC54060.2022.9929527
5. Gebre, B.G., Wittenburg, P., Lenkiewicz, P.: Towards automatic gesture stroke detection. In: LREC 2012: 8th international conference on language resources and evaluation. pp. 231–235. European Language Resources Association (2012)
6. Indriani, M.H., Agoes, A.S.: Applying hand gesture recognition for user guide application using mediapipe. 2nd International Seminar of Science and Applied Technology (ISSAT 2021) p. 101–108 (2021)
7. Kandoi, C., Jung, C., Mannan, S., VanderHoeven, H., Meisman, Q., Krishnaswamy, N., Blanchard, N.: Intentional microgesture recognition for extended human-computer interaction. In: Human-Computer Interaction, HCI 2023, Held as Part of the 25th HCI International Conference, HCII 2023. Springer (2023)

8. Kendon, A., et al.: Gesticulation and speech: Two aspects of the process of utterance. The relationship of verbal and nonverbal communication **25**(1980), 207–227 (1980)
9. Kranstedt, A., Lücking, A., Pfeiffer, T., Rieser, H., Wachsmuth, I.: Deixis: How to determine demonstrated objects using a pointing cone. In: Gesture in Human-Computer Interaction and Simulation: 6th International Gesture Workshop, GW 2005, Berder Island, France, May 18-20, 2005, Revised Selected Papers 6. pp. 300–311. Springer (2006)
10. Kranstedt, A., Wachsmuth, I.: Incremental generation of multimodal deixis referring to objects. In: Proceedings of the Tenth European Workshop on Natural Language Generation (ENLG-05) (2005)
11. Krishnaswamy, N., Alalyani, N.: Embodied multimodal agents to bridge the understanding gap. In: Proceedings of the First Workshop on Bridging Human–Computer Interaction and Natural Language Processing. pp. 41–46 (2021)
12. Krishnaswamy, N., Narayana, P., Bangar, R., Rim, K., Patil, D., McNeely-White, D., Ruiz, J., Draper, B., Beveridge, R., Pustejovsky, J.: Diana's world: A situated multimodal interactive agent. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 13618–13619 (2020)
13. Krishnaswamy, N., Narayana, P., Wang, I., Rim, K., Bangar, R., Patil, D., Mulay, G., Beveridge, R., Ruiz, J., Draper, B., et al.: Communicating and acting: Understanding gesture in simulation semantics. In: IWCS 2017—12th International Conference on Computational Semantics—Short papers (2017)
14. Lascarides, A., Stone, M.: Formal semantics for iconic gesture. Universität Potsdam (2006)
15. Lascarides, A., Stone, M.: A formal semantic analysis of gesture. Journal of Semantics **26**(4), 393–449 (2009)
16. Lücking, A., Bergman, K., Hahn, F., Kopp, S., Rieser, H.: Data-based analysis of speech and gesture: The bielefeld speech and gesture alignment corpus (saga) and its applications. Journal on Multimodal User Interfaces **7**, 5–18 (2013)
17. Lücking, A., Bergmann, K., Hahn, F., Kopp, S., Rieser, H.: The bielefeld speech and gesture alignment corpus (saga). In: LREC 2010 workshop: Multimodal corpora–advances in capturing, coding and analyzing multimodality (2010)
18. Lücking, A., Pfeiffer, T., Rieser, H.: Pointing and reference reconsidered. Journal of Pragmatics **77**, 56–79 (2015)
19. McNeill, D.: Hand and mind. Advances in Visual Semiotics **351** (1992)
20. McNeill, D.: Language and gesture, vol. 2. Cambridge University Press (2000)
21. McNeill, D.: Gesture and thought. In: Gesture and Thought. University of Chicago press (2008)
22. Narayana, P., Beveridge, R., Draper, B.A.: Gesture recognition: Focus on the hands. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5235–5244 (2018)
23. Roygaga, C., Patil, D., Boyle, M., Pickard, W., Reiser, R., Bharati, A., Blanchard, N.: APE-V: Athlete Performance Evaluation using Video pp. 691–700 (2022)
24. Singh, A.K., Kumbhare, V.A., Arthi, K.: Real-Time Human Pose Detection and Recognition Using MediaPipe pp. 145–154 (2022). https://doi.org/10.1007/978-981-16-7088-6_12
25. van der Sluis, I., Krahmer, E.: Generating multimodal references. Discourse Processes **44**(3), 145–174 (2007)
26. Wolf, K., Naumann, A., Rohs, M., Müller, J.: A taxonomy of microinteractions: Defining microgestures based on ergonomic and scenario-dependent requirements.

In: 13th International Conference on Human-Computer Interaction (INTERACT). pp. 559–575. No. Part I, Springer (2011)
27. Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G., Chang, C.L., Grundmann, M.: Mediapipe hands: On-device real-time hand tracking. arXiv preprint arXiv:2006.10214 (2020)