A Rapid Prototyping System for Hand Gesture Recognition

Tim Straubinger
University of British Columbia

Nam Hee Kim
University of British Columbia

ABSTRACT

Prototyping gesture recognition systems in a fast, reliable, and cheap way can make HCI applications easier to develop and more accessible. We present a simple system that combines a 3D hand pose estimator and a speed-invariant gesture recognizer. Our system allows for rapid prototyping of vision-based hand gesture recognition software without the burden of massive data collection.

Author Keywords

Hand gesture recognition; 3D hand pose estimation; rapid prototyping

INTRODUCTION

Hands and their gestures are used extensively in communication between people, and can convey much additional information about a person's intentions and feelings. They present a much more intuitive medium to communicate compared to the conventional keyboard and mouse used in modern computer interfaces which require a large amount of time to learn and gain familiarity with. The expressive potential of hand gestures is also clearly demonstrated by the viability of sign language as a primary means of communication in humans. Conceivably, gesture-driven computer interfaces could be as effective as or more effective than existing mice and keyboards for receiving user input.

Hand detection detection and pose estimation from a monocular image sequence has been explored with many unique approaches over the past few decades [10]. Many previous works of research employ highly specialized computer vision algorithms, and while many of these are data-driven, most of the existing literature predates the widespread use of convolutional neural networks (CNNs) in computer vision which have become highly effective at object detection and extracting high-level information from images. Thus, one of our goals is to explore the use of CNNs as hand pose estimators in monocular video-based gesture recognition.

The task of gesture recognition following hand detection has also been investigated diversely using a variety of classification and pattern-matching techniques. However, many of the techniques in use for dynamic gesture recognition are difficult to understand, implement, and train, and so we experiment with a dynamic gesture recognition technique which is easier in all of these regards.

Our main contributions are as follows. First, we present a system that performs both static and dynamic gesture recognition tasks. This is done by combining a 3D CNN-based hand pose estimator with a simple dynamic gesture recognizer. Next, we show that planar joint angles are viable features for dynamic gesture recognition. Finally, we introduce a preprocessing technique for static pose recognition that can correct for structured noise in the upstream pose estimator and greatly improve its stability.

For qualitative results, we refer the reader to the project website: https://sites.google.com/view/arpshand

RELATED WORK

Hand gesture recognition

Gesture recognition is by no means a new problem. Early work in gesture-based interfaces focused on hand-crafted solutions for classifying gestures [11, 2, 5]. Pattern matching methods have also been researched such Dynamic Time Warping [14], while many classical machine learning methods such as linear classifiers, support vector machines, hidden Markov models, recurrent neural networks and finite state machines [10]. In the early 1990s, Rubine developed feature-based automatic gesture recognition [12] as well as data-driven gesture recognition [13]. Later, in 2007, Wobbrock et al. developed a robust recognizer driven by examples, dubbed the $1 Recognizer [19]. Meanwhile, the development of hardware with hand-sensing capabilities enabled gesture recognition in hand movements [9]. More recently, classical vision-based techniques were combined for real-time hand gesture recognition [3]. Our

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

work borrows largely from the $1 Recognizer with modifications. Although the work cited here are mostly focused on one- or two-dimensional time-series data, we will deal with multi-dimensional input signal collected via 3D hand pose estimation.

3D hand pose estimation

Work in reconstructing 3D hand pose from monocular RGB images spans a few decades. Many techniques make use of purpose-built computer vision pipelines and are data-driven to vary extents. These methods include template-matching with data augmentation [15] and with silhouette and motion analysis [16], image brightness and motion estimation [5], skin-colour segmentation with depth estimation [17] and with edge detection [4], and dense trajectories [1].

As deep learning with artificial neural networks has gained much traction, almost all recent methods leverage deep learning in the hand pose estimation domain [20, 6, 7]. Our interest is in combining a 3D hand pose estimator with gesture recognition techniques, and we elect to use a fairly primitive model from [21] and investigate methods to supplement its performance.

Rapid prototyping

Rapid prototyping emerged as a strategy to produce a viable software product while optimizing the efficiency of the process [18]. [18] argued that rapid prototyping may offer dramatic advantages in instructional context as well as industrial context. Rapid prototyping has promising outlook in HCI applications. For example, [19] listed rapid prototyping as the main motivation behind the design of the $1 Recognizer. Our aim is to extend this motivation to hand gesture recognition domain.

METHOD

From a high-level view our system takes a monocular RGB video sequence as input, from which we extract 3D joint locations of the hand using a pre-trained CNN [21]. From this sequence of multiple 3D coordinates, we perform basic feature extraction. We then wait for a special static 'clutch' pose to appear which signifies the beginning of a dynamic gesture, and begin recording a sequence which is then classified using an adaptation of the $1 Recognizer [19].

Data Collection

We collect data from two different monocular RGB webcams[1], involving mainly two individuals. We capture videos of a single left hand performing a single gesture, beginning from the clutch pose. Hence, a data point in our dataset is a labeled pair $(\{x_1, x_2, \ldots, x_f\}, y)$, where $\{x_1, x_2, \ldots, x_f\}$ is the sequence of f frames in the video (after downsampling the video at a smaller framerate, i.e. choosing keeping one frame per every 3 frames, etc.) and y is the gesture label. For the purpose of limiting the scope of this project, we elect to use a set of 6 labels which are described shortly. An example for each gesture label can be viewed on the project website. We collected only a few videos for each gesture label.

[1]Dell XPS 13 9370's build-in webcam and IPEVO Point 2 View USB Document Camera

Name	# of videos
Paper	3
Scissors	3
Thumbs-Up	3
OK	7
Call-Me	3
Let's-Drink	3
Clutch	26

Table 1. Only a few videos are used for each gesture as a training example, showcasing the $1 Recognizer's viability in prototyping scenarios.

Gesture Labels

We use the following six gestures to demonstrate and test our system.

- *Paper* All fingers are out-stretched and slightly splayed out.
- *Scissors* All fingers remain in a fist pose, except for the pointer and middle fingers which are out-stretched and slightly bent away from one another.
- *Thumbs-Up* All fingers remain in a fist pose, while the thumb is out-stretched. Meanwhile, the hand is rotated to point the thumb upwards.
- *OK* The thumb and pointer finger are half-extended and their tips are brought together. The middle, ring, and pinky fingers are extended fully and splayed out.
- *Call-Me* The thumb and pink are out-stretched, while the other fingers remain in a fist pose. The palm is rotated away from the camera, and the thumb is raised while the pinky is lowered.
- *Let's-Drink* The thumb and pink are out-stretched while all other fingers stay lowered. The palm remains facing the camera while it is rotated to lift the pinky while lowering the thumb.

In addition to these dynamic gestures, we also defined a *clutch* pose, which is used to segment the input signal during runtime. A clutch pose can be defined as any configuration of the hand, as long as the pose does not intersect with any of the dynamic gestures significantly. In our implementation, we elect to use the fully closed fist facing the camera (see Figure 2). We record many videos of the clutch pose with varying hand root orientations and joint angles in attempt to increase the accuracy of the clutch detector.

3D Hand Pose Estimation

Zimmermann et al. [21] published a fairly primitive model that takes an RGB image as input and predicts the 3D coordinates of 16 keypoints of the left hand (consisting of 15 finger joints and the base of the hand). We use their available software package to extract the keypoint coordinates for each frame in our dataset.

3D Coordinates to Planar Angles

We elect to use a single quantity to characterize the angle defined by three consecutive joints in the hand. Given a triplet

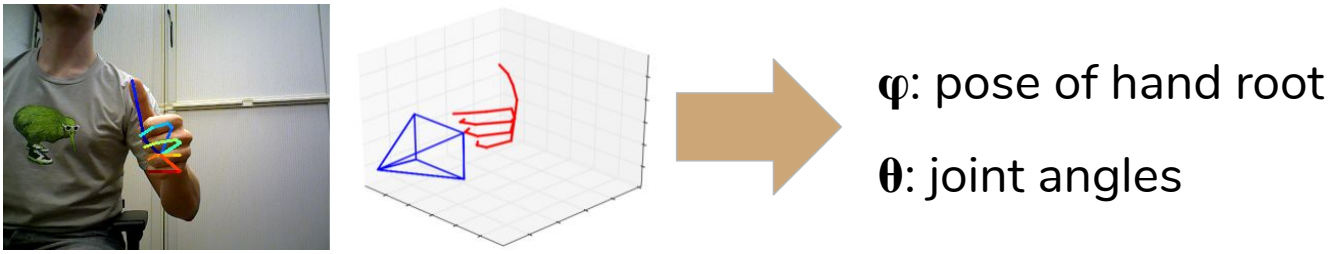

Figure 1. The feature vectors ψ and θ are extracted from the raw RGB frames (left) by using a 3D pose estimator, which estimates the coordinates of the joints of the hand (center). We further transform these coordinates into the joint angles θ as well as the pose parameters of the hand root ψ.

Figure 2. The reference clutch pose. Bias correction is applied based on the difference in overall hand orientation between the input pose and the reference pose.

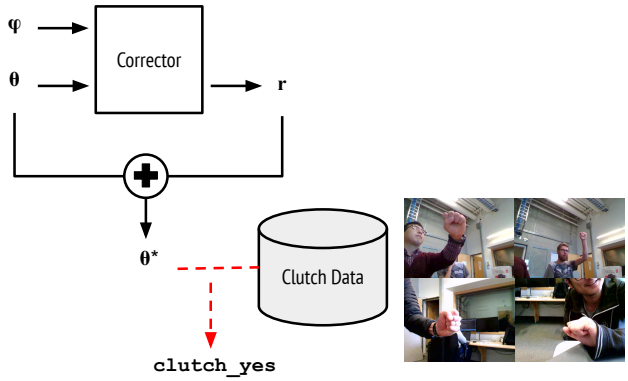

Figure 3. A bias correction with residuals is applied to the joint angles computed in Figure 1. We first compute the difference between the reference clutch joint angles and current joint angles, Then, we fit a model that predicts the difference based on the root pose and the current joint angles. The corrected joint angles vector is compared against the database of clutch joint angles to determine whether a given video input contains a clutch pose.

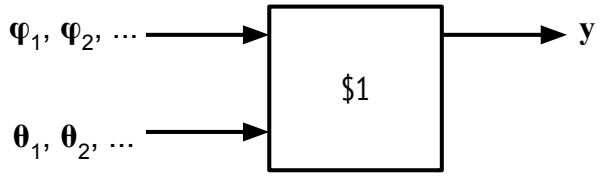

Figure 4. The $1 Recognizer takes a feature sequence $(\psi_1,\theta_1),(\psi_2,\theta_2),\cdots)$ as input and classifies the input sequence into a gesture label y.

of 3D coordinates (p_1,p_2,p_3), we compute

$$u = \frac{p_1 - p_2}{||p_1 - p_2||} \tag{1}$$

$$v = \frac{p_3 - p_2}{||p_3 - p_2||} \tag{2}$$

$$\theta = \arccos\langle u,v\rangle \tag{3}$$

where $\langle u,v\rangle$ is the dot product between the two 3-dimensional vectors. This angle θ is the angular displacement between p_1 and p_3 after treating p_2 as the origin, assuming (albeit naïvely) that the angular displacement happens along a single axis of rotation, which is orthogonal to the plane containing the three points. Hence, we call the angle θ a *planar angle*. We iterate through all triplets defined by consecutive keypoints of the Zimmerman model, extracting 15 joint angles in total. This provides a reduction from 45 original features (one *xyz*-coordinate per joint), and yields what we consider to be a more intuitive set of features, while the reduction in complexity helps to prevent over-fitting.

Residual Correction

The Zimmermann model's predictions are very sensitive to the orientation of the hand, such that the exact same joint configurations end up producing significantly different relative coordinates coordinates depending on whether the hand is rotated away from the camera. This is an issue especially for clutch detection, where slight rotations of the hands might completely throw off the gesture recognition performance. We observe that there is a hidden but structured relationship between the output coordinates and the input pose, and we elect to use a simple machine learning model to account for this artifact. The top potion of Figure 3 shows an abstract view of the bias corrector model (dubbed *Corrector*). Let the angle (θ_{ref}) be the joint angles representing the reference clutch pose shown in Figure 2. For each pose (ψ_i,θ_i), where $i \in 1,2,\cdots,n_{clutch}$ with n_{clutch} representing number of frames labeled as clutch, we generate the ground truth residue r_i:

$$r_i = \theta_{ref} - \theta_i. \tag{4}$$

This simple equation establishes that $\theta_{ref} = \theta_i + r_i$. After generating the residues corresponding to all frames labeled as clutch, we fit a predictor $\hat{f}$ that minimizes

$$\mathscr{L} = \sum_{i=1}^{n_{clutch}} \frac{1}{2}(\hat{f}(\theta_i,\psi_i) - r_i)^2 + \frac{1}{2}||v||^2 \tag{5}$$

where v is the parameters of the predictor $\hat{f}$. Consequently, for an input pose $(\tilde{\psi}, \tilde{\theta})$, we can compute the estimated residual $\tilde{r}$ with respect to the reference clutch pose:

$$\tilde{r} = \hat{f}(\tilde{\psi}, \tilde{\theta}). \tag{6}$$

Then, as established earlier, the estimated pose after the residual correction is simply

$$\tilde{\theta}^* = \tilde{\theta} + \tilde{r}. \tag{7}$$

Given that the input pose $(\tilde{\psi}, \tilde{\theta})$ is within the distribution of the clutch poses in the training data, the corrected joint angles $\tilde{\theta}^*$ should resemble the reference joint angles θ_{ref}. Then using $\tilde{\theta}^*$ as the features for clutch detection makes the detection robust to the differing hand root orientations. Figure 5 shows the effect of the residual correction in clutch poses with varying hand root orientations.

Clutch Detection

We segment the input stream of video by running clutch detection at every frame. This detection is a test of whether the given input pose $(\tilde{\psi}, \tilde{\theta})$ is within the distribution of the clutch pose data compiled by the user. We use a non-parametric method to detect the clutch pose. Notably, our method does not require negative examples and hence lifts the burden of labeling positive and negative examples in the training data.

We use the trained residual corrector to compute the corrected joint angles θ_i^* for every frame labeled as clutch (see Equation 7). We note that we are reusing the training set to generate these corrected joint angles, but as our task is to test whether a new data point is within the distribution of the training data, we argue that this is not a violation of the Golden Rule of Machine Learning. We use the $L_{\frac{1}{2}}$-distance between the corrected input joint angle $\tilde{\theta}^*$ and its nearest neighbor in the clutch dataset, and apply a threshold to declare whether a clutch is detected. More formally, we compute the distance

$$\mathscr{D}(\tilde{\psi}, \tilde{\theta}) = \min_i \left(\sum_{j=1}^{d} \left(\theta_{ij}^* - \left[\tilde{\theta}_j + \hat{f}(\tilde{\psi}, \tilde{\theta})_j \right] \right)^{\frac{1}{2}} \right)^2 \tag{8}$$

where d is the number of joint angles (15 in the case of planar angles), and with some threshold T, declare

$$is\_clutch(\tilde{\psi}, \tilde{\theta}) = \begin{cases} 1 & \text{if } \mathscr{D}(\tilde{\psi}, \tilde{\theta}) \leq T \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

We experimentally determine a good value for T. The bottom half of Figure 3 illustrates this distance-based detection.

Gesture Recognition

Recognition of dynamic gestures is a difficult task, but we simplify the problem by applying some restrictions. Firstly, because we rely on the clutch pose to delimit gestures, we do not need to segment gestures in time. Secondly, we restrict ourselves to gestures which do not contain a variable number of repeating motions. These two constraints allow us to use an implementation the $1 Recognizer introduced by Wobbrock et al. [19] with some modifications to its pre-processing and classification routines.

In summary, the $1 Recognizer as presented by Wobbrock et al consists of a number of feature transformations and preprocessing steps, followed by a very simple nearest-neighbour classification. The $1 Recognizer takes an arbitrary-length sequence of points in 2D space representing a path. This path is uniformly resampled to a fixed length, which discards speed and timing information but allows more intuitive comparison between paths. Next, the path is rotated to a standard configuration, then translated and scaled to fit perfectly inside a unit square. These transformations are applied to all training examples, and at test time, the 1$ Classifier simply chooses the label of the training gesture with the smallest distance to the test gesture. Here, the distance between paths is defined as the average L_2 distance between corresponding points in both paths

Figure 4 illustrates an abstract view of the gesture recognition model. Unlike the original $1 Recognizer, our gestures consist of sequences of points in a high-dimensional space rather than the two-dimensional space of pen-tablet drawings. Like the original $1 Recognizer, we resample this input sequence at uniform speed to yield a mono-spaced sequence with equal distance between all neighbouring pairs of points. However, we do not rotate, translate, or scale the gesture path, and our reasoning for this is as follows. We omit the rotation step described in the original algorithm because it is not meaningful in our high-dimensional space consisting of angles, whereas this is easily understood in the original 2D Euclidean space. We omit the translation step because this would discard information about the absolute position of each angle, and would cause our system to confuse gestures where the fingers are at completely different angles. We omit the scaling step because it is likely to cause degenerate behavior, namely when a gesture has extremely little variance along one axis, which causes division by an arbitrarily small number and can drastically amplify noise or cause division by zero. In the original $1 Recognizer, this is only a problem when the gesture appears to be a straight line which is relatively uncommon and easy to spot and avoid, but in our high-dimensional space, it is highly likely that the angles of one or more finger joints are staying very still throughout the gesture.

We also modify the classification algorithm of the $1 Recognizer. Instead of the L_2-norm used in the original implementation, we use the $L_{\frac{1}{2}}$ norm because it favours small differences, and we experimentally found that this improved classification accuracy. Additionally, because we have omitted the scaling step, we simply return the averaged L_frac12 distance instead of the original confidence measure which relies on preconditions established by the scaling step.

At training time, we simply resample each of the labeled training gestures and persist them, thus retaining the non-parametric nature of the original implementation. At test time, we compute the distance between a novel gesture and all known gesture, then choose the label of that training gesture which minimizes the distance between itself and the new gesture. Like the original implementation, this is effectively a k-nearest neighbours classifier with $k = 1$.

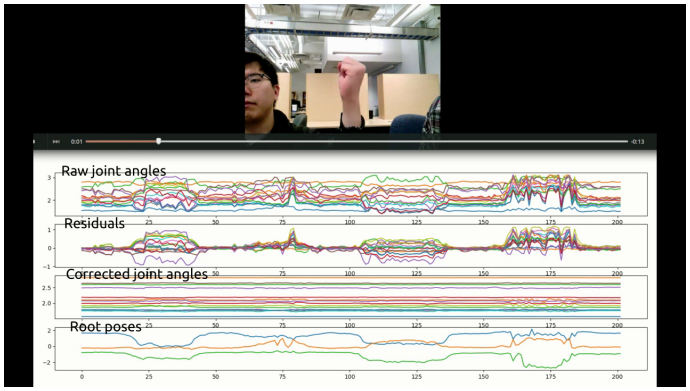

Figure 5. Image: the raw frame input of the clutch pose with a different hand root orientation than the reference clutch pose (Figure 2). First row: extracting planar angles from the image, a very different pattern in joint angles emerges although the only difference should be the hand root orientation. Second row: estimated residuals with respect to the reference clutch are computed as per Equation 7. Third row: after adding the residuals, the corrected joint angles are very stable and mostly invariant to the hand root orientation. Fourth row: the hand root orientations, in conjunction with the corrected joint angles, now parametrize the input clutch pose. Full video on the project website. Best viewed in color.

The resulting classifier performs very well with extremely few positive examples and no negative examples at all (although it does not attempt to classify an input sequence as a non-gesture), and relearning is as trivial as storing an additional gesture. If, during test time, a user is dissatisfied with an incorrect classification made by our $1 implementation, they can simply take the gesture they have just performed and instruct the classifier to learn it with the desired label. These capabilities, in addition to the simple implementation, demonstrate how our system encourages rapid prototyping.

State Machine

During user interaction, our system waits for the user's hand to appear making the clutch pose. Then it begins recording the user's hand pose and continuously attempts to classify the motion as a known gesture. After a successful classification or if the hand disappears from view at any time, the system resumes waiting idly. We implement this workflow using a state machine.

Our state machine begins in the *Wait* state where it remains idle. If at any time the hand is not detected, we return to this state. If during this state we detect the clutch pose as described earlier, we enter the *Clutch* state. To proceed from the *Clutch* state, we wait for the clutch pose to no longer be detected, at which point we transition to the *Record* state. Effectively, the user must begin and end the clutch pose before we begin recording.

Once we enter the Record state, we create an empty array of poses, and append the current hand pose to this array during each frame. As soon as the clutch pose is detected once more, we consider the gesture to have ended, and we transition to the *Classify* state. In the *Classify* state, we simply take the list of poses generated by the previous *Record* state and classify it using our adapted implementation of the 1$ Recognizer described earlier. It is at this point that the label may be passed as a notification or event to an external system that wants to use our gesture recognition. In our proof-of-concept, this system was simply a text label shown to the user, but the detected gesture could be used for a large variety of tasks such as navigating a user interface or interacting with a personal assistant. Finally, we return to the default *Wait* state and the process starts over.

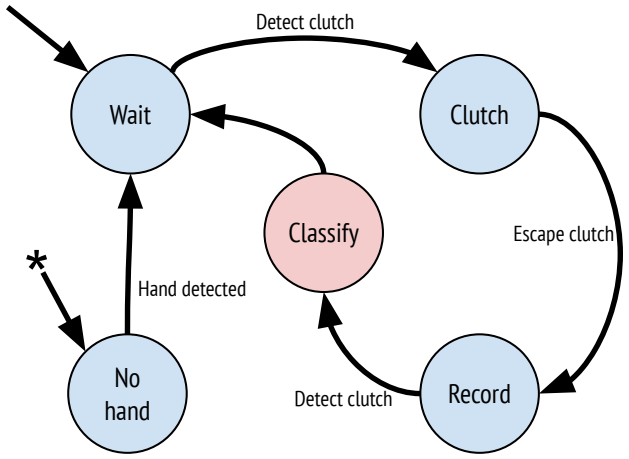

Figure 6. A simplified representation of the state machine we build to demonstrate the prototyping capabilities of our method. Clutch pose clearly marks the start and the end of a gesture window. At any time, if the 3D hand pose estimator fails to detect a hand, the machine faults to the *No hand* state which cancels all the parsing processes.

RESULTS

We evaluate our system manually by performing gestures in a randomized order and observing the system's behavior interactively. In poor lighting conditions and without careful movements, clutch detection and gesture classification suffer. Under reasonable lighting conditions and with some care taken to perform the gestures clearly and slowly, we achieve correct predictions a majority of the time. We achieve a high rate of correct labels when performing the *Scissors*, *Paper*, and *Thumbs-Up* gestures. Occasional ambiguity remains between the *Paper* and *OK* gestures. Notably, our implementation is able to distinguish between the *Call-Me* and *Let's-Drink* gestures, even though these are identical hand poses with different orientations and motions. For demonstrations of our system in

use, we direct users to the animated figures on the project website (link) which show a variety of gestures being classified in real-time after making a clutch pose. Unfortunately, we do not have quantitative results to illustrate the accuracy of our system.

CONCLUSION

We presented a system that can enable a rapid prototyping of real-time hand gesture recognition applications. As the qualitative results show, our system works well with very few training examples and can disambiguate gestures via motion, rather than simply classifying static poses. The clutch detection method is able to reliably distinguish clutch and non-clutch poses, even without the negative examples presented during training time. Our system is robust to the limb lengths of different users and varying lengths of the input sequences, which makes it a viable system for cross-user scenarios. Moreover, if the 3D hand pose estimator is running on a GPU-enabled server, the edge device does not need to have a GPU computing capacity.

Limitations and Future Work

We earlier noted that the upstream model published by Zimmermann et al. [21] is a relatively primitive model. The 3D hand pose estimator's performance is a major bottleneck in the system in practice. The Zimmermann model is especially sensitive to illumination, exposure, blurs, and many other factors, which makes the overall performance of the gesture recognition system unstable. Replacing the model with a more performant model, such as GANerated hands [8] or more recent hand pose estimators could alleviate this issue.

We also note that our recognition system depends on a manual segmentation via clutch pose. For each input gesture, the user must begin with a clutch and then end with a clutch in order to classify. Although it is possible in principle to use clutch only to mark the start of a signal sequence and produce online predictions in real time, we are unable to perform online predictions with the current implementation of the $1 Recognizer. Consequently, we are unable to use our system in conjunction with periodic gestures or unsegmented sequences of gestures. Using more sophisticated dynamic models such as hidden Markov models (HMM) or recurrent neural networks (RNN) could introduce online prediction capabilities, although implementing these models requires more development effort than the $1 Recognizer, as well as far more training data.

To further develop and improve the accuracy of our system, it would be useful to have a formalized test process, ideally involving datasets of multiple users performing multiple gestures, for the purposes validation and testing, so that we may more objectively determine the effectiveness of our techniques and to perform rigorous ablation studies.

ACKNOWLEDGEMENTS

The majority of this work was performed by both authors working together closely. Otherwise, additional credit is owed to Nam for his work on data collection and the runtime demo, while Tim contributed extra work to the $1 gesture classifier. We thank Prof. Robert Xiao for the meaningful discussions and supervision throughout the CPSC 554X course. We deeply appreciate the assistance of Prof. Michiel van de Panne and Zhaoming Xie for letting us borrow their GPU resources.

REFERENCES

[1] Lorenzo Baraldi, Francesco Paci, Giuseppe Serra, Luca Benini, and Rita Cucchiara. 2014. Gesture Recognition in Ego-Centric Videos using Dense Trajectories and Hand Segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

[2] William Buxton. 1995. Chunking and phrasing and the design of human-computer dialogues. In *Readings in Human–Computer Interaction*. Elsevier, 494–499.

[3] Qing Chen, Nicolas D Georganas, and Emil M Petriu. 2007. Real-time vision-based hand gesture recognition using haar-like features. In *2007 IEEE instrumentation & measurement technology conference IMTC 2007*. IEEE, 1–6.

[4] A. Chonbodeechalermroong and T. H. Chalidabhongse. 2015. Dynamic contour matching for hand gesture recognition from monocular image. In *2015 12th International Joint Conference on Computer Science and Software Engineering (JCSSE)*. 47–51. DOI: http://dx.doi.org/10.1109/JCSSE.2015.7219768

[5] Konstantinos G. Derpanis, Richard P. Wildes, and John K. Tsotsos. 2004. Hand Gesture Recognition within a Linguistics-Based Framework. In *Computer Vision - ECCV 2004*, Tomás Pajdla and Jiří Matas (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 282–296.

[6] Liuhao Ge, Zhou Ren, Yuncheng Li, Zehao Xue, Yingying Wang, Jianfei Cai, and Junsong Yuan. 2019. 3D Hand Shape and Pose Estimation from a Single RGB Image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 10833–10842.

[7] Shile Li and Dongheui Lee. 2019. Point-to-Pose Voting based Hand Pose Estimation using Residual Permutation Equivariant Layer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 11927–11936.

[8] Franziska Mueller, Florian Bernard, Oleksandr Sotnychenko, Dushyant Mehta, Srinath Sridhar, Dan Casas, and Christian Theobalt. 2018. GANerated Hands for Real-Time 3D Hand Tracking from Monocular RGB. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*. 11. https://handtracker.mpi-inf.mpg.de/projects/GANeratedHands/

[9] David L Quam. 1990. Gesture recognition with a dataglove. In *IEEE Conference on Aerospace and Electronics*. IEEE, 755–760.

[10] Siddharth S. Rautaray and Anupam Agrawal. 2015. Vision based hand gesture recognition for human computer interaction: a survey. *Artificial Intelligence*

Review 43, 1 (2015), 1–54. DOI: http://dx.doi.org/10.1007/s10462-012-9356-9

[11] James R. Rhyne and Catherine G. Wolf. 1987. Gestural interfaces for information processing applications.

[12] Dean Rubine. 1991a. *The automatic recognition of gestures*. Ph.D. Dissertation. Citeseer.

[13] Dean Rubine. 1991b. *Specifying gestures by example*. Vol. 25. ACM.

[14] Stan Salvador and Philip Chan. 2007. Toward Accurate Dynamic Time Warping in Linear Time and Space. *Intell. Data Anal.* 11, 5 (Oct. 2007), 561–580. http://dl.acm.org/citation.cfm?id=1367985.1367993

[15] Nobutaka Shimada, Kousuke Kimura, and Yoshiaki Shirai. 2001. Real-time 3D hand posture estimation based on 2D appearance retrieval using monocular camera. In *Proceedings IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems*. IEEE, 23–30.

[16] N. Shimada, Y. Shirai, Y. Kuno, and J. Miura. 1998. Hand gesture estimation and model refinement using monocular camera-ambiguity limitation by inequality constraints. In *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*. 268–273. DOI: http://dx.doi.org/10.1109/AFGR.1998.670960

[17] Jie Song, Fabrizio Pece, Gábor Sörös, Marion Koelle, and Otmar Hilliges. 2015. Joint Estimation of 3D Hand Position and Gestures from Monocular Video for Mobile Interaction. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 3657–3660. DOI:http://dx.doi.org/10.1145/2702123.2702601

[18] Steven D Tripp and Barbara Bichelmeyer. 1990. Rapid prototyping: An alternative instructional design strategy. *Educational Technology Research and Development* 38, 1 (1990), 31–44.

[19] Jacob O Wobbrock, Andrew D Wilson, and Yang Li. 2007. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*. ACM, 159–168.

[20] Fu Xiong, Boshen Zhang, Yang Xiao, Zhiguo Cao, Taidong Yu, Joey Tianyi Zhou, and Junsong Yuan. 2019. A2j: Anchor-to-joint regression network for 3d articulated pose estimation from a single depth image. In *Proceedings of the IEEE International Conference on Computer Vision*. 793–802.

[21] Christian Zimmermann and Thomas Brox. 2017. Learning to estimate 3d hand pose from single rgb images. In *Proceedings of the IEEE International Conference on Computer Vision*. 4903–4911.