SMASH-G: A System for Modeling, Analyzing, and Synthesizing Hand Gestures

CPSC 535P

Nam Hee Gordon Kim
Tim Straubinger

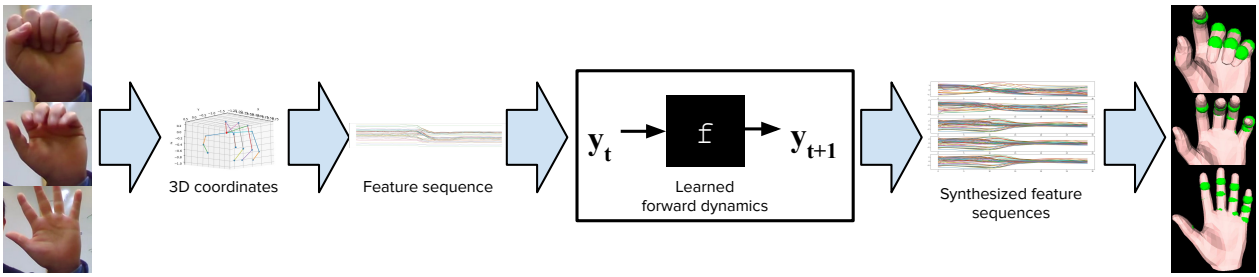

Figure 1: A high-level overview of our system. We begin with a video of a left hand making a specific gesture. We extract the 3D coordinates of the joints of the hand using a 3D pose estimator and convert the coordinates into rotation-invariant features. The resulting time series is then inputted to a machine learning model to learn a gesture-specific dynamics model. The dynamics model can be used to synthesize new feature sequences corresponding to the gesture from an unseen starting point. The synthesized feature sequences are finally rendered into a 3D hand mesh with inverse kinematics.

ABSTRACT

The task of modeling and animating for human hands is difficult due to the many intricacies in hand motions. Common articulated models contain a similar number of joints on a single hand when compared to rest of the body. While the motions of these joints are highly correlated in practice, many important gestures rely on distinct poses which require special care to capture. We present an automated and data-driven method that can animate a hand model to display a dynamic hand gesture. We use videos taken from a monocular RGB webcam and use a 3D hand pose estimator to extract the joint coordinates, which are matched with those of a hand model to reconstruct a desired pose. We also introduce a learned dynamics model that can encapsulate the machine understanding of the temporal pose variations associated with specific gesture labels. Finally, we render both real and synthesized gestures, using inverse kinematics to translate between hand models.

1 INTRODUCTION

One of the features that distinguish humans from other animals is the ability to use hands for a variety of manipulation tasks. Humans have evolved to use their hands to interact with the world and communicate with each other, and hence modeling the human hand is a crucial element in achieving a believable digital human simulation.

However, modeling the human hand can be challenging especially in the context of animation. Hundreds of hand gestures are integrated into human communication, and these can also vary in length and speed, demanding animators to pay constant attention to hands. Moreover, marker-based motion capture systems can be less cost-effective for capturing the movements of hands: a typical animation framework assigns 24 joints to the human body excluding hands, while each hand is assigned 15 joints. The sum of the two hands' degrees of freedom is larger than the rest of the body's.

In response to this challenge, we present a **s**ystem for **m**odeling, **a**nalyzing, and **s**ynthesizing **h**and **g**estures (SMASH-G). SMASH-G uses input from a commodity monocular RGB webcam to capture video data representing hand gestures and reconstructs a 3D mesh that displays learned hand gestures. Moreover, the dynamics model of each input gesture is learned to generate different instances of gestures with varying lengths and patterns in a data-driven manner.

Our main contributions are in two parts. First, we combine the existing techniques in computer vision and graphics literature to apply inverse kinematics for animating the human hand. Then, we propose a simple method to characterize and synthesize gestures via learning the dynamics. In this report, we combine the two contributions to present a step-by-step description of the pipeline that takes the labeled input video data and displays a 3D mesh showing synthesized gesture examples.

For qualitative results, we refer the reader to the project website, available in the following URL: https://sites.google.com/view/smash-g.

2 RELATED WORK

Hand models. Modeling the human hand has been studied extensively in graphics and computer animation communities. Early work can be traced to [11]. In more modern settings, linear blend skinning and pose space deformation [8] can be extended naturally to the hand, which is typically represented as a kinematic tree of

rigid bodies. [2] called for a more delicate treatment to animating the hands due to the complexity in hand motions. More recently, a number of different techniques were explored to model the human hand, including sum-of-Gaussians [20] and triangular mesh [1, 21]. The MANO model developed by Romero et al. [17] extended SMPL [10] to 3D hand scans and generated a data-driven, subspace-based hand model capturing both pose and shape of the hand. In this work, we leverage the MANO model and the associated software packages available to compute and display our animation and synthesis results.

Motion capture and inverse kinematics. Our work can be viewed as an instance of markerless motion capture for the hand. Vision-based motion capture [12] has been widely used in animation. While traditional motion capture focused on capturing the pose of the whole body, hand-specific methods were also explored [1, 18]. The motion capture data is typically used to animate the body via inverse kinematics [3, 5, 14]. We treat our raw video data as markerless motion captures and animate the MANO hand with inverse kinematics.

3D hand pose estimation. Work in reconstructing 3D hand pose from monocular RGB images can be tracked back to as early as 2001 [19]. As deep learning with artificial neural networks has gained much traction recently, almost all methods leverage deep learning in the hand pose estimation domain [4, 9, 22]. In this work, we use a fairly primitive convolutional neural network (CNN) model from [23] to convert raw video input into estimated hand poses.

Dynamical systems and learning dynamics. We adopt a dynamical view of hand motions, where a hand pose sequence follows the dynamics dictated by a gesture. Treating controllable bodies as dynamical systems is a well-established practice in robotics [6, 7]. The dynamics of a system is often estimated using state trajectory data (e.g. hidden Markov models [16]). In this work, we characterize a hand gesture as a dynamics function that can be learned from data, and show how the learned dynamics can be used to synthesize a gesture.

3 METHOD

Figures 1 and 2 provide an overview of the stages involved in our system. Broadly speaking, our workflow begins with monocular RGB video footage, from which we extract dynamic hand pose and orientation data. This data is then labeled, stored in a dataset, and learned from. We then render both ground-truth gestures and synthesized gestures to visualize the results of our learning model.

3.1 Data Acquisition

We first choose a small set of gesture labels for our system to learn. For simplicity, each gesture begins with a fist, with the palm facing the camera. Additionally, because the pose estimator [23] was only trained on left hands, we also restrict our gestures to left hands only. We have chosen the following six gestures.

- *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.

For each of these gestures, we have recorded a small set of videos of our own hands performing the gesture using an ordinary monocular RGB camera.

3.2 Feature Transformation

We use a pre-trained convolutional neural network published by Zimmermann et al. [23] to estimate the 3D coordinates of the joints of a hand in a monocular RGB image. We apply this model to every frame of an input video sequence to yield a series of points in space moving over time. While these coordinates specify the motion of the hand in great detail, they don't clearly distinguish the movements of the entire hand from those of individual fingers. To provide our learning model with a more intuitive representation of hand motion, we first transform the hand into a standard configuration and extract its position orientation as a separate feature. We translate the root of the hand $\bar{x}_{root}$ to the origin via a translation T. We then rotate the hand by taking two unit vectors $\vec{u}, \vec{v}$ from the base of the hand towards the pointer and pinky knuckles, then finding the orthonormal transformation R which moves these vectors into the XY plane, with the base of the hand at the origin and the X-axis passing through the pointer knuckle. Below, we provide the equations defining the matrices T and R.

$$T = \begin{bmatrix} I & -\bar{x}_{root} \\ \vec{0}^{\intercal} & 1 \end{bmatrix} \tag{1}$$

$$\vec{r}_1 = \vec{u}, \tag{2}$$

$$\vec{r}_2 = \vec{u} \times \vec{v}, \tag{3}$$

$$\vec{r}_3 = \vec{r}_1 \times \vec{r}_2 \tag{4}$$

$$R = \begin{bmatrix} \vec{r}_1 & \vec{r}_2 & \vec{r}_3 & \vec{0} \\ 0 & 0 & 0 & 1 \end{bmatrix}^{\intercal} \tag{5}$$

By applying the transformations RT to the entire hand, we can now analyze the hand's pose independently of its orientation. We extract Euler angles from the rotation matrix, and include these as an additional feature describing the hand's orientation.

Next, because the motion of fingertips is strongly affected by the orientation of proximal finger joints, we compute the displacement vectors between pairs of joints that are connected by a bone. This serves to decouple the overall direction of the finger from its curvature.

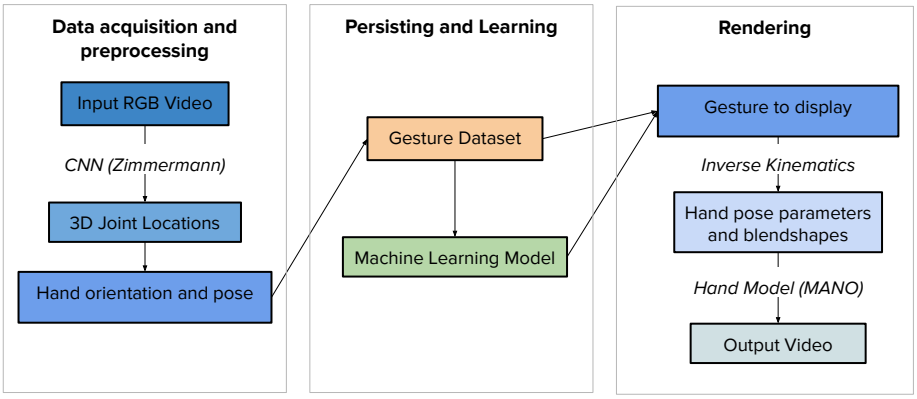

Figure 2: The general stages of our system. On the left, we convert video footage to a kinematic pose sequence. In the middle, we store and learn gestures. On the right, we pose a model of the hand and render images. The coloured rectangles represent distinct representations of the same hand gestures.

We additionally experimented with extracting angles between pairs of connected bones, and while this was sufficient for basic machine learning, it presented significant challenges with recovering the hand pose at rendering time due to degeneracies and lost information.

3.3 Learning Gesture Dynamics

We treat gestures as dynamical systems: a gesture's dynamics function determines the gesture-specific trajectory of the states of the hand "character". In other words, the task of making a gesture is equivalent to animating the hand with some dynamic force.

More formally (adopting CPSC 535P notation), we let $y_t \in \mathbb{R}^d$ and $y_{t+1} \in \mathbb{R}^d$ be the states of the hand character at time t and $t+1$, respectively (d is the number of features representing a hand state). A gesture is then a trajectory of hand states $(y_1, y_2, \ldots, y_T)$ where T is the number of timesteps required to reach the terminal state. Imposing a Markov assumption on the state transitions of this system, the dynamics function $f_g : \mathbb{R}^d \to \mathbb{R}^d$ for a gesture g satisfies:

$$y_{t+1} = y_t + f_g(y_t) \tag{6}$$

This is analogous to Euler's method involving position and velocity, except that we do not take acceleration or mass into account. The function f_g can be viewed as a compact and unique representation of the motion of gesture g: the temporal constraints, or the "allowed flow", of the hand state is determined by the behavior of f_g. Even if two gestures have the same starting state and terminal state, the path within the state space connecting the states in the trajectory will differ, and the behavior of the function f_g must encapsulate this difference.

Our task is to learn a function $\hat{f}_g$ that best approximates the function f_g given state trajectory data. We use Gaussian radial basis function (RBF) kernel regression with L2-regularization to minimize the objective function:

$$\mathcal{L} = \frac{1}{2} \sum_{\langle y_t, y_{t+1} \rangle} || \left[y_t + f_g(y_t) \right] - y_{t+1} ||^2 + \frac{\lambda}{2} ||v||^2 \tag{7}$$

where v is the parameters of the RBF kernel regression. We use the KernelRidge module available in scikit-learn [15].

3.4 Synthesizing Gestures

Once the learned dynamics model $\hat{f}_g$ is prepared, following $\hat{f}_g$ from some starting point $\tilde{y}_1$ produces a synthetic trajectory representing an instance of the gesture g, i.e. $(\tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_T)$, where T is the desired number of timesteps. We sample a starting point $\tilde{y}_1$ by selecting from the first states among the videos and adding a Gaussian noise to each feature, i.e.

$$\tilde{y}_1 = y_1 + \epsilon \tag{8}$$

$$\epsilon \sim \mathcal{N}(0, \sigma^2 I) \tag{9}$$

where $\epsilon \in \mathbb{R}^d$ is a noise sampled from a multivariate normal $\mathcal{N}$, σ^2 is a noise variance, and I is the $d \times d$ identity matrix. Once the starting point is chosen, we simulate the dynamics forward:

$$\tilde{y}_{t+1} = \tilde{y}_t + \hat{f}_g(\tilde{y}_t)\delta, \qquad t = 1, 2, \ldots, T-1 \tag{10}$$

where δ is the "speed" parameter. A high value of δ will make the system arrive at the terminal state faster, albeit with less stability, while a small value of δ achieves the opposite. This ability to control the speed of the synthesized gestures is a valuable consequence of the learned dynamics: even with the fixed frame rate of input video, using the learned dynamics allows us to interpolate the speed, hence more diverse instances of synthesized gestures.

3.5 Rendering Gestures

We use the MANO hand model [17] to render images of hands to aid our understanding of our motion capture and synthesis pipeline. This model is convenient for our purposes because it captures pose-dependent shape deformations and was easy to implement within our existing code base. Posing the model was a serious challenge however, because the MANO and Zimmermann hand models use very different conventions. While both represent the hand as a kinematic tree with the root located at the wrist, they differ in the number of joints, the presence of fingertips, and the sizes of homologous bones. This means that we are unable to directly apply

a Zimmermann hand pose to the MANO model because there is no well-defined conversion between the two. To get around this, we use inverse kinematics to automatically find a MANO pose that optimally matches a given Zimmerman pose. We use a two-stage optimization to account for the different hand shapes, and this optimization is performed independently for all frames of an animation.

First, we optimize the overall orientation and scale of the MANO hand in order to bring both hands into a similar orientation. This is achieved by moving the roots of both hands to the origin, then finding a combination of uniform scaling and rotation that brings the four knuckles of the MANO hand as close as possible to the corresponding four knuckles of the desired Zimmerman pose.

Secondly, we fix the scale and orientation of the MANO hand and find the pose parameters that maximize the colinearity of homologous bones between the MANO hand and the desired Zimmermann pose. We maximize the colinearity of the bones instead of minimizing the position error between joints, because this latter approach was found to cause the MANO model to enter extreme and unpleasant poses, particularly at the fingertips, whose poses it extrapolates. We acknowledge that this decision potentially introduces some distortion, and we suspect it is the cause for the misaligned thumb and pointer fingertips in the rendered *OK* gesture as seen in the results section.

4 IMPLEMENTATION

Our system is written in Python, as are the Zimmermann [23] and MANO [17] implementations we rely upon. Our source code is available online at the following URL:
https://github.com/namheegordonkim/smash-g.

We accelerate the performance of the Zimmermann hand pose estimator by offloading its work to a remote computer with a GPU. Input video frames are transmitted through the TCP protocol to a dedicated server machine, which evaluates the neural network and responds with 3D joint locations. This allows our pose prediction to run at interactive rates on devices with lesser hardware, such as a small laptop.

To perform inverse kinematics using the MANO hand model, we make use of the automatic differentiation facilities its implementation comes with to define simple loss functions which can be minimized automatically with respect to a custom set of variables. These features, along with the prevalent use of PCA subspaces to indirectly modify the hand, added significant difficulty to trying to pose the model with joint angles directly. We were surprised to find that the minimization routine provided by the implementation did not minimize its given loss function, but rather made it as close to zero as possible. This problem was solved by adding extremely large constants to our loss function.

5 RESULTS

5.1 Visualizing Learned Dynamics

For each of the 6 gestures we selected, we learn the dynamics as described in the Method section. We visualize each dynamics model as a vector field by creating a grid of points in a 2-dimensional PCA subspace, decoding the PCA data points to the original feature space, predicting the next states with the model, and encoding the next state predictions into the PCA subspace.

More formally, we first fit a PCA model with all of the video frames in the dataset. Let X be the $n \times d$ matrix representing the d rotation-invariant features across n frames stacked vertically. Then PCA factorizes X as

$$X = ZW \tag{11}$$

where $Z = \left[z^1, z^2, \cdots, z^k\right] \in \mathbb{R}^{n \times k}$ is the coefficients of k principal components and W is the conjugate transpose of principal component loadings. Assuming that the coefficients are sorted in the decreasing order of importance, we use z^1, z^2, i.e. the first two columns of Z, as the features for visualization.

We collect a 2D grid of z^1 and z^2 by generating vectors of 50 evenly spaced numbers within the range of data and taking a Cartesian product. For each of the resulting tuples $\left(z_i^1, z_i^2\right), \quad i = 1, 2, \cdots, 2500$, we decode into the original space by reconstructing:

$$\tilde{x}_i = \begin{bmatrix} z_i^1 & z_i^2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \tag{12}$$

where $w_1, w_2 \in \mathbb{R}^d$ are the first two rows of W. We use the resulting data point in the original feature space $\tilde{x}_i \in \mathbb{R}^d$ to predict the next state corresponding to the input. To this end, we first set $\tilde{y}_t = \tilde{x}_i$ and compute the prediction

$$\tilde{y}_{t+1} = \tilde{y}_t + \hat{f}_g(\tilde{y}_t)\delta. \tag{13}$$

We project $\tilde{y}_{t+1}$ back to the PCA space:

$$\tilde{z}_i = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \tilde{y}_{t+1}, \tag{14}$$

and then finally visualize $\tilde{z}_i - z_i$ as vectors in the PCA subspace.

The resulting vector fields are shown in Figure 3. We remark that each gesture dynamics has a characteristic "flow": the starting state and the terminal state of the hand, as well as the paths connecting between the two, are clearly defining the overall motion corresponding to the gesture label.

5.2 Gesture Synthesis and Analysis

After learning the dynamics, we generate synthetic hand state trajectories by using the dynamics models as described in Section 3.4. Figure 4 shows an example of 5 different synthetic trajectories corresponding to the gesture label *OK* with varying starting points. Examining the feature sequence plot, we detect that there are two distinct modes in the way the features evolve over time. Rendering the synthesized sequences, we get reasonable animated results where the middle finger, the ring finger, and the pinky are extended over time, while the thumb and the index finger stay bent.

We are able to synthesize trajectories with varying values of the speed parameter δ. This introduces an interesting trade-off between the speed and "smoothness" of the synthesized gestures. In Figure 5, a larger value of δ results in a fast transition from the starting state to the end state for the gesture *paper* while introducing jitters in the feature sequence. On the other hand, a smaller value of δ results in a more stable trajectory. This finding is analogous to the step size-stability trade-off often observed in Euler methods, and we remark that we are able to control this trade-off having

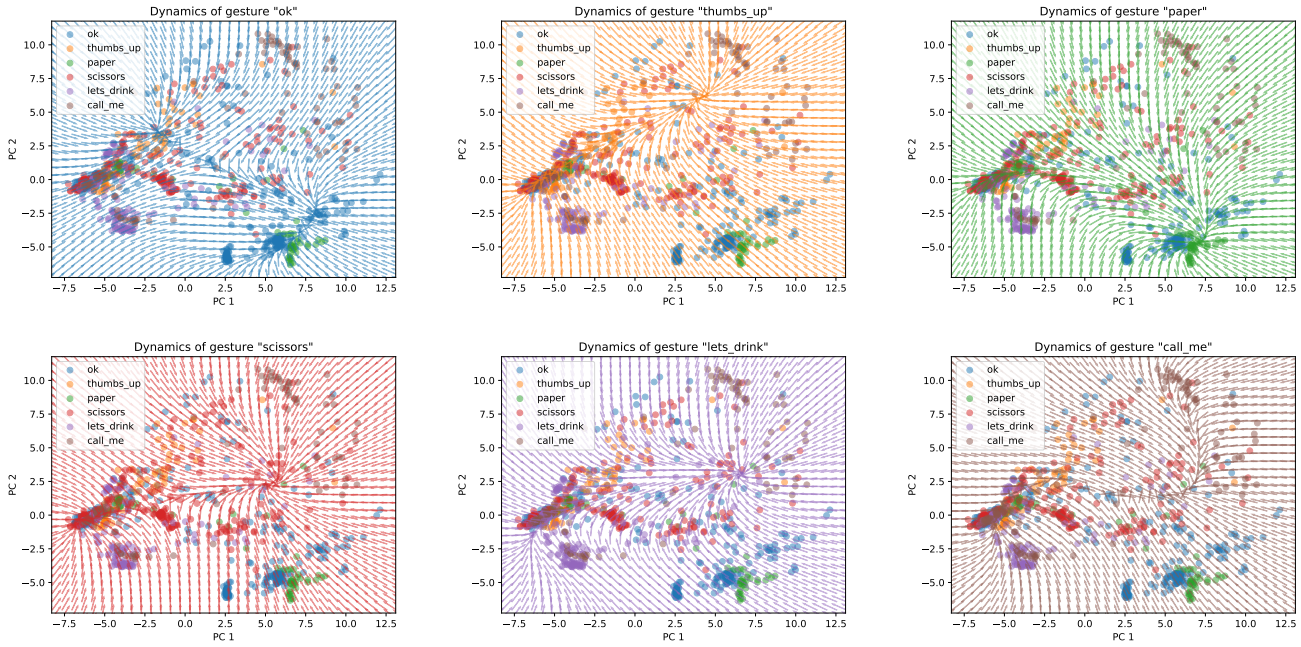

Figure 3: Learned dynamics of each function visualized as a vector field in each figure. The x and y axes are PCA coefficients corresponding to the first two principal components. Best viewed in color.

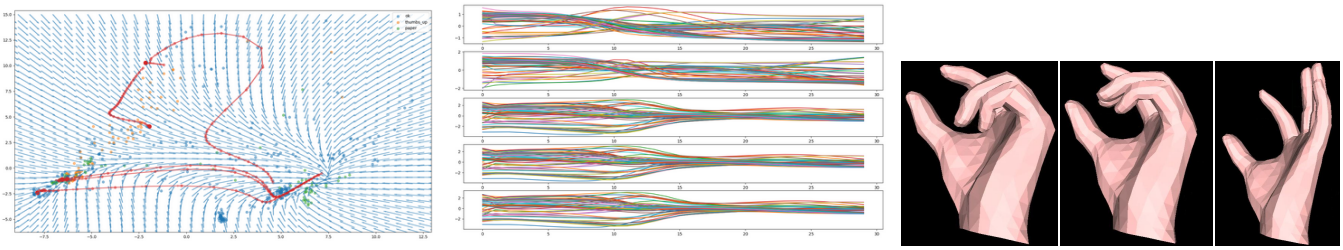

Figure 4: Results of synthesizing 5 different examples of *OK*. Left: the red arrows show the result of initializing at some start point and following the dynamics for 30 timesteps (visualized in the PCA subspace). Middle: the feature sequence of the 5 synthesized *OK* examples. Right: rendering the synthesized *OK* gestures show an animation that resembles the input data.

learned the dynamics purely from data. The qualitative results of the synthesized gestures with varying δ are available on the project website.

6 CONCLUSION

We presented SMASH-G, which takes input videos from a monocular RGB webcam and displays a 3D hand model making gestures synthesized based on the input. Although the results are primitive, our work provides a proof of concept that vision-based techniques with inverse kinematics can be successfully applied to animate digital representations of the human hand with gestures. Even with an extremely small amount of data, the RBF kernel-based learned dynamics is able to synthesize visually coherent sequences, showing that data-driven animation is a viable alternative to synthesizing motion manually.

6.1 Limitations and Future Work

We earlier noted that the upstream model published by Zimmermann et al. [23] 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 made the result of rendering less visually appealing. Replacing the model with a more performant model, such as GANerated hands [13] or more recent hand pose estimators could alleviate this issue.

We also note that the inverse kinematics method we used is computationally expensive. Our processing rate was around 8 seconds per frame, making our system infeasible for real-time applications. However, we note that the inverse kinematics can be applied in parallel in some contexts, which can boost the speed of computation.

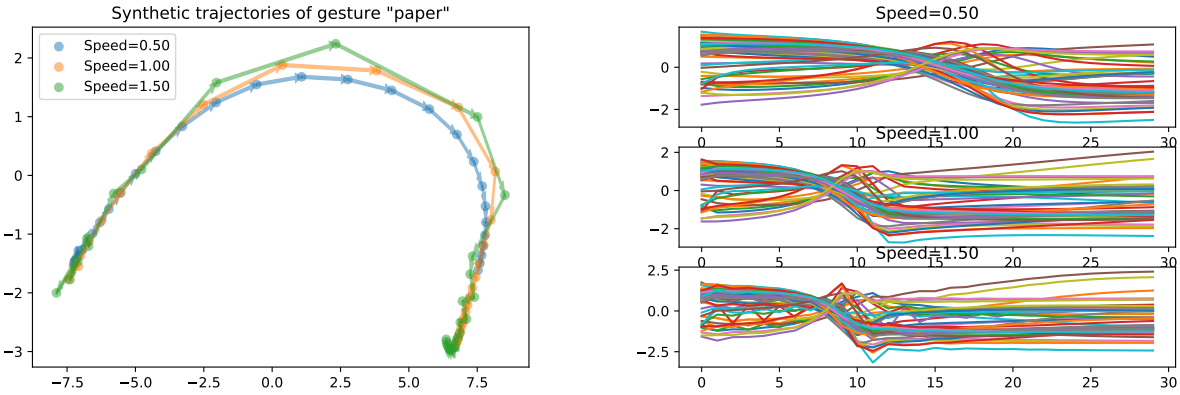

Figure 5: Left: 3 trajectories (visualized in the PCA subspace) of the gesture *paper* from similar starting points but different values of the speed parameter δ are synthesized with the learned dynamics. Right: corresponding feature sequences show that $\delta = 0.5$ has the smoothest transition although it is slower.

Given the experimental results showing the models' ability to interpolate and synthesize, the learned dynamics approach points to some promising directions. While the Markov assumption on the hand gesture is not thoroughly justified, using positional features with a Markovian dynamics has shown to be able to synthesize many of our gestures even with an extremely small amount of training data. One may improve the performance of the learned dynamics by including velocities as well as root poses into the representation of hand states. One interesting extension of our work is whether the learned dynamics can be used interchangeably to produce a sequence of gestures (e.g. rock-scissors-paper) or even transitions between gestures mid-action. Also, it would be interesting to see whether the learned dynamics models can act as motion primitives that mix together to create more complex gestures.

7 ACKNOWLEDGMENTS

The majority of this work was performed by both authors working together closely. Otherwise, additional credit is owed to Nam for his work on gesture synthesis, while Tim contributed extra work to the inverse kinematics and rendering. We thank Professor Dinesh Pai for helpful discussions and supervision through the CPSC 535P course.

REFERENCES

[1] Luca Ballan, Aparna Taneja, Jürgen Gall, Luc Van Gool, and Marc Pollefeys. 2012. Motion capture of hands in action using discriminative salient points. In *European Conference on Computer Vision*. Springer, 640–653.

[2] George ElKoura and Karan Singh. 2003. Handrix: animating the human hand. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association, 110–119.

[3] Doug Epps and Nate Reid. 2012. Inverse kinematics for motion-capture characters. US Patent 8,334,872.

[4] 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. arXiv:cs.CV/1903.00812

[5] Andrew Goldenberg, Beno Benhabib, and Robert Fenton. 1985. A complete generalized solution to the inverse kinematics of robots. *IEEE Journal on Robotics and Automation* 1, 1 (1985), 14–20.

[6] Micha Hersch, Florent Guenter, Sylvain Calinon, and Aude Billard. 2008. Dynamical system modulation for robot learning via kinesthetic demonstrations. *IEEE Transactions on Robotics* 24, 6 (2008), 1463–1467.

[7] J Klamka. 2013. Controllability of dynamical systems. A survey. *Bulletin of the Polish Academy of Sciences: Technical Sciences* 61, 2 (2013), 335–342.

[8] John P Lewis, Matt Cordner, and Nickson Fong. 2000. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 165–172.

[9] 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.

[10] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. 2015. SMPL: A skinned multi-person linear model. *ACM transactions on graphics (TOG)* 34, 6 (2015), 248.

[11] Laurent Moccozet. 1996. *Hand modeling and animation for virtual humans*. Ph.D. Dissertation.

[12] Thomas B Moeslund and Erik Granum. 2001. A survey of computer vision-based human motion capture. *Computer vision and image understanding* 81, 3 (2001), 231–268.

[13] 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/

[14] Richard P Paul and Bruce Shimano. 1979. Kinematic control equations for simple manipulators. In *1978 IEEE Conference on Decision and Control including the 17th Symposium on Adaptive Processes*. IEEE, 1398–1406.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[16] Lawrence R Rabiner and Biing-Hwang Juang. 1986. An introduction to hidden Markov models. *ieee assp magazine* 3, 1 (1986), 4–16.

[17] Javier Romero, Dimitrios Tzionas, and Michael J. Black. 2017. Embodied Hands: Modeling and Capturing Hands and Bodies Together. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)* 36, 6 (Nov. 2017).

[18] Matthias Schröder, Jonathan Maycock, and Mario Botsch. 2015. Reduced marker layouts for optical motion capture of hands. In *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games*. ACM, 7–16.

[19] 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.

[20] Srinath Sridhar, Antti Oulasvirta, and Christian Theobalt. 2013. Interactive markerless articulated hand motion tracking using RGB and depth data. In *Proceedings of the IEEE international conference on computer vision*. 2456–2463.

[21] Dimitrios Tzionas, Luca Ballan, Abhilash Srikantha, Pablo Aponte, Marc Pollefeys, and Juergen Gall. 2016. Capturing hands in action using discriminative salient points and physics simulation. *International Journal of Computer Vision* 118, 2 (2016), 172–193.

[22] 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.

[23] Christian Zimmermann and Thomas Brox. 2017. Learning to Estimate 3D Hand Pose from Single RGB Images. In *IEEE International Conference on Computer Vision (ICCV)*. https://lmb.informatik.uni-freiburg.de/projects/hand3d/ https://arxiv.org/abs/1705.01389.