Pose2Pose: Pose Selection and Transfer for 2D Character Animation

Nora S Willett
Princeton University
Pixar Animation Studios

Hijung Valentina Shin
Adobe Research

Zeyu Jin
Adobe Research

Wilmot Li
Adobe Research

Adam Finkelstein
Princeton University

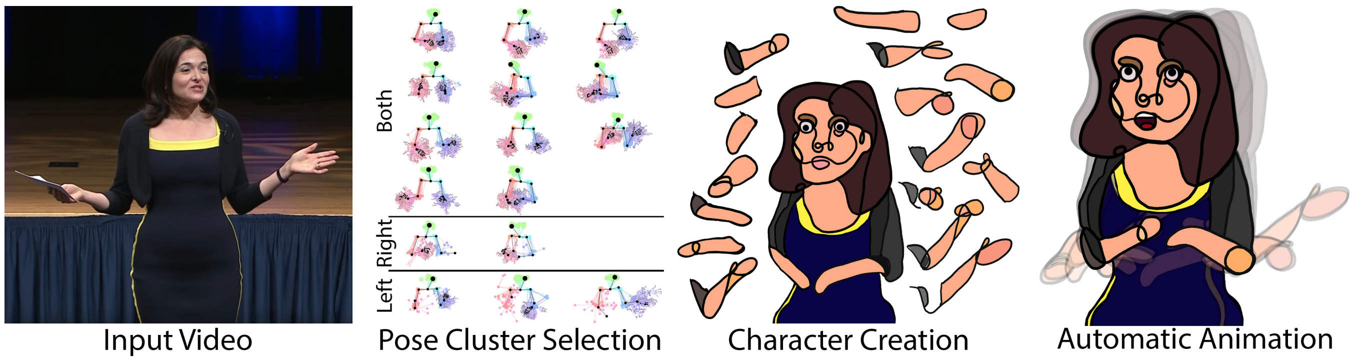

Figure 1: An example of selecting poses and transfering them to a 2D character. From an input video, we track the performer's poses and cluster them. With our UI, an artist selects pose clusters and uses those clusters as reference for creating a character. Then, our algorithm automatically drives an animation using pose data from a new performance video.

ABSTRACT

An artist faces two challenges when creating a 2D animated character to mimic a specific human performance. First, the artist must design and draw a collection of artwork depicting portions of the character in a suitable set of poses, for example arm and hand poses that can be selected and combined to express the range of gestures typical for that person. Next, to depict a specific performance, the artist must select and position the appropriate set of artwork at each moment of the animation. This paper presents a system that addresses these challenges by leveraging video of the target human performer. Our system tracks arm and hand poses in an example video of the target. The UI displays clusters of these poses to help artists select representative poses that capture the actor's style and personality. From this mapping of pose data to character artwork, our system can generate an animation from a new performance video. It relies on a dynamic programming algorithm to optimize for smooth animations that match the poses found in the video. Artists used our system to create four 2D characters and were pleased with the final automatically animated results. We also describe additional applications addressing audio-driven or text-based animations.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
IUI '20, March 17–20, 2020, Cagliari, Italy
© 2020 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-7118-6/20/03…$15.00
https://doi.org/10.1145/3377325.3377505

CCS CONCEPTS

• **Human-centered computing** **Gestural input**; **Systems and tools for interaction design**.

KEYWORDS

Pose selection; animation; 2D character creation.

ACM Reference Format:
Nora S Willett, Hijung Valentina Shin, Zeyu Jin, Wilmot Li, and Adam Finkelstein. 2020. Pose2Pose: Pose Selection and Transfer for 2D Character Animation. In *25th International Conference on Intelligent User Interfaces (IUI '20), March 17–20, 2020, Cagliari, Italy.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3377325.3377505

1 INTRODUCTION

Designing the motion of 2D or 3D characters is a critical part of creating animated stories. Relative to traditional keyframe-based workflows, performed animation provides a more convenient way to specify how characters move. Instead of authoring individual frames or specifying motion curves, an actor simply demonstrates the desired movements, which are acquired via video or motion capture [36] and transferred to the character. For most 3D characters, the motion transfer is relatively straight forward. Each joint on the actor typically corresponds to a matching joint on the character, which allows the continuous motion of the performer to be mapped directly to the 3D character producing convincing, nuanced animations. As a result, performance animation is used for a wide range of 3D animation scenarios, from feature films to games to special effects.

In contrast to 3D animation, this paper focuses on a 2D style traditionally called "cutout animation." Characters are represented by layered artwork that is animated through a combination of both continuous deformations (e.g. an arm bending at the elbow) and discrete transitions (e.g. a finger point replaced with a thumbs-up, or an open arm pose replaced with crossed arms) [66]. This style is prevalent in popular TV shows, such as South Park, The Late Show [19], Our Cartoon President [54], and streaming content like Twitch [47] and Final Space Live [28] that include animated avatars. Cutout animation has also been the subject of recent graphics and HCI research [60, 66–68]. While traditional cutout animation relies on keyframes, newly emergent approaches use human performance to drive animations [19, 67]. Some existing performance-driven tools transfer face/head motions to 2D characters [1, 10, 50, 68] or convert audio input to mouth animations [18].

However, no prior 2D animation system supports automatic triggering of arm/hand poses through human performance which significantly increases the expressiveness of performance-driven 2D animation. Directly applying performance animation to 2D characters is more challenging than the 3D case because of a mismatch between the degrees of freedom of the actor and character. While the continuous motion of an actor can theoretically be projected into 2D and then mapped directly to continuous deformations of the relevant layers, such deformations produce awkward results for all but the smallest changes in pose. These continuous 2D deformations can produce subtle movements, as shown in our results and [22, 46], but many pose transitions involve larger changes such as a finger point to thumbs up or crossing arms. As a result, when generating 2D animations by hand, animators often combine continuous motion with discrete artwork swaps that represent larger pose changes [52]. Applying this technique in a 2D performance animation workflow involves two key challenges: 1) at character design time, the artist must select (and draw) a representative set of poses that cover the desired range of motions for the character; and 2) at animation time, the system must carefully time the discrete transitions between the representative poses based on the movements of the actor.

To address these challenges, we propose a new approach to 2D performance animation that facilitates the design and animation of 2D characters from reference videos (Figure 1). We start with a *training video* of an actor demonstrating the personality and typical movements of the character. From this video, we analyze the actor's poses and provide a *pose selection interface* that helps an artist browse and select a set of representative poses to draw. Finally, given a 2D character with representative poses and a *performance video* where the actor demonstrates the desired character motion, our system automatically generates an animation that transitions between the appropriate poses via dynamic programming. Since hand and arm motions are often the most expressive aspects a performance, our current implementation supports the transfer of upper body movements to 2D characters.

An important element of our approach is the use of a training video during the character design process. Not only does this video help artists select representative poses to draw, it also allows our system to acquire a mapping from the drawn poses to a set of corresponding tracked poses from the training video. We leverage this mapping to automatically trigger pose transitions based on new performances and to add subtle continuous motion to the resulting animations based on the pose variations in the training video.

To evaluate our approach, we recruited four artists to design and animate 2D characters using our system. All of the animation examples in our submission were generated based on their designs. We collected qualitative feedback on their experiences, and their responses indicate that our pose selection interface helped them understand the variety of movements in the training video and decide what representative poses to draw. In addition, we evaluated the quality of our synthesized animations against manually-authored results created by a professional 2D animator and the output of two variations of our animation algorithm. Finally, we demonstrate two additional applications that our approach supports: audio-driven and text-based animation.

2 RELATED WORK

Prior work focuses on 3D and 2D performed animations as well as creating 2D characters.

3D Performed Animation: Many existing techniques have been proposed for generating 3D animation from performances. The most common method is through motion capture [36] to control all parts of the character. In addition, researchers have explored other ways to synthesize character motion, through analyzing and classifying performed gestures [49], directly sketching limb movements [57], and generating gestures from audio features [5, 13, 29, 30, 32, 41, 58] and text [8]. Our work focuses on animating 2D drawn characters rather than 3D models. Hence, we require different techniques for mapping human performances to animation.

2D Performed Animation: Prior work in 2D performed animation presents several different techniques for controlling 2D artwork. One simple example is mapping the mouth of 2D characters to a hand opening and closing creating a new version of a sock puppet with the mixed reality app YoPuppet [23]. Another way is to manually trigger artwork during a performance [67]. Numerous systems [1, 10, 50] and previous research [18, 68] focus on automatic head and mouth animations. Template-based deformation systems allow for full body movement but are limited to small deformations of the original template and can not handle larger changes in appearance such as an open hand to a closed fist [22, 46, 53]. In addition, there are a variety of methods to animate 2D characters through other techniques [3, 25, 26, 59]. In contrast, we are interested in handling large pose changes while automatically animating a 2D character from an upper body performance.

2D Character Creation: Our 2D characters consist of a multi-layer representation that defines the relationships between layers. This representation has been used in previous research and commercial 2D animation systems, including [66] and [1]. In most workflows, artists design and create layered 2D characters by drawing from scratch [20] or by reusing existing static images. Fan et al. enable the creation of 2D layered characters from multiple still images of the character [11]. In contrast, we provide a user interface that leverages an input video to assist artists with the creation of animated characters.

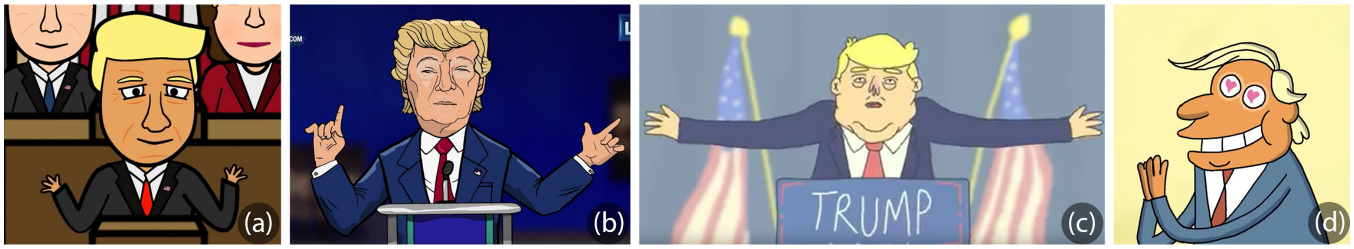

Figure 2: Caricatures of Donald Trump from 2D animations. (a) Cartoon Trump giving the State of the Union address. [34]. (b) Cartoon Trump participating in the GOP debates [9]. (c) Cartoon Trump at a rally [62]. (d) Cartoon Trump watching TV [38].

3 CHALLENGES

Animating a 2D character through performance involves multiple challenges at different stages of the process.

Capturing Performances: The first challenge is how to capture performances. Most motion capture systems require specialized hardware, like UV cameras and tracking markers, that may be inaccessible or inconvenient for many artists. As a result, we record performances with monocular video and extract the pose of the performer through markerless, vision-based pose tracking. This approach enables artists to easily capture performances using standard smart phones and digital cameras, or to leverage the plethora of videos on streaming platforms such as YouTube.

Designing Characters: Another challenge for the artist is how to encapsulate the personality of a character. When caricaturing a person for an animation, artistic visions vary widely. Every artist chooses different aspects of the character to emphasize. For instance in Figure 2, some artists emphasize Trump's hair (b,d), other's his mouth (c,d) or hands (a). Even with similar poses, such as the arms open, artists will interpret it differently (Figure 2a,b,c). While the design of a character's appearance is part of our workflow to create a character, we leave the style to the discretion of the artists.

Instead, the challenge that we focus on is the selection of a character's poses that best emphasize the artist's message. Identifying a key set of poses to draw is hard since the poses must be expressive enough to cover a range of scenarios and emotions. However, drawing these poses is labor intensive, so we want the set to be minimal as well. In addition, if the character is modeled after a performer, the artist may have to carefully watch the performer for some time to capture the right set of poses. Choosing a representative set of poses for an animated character is a subjective process motivated by the aesthetics and creative goals of the artist. Thus, while previous work on automatic pose selection [2] could provide alternatives to our specific pose clustering technique, our main goal is to design an interactive system that assists artists in the selection process. Our system addresses these problems by proposing a pose-centered video browsing interface that helps artists identify common poses in the input video. In principle, the artist could select poses simply by watching the videos linearly and marking specific frames, instead of using our clustering based interface. However, this manual workflow would be cumbersome given that the input videos in our experiments ranged from 26 to 49 minutes long and included extended periods where the actor was not visible. Moreover, even after finding the relevant parts of the footage, the artist may still need to re-watch sections to determine the most typical and distinct poses.

Generating Animations: When animating, another challenge is how to transfer human motion to a character. In the 3D world,

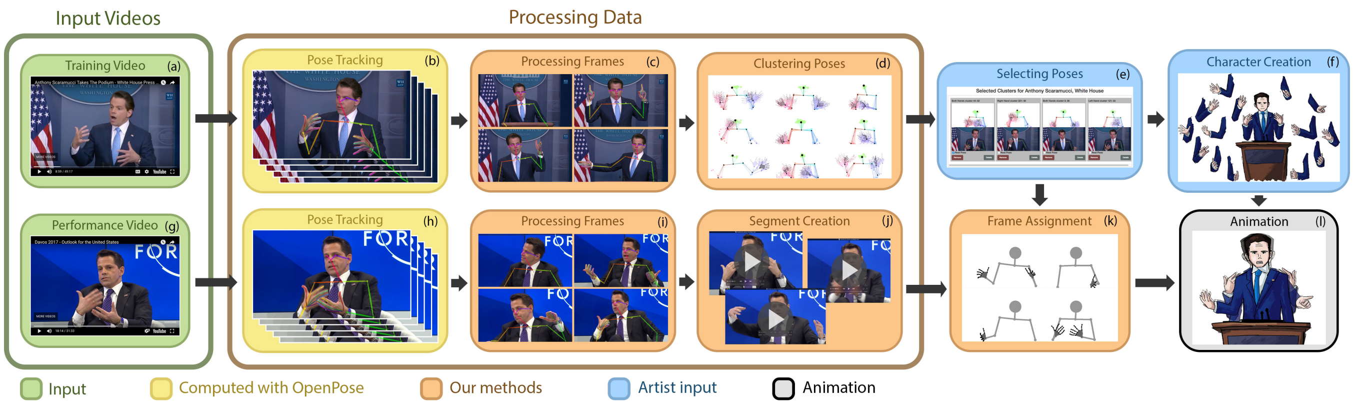

Figure 3: System pipeline. (a) The artist obtains an input training video. (b) Acquire pose tracking data from OpenPose [21]. (c) Process the frame pose data into *frame groups*. (d) Cluster the poses. (e) The artist selects some pose clusters. (f) The artist designs a character with the selected poses. (g) The artist finds another character input video of which parts will be animated. (h) Use Openpose on the new video. (i) Process the frames into *frame groups*. (j) Segment the video into smaller result parts. (k) Assign frames from the segments to one of the artist selected pose clusters based on the pose data. (l) Animate the final character.

motion capture is easily mapped to a 3D character model because human joints and their movement are directly transferable to the joints on the humanoid 3D character. However, when transferring the continuous input of joint movements to the discrete output of displaying a 2D artwork layer, there is more ambiguity due to a character's design.

For example, during the design process, an artist draws a character with a pose of wide open arms and hands. In addition, they draw artwork for clasping hands together showing the backs of the character's forearms with fingers intertwined. When animating, a human's pose of wide open arms and hands can be directly mapped to the character's first pose. With slight human movement, the 2D character can copy the movement through deformations. For instance, the human flaps their arms and the character's arms deform up and down as well. However, if the actor suddenly brings their hands down and clasps them, the question becomes at what point during the human movement do the separate artwork layers transition. In addition, the artist only draws artwork for a subset of all possible human movement. So then, we must determine which piece of artwork to switch to in order to best approximate the current human pose.

4 WORKFLOW

To address these challenges, we propose a new approach for designing and animating 2D characters (Figure 3). In the description below, we refer to the person designing the character as an artist, and the person animating the character as an actor or performer. While it is possible for the same user to carry out both of these tasks, we distinguish the roles to clarify the different stages of our proposed worfklow.

4.1 Designing Characters

The artist starts by obtaining a *training video* of a human performer demonstrating the range of poses for the character (Figure 3a). This video may be created by the actor who will eventually drive the final animation, or if the goal is to create an animated version of a celebrity, it may be possible to find existing recordings to use for training. To facilitate pose tracking, the performer in the video should mainly be facing the camera with their hands in full view as much as possible.

Next, the artist uses our *pose selection interface* to identify representative poses for the character based on the training video. After loading the video into our system, the artist sees a set of candidate poses extracted from the video (Figure 3e and 4). The system categorizes poses based on which hands are visible (*Both Hands*, *Left Hand*, or *Right Hand*) to help artists quickly browse related poses. In addition, since every pose generally corresponds to several frames in the video (i.e., poses represent *clusters* of similar video frames), our interface sorts the poses based on the cluster sizes, so that common poses are easy to identify. For each pose, the system shows a visualization of the extracted joint positions from up to 50 frames within the cluster and a looping sequence of actual video frames (represented as an animated gif) that map to the same pose. To explore a candidate pose in more detail, the artists clicks the *Details* button to see a new page showing gifs of several different video frame sequences associated with the pose, sorted based on their similarity to the "average" pose within the cluster.

Using this interface, the artist selects the desired number of representative poses for the character. By categorizing, sorting and visualizing poses as described above, our interface helps users focus on promising candidate poses without examining every individual pose cluster. However, the artist still ultimately has the freedom to pick whichever poses they decide will best capture the character they are designing. For most animations, it is useful to have a designated *rest pose* that the character returns to whenever the performer is not explicitly striking a pose. By default, the system labels the first selected pose as the rest pose, but the artist can override this choice.

Finally, after selecting representative poses, the artist creates a drawing for each pose cluster (Figure 3f). Our system associates these drawings with the corresponding poses.

4.2 Generating Animations

Generating a new animation of the 2D character requires a *performance video* where a human actor demonstrates the desired performance (Figure 3g). This video can either be recorded by the actor or obtained from existing footage, just like the training video. Our system automatically synthesizes an animation that moves through the representative poses based on the recorded performance (Figure 3k). While we create a simplified stick figure previz version of the animation (Figure 3k), the most compelling results are created by artists (Figure 3l and 9).

5 METHODS

Our approach includes automated algorithms for extracting pose data, clustering similar poses, and generating pose-to-pose animations. Here, we describe these methods and explain how they support the proposed workflow.

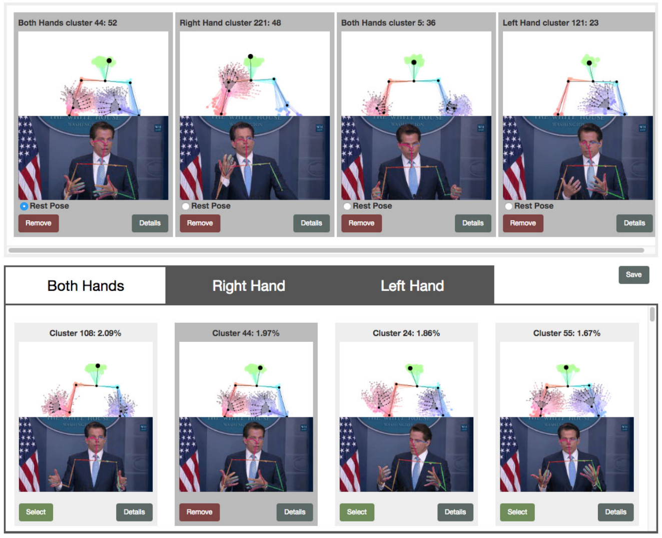

Figure 4: The user interface for selecting clusters. Top, the clusters selected by the artist. Bottom, all the pose clusters separated by hand type.

5.1 Extracting Poses

Our pose selection interface and performance-driven animation technique require analyzing the pose of a human actor in training and performance videos. Recent work on pose estimation from monocular video suggests that neural networks can be very effective for this task [7, 12, 37, 42]. We use the open source library of OpenPose [21], based on the publications of Simon et al. [55] and Wei et al. [65], to extract pose information for each video frame (Figure 3b). Since we focus on upper body performances, we only consider the (x, y) coordinates and confidence scores for the hand, arm and neck joints.

5.2 Processing Poses

Given that poses in consecutive frames are typically very similar, we segment videos into *frame groups* that represent distinct poses (Figure 3c). Operating on frame groups significantly reduces the computational cost of the clustering and frame assignment algorithms described below.

As a first step, we filter out frames with either no detected people or more than two people. For each frame containing one to two people, we select the one whose bounding box centroid is closest to the center of the image as the primary subject. We then compute a feature vector that encodes the upper body pose of the primary subject. The dimensionality of the vector depends on the number of hands that are visible in the frame (i.e., $\geq 75\%$ of the hand joints are within the image bounds). If both hands are visible and three or more finger tip joints on each hand are detected with high confidence (greater than 1%) the frame is a *Both Hands* pose. If there are three or more high confidence finger tips for only one visible hand, the frame is either a *Left Hand* or *Right Hand* pose. Otherwise, we label the frame as having *No Hands*. As noted earlier, our pose selection interface organizes the training video into these pose types to facilitate browsing.

We compute a pose feature vector for all *Both Hands*, *Left Hand*, and *Right Hand* frames. The feature vector includes the (x, y) values of each wrist joint relative to the neck joint and the distance from each finger tip to the corresponding wrist joint. The wrist positions capture gross arm movements, and the finger distances encode fine-grained information about the hand shape. We also experimented with other feature vectors that included individual finger joint angles, but these suffered from unreliable joint data, projection ambiguities, and the curse of dimensionality (i.e., too many degrees of freedom). We found that finger distances provided a compact representation with sufficient expressive power to distinguish between different hand poses.

Next, we normalize the feature vectors to account for camera changes or cuts that make the person bigger or smaller in relation to the frame. We estimate the scale of the subject in each frame by computing the inter-shoulder pixel distance relative to the frame diagonal and construct a histogram based on this measure, which represents the distribution of such scales across the video. To reduce noise, we smooth the distribution with a Gaussian filter ($\sigma = 5\text{px}$), as shown in Figure 5. Then, we identify sets of frames with approximately the same subject scale by splitting the histogram into sections with boundaries at the local minima of the smoothed counts. For each section, we compute a normalization factor by taking the median inter-shoulder distance and dividing it by the median distance for the entire video. We then normalize the feature vector for each frame with the corresponding normalization factor. Sections that have very low counts ($< 1\%$ of all frames) typically arise from quick zooms or cuts to different people or the subject turning to the side. We treat these as outliers and label them as *No Hands* poses.

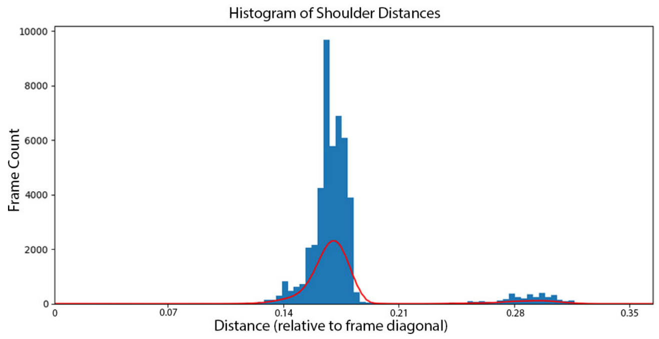

Figure 5: A histogram of shoulder pixel distances relative to the frame diagonal for normalizing the feature vectors. All frames with good hand data were counted. The red line is the smoothed distribution used to split the counts into sections. There is a camera shift represented by the gap of distances between 0.21 and 0.25.

Once the feature vectors are computed and normalized, we split the frames into *frame groups*, which are consecutive frames that represent a pose. To construct these groups, we visit each frame in order, adding it to the current frame group until either the hand type changes or the Euclidean distance between the current pose feature vector and the first one in the frame group is greater than a threshold of 60. If either of these conditions are met, we start a new group. We use the feature vector of the middle frame to represent each frame group.

5.3 Clustering Poses

Our pose selection interface presents artists with candidate poses that cover the range of motions in the training video. To compute these candidates, we cluster the pose feature vectors of all frame groups for each hand type into sets of similar poses (Figure 3d). We note that the notion of "similar" here is not well-defined. For instance, should a pose with open fingers and open straight arms be grouped with closed fists and open straight arms? Or should the closed fists and open straight arms be grouped instead with closed fists and slightly bent arms? Depending on the scenario, either grouping may be preferred. Since there is no one correct answer, our goal is to provide a reasonable clustering to help artists select representative poses when designing a character.

To this end, we experimented with various clustering methods (DBSCAN, MeanShift and Affinity Propagation) using a range of parameters and visualized the results using t-SNE [31]. We restricted our exploration to techniques that adaptively determine the total number of clusters. In the end, we found that sklearn's Affinity Propagation [17] (with damping set to 0.51) provided the best balance between too few and too many clusters. Figure 6 shows example clusters and some of the frame groups they contain. For our results, we had 150 to 300 clusters per character.

5.4 Mapping Performance to Animation

To generate an animation from a given performance video (Figure 3g), our system automatically maps the video frames to the representative poses of the character (Figure 3k). In addition, we introduce a simple technique to add subtle continuous motion to the animation based on the pose variations from the input training video.

5.4.1 Frame Assignment. As a first step, our system processes the performance video in the same manner as the training video to extract frame groups (Figure 3h,i). We then map the sequence of frame groups to a sequence of representative character poses via dynamic programming [4] (Figure 3k).

Assignment Energy: Our optimization solves for an assignment $\{a_1, a_2, \dots, a_n\}$ where n is the number of frame groups in the performance video, and a_i is the assigned pose cluster for frame group i that minimizes the following energy:

$$E(\{a_1, a_2, \dots, a_n\}) = \sum_{j=1}^{m \ll n} k_L \cdot E_L(a_j^*) + k_P \cdot E_P(a_j^*)$$

where a_j^* is the jth contiguous run of frame groups with the same held pose assignment, which we refer to as a segment; m is the number of segments; E_L is a length energy that prevents each segment from being too short; E_P is a pose energy that measures the similarity of an assigned pose to the set of tracked poses in the corresponding frame groups; and (k_L, k_P) are weights that trade off the importance of the two energy terms.

Length Energy: The length energy applies a quadratic drop off to penalize segments that are shorter than a threshold ϕ_L:

$$E_L(a_j^*) = \begin{cases} \left(\left(\dfrac{3n_j - \phi_L}{\phi_L}\right)\right)^2 & \text{if } n_j < max(c_L, \phi_L) \\ 0, & \text{otherwise} \end{cases}$$

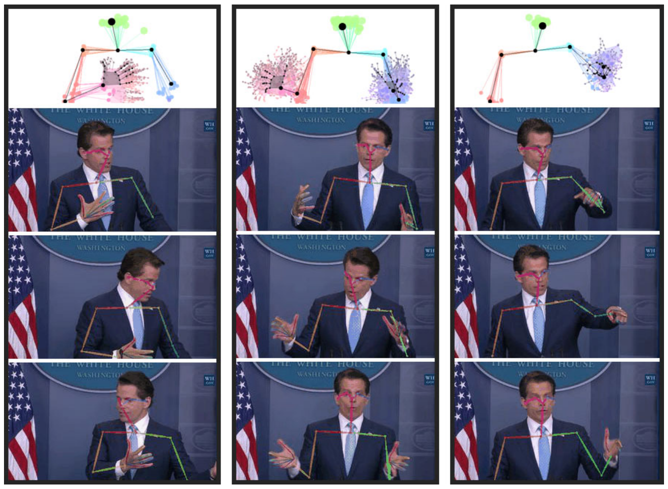

Figure 6: Three clusters (above) and three example frame groups from each cluster (below).

where n_j is the total number of frames in the segment, and c_L is the average length of all the training frame groups in the corresponding pose cluster.

Pose Energy: We define the pose energy of a segment as

$$E_P(a_j^*) = \sum_{i=0}^{n_j} d(p(a_j^*), p_{j,i})$$

where $p_{j,i}$ is the extracted pose from the ith video frame of the jth segment in the assignment; $p(a_j^*)$ is the pose closest to the assigned cluster's centroid; and $d(p_1, p_2)$ is the distance between two poses. We compute this pose distance based on the hand types for the two poses. If they match, then we use the Euclidean distance between the feature vectors. If they do not match, we set the pose distance to a constant k_D. If the ith video frame does not have a pose (no people or hands in the frame), then we use the Euclidean distance between the assigned pose cluster and the rest pose cluster.

Constants: Our frame assignment algorithm includes constants (k_L, k_P, k_D, ϕ_L) that control the relative strengths of the energies, how likely a pose is to switch to a different hand type, and how short segments should be. As we discuss in the Evaluation section, we searched over a range of potential parameter values and found that defaults of $k_L = 2, k_P = 1, k_D = 0.7, \phi_L = 15$ generally produced the best results.

5.4.2 Continuous Movement. The optimization above generates a sequence of character poses based on the performance video. To increase the expressiveness of the resulting animation, we add continuous movement to the character's hands for each held pose.

First, we determine the range of hand motions within a cluster from the training video poses that correspond to each held character pose. More specifically, we align all the cluster poses spatially based on the position of the neck joint so that they occupy a shared space and then compute the convex hull of the wrist joint positions (Figure 7). Then, we generate a randomized motion path for each hand by constructing a Bezier curve inside the convex hull. Starting at the wrist position for the center frame of the cluster, we add Bezier control points one at a time by translating along a randomly varying vector. At each iteration, we rotate the vector by a random angle between -90 and 90 degrees and scale the vector by a sine function whose amplitude is between 50% and 175% of the square root of the convex hull's area. If the resulting vector moves the control point outside the convex hull, we construct a new random vector until we obtain one that stays within the boundary. Finally, we deform the

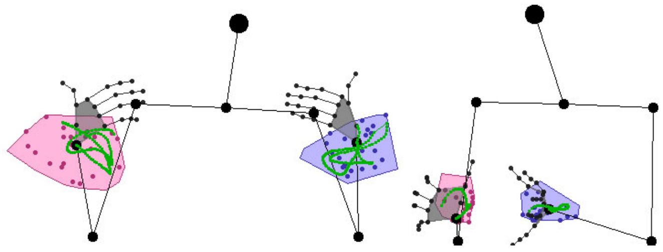

Figure 7: Visualization of continuous movement in clusters. The pink and purple areas are convex hulls. The green lines are motion paths.

artwork by moving a control handle positioned near the character's wrist along the motion path [66].

Note that we could also generate similar continuous movements by attempting to map hand motion from the performance video directly to the character poses. However, our technique allows us to generate subtle continuous movements even if the performer does not move their hands. Moreover, as discussed in the Other Applications section, our approach can be applied to synthesize continuous motion in settings where there is no explicit pose data driving the animation (e.g., using audio or text as input).

5.5 Implementation

Our system is implemented in Python and run on a MacBook Pro, 2.5 GHz Intel Core i7, 16GB DDR3. We use the numpy[1] and scikit-learn[2] libraries for processing and clustering algorithms. In addition, we use Adobe Character Animator as the animation engine for creating our results [1].

6 RESULTS

To showcase our method, we generated animations of four different characters that were designed based on famous personalities. Here, we outline the process for creating those results.

6.1 Input Videos

We searched for existing footage of individual subjects "performing" in front of a camera to use as training and performance videos. Broadcasts of presentations and speeches were a convenient source of data because speakers typically face the camera and move through a wide range of poses (some more so than others!). In the end, we selected footage of Anthony Scaramucci, Donald Trump, Hillary Clinton and Sheryl Sandberg. The videos of Scaramucci are from a White House Press Briefing (*ScarWH*, 49 minutes) [39] and the Davos Conference (*ScarDavos*, 31 minutes) [15]. Donald Trump's videos are from the Arab Islamic Summit (*TrumpAIS*, 34 minutes) [44] and the Davos Conference (*TrumpDavos*, 16 minutes) [16]. Hillary Clinton spoke at rallies in Nevada (*ClintonNV*, 26 minutes) [45] and Florida (*ClintonFL*, 24 minutes) [43]. The video of Sheryl Sandberg is from a presentation at the Pentagon (*SandPent*, 46 minutes) [63].

We used *ScarWH*, *TrumpAIS*, and *ClintonNV* as training videos and reserved the remaining footage to extract performance videos. *SandPent* was used as both the training and performance video. Figure 8 shows frames from the training videos which were processed into frame groups and then clustered. Table 1 shows the distribution of pose types in the training videos. In most cases, the speaker was the only person in the frame for the majority of the video. In all of the videos, both hands appeared in the frame more frequently than just a single hand.

[1]http://www.numpy.org
[2]https://scikit-learn.org/stable

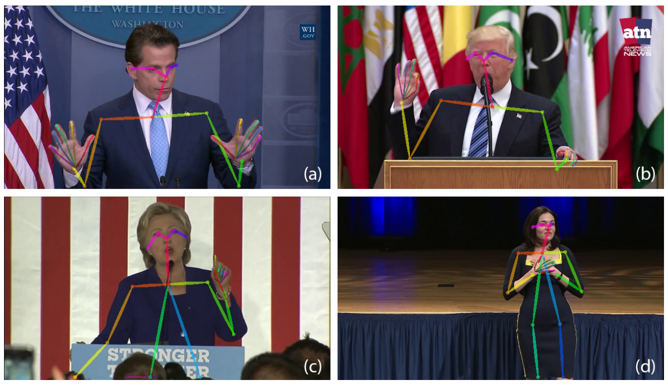

Figure 8: Pose tracked frames from training videos. (a) Anthony Scaramucci (*ScarWH*) [39]. (b) Donald Trump (*TrumpAIS*) [44]. (c) Hillary Clinton (*ClintonNV*) [45]. (d) Sheryl Sandberg (*SandPent*) [63].

	Scaramucci	Trump	Clinton	Sandberg
Too many or few people	23%	1%	1%	17%
No Hands	35%	8%	15%	7%
Both Hands	23% (6)	51% (7)	60% (7)	74% (11)
Left Hand	2% (1)	31% (0)	15% (1)	1% (3)
Right Hand	17% (3)	9% (1)	9% (1)	1% (2)

Table 1: Distribution of frame types in each training video and the number of artist selected representative poses in parentheses.

6.2 Character Creation

We asked four artists to use our system to create 2D characters. First, the artists chose representative poses for each character using our pose selection interface. Table 1 shows the distribution of selected poses across hand types. Then, the artists created the corresponding artwork for these poses (Figure 10). For Scaramucci and Sandberg, the artists created the artwork from scratch. For Trump and Clinton, the artists were given the source artwork for the cartoon versions of the politicians from *The Late Show with Stephen Colbert* to edit [19, 67]. Figure 9 shows the different styles of these characters.

6.3 Segment creation

To create performance videos for our results, we extracted short segments from the footage that we reserved for generating animations (Figure 3j). We considered 10 second clips that contained a single person, which is long enough to convey the personality of the character but short enough to prevent fatigue when viewed multiple times in our user evaluations, which we describe in the next section. In order to cover a range of animated results, we selected segments with different amounts of movement, which we estimate as the average standard deviation of the hand-related feature vector dimensions across all segment frames normalized by the standard deviation of all performance frames. To ensure that we can compute this movement score reliably, we only consider clips where 90% of the frames contain a valid hand pose. After computing the movement score for all 10 second clips in each reserved piece of footage, we discard those with scores in the bottom half since many such clips included almost no movement at all. From the remaining segments, we chose low, medium and high scoring segments that do not overlap by more than half of their duration to use as performance videos for generating animations.

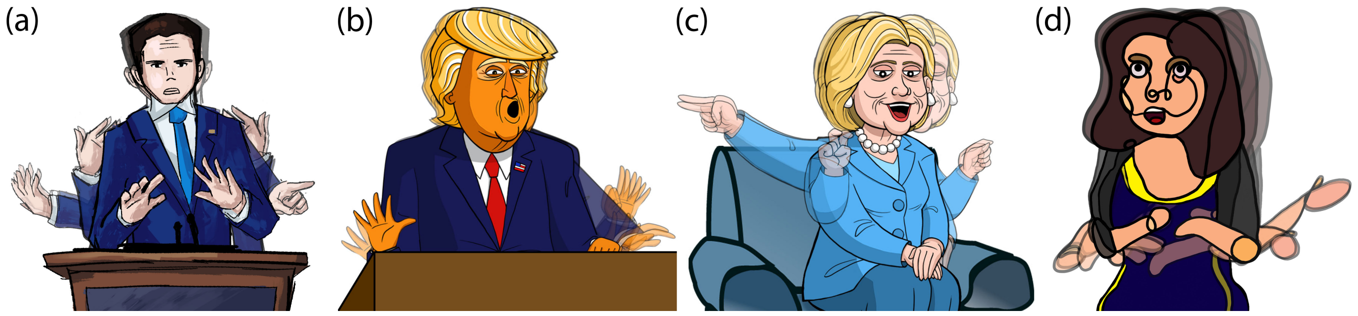

Figure 9: Characters created by artists. For animations and other details, see our supplemental materials and video. (a) Anthony Scaramucci ©Diano Lao. (b) Donald Trump*. (c) Hillary Clinton*. (d) Sheryl Sandberg ©Raya Ward. *Elements from Our Cartoon President - Courtesy of CBS Television Studios.

6.4 Final Animations

For each chosen segment, we generated a final animation using our dynamic programming optimization with the parameters $k_L = 2, k_P = 1, k_D = 0.7, \phi_L = 15$. For comparison, we created results with and without continuous hand movement. Please see our supplemental materials and video for the animations.

7 EVALUATION

To evaluate our system and assignment algorithm, we gathered qualitative feedback from the artists. In addition, we ran surveys exploring the assignment energy's parameters as well as comparing the final animations.

7.1 Artist Study

Four experienced artists, with backgrounds focusing on illustrations and painting, were recruited to draw the characters. First, we asked each artist to watch the character's training video. This process took between 10 to 30 minutes, with some artists just skimming through the video since they were already familiar with mannerisms of the person. Next, artists selected clusters with our UI (Figure 4). Three out of four artists took 15 minutes or less, and found the process "easy", "straightforward", "enjoyable" and "simple." Clusters were selected based on characteristics of the actor, uniqueness, and emphasis.

The following step for the artists was to design and draw the characters. When creating a character from scratch (Scaramucci and Sandberg), the artists took between five to twelve hours. They spent most of that time drawing arm and hand poses and found the cluster summary images helpful. However, artists indicated that drawing the face was also time consuming. When editing an existing character (Clinton and Trump), artists took between three to four hours. For those characters, the artists were able to reuse portions of existing drawings to match the pose clusters. While these artists did not iterate on their characters due to time constraints in our study, our system does allow for iteration when selecting poses and designing a character.

The artists also filled out a questionnaire regarding their opinions about their character's animations. Overall, they thought that the animations matched the characters' movements and the continuous motion added richness by smoothing the transitions between poses. Given the limited number of poses, they felt that the animations captured the personality of the character. While the pose animations matched performance video clips the best, one artist felt that the audio driven animations (introduced below) looked more natural. Additional areas for improvement were also mentioned by the artists. For some of the results, the animations were a little jittery and deformations made the arms and head look uncanny. Causing additional concern was when the animations transitioned between very different looking poses too quickly making the character appear robotic. If given the opportunity to revise their character, one artist wished to decouple the left and right hand pose drawings for more expressiveness. Two artists mentioned that they would create additional hand poses for more variation and smoothness.

7.2 Parameter Tuning

To determine the sensitivity of the animation quality to the different parameters in our energy function, we ran a survey varying the constants k_L, k_D, ϕ_L while keeping the pose energy weight k_P fixed at 1. For k_L, the length energy weight, we tested the values 0, 2, and 4. For k_D, we tested the values 0.5, 0.7, and 0.9. The ϕ_L values, which represent the minimum sequence length, consist of

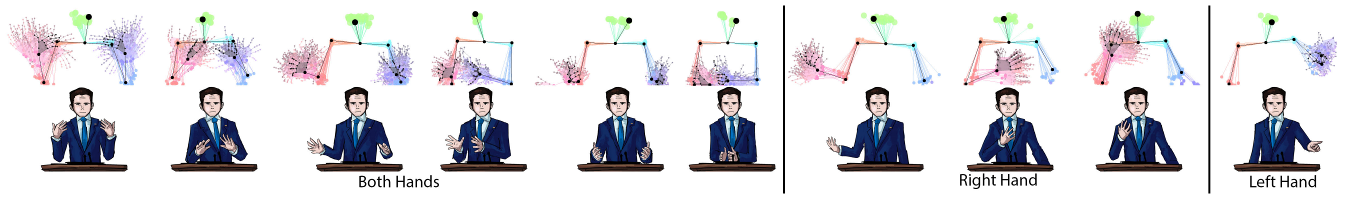

Figure 10: The artist selected pose clusters (top) and renderings (bottom) for Anthony Scaramucci. ©Diano Lao

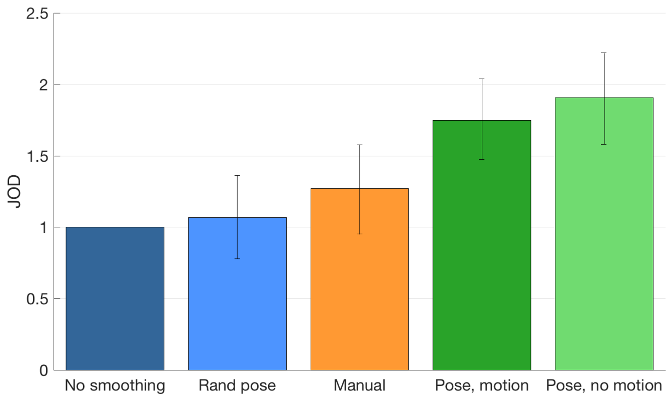

Figure 11: Comparing different types of animation. We determine the relative rankings from pairwise comparisions by using the Perez-Ortiz method [48]. They use an inverse cummulative normal distribution to map probabilities into JOD distances.

5, 15, and 30 frames. When changing one variable, the others were set to the default values of $k_L = 2$, $k_D = 0.7$, and $\phi_L = 15$. As a baseline for comparison, we included videos where the audio track and animation were generated from different performance video clips.

For the study, we showed the viewers three videos, the 10 second performance video clip and two animated results side-by-side. Since we wanted to understand the impact of the parameters independent of the artwork style for a given character, we generated all animations in a stick figure previz style (Figure 3k). We asked users "Which animation best characterizes the poses of the person in the original video?" Each user performed 24 comparisons, four of which were sentinels consisting of a constant pose for the whole 10 seconds. The comparisons rotate between characters so that the user does not see a character twice in a row. Within a character, the parameter comparisons and location on the screen (left or right) were randomized.

We ran the experiment using Amazon Mechanical Turk (AMT) [6] with 60 participants. We paid each participant one dollar for their work, and we collected 4 separate judgments for each individual comparison. All of our animations were preferred to the switched audio baseline. In addition, participants liked shorter sequence lengths (ϕ_L), resulting in more movement during the animation. However, they preferred animations with some smoothness measured by $k_L = 2, 4$ to those without it ($k_L = 0$). In addition, participants liked higher values of k_D which correspond to more closely matched hand pose types instead of switching to a different type.

7.3 Final Animations

To evaluate the quality of our results, we ran a similar study, with different participants, with animations using the artist created characters. We used our system to generate animations with (*Pose, motion*) and without (*Pose, no motion*) continuous hand motion. As baselines, we created a *No smoothing* result by setting the length energy weight k_L to zero and a *Random pose* result by generating a random value for the pose distance $d(p(a_j^*), p_{j,i})$ at every frame. These baselines did not have continuous motion. We also compared against manually authored results (*Manual*) from a professional animator who specified when to transition between representative poses based on the input performance video. The animator chose not to use continuous motion in their results.

To compare the animations to each other, we ran an AMT experiment with the same format as the parameter tuning study described above. In this study, we collected five separate judgments for each pair of animations. To analyze the pairwise comparisons, we used the Thurstone method [61] modified by Perez-Ortiz et al. [48] to determine the relative rankings of each animation type (Figure 11).

The findings suggest that our animations better characterized the poses in the video than either of the baselines or the manual versions. *Pose, no motion* was preferred with a probability of ~70% (0.85 and 0.9 JOD) over the *No smoothing* and *Random pose* baselines, and ~60% (0.6 JOD) over the *Manual* results. *Pose, no motion* was also preferred with a probability of ~55% (0.15) compared to *Pose, motion*. However, participants mentioned that they had trouble determining the subtle differences between the pose animations with and without motion.

8 OTHER APPLICATIONS

In addition to generating animations based on performance videos, our approach supports the synthesis of 2D animations from other inputs. We have implemented prototypes that use vocal performances and annotated scripts to drive animations.

8.1 Audio-driven Animations

Using speech to drive animation is a well studied topic [5, 13, 29, 30, 41, 58]. While all of these methods use various acoustics features to produce continuous 3D movements, we use similar features (prosody, intensity, timbre) to transition between the drawn poses of a 2D character.

We develop a prediction network that takes audio samples as input and generates a corresponding sequence of pose assignments. Once the artist has selected representative poses with our pose selection interface, we train the network with the audio track from the training video. By running our pose-driven animation synthesis algorithm on the training video, resulting in sequences of representative poses, we obtain ground truth output for the network. The network architecture is akin to the audio prosody prediction network presented by Wang et al. [64]. First, we extract acoustic features from the audio at each video frame. Three features are extracted using SPTK [14] with a window size of 1024: intensity in dB, a 13-coefficient MFCC [40] representing timbre information, and fundamental frequency (F0) representing the prosody. These features are concatenated together and serve as the input to the network.

The neural network consists of three convolutional layers and two LSTM layers. The first convolutional layer is a size-1 convolution of 128 channels that transforms the input features independently into 128 channels. The next two convolutional layers are of size 5 and 256 channels. For the LSTM layers, the first is a bi-directional LSTM that summarizes the input and the second is uni-directional to which a fully connected layer is attached at each time step to produce logits for the final prediction. The logits are passed through a softmax layer to output the posterior distribution for each representative pose. A dropout rate of 20% is added to the LSTM to prevent overfitting. To train the network, we randomly extract 8–12s training video segments and use the audio track and corresponding representative poses as an input-output pair. In addition, an Adam optimizer with a learning rate of 0.001 is used [27]. After the network is fully trained with about 50 epoches, we use it to generate pose probabilities for a new performance audio clip.

For a direct comparison, we split the new audio into "frames" based on the training video's frame rate. To assign a frame to a representative pose, we apply our dynamic programming algorithm using the pose probabilities from the audio network (which we refer to as the *audio energy*) in place of the pose energy in our pose-driven technique. In particular, we expand our previously defined assignment energy by adding $k_A \cdot E_A(a_j^*)$, where E_A is the audio energy and k_A is the corresponding weight. We define the audio energy as:

$$E_A(a_j^*) = \sum_{i=0}^{n_j} 1 - P_i(a_j^*)$$

where $P_i(a_j^*)$ is the probability from the network of assigning the ith frame of the jth segment. When using audio to create an animation, we set $k_L = 2, k_P = 0, k_A = 1$, and $\phi_L = 15$. The final animations are available in our supplemental materials.

In addition to acoustic features, we also experimented with linguistic features. We used a speech recognizer to extract words and timings from the audio. For each training video frame, we compute the frame position inside a word and the word position inside the sentence. For example, if a frame is the 3rd frame of a word that lasts 10 frames long, its frame position is 0.3. We also add a binary feature to represent whether there is vocal activity at a frame. To characterize the functionality of a word, we use Google's word2vec [35] embedding to transform a word into a real-valued vector. Then, we concatenated the position information and the word embedding together to use as the network input. We ran some preliminary experiments comparing animations generated with the acoustic versus linguistic features and found that the acoustic features generally produced better results.

8.2 Text-based Animations

Another way to animate characters is to associate words in a script with specific motions, as demonstrated in the TakeToons system [60]. Our approach makes it possible to augment text-based animations using continuous motions for each triggered pose. To demonstrate this idea, we implemented a simple prototype based on TakeToons. Given a speech recording, we created an annotated transcript that emphasizes specific words with representative poses of the target character. We use a forced alignment algorithm [56] to generate timings for each word in the audio [51] and then generated a corresponding animation that triggers the specified pose at the start of each annotated word. We then hold that pose until the next annotated word is spoken. For each held pose, we add continuous movement using the method described earlier in our pose-driven animation algorithm. For our text-based results, we exaggerated the continuous movement by doubling the size of the convex hulls that bound the motion paths for the hands. Please see our supplemental materials and video for example animations.

9 LIMITATIONS AND FUTURE WORK

While our system works well for a variety of input videos and drawn characters, several aspects offer opportunities for improvement. First, when selecting input videos, we prefer those in which the actor mostly stays stationary and faces the camera. Our system can handle slight changes in body position or camera cuts to different scenes, but artists tend to have better selection and assignment of poses if their characters and camera movements are relatively stationary (aside from arm and hand motion). While our examples focus on frontal poses, our approach would also work with other views (e.g., profile, three-quarters) provided the performer stays in that orientation. Handling twists, 3D rotations, and full-body animations would require extending our pose feature vector and distance computation to capture the relevant 3D spatial and temporal information. These are interesting directions for future work.

Second, our system relies on the pose data acquired from OpenPose [21]. While OpenPose is highly accurate, it will often fail to detect hands and fingers in frames that suffer from considerable motion blur. In such cases, we can only classify the frame as a single hand or no hands even though there are in fact both hands visible. This misclassification results in some left or right hand clusters having frame groups with both hands visible.

When selecting pose clusters, our artists offered other suggestions for improvement. Some felt that there were too many clusters displayed especially since some clusters are visually similar. One artist thought that the display of selected clusters was insufficient and wished for a better interface to determine if a cluster that they were considering was close to another that was already selected. Finally, when commenting about the quality of animations, artists found that it was jarring to transition from one pose to another very different pose. One way to address this concern would be to add another term into our assignment energy that favors transitions between clusters that are visually similar.

10 CONCLUSION

We present a system to help artists design and animate 2D characters based on reference videos of an actor. In particular, our tool assists artists in choosing representative hand and arm poses to design a character that encapsulates the personality of the actor. Then, we automatically trigger those poses to synthesize animations based on other reference videos, audio input or text. This process enables both amateurs and professional animators to easily animate the arm and hand poses of 2D characters. Four artists used our system to create 2D characters and were pleased with the final automatically synthesized animations. In addition, users felt that our automatically synthesized results characterized the personality of the actor in the reference video better than manually created animations.

11 ACKNOWLEDGMENTS

We would like to thank CBS Television Studios for the use of Cartoon Trump and Hillary from *Our Cartoon President*. We would also like to acknowledge the artists: Diano Lao (Scaramucci), Raya Ward (Sandberg), Nora Wildberg (Hillary), and Ilene E (Trump).

REFERENCES

[1] Adobe. 2019. Character Animator. (2019).

[2] Jackie Assa, Yaron Caspi, and Daniel Cohen-Or. 2005. Action synopsis: pose selection and illustration. In *ACM Transactions on Graphics (TOG)*, Vol. 24. ACM, 667–676.

[3] Connelly Barnes, David E. Jacobs, Jason Sanders, Dan B Goldman, Szymon Rusinkiewicz, Adam Finkelstein, and Maneesh Agrawala. 2008. Video Puppetry: A Performative Interface for Cutout Animation. *ACM Trans. Graph.* 27, 5, Article 124 (Dec. 2008), 9 pages.

[4] Richard Bellman. 2013. *Dynamic programming*. Courier Corporation.

[5] Elif Bozkurt, Engin Erzin, and Yucel Yemez. 2015. Affect-expressive hand gestures synthesis and animation. In *2015 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 1–6.

[6] Michael Buhrmester, Tracy Kwang, and Samuel D Gosling. 2011. Amazon's Mechanical Turk: A new source of inexpensive, yet high-quality, data? *Perspectives on psychological science* 6, 1 (2011), 3–5.

[7] Adrian Bulat and Georgios Tzimiropoulos. 2016. Human pose estimation via convolutional part heatmap regression. In *European Conference on Computer Vision*. Springer, 717–732.

[8] Justine Cassell, Hannes Högni Vilhjálmsson, and Timothy Bickmore. 2004. Beat: the behavior expression animation toolkit. In *Life-Like Characters*. Springer, 163–185.

[9] Steven Crowder. 2019. The Real FoxBusiness GOP Debate — Louder With Crowder. (2019). https://youtu.be/-19XqkyYeuI

[10] Live 2D Cubism. 2019. Live 2D. (2019).

[11] Xinyi Fan, Amit H Bermano, Vladimir G Kim, Jovan Popović, and Szymon Rusinkiewicz. 2018. Tooncap: a layered deformable model for capturing poses from cartoon characters. In *Proceedings of the Joint Symposium on Computational Aesthetics and Sketch-Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering*. ACM, 16.

[12] Haoshu Fang, Shuqin Xie, Yu-Wing Tai, and Cewu Lu. 2017. Rmpe: Regional multi-person pose estimation. In *The IEEE International Conference on Computer Vision (ICCV)*, Vol. 2.

[13] Adso Fernández-Baena, Raúl Montaño, Marc Antonijoan, Arturo Roversi, David Miralles, and Francesc Alías. 2014. Gesture synthesis adapted to speech emphasis. *Speech communication* 57 (2014), 331–350.

[14] Source Forge. 2017. Speech Signal Processing Toolkit (SPTK). http://sp-tk.sourceforge.net/. (2017).

[15] World Economic Forum. 2017. Davos 2017 - Outlook for the United States. (2017). https://youtu.be/yDFf93xoV5k

[16] World Economic Forum. 2018. Donald Trump Speaks at Davos 2018. (2018). https://youtu.be/UT7GlaDc060

[17] Brendan J Frey and Delbert Dueck. 2007. Clustering by passing messages between data points. *science* 315, 5814 (2007), 972–976.

[18] Shoichi Furukawa, Tsukasa Fukusato, Shugo Yamaguchi, and Shigeo Morishima. 2017. Voice Animator: Automatic Lip-Synching in Limited Animation by Audio. In *International Conference on Advances in Computer Entertainment*. Springer, 153–171.

[19] Michelle Gallina. 2016. Cartoon Donald Trump Delights Audiences on The Late Show with Stephen Colbert. *Adobe Creative Cloud* (sept 2016). https://blogs.adobe.com/creativecloud/cartoon-donald-trump-late-show-with-stephen-colbert/

[20] GraphicMama. 2016. Adobe Character Animator Puppets. (2016). https://graphicmama.com/blog/adobe-character-animator-puppets/

[21] Gines Hidalgo and others. 2018. OpenPose: Real-time multi-person keypoint detection library for body, face, and hands estimation. *Retrieved April* (2018).

[22] Alexander Hornung, Ellen Dekkers, and Leif Kobbelt. 2007. Character animation from 2D pictures and 3D motion data. *ACM Transactions on Graphics (ToG)* 26, 1 (2007), 1.

[23] 2020CV Inc. 2019. YoPuppet. http://yopuppet.com/. (2019).

[24] Sophie Jörg, Jessica Hodgins, and Alla Safonova. 2012. Data-driven finger motion synthesis for gesturing characters. *ACM Transactions on Graphics (TOG)* 31, 6 (2012), 189.

[25] Rubaiat Habib Kazi, Fanny Chevalier, Tovi Grossman, Shengdong Zhao, and George Fitzmaurice. 2014. Draco: bringing life to illustrations with kinetic textures. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*. ACM, 351–360.

[26] Rubaiat Habib Kazi, Tovi Grossman, Nobuyuki Umetani, and George Fitzmaurice. 2016. Sketching Stylized Animated Drawings with Motion Amplifiers. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems (CHI EA '16)*. ACM, 6–6.

[27] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[28] Robert Lester. 2018. TBS: Final Space Cards with Gary: Case Study. (2018). https://vimeo.com/295195953

[29] Sergey Levine, Philipp Krähenbühl, Sebastian Thrun, and Vladlen Koltun. 2010. Gesture controllers. In *ACM Transactions on Graphics (TOG)*, Vol. 29. ACM, 124.

[30] Sergey Levine, Christian Theobalt, and Vladlen Koltun. 2009. Real-time prosody-driven synthesis of body language. In *ACM Transactions on Graphics (TOG)*, Vol. 28. ACM, 172.

[31] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.

[32] Stacy Marsella, Yuyu Xu, Margaux Lhommet, Andrew Feng, Stefan Scherer, and Ari Shapiro. 2013. Virtual character performance from speech. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. ACM, 25–35.

[33] David McNeill. 1992. *Hand and mind: What gestures reveal about thought*. University of Chicago press.

[34] Daniel Meeks. 2019. Cartoon Donald Trump gives The State of the Union 2019. (2019). https://youtu.be/BXxjM2wsX-Y

[35] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[36] Thomas B Moeslund, Adrian Hilton, and Volker Krüger. 2006. A survey of advances in vision-based human motion capture and analysis. *Computer vision and image understanding* 104, 2-3 (2006), 90–126.

[37] Pavlo Molchanov, Shalini Gupta, Kihwan Kim, and Jan Kautz. 2015. Hand gesture recognition with 3D convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 1–7.

[38] Face Monde. 2018. Donald Trump Watching The Voice Cartoon Box 107. (2018). https://youtu.be/7MbzezfUw4M

[39] Millennial Monitor. 2017. Anthony Scaramucci Takes The Podium - White House Press Briefing. (2017). https://youtu.be/wOyrlobafBU

[40] Lindasalwa Muda, Mumtaj Begam, and Irraivan Elamvazuthi. 2010. Voice recognition algorithms using mel frequency cepstral coefficient (MFCC) and dynamic time warping (DTW) techniques. *arXiv preprint arXiv:1003.4083* (2010).

[41] Michael Neff, Michael Kipp, Irene Albrecht, and Hans-Peter Seidel. 2008. Gesture modeling and animation based on a probabilistic re-creation of speaker style. *ACM Transactions on Graphics (TOG)* 27, 1 (2008), 5.

[42] Alejandro Newell, Kaiyu Yang, and Jia Deng. 2016. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*. Springer, 483–499.

[43] American News. 2016a. American News: Hillary Clinton Speech Today Rally in Wilton Manors, Florida. (2016). https://youtu.be/VOfnBH9Qz0M

[44] American Television News. 2017. President Donald Trump Speech to the Arab Islamic Summit. (2017). https://youtu.be/u225jcAfq-0

[45] The World Hot News. 2016b. Hillary Clinton Speech Today Rally in Las Vegas, Nevada. (2016). https://youtu.be/PNriLB20l4w

[46] Junjun Pan and Jian J Zhang. 2011. Sketch-based skeleton-driven 2D animation and motion capture. In *Transactions on edutainment VI*. Springer, 164–181.

[47] Iveta Pavlova. 2018. Streaming on Twitch with Adobe Character Animator. (2018). https://graphicmama.com/blog/streaming-on-twitch-with-adobe-character-animator/

[48] Maria Perez-Ortiz and Rafal K Mantiuk. 2017. A practical guide and software for analysing pairwise comparison experiments. *arXiv preprint arXiv:1712.03686* (2017).

[49] Ajay Sundar Ramakrishnan and Michael Neff. 2013. Segmentation of hand gestures using motion capture data. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1249–1250.

[50] Reallusion. 2019. Cartoon Animator 4. (2019).

[51] Steve Rubin. 2018. p2fa-vislab. https://github.com/ucbvislab/p2fa-vislab. (2018).

[52] Okay Samurai. 2019. Game Subscriptions Are the Future. https://youtu.be/rzzXqKTC5bI. (2019).

[53] Scott Schaefer, Travis McPhail, and Joe Warren. 2006. Image deformation using moving least squares. In *ACM transactions on graphics (TOG)*, Vol. 25. ACM, 533–540.

[54] SHOWTIME. 2018. Our Cartoon President. (2018). https://www.youtube.com/watch?v=260mj1dmJhU

[55] Tomas Simon, Hanbyul Joo, Iain Matthews, and Yaser Sheikh. 2017. Hand Keypoint Detection in Single Images using Multiview Bootstrapping. In *CVPR*.

[56] Kåre Sjölander. 2003. An HMM-based system for automatic segmentation and alignment of speech. In *Proceedings of Fonetik 2003*. 93–96.

[57] Huaguang Song and Michael Neff. 2014. Design and evaluation of a sketch-based gesture animation tool. In *Proceedings of the Seventh International Conference on Motion in Games*. ACM, 183–183.

[58] Matthew Stone, Doug DeCarlo, Insuk Oh, Christian Rodriguez, Adrian Stere, Alyssa Lees, and Chris Bregler. 2004. Speaking with hands: Creating animated conversational characters from recordings of human performance. In *ACM Transactions on Graphics (TOG)*, Vol. 23. ACM, 506–513.

[59] Qingkun Su, Xue Bai, Hongbo Fu, Chiew-Lan Tai, and Jue Wang. 2018. Live Sketch: Video-driven Dynamic Deformation of Static Drawings. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, Article 662, 12 pages.

[60] Hariharan Subramonyam, Wilmot Li, Eytan Adar, and Mira Dontcheva. 2018.

TakeToons: Script-driven Performance Animation. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology (UIST '18)*. ACM, 663–674.

[61] Louis L Thurstone. 1927. The method of paired comparisons for social values. *The Journal of Abnormal and Social Psychology* 21, 4 (1927), 384.

[62] Trending Videos. 2019. Trump Builds a Wall. (2019). https://youtu.be/geKHVVbaOfA

[63] U.S. Army Website Videos. 2014. Lean in: A Discussion on Leadership with Sheryl Sandberg. (2014). https://youtu.be/LTZPvLi3Hdc

[64] Xin Wang, Shinji Takaki, and Junichi Yamagishi. 2017. An RNN-Based Quantized F0 Model with Multi-Tier Feedback Links for Text-to-Speech Synthesis.

[65] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. 2016. Convolutional pose machines. In *CVPR*.

[66] Nora S Willett, Wilmot Li, Jovan Popovic, Floraine Berthouzoz, and Adam Finkelstein. 2017b. Secondary Motion for Performed 2D Animation. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. ACM, 97–108.

[67] Nora S Willett, Wilmot Li, Jovan Popovic, and Adam Finkelstein. 2017a. Triggering Artwork Swaps for Live Animation. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. ACM, 85–95.

[68] Xiang Yu, Jianchao Yang, Linjie Luo, Wilmot Li, Jonathan Brandt, and Dimitris Metaxas. 2016. Customized expression recognition for performance-driven cutout character animation. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 1–9.