Piku Piku Interpolation

An artist-guided sampling algorithm for synthesizing detail applied to facial animation

Richard A. Roberts
CMIC, Victoria University of Wellington

Rafael K. dos Anjos
CMIC, Victoria University of Wellington

Ken Anjyo
CMIC, Victoria University of Wellington

J.P. Lewis
Victoria University of Wellington

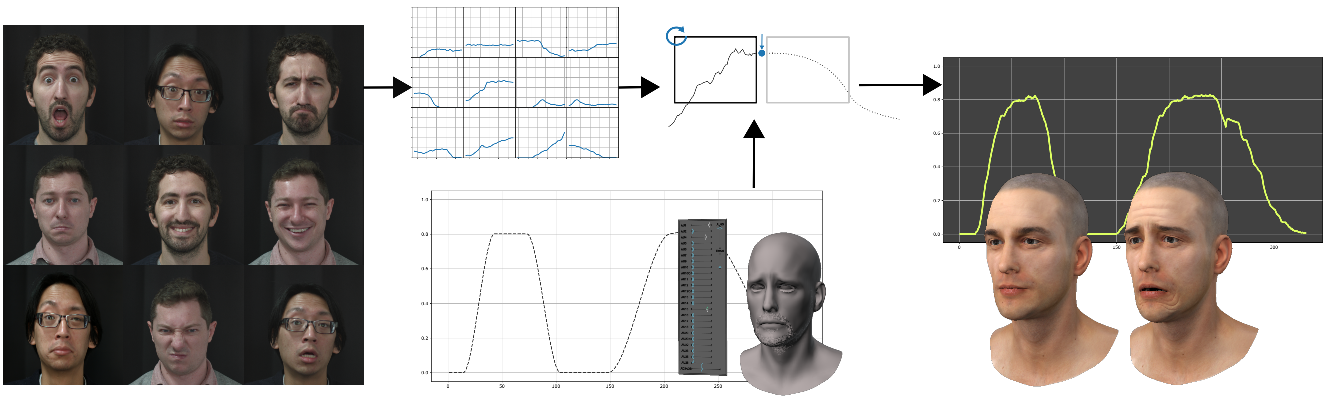

Figure 1: Our system samples FACS data to create detailed facial motion from early-stage animation.

ABSTRACT

We propose a new sampling algorithm that reassembles real-life movements to add detail to early-stage facial animation. We examine the results of applying our algorithm with FACS data extracted from video. Using our algorithm like an interpolation scheme, animators can reduce the time required to produce detailed animation.

CCS CONCEPTS

• **Computing methodologies → Motion processing**.

KEYWORDS

non-parametric sampling, facial motion, facial animation, FACS

ACM Reference Format:
Richard A. Roberts, Rafael K. dos Anjos, Ken Anjyo, and J.P. Lewis. 2020. Piku Piku Interpolation: An artist-guided sampling algorithm for synthesizing detail applied to facial animation. In *SIGGRAPH Asia 2019 Technical Briefs (SA '19 Technical Briefs), November 17–20, 2019, Brisbane, QLD, Australia.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3355088.3365156

SA '19 Technical Briefs, November 17–20, 2019, Brisbane, QLD, Australia

ACM ISBN 978-1-4503-6945-9/19/11…$15.00
https://doi.org/10.1145/3355088.3365156

1 INTRODUCTION AND MOTIVATION

The latest rendering techniques enable us to render characters near photo-realistically. Despite those advances, creating detailed, varied and plausible motion remains a difficult problem. While motion capture (MoCap) technology and keyframe-based animation can both produce detailed and varied motion, these approaches are generally expensive and require intensive labour. MoCap offers accurate tracking of points on the body and face, but the resulting data is difficult to edit, and specialized equipment may be required. Keyframe-based animation is more flexible, but creating the large number of fully detailed and varied animations for a production can require teams of animators to work for many years; consequently, while important characters appear detailed and varied, other characters often lack interesting detail.

To help address this problem we propose a simple system (illustrated in Fig. 1) that automatically details facial animation based on real-life movement. Our approach is inspired by the Japanese phrase *Piku Piku* (meaning twitching during motion), which is one type of facial movement that is particularity hard to recreate in hand crafted animation. Specifically, we present a new non-parametric sampling algorithm: the animator provides a blocked animation[1], from which our algorithm creates a new detailed animation that traces their input while reproducing movements from a reference motion; in our case, we use FACS[2] data extracted from video via

[1]A blocked animation is hand crafted animation that uses a few keyframes to coarsely outline an envisioned motion. See discussion on pose-to-pose animation in [11].
[2]The Facial Action Coding System (FACS) [4] has become a popular model for encoding facial movements. In FACS, a set of action units (AUs) describe how different facial muscles activate to produce facial expressions. Facial animation can be encoded with this model by recording AU configuration over time.

OpenFace for this reference. When used effectively, our system helps animators to reassemble real-life movements that improve the realistic appearance of their animation.

The advantages of our algorithm are: (1) it can be integrated with the time-tested keyframe-based approach to animation that is well support by commercial software, (2) it is simple to implement, and (3) it is driven by input data that is easy to obtain, even for non-professional use. When compared to previous work that uses noise to add naturalness to keyframe-based animations, our algorithm produces different activation and relaxation profiles, produces realistic twitches when holding expressions, and also factors in the effect of time when creating these effects.

2 BACKGROUND

Jerk is a subtle effect (distinct from *variation*).[3]Although changes in facial expression can appear smooth, they contain twitches. One can observe jerk by trying to smile gradually over an extended period, such as 10 seconds. Harris & Wolpert theorize that noise in neural commands leads to fine scale perturbations in movement [5]. Jerk might also be related to how quickly transitions are performed, how much different facial muscles are strained, and other factors such as the elasticity of the skin.

We hypothesize that the uncanny feeling of some character animations is due to a lack of jerk. To investigate this, we examine how natural facial movements exhibit changes at a fine scale, as illustrated by Figure 2. The figure displays the activation of two action units that we observed from FACS data (extracted from video using *OpenFace*) for a smiling expression performed at three different speeds. Both action units display distinct behavior for each performance. While the transition from a neutral face to a smiling expression (activation) may happen smoothly when done quickly (Figure 2B, C, top), when done in a slower fashion (Fig. 2B, C, bottom) it displays increasing levels of jerkiness. This is also observed, with a different profile, when transitioning back to the neutral face. Additionally, we can see that holding expressions for a period of time also creates jerky movement (Fig. 2, red area). Other literature makes similar observations [8, 9].

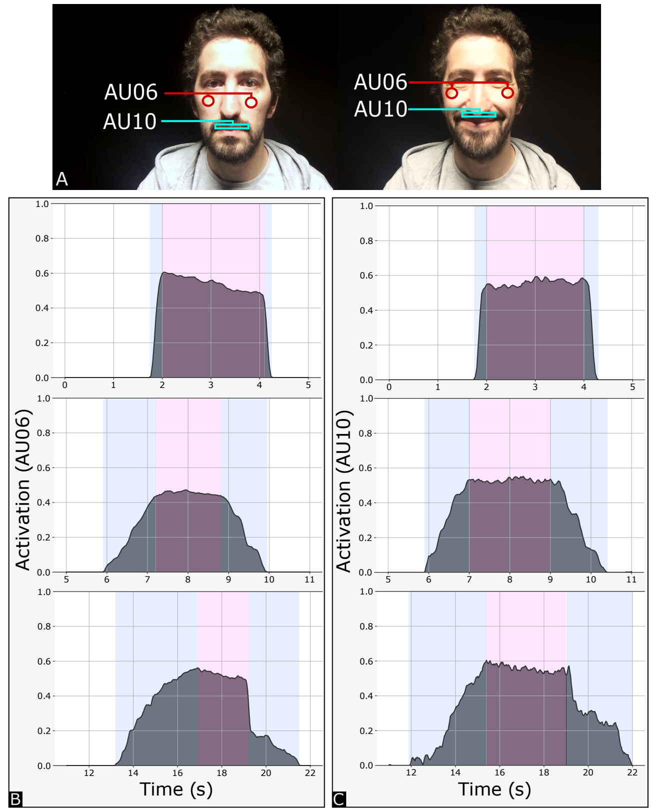

Figure 2: Activation curves of AU06 (cheek raiser, B) and AU10 (upper lip raiser, C) for a person smiling (A) at three different speeds (from top to bottom: fast, medium, and slow). Jerkiness is noticeable throughout the movement.

2.1 Limitations of Previous Work

Automating the process of adding detail and variation has been a common goal of previous work (see Section 5 in [10]), although with a focus on animation for the body. Bodenheimer et al. described the problem of varying walking motions [2], proposing a simple approach where noise is added to variables in a simulation creating the walking animation. Their experimental results show that animations were perceived as most natural when some noise was added. More recently, deep neural networks have been used for motion synthesis of walking animation [6]. While these approaches can produce a rich variety of outputs from a simple input, they do not enable an animator to control over how details are added.

Most similar to our work, [8] use a linear dynamic system (LDS) to add noise to the movement of facial landmarks. Following from Harris & Wolpert's observations, the model alters the amount of noise during transitions. However, as an LDS it has limited modeling power, and their approach does not add detail when an expression is held approximately constant, contrary to the results seen in Figure 2.

Importantly, configuring noise parameters to change correctly with respect to motion is non-trivial (refer to supplementary material for a basic illustration). This motivates our non-parametric approach, where no parameters relating to a noise function are used. Like our approach, [7] also present a non-parametric algorithm, where they divide an input animation into fragments and then replace each fragment with a similar fragment of MoCap. Their multi-scale implementation was developed for body motion and features a more complex structure than ours.

3 NON-PARAMETRIC SAMPLING

Our non-parametric sampling algorithm is inspired by Efros & Leung's approach for texture synthesis where pixels are synthesized one at a time from a input image [3]. Much like how they grow pixels by reassembling them from an input texture, we generate detail by reassembling observed motion data. However, distinct

[3]Variation refers to the concept that repeated human movements naturally differ from each other, even if someone is consciously attempting to replicate the same movement. MoCap captures such variations and the result often appears natural. Variation is well recognized [10], and adding variation to keyframe-based animation is a common research problem.

from Efros & Leung's algorithm, we create the detail in a way that traces a user-defined input.

The inputs to our algorithm are the blocked animation together with the reference motion. For our case of facial animation, we represent both inputs as a set of independent 1D curves that correspond to FACS (each curve specifies an AU over time). For the reference motion, we obtain the set of 1D curves directly by applying OpenFace [1] to video recordings of a person performing a variety of facial expressions.

Our system works in four steps: (1) It first divides the observed the data into a collection of *patches* using a chosen window size. (2) It then chooses a starting patch to initialize the synthesis, using only the right part of the sampling window (Fig 3, described below). (3) It then scans through the blocked animation and repetitively adds frames until most of motion has been synthesized. (4) To finish, it chooses a patch using only the left part of the sampling window and copies from it the frames required to complete the animation.

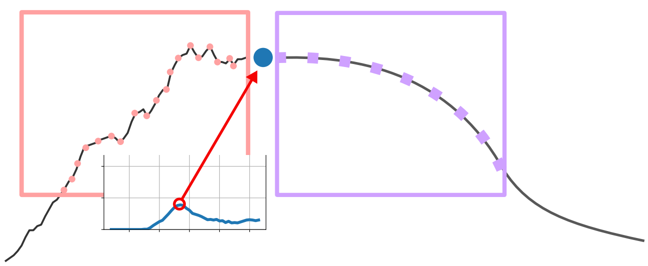

Figure 3: The sampling window. Left: synthesized data. Right: blocked animation. The window is used during a nearest neighbour step to find a patch from the reference motion, which provides the next frame for the synthesized animation.

After initialization, the algorithm needs to choose a patch to sample a frame from. To choose a patch, we need to consider both the already synthesized detail and also the upcoming part of the blocked animation. Figure 3 illustrates our sampling *window* that considers both the already synthesized detail (left part, red) and also the next part of the blocked animation (right part, purple). Our algorithm uses a high-dimensional nearest neighbour solution to find a patch from the reference motion with values most similar to those seen in the window.[4] The nearest neighbour algorithm selects a particular patch P^k to minimize:

$$\alpha \sum_{i=1}^{\lfloor N/2 \rfloor} G_i \cdot |W_i - P_i^k| \; + \; (1-\alpha) \sum_{i=\lfloor N/2 \rfloor+1}^{N} G_i \cdot |W_i - P_i^k|$$

where W_i denotes samples from the window (containing previously constructed points $i \leq \lfloor N/2 \rfloor$ on the left, along with points from the animator's input on the right) and P_i^k denotes samples from a candidate patch P^k. The vectors representing the window and patch are multiplied by a Gaussian falloff G_i that de-emphasizes samples away from the center, i.e. the next point to be sampled.

[4] We use FLANN: https://www.cs.ubc.ca/research/flann. We also experimented with randomly selecting a sample from among the k nearest neighbors, but found this usually did not improve results.

This reduces the possibility of discontinuity due to sampling from non-neigbouring patches. Finally, the contribution of left and right parts are also scaled by the coefficient α (a constant selected by the animator). See Section 5 for further discussion on the use of α.

Once a patch has been selected we copy the frame at the center of that patch into the output motion. We repeat this search and sample process until the right edge of the sampling window reaches the end of the block animation. At this point, we complete the output animation by copying all frames occurring after the center frame of the last used patch into the output motion.

In summary, our two-part sampling window design can be applied to create new animation that preserves the characteristics of the reference motion while roughly tracing the blocked animation. An animator might choose to use a number of different reference motions to produce a variety of detail (perhaps using videos of different people). Importantly, the animator has control over how closely the added detail traces the animation. The result is a detailed and varied interpretation of the blocked animation.

4 EVALUATION

In order to validate our algorithm, we use a FACS-based character that can be controlled by setting values for a set of AUs. For testing we first created three blocked animations (a happy expression, a sad expression, and an angry expression) that start with a neutral face, transition into an expression, and then return back to a neutral face. We applied our algorithm to each of the animated AUs, thus creating a detailed version of each blocked animation.

Figure 4 shows plots of certain AUs for the results obtained for two motions: anger and sadness. See the supplementary video for comparison between the original and resulting animations.

Figure 4A shows the outputs of four AUs in the anger example. Similar to [8], our algorithm correctly generates distinctive activation profiles for each action unit. While AU04 and AU07 rise smoothly, that is not the case for AU05 and AU15. The output AUs also display unique behaviour through the apex phase, with different scales of jerkiness. Moreover, this figure shows that our algorithm produces different shapes for activation and relaxation (e.g. AU05 has jerky activation and abrupt relaxation). These differing temporal and intensity qualities are related to the way that facial muscles behave when contracting and relaxing.

Figure 4B shows the results obtained when our algorithm is applied to two animation curves of different lengths for AU01. To demonstrate the effect of our α variable, we ran the algorithm with three different values: $\alpha = 0.8$, $\alpha = 0.9$, $\alpha = 0.95$. Regarding duration of the movement, the obtained results feature different jerk profiles for curves of differing lengths, thus replicating the same behavior that we observed earlier in Figure 2. Finally, when α is high (0.95), the generated curve is based more strongly on the reference motion, whereas, for lower values (0.9, 0.8), the generated curve traces the input animation more closely.

5 DISCUSSION

There are a two variables to consider when applying the algorithm. The size of the sampling window determines how much of the observed motion is recognizable in the output: smaller windows enable the synthesized data to appear more distinctive while larger

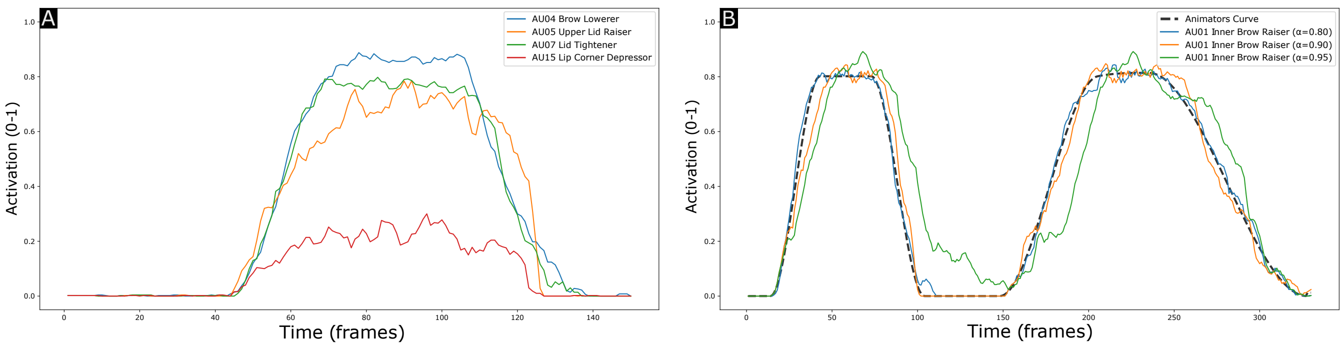

Figure 4: Algorithm results: (A) Four different action units for an anger expression. Each AU produces a different jerk profile. (B) Plot of AU01 for a sadness expression at two different speeds and three different parameters for α. The user can choose how close they want to stay to the animator's curve.

windows ensures that the output is more faithfully to the reference motion (see Figure 2 in [3]). Unique to our two-part sampling window design, we also need to consider the balance between the left and right parts of our sampling window. Placing more importance on the right part enables us to produce detail that traces the animator's outline more closely, while placing more importance on the left part produces detail that is closer to the reference motion. By controlling these parameters (window size and α), an animator can choose to apply our algorithm to add just a small amount of jerk to a detailed curve, to generate a variety of distinct detail, or anything in between. Additionally, small changes in α can be used to achieve variation.

Our algorithm has some limitations. First, it cannot extrapolate detail beyond that which is present in the reference motion. In the current work, this was addressed by capturing enough data to cover a wide range of motion for each action unit. Another problem is the speed of our implementation, which is impacted by the number of the patches in our collection. While Deep Neural Networks could potentially overcome some of these limitations – they have been successfully used for full body animation [6] – they also have disadvantages (with a few exceptions they require extensive data, long training times, and they typically generate a point estimate rather than a predictive distribution and thus are not suitable for generating a random signal, which essentially involves sampling from a distribution).

Another key limitation of our present implementation is the reliance on AU estimation through computer vision algorithms. Both *OpenFace* and other state of the art techniques produce erroneous noise when tracking AUs. Consequently, it may be necessary to use smoothing algorithms to reduce artifacts in the motion that result from this noise. Unfortunately, such smoothing can also remove the details added by our algorithm. Future developments in facial detection will likely resolve this issue.

In future work we aim to extend our algorithm to extrapolate away from the reference motion, examine the effects of using different reference motions, explore how spatial partitioning and parallel processing of the patch collection may improve performance, evaluate resulting animations in a formal user study, and get feedback from professional animators to determine the viability of our approach in a production setting.

6 CONCLUSIONS

We have presented a new non-parametric sampling algorithm to synthesize detailed motion from blocked animation, in which the added detail replays movements observed from real-life video. The result improves the appearance of realism and naturalness by correctly recreating time and intensity related aspects of the observed facial motion. Our algorithm has simple setup and can be used as an extension to existing keyframe interpolation methods available in commercial animation software.

ACKNOWLEDGMENTS

Special thanks to Ayumi Kimura and Ian Loh. This project was supported by the Entrepreneurial University Programme funded by the Tertiary Education Commission, New Zealand.

REFERENCES

[1] T. Baltrušaitis, P. Robinson, and L. Morency. 2016. OpenFace: An open source facial behavior analysis toolkit. In *IEEE Winter Conference on Applications of Computer Vision*.
[2] Bobby Bodenheimer, Anna V. Shleyfman, and Jessica K. Hodgins. 1999. The Effects of Noise on the Perception of Animated Human Running. In *Computer Animation and Simulation*.
[3] Alexei A. Efros and Thomas K. Leung. 1999. Texture Synthesis by Non-Parametric Sampling. In *Proceedings of the International Conference on Computer Vision*.
[4] Paul Ekman and Wallace V. Friesen. 1976. Measuring facial movement. *Environmental psychology and nonverbal behavior* 1, 1 (1976).
[5] Christopher M Harris and Daniel M Wolpert. 1998. Signal-dependent noise determines motor planning. *Nature* 394 (1998).
[6] Daniel Holden, Jun Saito, and Taku Komura. 2016. A Deep Learning Framework for Character Motion Synthesis and Editing. *ACM Trans Graph.* 35, 4 (2016).
[7] Katherine Pullen and Christoph Bregler. 2002. Motion Capture Assisted Animation: Texturing and Synthesis. *ACM Trans. Graph.* 21, 3 (2002).
[8] N. Stoiber, G. Breton, and R. Seguier. 2010. Modeling Short-Term Dynamics and Variability for Realistic Interactive Facial Animation. *IEEE Computer Graphics and Applications* 30, 4 (2010).
[9] M. Valstar and M. Pantic. 2006. Fully Automatic Facial Action Unit Detection and Temporal Analysis. In *2006 Conference on Computer Vision and Pattern Recognition Workshop*.
[10] H. Van Welbergen, B. J. H. Van Basten, A. Egges, Zs. M. Ruttkay, and M. H. Overmars. 2010. Real Time Animation of Virtual Humans: A Trade-off Between Naturalness and Control. *Computer Graphics Forum* 29, 8 (2010).
[11] Richard Williams. 2001. *The Animator's Survival Kit.* Faber.