# Face Stabilization by Mode Pursuit for Avatar Construction in the Universe

Mathieu Lamarre
*SEED, Electronic Arts*
Montreal, Canada
mlamarre@ea.com

J.P. Lewis
*SEED, Electronic Arts*
Los Angeles, USA
noisebrain@gmail.com

Etienne Danvoye
*SEED, Electronic Arts*
Montreal, Canada
edanvoye@ea.com

***Abstract*—Avatars driven by facial motion capture are widely used in games and movies, and may become the foundation of future online virtual reality social spaces. In many of these applications, it is necessary to disambiguate the rigid motion of the skull from deformations due to changing facial expression. This is required so that the expression can be isolated, analyzed, and transferred to the virtual avatar. The problem of identifying the skull motion is partially addressed through the use of a headset or helmet that is assumed to be rigid relative to the skull. However, the headset can slip when a person is moving vigorously on a motion capture stage or in a virtual reality game. More fundamentally, on some people even the skin on the sides and top of the head moves during extreme facial expressions, resulting in the headset shifting slightly. Accurate conveyance of facial deformation is important for conveying emotions, so a better solution to this problem is desired. In this paper, we observe that although every point on the face is potentially moving, each tracked point or vertex returns to a neutral or "rest" position frequently as the responsible muscles relax. When viewed from the reference frame of the skull, the histograms of point positions over time should therefore show a concentrated mode at this rest position. On the other hand, the mode is obscured or destroyed when tracked points are viewed in a coordinate frame that is corrupted by the overall rigid motion of the head. Thus, we seek a smooth sequence of rigid transforms that cause the vertex motion histograms to reveal clear modes. To solve this challenging optimization problem, we use a coarse-to-fine strategy in which smoothness is guaranteed by the parameterization of the solution. We validate the results on both professionally created synthetic animations in which the ground truth is known, and on dense 4D computer vision capture of real humans. The results are clearly superior to alternative approaches such as assuming the existence of stationary points on the skin, or using rigid iterated closest points.**

***Index Terms*—computer vision, face tracking, avatars, animation**

## I. Introduction

Motion capture (mocap) is widely used in movies and games and may become ubiquitous if anticipated online social platforms are realized. A fundamental task in these applications is to transfer the performance to the virtual avatar. In some cases it is sufficient to simply reproduce the movement of the user or actor as accurately as possible and no analysis of the facial expression is needed. In other cases, however, the actor's performance must be transferred to a fantasy character, or the user may wish to take on the appearance of a warrior, princess, or alien.

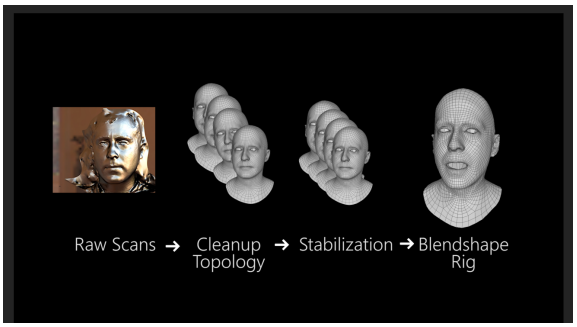



Fig. 1: A high-level view of constructing an avatar model, showing the role of stabilization.

The most common approach to this *expression cloning* or performance transfer problem [1] is to represent the original performance using a low-dimensional set of parameters and drive the avatar with these same parameters. Typically the parameterization is in terms of approximate muscle movements inspired by the Facial Action Coding System (FACS) [2], [3]. In the simplest case, the approximate muscle movements form a linear basis, and the corresponding muscle activations at each frame can be obtained by solving a small linear system [4]. These solved parameters can then be used to drive an avatar model with arbitrary geometry provided it is similarly parameterized.

A small but crucial problem in the construction and use of such avatars is the *stabilization* problem: motion due to facial expressions must be isolated from motion due to overall head movement. Failure to fully solve this problem results in expression parameters that are corrupted by head movement, with the result that moving the head may incorrectly cause the facial expression to change. A simplified view of how stabilization is used is shown in Fig. 1. It should be noted that, depending on the particular pipelines involved, the stabilization problem can occur in *both* the avatar construction process and in driving the final avatar. See [5], [6] for recent approaches to the more general problem area.

An evident solution to this problem is to identify three or more relatively rigid points on the face, such as the eye corners and tip of the nose, and compute the rigid transform from these points. Alternately, the motion of a tightly attached headset or

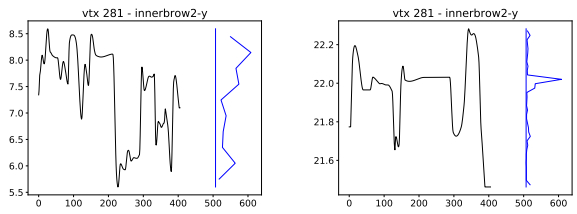



Fig. 2: Movement of the $y$-coordinate of vertex on the inner brow in a 405 frame artist-created animation (left), and movement of the *same* vertex with the head motion removed (right). The $y$-coordinate on the right corresponds to raising the eyebrow, but the signal in the left plot also contains the overall head motion (note differing scales). In each plot the movement of the vertex is shown in black, while the blue curve is a histogram of the $y$ values. The histogram is rotated 90 degrees to align with the $y$-movement function. *Please enlarge figures to see details.*

helmet can be used to approximate the rigid movement of the skull. However, these solutions are only approximate. Thick hair makes it difficult to tightly attach the headset, and in any case it may move during vigorous acting or gameplay. More fundamentally, on some people it is difficult to find *any* visible points on the skin that do not move during facial expressions. For example, forcefully raising the eyebrows often causes the tip of the nose to move slightly, and even results in skin on the top of the head moving on some people.

*While the resulting error is quantitatively small, it is subjectively quite important*–consider that the difference in facial geometry between the expressions of "calm" and "contempt" may be very small, perhaps on the order of a millimeter. Conveying the correct emotion is important both for acting in movies and games, and for social interaction in virtual online spaces. As a result, manual correction of transferred performances is frequently required in high-quality entertainment applications, while uncorrected errors contribute to the "creepy" character of some real-time avatars. Further discussion of existing approaches is provided in Section II.

Our approach to stabilization starts from the hypothesis that points on the face are close to a "rest" position more frequently than to any other single position. The idea is that muscles contract to produce a variety of facial expressions, but then relax, returning to a common rest shape. As a consequence, the histogram of positions of a particular point tracked over time should show a clear approximate *mode* corresponding to the rest position. On the other hand this statement *is only true when viewed from the coordinate frame of the skull* – in the world coordinate system the histogram of point positions also has contributions from the overall head and body motion.

Since the ground-truth skull motion is not easily obtained, we explored this hypothesis by examining professionally authored facial animations. In this synthetic case, the head's rigid frame is explicitly animated, and zeroing this rigid animation leaves expressions that are moving relative to a stabilized head. We found that some vertices show an approximate mode corresponding to the rest pose Fig. 2. Other vertices do not show a clear mode, however when all vertices on the face region are considered a very clear mode is visible (see Fig. 5 in Section IV). Further, the addition of even small amounts of rigid rotation destroys the mode (Fig. 5).

Consequently, our stabilization algorithm seeks to find a series of rigid transforms $\mathbf{M}_i \in SE(3)$ that, when applied to the reconstructed moving ahead, recover a sharp mode in the histogram of positions-over-time of many points in the face region. This is a challenging optimization problem for several reasons: 1) recovering the mode involves a non-convex loss and the problem has multiple optima (for example, a poor solution is to hold several points on the chin stable), and 2) at some times during a facial performance every point on the face may be moving.

Our *mode pursuit* algorithm solves this problem in a coarse-to-fine manner: Starting from plausible initialization using an existing algorithm such as Iterated Closest Point (ICP), we find a rigid transform at every frame that causes as many vertices as possible to approximately align with their rest position across time. The allowable tolerance in the approximate alignment is then reduced and the process is repeated. See Section III-A.

Optimization of this objective is not enough, however, since there may be temporal samples (frames) for which *every* point on the face is moving. The rigid motion of the head is slow relative to facial deformation (for example, by viewing video frame-by-frame we know that the mouth can move from fully closed to fully open in $1/30$ sec.), so a smooth rigid motion is preferred. We *enforce* it by parameterizing the solution in terms of a dual-quaternion spline (Section III-B). The stabilization process is subtractive in the sense that the estimated rigid motion is removed from the measured motion. Subtracting a smoothed modeled trajectory from the original motion implements a high-pass filter. For this reason, the optimization includes a term penalizing vertex motion. In the absence of vertices close to their rest pose modes, this minimizes high frequency residuals. We originally considered adding a term favoring smoothness on the solved rigid motion parameters (the dual quaternion curves), but found that penalizing vertex velocity is sufficient for good results. The resulting optimization is solved using a coarse-to-fine scheme. The algorithm is described in more detail in Section III.

## II. Related Work

As mentioned earlier, the obvious stabilization approach of identifying the rigid transform from a triple of points usually does not work because it is difficult to find three points that are truly rigid with respect to the skull. This can be partially addressed by averaging over a larger number of points, such as selected regions on the upper face (avoiding the jaw), with the hope that errors at individual points average out. In the case where correspondences across frames are known (true in our case), Procrustes alignment has been applied for this purpose [7]. In this procedure the translation is first trivially handled by removing the mean of the tracked points. The rotation is

then obtained as the solution to the problem $\min \|\mathbf{A}-\mathbf{BQ}\|_F^2$ where in the stabilization problem $\mathbf{A},\mathbf{B}$ contain the points from two frames to be aligned and $\mathbf{Q} \in SO(3)$. The solution $\mathbf{Q}=\mathbf{UV}^T$ is obtained from the SVD of $\mathbf{B}^T\mathbf{A}$ [8]. See [9] for further variants.

Iterated Closest Point (ICP) algorithms have been used in the case where correspondences across frames are not known [10]. This family of algorithms alternates between assigning correspondences using closest points, and solving a rigid alignment (Procrustes) problem given these correspondences. Classic algorithms minimizing squared error perform poorly when there are outliers in the data; this has been addressed with the use of robust (e.g. $\ell_1$ or Huber) loss functions. Bouaziz et al. [11] introduced an optimization algorithm that achieves a highly robust "near $\ell_0$" loss, providing superior results.

The only published work that specifically addresses our problem is [12], [13]. In Beeler and Bradley [12] an underlying skull is first explicitly estimated by morphing a template skull to fit a given neutral expression scan assuming expected flesh thicknesses. Then, a nonlinear optimization solves for the skull position that approximately preserves the volume of the flesh between the skin and skull. This approach was compared to manual stabilization and worked well. A drawback of the method is that it is rather complex to implement. Also, the several approximations involved (estimates of unknown flesh thickness and skull shape, and the approximation of volume preservation) do not guarantee that the results are always correct. Fyffe et al. [13] approach the stabilization problem by assuming that, following an initial Procrustes alignment, the rigid rotations are small and hence can be linearly approximated as a weighted sum of the Lie algebra rotation generators. An iterative algorithm removes motion that is better explained as linearized rotation than as deformation, and a final rotation at each frame is computated as the best rigid alignment of the resulting and original meshes.

Another approach to the stabilization problem is to incorporate estimation of head orientation into the optimization that solves for the facial parameters. In the simplest case of a linear "blendshape" model, the parameters can be obtained for a stabilized head as a constrained linear system solved with quadratic programming, whereas incorporating stabilization requires a nonlinear solve. A more fundamental problem is that if the parametric model does not exactly fit the performer, the fitting error will be distributed in an unknown way between the expression and head motion parameters. In fact it is common that the model does not exactly match the performer even if it is based on a prior scan of the same person. Several algorithms address this with schemes that adapt an existing model to better fit the face shape being tracked [14]–[16]. On the other hand, the principle used in our stabilization approach does not reference the parametric model and so does not suffer from this problem.

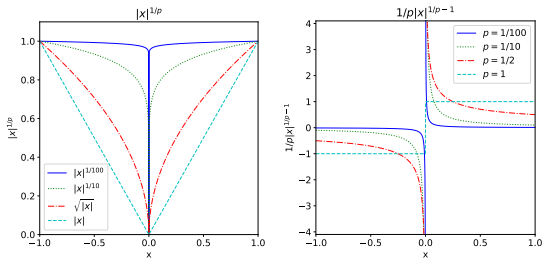



Fig. 3: Family of penalties corresponding to the $p$-norm (left) and derivative (right).

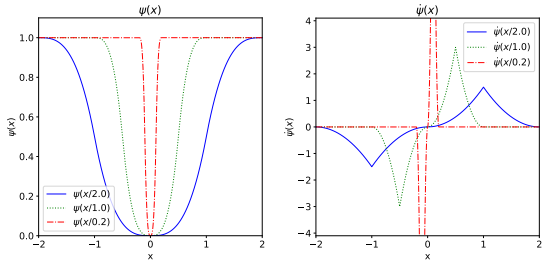



Fig. 4: Our custom penalty function $\psi$ with $n = 2$ has an adjustable width or "tolerance" (left); derivative (right).

.

## III. Method

Our solution to the stabilization problem requires a smooth set of rigid transforms that, when applied to a "4D tracked" mesh, will cause the motion of many vertices to have an approximate but clear mode. While our 4D tracking system is proprietary, the stabilization algorithm presented here does not rely on details of the tracking and could be used with published algorithms [17]–[19]; see [5] for one recent survey.

### A. Mode pursuit

Recall that the mean is the central value $c$ that minimizes the sum of squared errors $\arg\min_c \sum (c-x_k)^2$, whereas the median is the value that minimizes the sum of absolute errors, $\arg\min_c \sum |c-x_k|$. These measures of the distance between $c, x_k$ are generalized to a continuous family in the $p-$norm (or $\ell_p$-norm) $\|\mathbf{d}\|_p = (\sum |d_i|^p)^{\frac{1}{p}}$ where $\|\mathbf{d}\|_p$ is the multidimensional $p$-distance and minimizing $\ell_2, \ell_1$ reduce to the mean and median respectively. Similarly, the "$\ell_0$ norm" corresponds to the mode. However, the $\ell_p$ norm expression is not a true norm for $p < 1$, and $\ell_0$ should be understood as $\lim_{p\to 0} \ell_p$. Fig. 3 (left) plots the penalty $|x|^p$ for several values of $p$ approaching zero. While the median corresponds to a loss in which the cost increases linearly with distance, as $p \to 0$ *any* discrepancy is equally penalized. Thus the "$\ell_0$" penalty is conceptually an approach to finding the mode.

While the fact that $\ell_p$, $p < 1$ is not a true norm is not crucial for our purpose, there are several other problems with directly applying this idea. Optimization requires the derivative of the penalty function, and as shown in Fig. 3 (right) the derivative is poorly behaved for $p$ near zero. Further, finding

the minimum $\ell_0$ norm is both a combinatorial search problem, and is ill-posed: a small amount of noise change the solution completely.

Our mode pursuit algorithm addresses these issues with the $\ell_0$ penalty using two ideas. First, we adopt a penalty that is blind to some amount of noise. Through experimentation we designed a symmetric sigmoidal penalty composed from a piecewise polynomial

$$\psi_{\text{even n}}(x) = \begin{cases} \frac{(2x)^n}{2} & \text{if } |x| \leq 0.5; \\ 1 - \frac{(2x-2)^n}{2} & \text{if } 0.5 < |x| \leq 1; \\ 1 & \text{otherwise} \end{cases}$$

$$\psi_{\text{odd n}}(x) = \begin{cases} \frac{(2x)^n}{2} & \text{if } |x| \leq 0.5; \\ 1 + \frac{(2x-2)^n}{2} & \text{if } 0.5 < |x| \leq 1; \\ 1 & \text{otherwise} \end{cases} \quad (1)$$

(see Fig. 4); the particular polynomial function has been used as for a smooth fade function in computer graphics [20]. This function evaluates the absolute difference between the rigidly transformed rest pose mesh vertex coordinates and the measured coordinates. It gives the same penalty to large outliers while allowing a certain amount of deviation from the mode to be unpenalized. The derivative of the function is well behaved (Fig. 4 (right)).

Second, the solution is "steered" in a coarse-to-fine manner, starting with an approximate initialization and then gradually reducing the width of $\psi$ to approximate the $\ell_0$ norm. Iterations are stopped at a width that allows the expected amount of normal noise. In our experiments we used the schedule of 0.8, 0.4, 0.2, 0.1, and 0.05 cm for positions and 0.2, 0.1, 0.05, 0.025, 0.0125 cm/frame for velocities

### B. Motion representation

Kavan [21] has shown that dual-quaternion have desirable properties to model sequences of rigid transformations, namely: constant speed, shortest path and coordinate system invariance. They also demonstrate empirically that an approximate algorithm, dual-quaternion linear blending (DLB), preserves the shortest path and coordinate system invariance properties. The constant speed property, which means that the rotation angle varies linearly with the time parameter, is useful for artistic control but has no incidence on data fitting. DLB is easy to implement with high throughput tensor processing software like Pytorch [22].

$$\mathbf{DLB}(w; \mathbf{q}_1, ..., \mathbf{q}_n) = \frac{w_1\mathbf{q}_1 + ... + w_n\mathbf{q}_n}{\|w_1\mathbf{q}_1 + ... + w_n\mathbf{q}_n\|}$$

The blending weights are computed with a B-spline basis, resulting in spline dual-quaternion linear blending (SDLB):

$$\mathbf{SDLB}(\mathbf{q}_i, B_i, x) = \text{normalize}\left(\sum_i \mathbf{q}_i B_{i,p}(x)\right)$$

where the basis vectors are computed with DeBoor's algorithm [23].

$$B_{i,0}(x) := \begin{cases} 1 & \text{if} \quad t_i \leq x < t_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$B_{i,p}(x) := \frac{x - t_i}{t_{i+p} - t_i} B_{i,p-1}(x) + \frac{t_{i+p+1} - x}{t_{i+p+1} - t_{i+1}} B_{i+1,p-1}(x).$$

The rigid transform $\mathbf{r}_f$ at every frame is represented as (2)

$$\mathbf{r}_f(\mathbf{q}_i) = \mathbf{SDLB}(\mathbf{q}_i, B_i, x_f) \quad (2)$$

where $i \in \{1, \ldots, n_{dqs}\}$ with $n_{dqs}$ the number of control dual-quaternions and $x_f$ denotes the spline parameter corresponding to frame $f$.

We refine the dual-quaternion curve in coarse to fine fashion using the shape preserving midpoint refinement [23]. In practice, the optimal initial number of control points depends on the quality of the initialization which is discussed in Section III-D. The final number of control points parameterizes the smoothness of the solution.

### C. Optimization

Let $S_f$ be the stabilized mesh at frame $f$ as a function of the control dual quaternions $\mathbf{q}_i$ using (2)

$$S_f(\mathbf{q}_i) = \mathbf{r}_f(\mathbf{q}_i) X_f \overline{\mathbf{r}_f(\mathbf{q}_i)} \quad (3)$$

where $X_f$ denotes the tracked mesh at frame $f$. Our overall optimization problem is then

$$\arg\min_{\mathbf{q}_i} \sum_{f \in \text{frames}} \sum_{c}^{3} \psi_{pos}\left(|S_f(\mathbf{q}_i) - X_r|_c\right) + \psi_{vel}\left(|\dot{S}_f(\mathbf{q}_i)|_c\right) \quad (4)$$

$X_r$ is the *static* rest mesh, $\dot{S}_f$ is the stabilized mesh vertex velocity computed using central finite differences, $|\cdot|_c$ denotes the distance in the $c$-th coordinate (x,y, or z), and $\psi_{pos}$ and $\psi_{vel}$ are penalty function for vertex position and velocity.

The energy minimization problem (4) is implemented with Pytorch as two main functions: a forward function computing the stabilized mesh coordinates from the dual-quaternion parameters (3) and the penalty function implementing $\psi$ per (1). Like other automatic gradient libraries Pytorch is flow graph based. The graph is built on-the-fly by evaluating functions in a model class forward method. Our algorithm model builds a graph with the dual-quaternion control vertices as leaf variables. Leaf variables are the parameters optimized by Pytorch to minimize the loss between the forward method output and the expected values. The tracked coordinates are input to the forward method. The stabilized coordinates and stabilized velocities are the output. The position loss function is evaluated on the absolute difference between the stabilized coordinates and static rest pose coordinates. Stabilized vertex velocities are estimated with a convolution filter with 7 central difference coefficients (accuracy $O(h^6)$). Absolute velocities are evaluated to penalize any motion of the stabilized mesh. The energy is minimized using the Pytorch L-BFGS optimizer. In our experiments, the optimizer stops at a prescribed number of iterations rather than reaching its internal convergence criterion. The optimization is restarted multiple times when

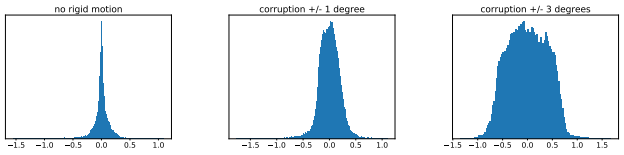



Fig. 5: Effect of adding rigid motion on the movement histogram. From a 405-frame artist-created animation, with the head motion removed: (Left) histogram of the y (vertical) movement of 832 face vertices from the face region concatenated into a single signal by aligning their means. The combined signal has a length of $405 \times 832 = 336960$. (Middle) the histogram resulting after $\pm 1$ degree of random rigid head rotation is added every 5 frames, with spline interpolation of the rotation in between. (Right) result after adding $\pm 3$ degrees of random rotation.

reducing the width of $\psi$ and when refining the B-spline curve (i.e. this is a numerical continuation scheme).

Benchmarks are run on a Titan-V GPU. The implementation of the forward and loss functions is vectorized but not optimized to fully utilize the GPU. The runtime depends mostly on the number of iteration and very little on the problem size. It varies by large amount between repeated experiment (+/- 10 seconds). With our datasets GPU processing is not a bottleneck. The fixed costs inside Pytorch for gradient estimation dominate the runtime. This indicates there is a large potential for optimization, even though the runtime are already reasonable. Sample timing results are shown in Table I.

### D. Initialization

The Initialization method depends on the application. If the head is restrained by some means like a head rest, the guess solution should be initialized to identity transform. For free head motion, the solution should be initialized with previous simple methods such as three-point alignment or ICP.

## IV. Results

### A. Performance on "ground truth" animations

We validated the approach using two convincing synthetic "semi-professional" facial animations (Figs. 6) for which the ground-truth head motion is provided by the animator (see Acknowledgments). Figs. 2, 5 validate the hypothesis that vertex motion exhibits a mode when viewed in the skull-rigid frame. On an animation with 651 of frames of rapid speech and head movement, using our algorithm to align the rigid-removed animation to the original animation, based on only vertices in the face region, resulted in a median error of 0.01cm and mean error of 0.03cm between the true and aligned vertices assuming the model has a standard inter-ocular distance of about 6cm. The three-point stabilization median error is 235% higher and the mean error is 79% higher.

### B. Performance on captured facial performances

Conveying results on real performances is difficult in a paper format, both because of the temporal nature of the problem and the lack of a quantitative ground truth (in standard mocap or video face reconstruction the true motion of the skull is unknown). Please see the accompanying video to fully assess the results. The video compares the results three-point stabilization, an ICP method, and our algorithm. The algorithms are shown running on short performances of the sort that might be used to build primitive expressions for an avatar. We selected several sequences that highlight the challenges of stabilization.

TABLE I: Timings

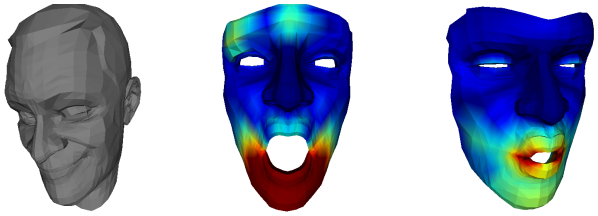
| Sequence | # Frames | # Vertices | Array size | Iters | Time (secs.) |
|---|---|---|---|---|---|
| Eyebrow | 63 | 2466 | 466074 | 12 | 105 |
| | | | | 24 | 198 |
| Jaw slide | 96 | 2466 | 710208 | 12 | 109 |
| | | | | 24 | 193 |
| Synthetic | 651 | 1185 | 2314305 | 12 | 126 |
| | | | | 24 | 261 |

For the three-point stabilization, we selected the corners of the eyes, and a point slightly below the tip of the nose that we observed to be relatively stable on the particular individual. The ICP algorithm considers only the forehead and nose bridge geometry, selected with the mask shown in Fig. 7. Our ICP algorithm also includes a temporal noise filtering component.

## V. Discussion and Conclusion

Our problem is different from, but share some concepts with, the problems of independent component analysis (ICA) and compressive sensing. In the classic form of ICA, several instances of a class of signals (e.g. several audio signals) are linearly combined. The central limit theorem effect of this linear combination causes the probability density of the mixture to be more Gaussian. Thus ICA seeks an un-mixing matrix that makes the recovered signals as non-Gaussian as possible. In our case, histograms with a clear mode are similarly non-Gaussian, however the "mixture" involves dissimilar signals (rigid transforms and expressions), and the rigid rotation is an *operator* on the expression. The classic form of compressed sensing seeks a solution to an under-determined linear system in which the solution is as sparse as possible. While an $\ell_1$ cost on the solution is typically used as a surrogate for sparsity, algorithms that seek a sparse "$\ell_0$" solution [11] involve considerations similar to the optimization presented in Section III.

In this paper we introduced a solution to the difficult problem of stabilizing facial motion capture. It proceeds from a simple and easily stated principle that has not previously appeared in the literature. Although the optimization approach requires care, it is easily implemented and our complete implementation is roughly 500 lines of Pytorch. Limitations of this work include the fact that it is oriented toward high-quality offline stabilization for avatar construction and may not be suitable for stabilization of real-time mocap. As with most ICP and related algorithms, careful initialization is needed. We have also observed it failing to stabilize a few "extreme" expressions such as Fig. 8. Nevertheless it generally works well, provides a clear improvement over existing approaches to



(a)

(b)

Fig. 6: Images from some sequences used in the evaluation. (a) Synthetic facial animation used to compute ground truth error. (b) Frame from a 4D-tracked human performance, showing the asymmetric jaw-slide expression that is difficult to stabilize.

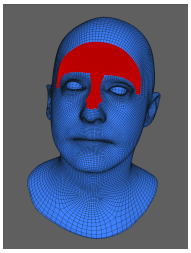


Fig. 7: Our baseline ICP algorithm is applied only to vertices in the masked area.

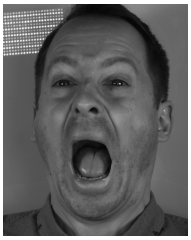


Fig. 8: With default settings the stabilization fails on this extreme pose. Changing the penalty width from the default 0.5mm to 1mm results in adequate results on this case, but we prefer to not manually adjust this parameter.

rigid alignment, and is considerably simpler than the previous published approach to this problem.

## Acknowledgments

The model used in the 405-frame animation was obtained from Prof. Hiroki Itokazu, California State University. The model for the "Strangelove" 651-frame animated sequence is from [24]. Animations were provided by Nickson Fong and Egg Story Creative Productions. We acknowledge valuable feedback from Binh Le and Javier von der Pahlen.

## References

[1] J. Noh and U. Neumann, "Expression cloning," in *SIGGRAPH 2001, Computer Graphics Proc.*, E. Fiume, Ed. ACM, 2001, pp. 277–288.

[2] P. Ekman and W. Friesen, *Manual for Facial Action Coding System*. Palo Alto, CA: Consulting Psychologists Press Inc., 1978.

[3] M. Sagar and R. Grossman, "Facial performance capture and expressive translation for King Kong," in *SIGGRAPH 2006 Sketches*, 2006.

[4] J. Lewis, K. Anjyo, T. Rhee, M. Zhang, F. Pighin, and Z. Deng, "Practice and theory of blendshape facial models," in *Eurographics*, 2014.

[5] M. Zollhöfer, J. Thies, P. Garrido, D. Bradley, T. Beeler, P. Pérez, M. Stamminger, M. Nießner, and C. Theobalt, "State of the art on monocular 3d face reconstruction, tracking, and applications," *Computer Graphics Forum*, vol. 37, no. 2, pp. 523–550.

[6] A. Smith, S. Pohle, W.-C. Ma, C. Ma, X.-C. Wu, Y. Chen, E. Danvoye, J. Jimenez, S. Patel, M. Sanders, and C. A. Wilson, "Emotion challenge: Building a new photoreal facial performance pipeline for games," in *DigiPro*. New York, NY, USA: ACM, 2017, pp. 8:1–8:2.

[7] D. Vlasic, M. Brand, H. Pfister, and J. Popović, "Face transfer with multilinear models," in *ACM Transactions on Graphics (TOG)*, vol. 24. New York, NY, USA: ACM Press, 2005, pp. 426–433.

[8] G. Golub and C. Van Loan, *Matrix Computation, third edition*. Baltimore and London: The John Hopkins University Press, 1996.

[9] J. Gower and G. Dijksterhuis, *Procrustes Problems*. OUP Oxford, 2004.

[10] T. Weise, H. Li, L. V. Gool, and M. Pauly, "Face/off: Live facial puppetry," in *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer animation (Proc. SCA'09)*. ETH Zurich: Eurographics Association, August 2009.

[11] S. Bouaziz, A. Tagliasacchi, and M. Pauly, "Sparse iterative closest point," in *Eurographics/ACMSIGGRAPH Symposium on Geometry Processing*, ser. SGP '13, 2013, pp. 113–123.

[12] T. Beeler and D. Bradley, "Rigid stabilization of facial expressions," *ACM Trans. Graph.*, vol. 33, no. 4, pp. 44:1–44:9, Jul. 2014.

[13] G. Fyffe, K. Nagano, L. Huynh, S. Saito, J. Busch, A. Jones, H. Li, and P. Debevec, "Multi-View Stereo on Consistent Face Topology," *Computer Graphics Forum*, vol. 36, no. 2, pp. 295–309, 2017.

[14] H. Li, J. Yu, Y. Ye, and C. Bregler, "Realtime facial animation with on-the-fly correctives," *ACM Transactions on Graphics*, vol. 32, no. 4, pp. 42:1–42:10, July 2013.

[15] S. Bouaziz, Y. Wang, and M. Pauly, "Online modeling for realtime facial animation," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 40:1–40:10, 2013.

[16] K. S. Bhat, R. Goldenthal, Y. Ye, R. Mallet, and M. Koperwas, "High fidelity facial animation capture and retargeting with contours," in *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '13)*. New York, NY, USA: ACM, 2013, pp. 7–14.

[17] A. H. Bermano, D. Bradley, T. Beeler, F. Zund, D. Nowrouzezahrai, I. Baran, O. Sorkine-Hornung, H. Pfister, R. W. Sumner, B. Bickel, and M. Gross, "Facial performance enhancement using dynamic shape space analysis," *ACM Trans. Graph.*, vol. 33, no. 2, pp. 13:1–13:12, Apr. 2014.

[18] G. Fyffe, T. Hawkins, C. Watts, W. Ma, and P. E. Debevec, "Comprehensive facial performance capture," *Comput. Graph. Forum*, vol. 30, no. 2, pp. 425–434, 2011.

[19] G. Fyffe, A. Jones, O. Alexander, R. Ichikari, P. Graham, K. Nagano, J. Busch, and P. Debevec, "Driving high-resolution facial blendshapes with video performance capture," in *ACM SIGGRAPH 2013 Talks*. ACM, 2013, p. 33.

[20] G. Levin, http://www.flong.com/texts/code/shapers_poly.

[21] L. Kavan, S. Collins, C. O'Sullivan, and J. Zara, "Dual quaternions for rigid transformation blending," *Trinity College Dublin, Tech. Rep. TCD-CS-2006-46*, 2006.

[22] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[23] C. De Boor, "A practical guide to splines, revised edition, vol. 27 of applied mathematical sciences," *Mechanical Sciences, year*, 2001.

[24] J. Osipa, *Stop Staring: Facial Modeling and Animation Done Right, 2nd Ed.* Sybex, 2007.