Interactive Exploration and Refinement of Facial Expression using Manifold Learning

Rinat Adbrashitov[†] **Fanny Chevalier** [†‡] **Karan Singh** [†]
[†] Department of Computer Science, [‡] Department of Statistical Sciences
University of Toronto
{rinat | fanny | karan}@dgp.toronto.edu

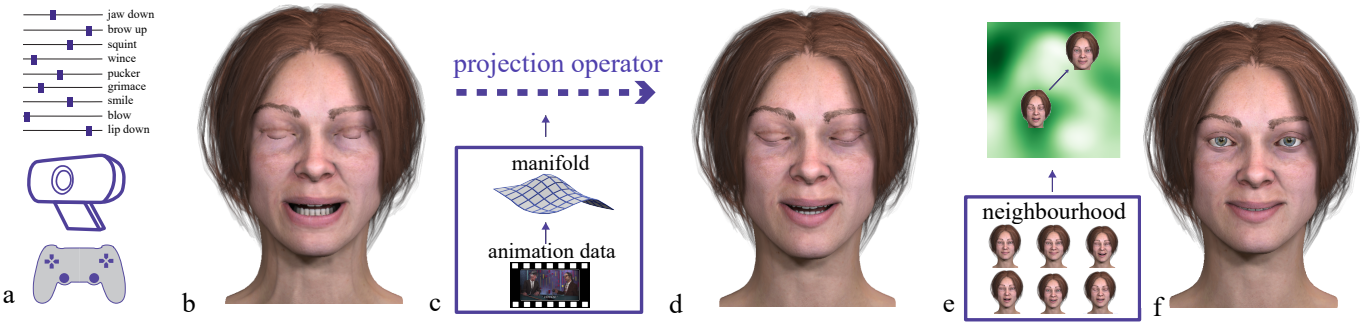

Figure 1. Interacting with high-dimensional face parameters using a variety of devices, is a challenging task (a), often resulting in implausible facial expressions: the eyelids inter-penetrate and the neck muscles are overstretched in (b). We address this problem by interactively projecting user created expressions onto a subspace of natural or stylized expressions, called a face manifold (c), learned using an autoencoder from existing facial animation data. Manifold projection enables rapid user exploration of the rich multidimensional subspace of expressive faces (d). These exploratory expressions can then be further adjusted interactively via a 2D embedding (e) that facilitates incremental refinement of expression on and off the manifold (f).

ABSTRACT

Posing expressive 3D faces is extremely challenging. Typical facial rigs have upwards of 30 controllable parameters, that while anatomically meaningful, are hard to use due to redundancy of expression, unrealistic configurations, and many semantic and stylistic correlations between the parameters. We propose a novel interface for rapid exploration and refinement of static facial expressions, based on a data-driven face manifold of "natural" expressions. Rapidly explored face configurations are interactively projected onto this manifold of meaningful expressions. These expressions can then be refined using a 2D embedding of nearby faces, both on and off the manifold. Our validation is fourfold: we show expressive face creation using various devices; we verify that our learnt manifold transcends its training face, to expressively control very different faces; we perform a crowd-sourced study to evaluate the quality of manifold face expressions; and we report on a usability study that shows our approach is an effective interactive tool to author facial expression.

Author Keywords

3D face modeling; blendshape; manifold learning.

INTRODUCTION

Emulating the complex expression conveyed by a human face, on digital 3D models, is a difficult task for actors and animators alike. Subtle mistakes in facial nuance can plunge an animated character into the Uncanny Valley, where the audience loses trust and empathy with the character [40]. The importance of realistic CG human faces, however, cannot be overstated. A diverse set of applications ranging from entertainment, medicine, education, and avatars, are all empowered by the ability to model, pose, and simulate human faces.

The facial rig is often comprised of a large (typically 30+) set of target blendshapes representing various facial configurations, for eg. those of the Facial Action Coding System (FACS) [17, 34]. Facial expressions result from a user controlled weight interpolation of these targets. The onus of manipulating these weights to author complex facial expressions, however, remains tedious in the hands of the user.

Individual blendshapes while semantically meaningful, often exhibit correlations, conflicts and significant redundancy. Even identifying a constituent subset of active blendshapes that model an expression, can be ambiguous and difficult. Sequentially experimenting with such a parameter space through independently operated weight sliders is particularly painful since only a small non-linear subspace of blendshape weights correspond to meaningful facial expressions. This makes unconstrained exploration of all blendshape weight configurations prone to generating implausible expressions. It is useful to restrict exploration to the subspace of plausible face configurations only. This allows users with limited understanding of blendshapes, face anatomy, or facial animation experience to produce complex expressions faster and easier.

Real-world high dimensional data (such as images or blendshape weights) lie on a low-dimensional subspace embedded in the high-dimensional space. In the data science literature that low-dimensional subspace is referred to as a manifold. We do not presume any user expertise in facial anatomy or animation. Instead, we capture this domain expertise as a manifold of **"natural"** (realistic or stylized) faces, as represented by a training corpus of sample expressive faces. Our face manifold is learnt from the corpus using a denoising autoencoder, and encoded by the latent space of the network [51]. Restricting interactive manipulated faces to this subspace or manifold of plausible configurations (see Fig. 1) addresses the above parameter space problems, and keeps facial expressions consistent for a given character. Current face manipulation interfaces ranging from a simple array of weight sliders, to image or video based performance capture [26, 52], direct manipulation [35], and sketch-based posing of faces [22], can all benefit from our approach by minimally altering their output to project onto a desired face manifold (see Fig. 1(a-c)).

Limiting faces to always lie within the potentially incomplete space of a data-driven manifold however, can be over constraining. We therefore allow iterative refinement of expression, by sampling the neighbourhood of an exploratory face, both on and off the manifold to create an interactive 2D embedding of nearby faces, with qualitative feedback on their distance to the face manifold (see Fig. 1(e)). The refined face can be used to produce subsequent neighbourhood embeddings, as well as bookmark specific faces for future use. A collection of expressive face bookmarks can be added to the data corpus to re-train the face manifold, bootstrapping the construction of a character face manifold.

Our implementation of manifold construction, projection and refinement is shown on a variety of inputs such as a face camera, game controller, and traditional sliders. Our manifold trained on a FACS-like rig, is also shown to be face agnostic, i.e. the learnt manifold of blendshape weights can meaningfully control face geometries quite different from the one used in training. We use a crowd-sourced study to evaluate the quality of the manifold in capturing expressive faces, and its impact on fixing unnatural faces. Finally, we perform a 10-user usability study that shows our approach to be an effective interactive tool to author facial expression, that compares favorably to the interactive control of independent face parameters.

Our main contributions are: 1) A data-driven model to capture a subspace of natural face expressions, and an operator to project faces onto this manifold; 2) A demonstration that interactive manifold projection can enable the rapid exploration of meaningful facial expressions by untrained novices; 3) A refinement interface to interactively manipulate the high-dimensional face space in the vicinity of an expression, both on and off the manifold.

RELATED WORK

We review frameworks and tools for creating facial expressions (Fig. 2), and discuss prior work on denoising methods, and systems for the exploration of large dimensional spaces.

Method \ Feature	Blendshape based	Doesnt require hardware	Doesnt manipulate geometry	Restricts exploration to natural faces	Allows refinment within neighbourhood	Doesnt require crowd source comparisons
Motion Capture	●	●	●	●	●	●
RGBD cameras	●	●	●	●	●	●
Slider interfaces	●	●	●	●	●	●
Sketch interfaces	●	●	●	●	●	●
Direct manipulation	●	●	●	●	●	●
Motion Attenuation	●	●	●	●	●	●
2D Face manifolds	●	●	●	●	●	●
Design galleries	●	●	●	●	●	●
VisOpt interface	●	●	●	●	●	●
Our method	●	●	●	●	●	●

Figure 2. Comparing our method to other approaches for posing faces.

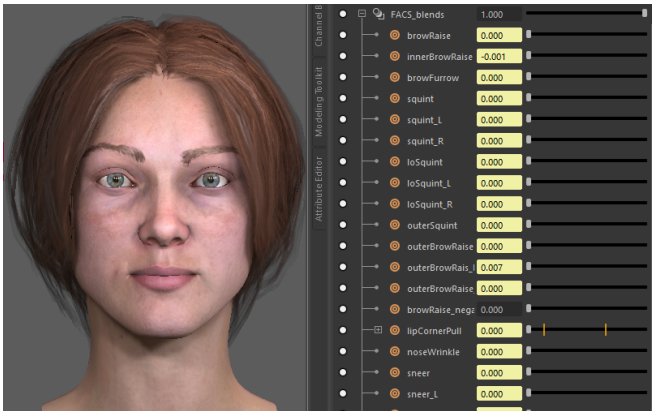

Figure 3. A face rig in Maya® with sliders for blendshape weights.

Animating 3D face geometry

Blendshape rigs are arguably the most popular choice for realistic or stylized character animation [34]. Each blendshape is a target face, such as a smile, pucker, or squint, representing the result of facial action(s) on a neutral face. A facial expression or pose is the weighted linear combination of these blendshapes. Traditional face posing interfaces in software like Blender or Autodesk Maya®, are thus simple arrays of sliders and widgets, for animators to control blendshape weights (Fig. 3). Individually and sequentially manipulating blendshape weights, and even locating a desired target in the widget array is tedious. Over time animators thus memorize the function and layout of hundreds of sliders to efficiently pose a face. Professional 3D animation unsurprisingly requires about one hour of labor per second of animation [35].

Face posing systems

Performance or motion capture systems use the motion of a human actor to drive a 3D face model [53]. These methods track 3D markers on an actor's face via a multi-camera system, and compute time-varying blendshape weights that match the moving markers, to drive the animated expression of a synthetic 3D face [6, 16, 59]. High-end performance capture typically requires substantial imaging hardware and manual clean-up of the resulting face animation. Approaches based on ordinary web cameras [9, 19, 26] or low-cost depth cameras [7, 37, 38, 52] have been proposed to overcome cumbersome motion capture setup. These methods are easier to deploy, but produce coarser results and can be non-trivial to map to arbi-

trary 3D face geometry. Motion capture methods are strongly dependent on camera hardware and actor performance. They also do not inherently restrict geometry to lie within any subspace of meaningful expressions, and can thus benefit from our work as a post projective step (Fig. 9a).

Sketch-based, and direct manipulation techniques have been proposed to overcome the tedium of sequentially controlling individual sliders. Face Poser [31] combines user input with a facial prior learned from a prerecorded facial expression database. Such systems use screen-space 2D point, stroke and curve constraints to pose a 3D facial model. Sketch Express [43] is a facial sketching interface based on drawing simple strokes in predefined 2D drawing regions. Direct manipulation methods [35] pose 3D faces by manipulating points on the geometry and inverse computing blendshape weights to match the points. Similarly, the motion of user specified points can be minimized, when optimizing blendshape weights to mitigate self-collisions and geometric mangling due to blendshape interference [36]. 2D portrait manipulation interfaces [48] map the manipulation of predefined curves of a 2D portrait, to drive 3D marker-based facial expressions. These approaches map user interaction to a subspace of plausible facial expressions using geometric and collision constraints. While these approaches are better than unconstrained manipulation, they tend to be face geometry specific and are unable to capture non-geometric or stylistic constraints. Our data-driven manifold approach addresses both these issues, and integrates well with existing animation workflows.

Face Manifolds

Face manifolds have been explored in various 2D contexts based on 2D landmarks or image pixel data [11, 12, 13, 45, 55], and based on the geometric sampling of the 3D facial surface in 3D [15]. In contrast, our approach is blendshape based: an animation-data-driven, sparse, and compact subspace of blendshape weights that capture facial expression.

Denoising autoencoders

Machine learning denoising methods are increasingly applied in computer graphics for a variety of purposes. Denoising is an unsupervised learning technique used to train autoencoders to detect structure in the given input [51]. Denoising autoencoders have been applied to problems such as in-painting [57, 58], denoising Monte-Carlo renderings [3, 10] and fixing corrupted animation data [23, 41, 54].

Our work is inspired by recent advances in skeletal-based animation [24, 25], that use convolutional autoencoders to learn a manifold from a motion capture database. Projecting invalid or corrupt data onto this manifold can fix motion errors and reconstruct natural movement. Additionally, high-level character constraints such as a locomotion trajectory or the target of a punch, can be mapped to the motion manifold, to produce motion that is both goal-directed and natural. We use a similar approach, utilizing denoising autoencoders to learn a manifold of static facial expressions.

Exploration of high-dimensional spaces

Exploring high-dimensional design spaces for creative authoring tasks is difficult. The baseline approach, is still to manipulate a collection of uni-dimensional sliders. Smart Suggestion and VisOpt Slider interfaces [29] use a goodness function, to interactively visualize the distribution of goodness in the design space. The goodness function is obtained via crowdsourcing pairwise comparisons between different parameter configurations. High-quality 3D models can be created by estimating the distribution of good models in a design space, where goodness is based on tracking the modeling activity of a distributed community of users [49]. In contrast, we directly learn the space of natural facial expressions from pre-existing datasets, an approach that is cost-effective and scales well to high dimensional parameter spaces.

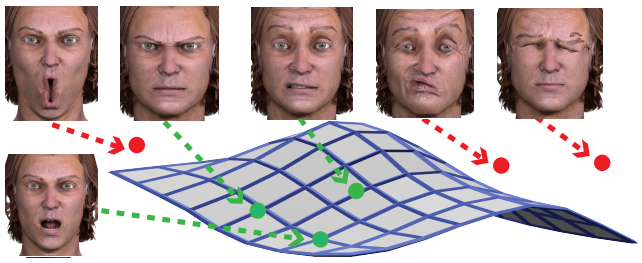

Figure 4. Multiple conflicting blendshapes are activated which produces unnatural face (outside of the face manifold). While faces that belong to the manifold are natural.

Design galleries present suggestions, to iteratively guide users towards desired parameter configurations, by asking them to rank or choose from multiple candidate configurations [20, 33, 8, 60]. In contrast, we want the user to be able to rapidly explore the face space interactively and experiment with, or further refine their choices.

FACE MODEL

Creating a particular facial expression usually involves the manipulation of a *face model* — a reference face rig featuring a neutral expression that can be modified using a set of user controls, which are typically blendshape weights. Mathematically, such a face model is a simple vector sum as follows:

$$\mathbf{f} = \mathbf{b}_0 + \sum_{k=0}^{n} w_k(\mathbf{b}_k - \mathbf{b}_0) \quad (1)$$

where $\mathbf{f} \in R^n$ is the resulting face, $\mathbf{b}_0$ is the neutral face, and $\mathbf{b}_k \in \mathbb{R}^n$ are target blendshapes with weights $w_k \in \mathbb{R}$, (usually $[0,1]$), that modify $\mathbf{b}_0$.

Blendshapes are user intuitive but non-orthogonal, and sparse representation. This means that different blendshapes may be redundant in their action as well as conflict with one another, for eg. activating a "squint" and "eyebrows up" simulataneously (Fig. 5(right)). It is thus important to restrict exploration to the subspace of plausible face configurations, which we call the face manifold (Fig. 4).

MANIFOLD LEARNING

Our definition of realistic or stylized "natural" expressions is a subspace or manifold based on an input corpus of posed faces. Therefore, the degree of naturalness and plausibility is fully determined by the training data. Formally, given a facial expression $\mathbf{x} \in \mathbb{R}^n$ where x_i is the weight of each of

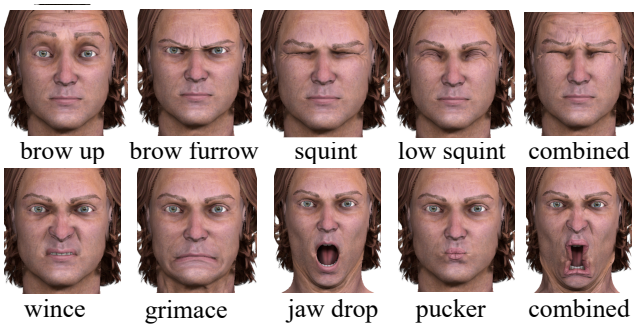

Figure 5. Top: Example of the blendshapes affecting the top part of the face where the last face is a combination of the 4 blendshapes. Bottom: Example of the blendshapes affecting the bottom part of the face where the last face is a combination of the 4 bottom blendshapes.

n blendshapes, we wish to learn a manifold projection operator $\mathbf{y} = \Phi(\mathbf{x}), \mathbf{y} \in \mathbb{R}^m$ and an inverse projection operator $\hat{\mathbf{x}} = \hat{\Phi}(\mathbf{y}), \hat{\mathbf{x}} \in \mathbb{R}^n$. The inverse projection operation $\hat{\Phi}$ tries to produce an $\hat{\mathbf{x}}$ which lies within the subspace of valid facial expressions. We learn the projection operation Φ and its inverse $\hat{\Phi}$ using a denoising autoencoder and therefore $\mathbf{x}$ corresponds to the visible units and the hidden units at the deepest layer correspond to $\mathbf{y}$ (Fig. 6). Propagating the resulting values of the hidden units at the deepest level backward to reconstruct visible units corresponds to the inverse operation $\hat{\Phi}$.

Data pre-processing and representation

In our implementation, we collected facial animation data from the animated movie *Subconscious Password* [30], which encompasses a collection of shots, each ranging from 20 to 1,000 frames at 24 frames per second. We exclude blendshapes that were not used in the animation process, obtaining a final set of 31 blendshapes and the neutral face. We assume that all faces contained in the training data are "natural" (realistic or stylized) because they were manually authored.

Each frame provides a 31D weight vector, which we separate into two groups affecting the top (10 blendshapes), and bottom part (21 blendshapes) of the face (Fig. 5). This decision is motivated by multiple factors: the motion affecting the top and bottom parts of the face are largely separable and under independent anatomic control [1]; input data will likely be incomplete in terms of capturing all combinations of upper and lower face expressions; training a manifold where samples have the same top and varying bottom expressions could lead to a single average bottom expression as the network output; users often wish to control the expression on the two parts in isolation from each other. Following these observations, we train separate autoencoders for the two parts of the face.

While our dataset contains a large number and variety of facial expressions, it is imbalanced as some expressions appear more often that others. For example, close-to neutral expressions greatly outnumber smiling expressions. Imbalanced datasets can result in bias and overfitting, where the network learns to perform well on one set of expressions and ignore others. We mitigate this problem by first running a k-means algorithm [39] to separate the data into clusters. We force all clusters to have a similar number of samples by oversampling the minority clusters and undersampling the majority clusters, using the

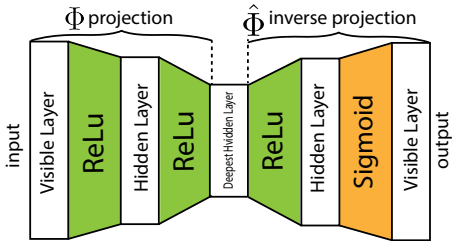

Figure 6. Overview of our denoising autoencoder. We train two such autoencoders for the top and bottom part of the face.

Synthetic Minority Over-sampling Technique (SMOTE) [14]. To undersample a cluster, we randomly pick samples to be removed. We use a balanced set of 10,000 samples to train our face manifold.

Denoising Autoencoder for Learning Motion Data

Autoencoders exploit the idea that the data is concentrated around a low-dimensional manifold or a small set of manifolds, and aim to learn the structure of those manifolds [21]. Autoencoders impose a bottleneck at the deepest hidden layer which forces a compressed knowledge representation of the original input, to capture any structural correlations in the training data. Denoising autoencoders are an extension of classical autoencoders but are exposed to partial corruption of the input pattern during training, to better handle noisy device input in our context.

We train 2 denoising autoencoders, one for the top and one for the bottom half of the face. The input size for the top is 10 dimensional and for the bottom is 21. Each autoencoder contains 3 hidden layers with different number of units (top: 7, 4, 7; bottom: 12, 7, 12). For a single hidden layer k, filter weights $\mathbf{u}_k$ and biases $\mathbf{t}_k$, the projection operation from input vector $\mathbf{x}$ to hidden units is given by:

$$\Phi_k(\mathbf{x}) = ReLU(\mathbf{u}_k * \mathbf{x} + \mathbf{t}_k) \tag{2}$$

except for the last layer, where we use a sigmoid activation function (Fig. 6). The number of hidden layers and the size of each hidden layer in the autoencoder was determined during training by choosing the best combination that produced the smallest loss error (Eq. 3) on the validation data.

Training

Denoising autoencoders are typically trained to retrieve an original input, given an altered version of such input. To train our denoising autoencoder, we corrupt input vectors $\mathbf{x}$ to a new vector $\mathbf{x}_c$ by adding Gaussian noise, and the network is trained to reproduce the original input $\mathbf{x}$. We find network parameters $\theta = (\mathbf{u}, \mathbf{t})$ for each layer such that the following loss function measuring the squared reproduction error and the L_1 norm of the output vector of activations ($\hat{x}$) is minimized. The L_1 norm term acts as a regularizer to ensure that the final output of the blendshapes weight is sparse. In other words we want the output expressions to be achieved by activating the least number of blendshapes as possible [34]. The amount of sparsity is controlled by the λ parameter (0.1 in our experiments).

$$Loss(\mathbf{x}) = ||\mathbf{x} - \hat{\Phi}(\Phi(\mathbf{x}))||_2^2 + \lambda \sum_i |\hat{x}_i| \tag{3}$$

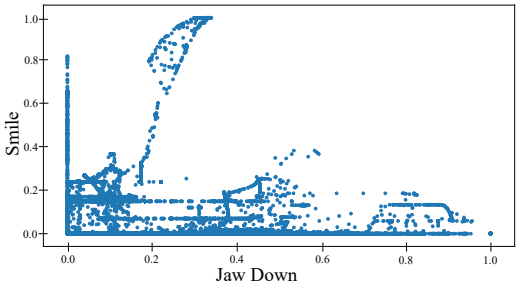

Figure 7. Training data for 2 different blendshapes (jaw down and smile). The relationship between the blendshapes is highly nonlinear.

Figure 8. Comparison of our method vs. PCA. Left: Unnatural face. Middle: Result of applying the PCA still seems unnatural. Right: Our method brings the expression closer to the input dataset.

To minimize this function we perform a stochastic gradient descent. We input the corrupted elements $\mathbf{x}_c$ from the database, and using autograd system in Pytorch to calculate derivatives of each node in the network, we update the network parameters θ [44]. We use the adaptive gradient descent algorithm Adam [27] to improve training speed and quality. Training is performed for 150 epochs on an Nvidia GeForce GTX 1080 GPU.

Overfitting

Overfitting can be a concern when the training dataset size is not large. While the size of our dataset is comparable to other training datasets described in learning literature [50, 2], we would like our approach to be applicable to smaller or evolving datasets of character animation data.

We address overfitting by: balancing the dataset to avoid over-representation of certain expressions; training two separate autoencoders for the top and bottom face, that have a lower dimensionality than a single autoencoder for the full face (Fig. 6); and using a Dropout [47] of 0.2 during training.

Design choices

Face manifolds tend to be non-convex in shape and capture complex non-linear relationships (Fig.7) between blendshapes. Autoencoders provide a more powerful representation for such structure than baseline approaches such as PCA (Principal Component Analysis) (Fig. 8). Variational autoencoders learn the parameters of a probability distribution representing the training data. This learned distribution can then be sampled to produce novel data but is not well suited to our goal of denoising the corrupted input. We experimented with general neural networks and autoencoders (without Gaussian noise) but given that we want our approach to be applicable to a variety of interfaces and device inputs, we found denoising autoencoders to produce the smallest reconstruction error and show the most resilience to imperfect input.

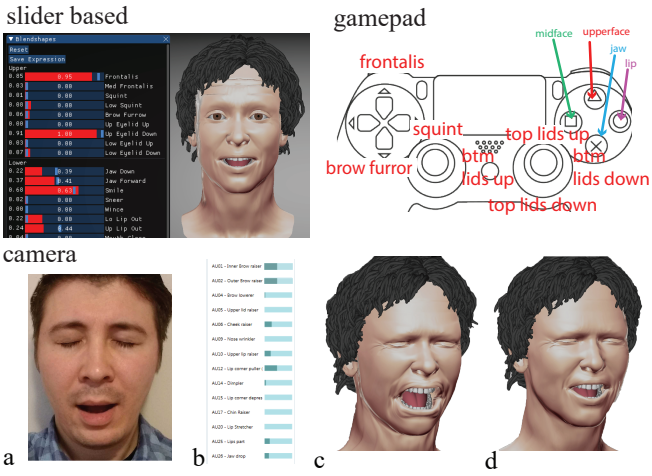

Figure 9. *Slider, Gamepad*: Manifold projection augmented slider and gamepad interfaces for posing faces. *Camera*: from an image of a face (a), face muscle activations can be estimated (b), that may produce implausible results (c), corrected by manifold projection (d).

FACE EXPLORATION

Given a face manifold trained on pre-existing data, we can project any face input onto the manifold. Faces may be posed using input from sources such as web cameras, hand or face trackers, game controllers, or a traditional weight slider based interface. The subspace of meaningful faces is small relative to space of all configurations and projecting input onto this manifold keeps a user from getting lost in a sea of meaningless and mangled faces. We experimented with 3 manifold projection augmented interfaces: a face camera, traditional sliders, and a gamepad controller.

Slider interface

Slider-based interfaces are dominant in facial animation practice. We augment such interfaces by automatically applying our manifold projection operator to ensure that users never leave the subspace of natural facial expression (Fig.9). This enables rapid exploration on the manifold and also aids novices in understanding blendshape correlations when posing faces. For example, increasing "Jaw forward", also visibly increases "Jaw drop" upon manifold projection, to capture the natural behavior of an open jaw. The current projected values used to pose the face are shown beside the slider, and as red bars in the slider background.

Camera interface

There is extensive research in analyzing facial expressions from from images [42]. OpenFace [4, 5] is able to estimate values for 15 blendshapes from an image of a face. Directly using these blendshape weights to pose 3D faces does not always produce plausible results since the outcome depends on quality of the image, lighting conditions, quality of the data set used for training, and the robustness of analysis algorithm. We apply our manifold projection technique to produce more natural facial expressions and fix corrupt input (Fig.9).

Gamepad controller

Gamepad controllers can simultaneously control multiple blendshape weights to pose avatar faces in games. Figure 9 shows an example layout for controlling blendshapes using a gamepad. Simultaneous control of many blendshapes is fast but more susceptible to producing unnatural faces (see accompanying video), which are readily corrected by manifold projection. Our approach thus enables the use of a wider range of device input for posing faces.

FACE REFINEMENT

Most approaches in facial animation solely encompass a face exploration stage (without correction by a manifold). However, regardless of the input method used, and even with our correction through projection onto the manifold, face posing via manipulation of blendshape weights remains imprecise due to the complexity of combining multiple blendshapes. After posing the face, the user may find that the result is close to an envisioned facial expression, but may still require fine-tuning to fit their artistic vision. We thus propose a novel interface to face refinement by learning a latent space embedding only within the vicinity of an explored face. This embedding allows users to visualize and interactively refine the current facial expression or produce new highly exaggerated expressions.

Latent space embedding

We create a 2D latent space around a given facial expression where similar expressions can be embedded and intuitively explored. More formally, given a facial expression $\mathbf{x}_0$, we generate a set of similar expressions $[\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, ..., \mathbf{x}_k]$ where $\mathbf{x}_i \in \mathbb{R}^n$. A naive approach to generating the set of similar expressions is to sample the subspace around a given expression. However, since face expressions are high dimensional, where $\mathbf{x}_0 \in \mathbb{R}^n$ with possibly $n > 30$, a lot of samples would need to be produced and most of them would not look natural. Instead, we sample the neighbourhood following these 3 rules:

1. We identify the list of activated blendshapes whose weight is above a certain threshold $\tau = 0.15$. We uniformly sample along each activated blendshape axis, such that if i is an index of the blendshape verifying $x_i > \tau$, then the sampling boundary is defined as $[x_i - \tau, x_i + \tau]$. We repeat the sampling for each activated blendshape individually to produces a set of samples $S1$. This ensures that users will be able to refine their facial pose along each activated blendshape individually.

2. We build a set $S2$ by uniformly sampling along all activated blendshape axes at the same time. This ensures that we have samples where multiple blendshape weights are varying.

3. We look at the set of previously user-created faces which we call *bookmarks*. We find a set of bookmarked faces that are within a small threshold away from the given facial expression to produce a set S3. Adding *bookmarks* allows user to effortlessly bring previous faces that they considered plausible and quickly generate similar results.

The final set of neighbours S is the union of sets $S1-3$. To enable interactive exploration of this space, we require an adequate representation that both conveys information to the user, while supporting easy navigation. An obvious solution consists of mapping this result space on a 2D representation that can be visualized and interacted with. Non-linear manifold learning techniques come as a natural choice for this kind of problem. Gaussian process latent variable models (GPLVM)[32] is a probabilistic dimensionality reduction method that is able to find a lower dimensional non-linear embedding of high dimensional data.

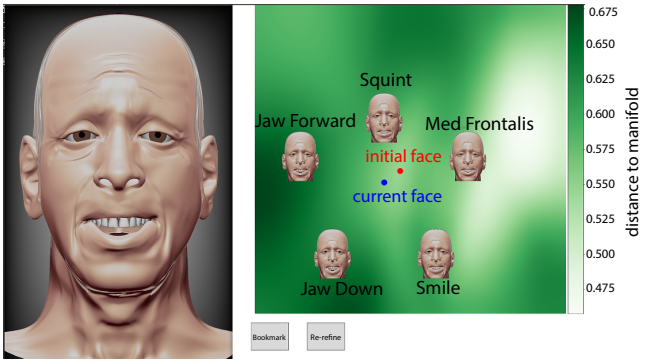

Figure 10. Refinement UI. The user can interactively explore 2D latent space and see the resulting face on the left. Before the user starts to drag, a set of neighbouring faces are shown to help build expectations of what will changed if moving in different directions.

Our projection operator can be used to measure the distance between the facial expression and the face manifold to determine how plausible the facial expression is. This enables an informative visualization of the latent space where the distance to the facial manifold presented in the form of color coding and highlights regions of the latent space that move far away or lie closer to the manifold (Fig. 10).

Interactive refinement in latent space

Given the 2D latent space visualized as a distance to manifold color map, we enable interactive navigation through dragging a pointer inside the 2D space to generate new latent coordinates which are projected back to the high dimensional blendshape space. This output is visualized dynamically, which provides real-time feedback supporting efficient exploration of the neighbourhood of the initial facial pose (Fig. 10).

To further guide the user in their exploration, we provide an additional visual aid in the form of images of the neighbouring faces. These images are shown when the current face position is clicked on, and aim to give an intuitive cue on how the face would change if the user were to move in that direction, guiding them the right way to obtain the desired pose. The location of each image is determined by sampling the neighbourhood along 5 different directions and determining the point along each direction that is the furthest distance away from the current pose. Above each image we show the name of the muscle that changes the most when moving to that area from the current position. If the user finds a plausible face but wants to see additional options, they can click the re-refine button to generate a new latent space from this face.

Design Choices

While the face manifold can be trained offline on a large dataset, the face refinement interface needs to be able to gener-

Figure 11. The projection operator fails to find a satisfactory result when the expression is missing from the training data. (a) Neutral expression (b) *Buccinator* muscle activated. (c) The projection is far from (b).

ate its latent space "on the fly" with little data of only the neighbouring faces (sets $S1$-3). Therefore, deep learning approaches used to build the face manifold are not suitable. GPLVM meets various criteria needed for our refinement interface: creates a non-linear mapping between 2D latent space and the blendshape space, requires little data (about 10 faces), requires short training time (seconds) to quickly re-refine around a chosen expression or add new custom expressions, interpolates between the embedded expressions for continuous exploration. PCA is not well suited to model the non-linear mapping from the very low dimensional latent space to the high dimensional blendshape space and other non-linear dimensionality techniques like Eigenmaps, t-SNE, LLE are less suited to interpolating between the embedded expressions.

MANIFOLD QUALITY ASSESSMENT

The quality of the produced manifold can be evaluated in multiple ways: a "good" manifold should be expressive, exclude unnatural faces, and generalize to a variety of rigs.

Expressiveness. Our manifold captures the expressive range of the given face, as represented by the input data. A limited dataset can unintentionally leave out meaningful expressions, or intentionally impose an expressive style on a face since the learned manifold is highly representative of the input data. For example, our training data is missing expressions where the *buccinator* muscle—the major facial muscle underlying the cheek that aids whistling and smiling—is highly activated. When applying the projection, the resulting face is closer to a neutral expression (see Fig. 11). Incomplete manifolds can always be improved with a richer data-set, or by adding novel expressions created by our refinement interface to the original training data and recomputing a more expressive manifold.

Unnatural faces. Another quality measure is the representation of undesired or unnatural faces by the manifold. Our manifold is trained by minimizing the error difference between facial expressions in the input data and their corrupted counterparts. We comprehensively evaluated the presence of unnatural faces in the manifold and the quality of the projection operator via a crowd-sourced study on Amazon Mturk (40 participants, excluding subjects who failed the understanding test below). Each participant was shown a different set of 100 images of facial expressions, one image at a time, and asked (yes/no) if the face looked natural (example faces were shown to provide meaning to the word "natural"). Images of facial expressions were generated as follows:

- *Random sample set:* 35 randomly sampled faces. Each expression $f \in R^n$ ws generated by uniformly sampling the n dimensional face space.

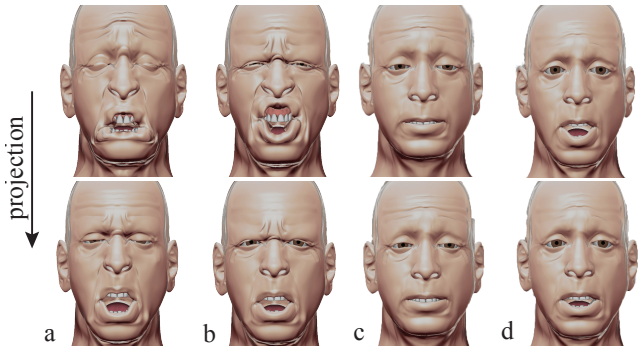

Figure 12. Unnatural faces (a,b) are projected onto the manifold to find the closest natural expression. Natural expressions (c,d) remain natural after the projection.

- *Projection set:* projections of the 35 faces from the *random sample set* onto the manifold.
- *Repeat set:* 10 faces repeated from each of the above two sets, to assess consistency of participant response. Data of subjects (2) who inconsistently answered on 5 or more repeats was excluded.
- *Understanding set:* 5 natural (fully activating a single blendshape corresponding to a natural expression) and 5 unnatural faces (extreme self-intersecting geometry), manually chosen to test user understanding of the task. We excluded the data of participants (4) who incorrectly answered to 2 or more of these trials.

Table 1 shows a summary of the study results. Of the 1,400 random samples (40 participants presented with a different set of 35 random faces), only a few were found to be natural. In contrast, after projection on the manifold, these random faces were largely perceived as natural, suggesting that our projection operator finds the closest natural expression with an average distance in blendshape space of 1.8 between to original face, and its projected counterpart (Fig. 12a,b). Further, the faces from the random sample set that were found natural, were also found natural after projection, with an average distance of 0.24 between the pairs (Fig. 12c,d). This suggests that while our manifold is a good proxy for natural faces, there are also meaningful facial expressions that lie off the manifold, which can be reached by our refinement interface.

% of natural faces in the *Random set*	8% (112/1400)
% of natural faces in the *Projection set*	87% (1218/1400)
% of unnatural faces in the *Random set* that became natural after projection	89% (1146/1288)
% of natural faces in the *Random set* that remained natural after projection	98% (109/112)

Table 1. Results of the crowd-sourced Amazon Mturk study evaluating the quality of the manifold based on the unnatural faces criteria.

Generality. Our manifold was trained on the animation data for the character in Figure 10. However, as long as the source and target rigs have corresponding/similar blendshape semantics, face manifold does generalize well across different human rigs (see Fig. 1, 8, 9, 13 and accompanying video).

Figure 13. Our manifold generalizes well to other facial rigs that are based on Facial Action Coding System.

INTERFACE EVALUATION

We conducted a study with participants who had a limited understanding of blendshapes, face anatomy, or any facial animation experience, in order to evaluate the potential of our approach in aiding such users in posing facial expressions. The study lasted 1 hour and we compensated participants with a $25 Amazon gift card.

Apparatus

We use the Pytorch Python library to perform data preprocessing tasks and train our autoencoder model. Once the model is trained it is serialized and ready to be loaded in a C++ application. The main exploration interface was written in C++/OpenGL using ImGui for the UI components. A separate Python-based application showed the refinement UI (the 2D latent space). Both applications were run on the same machine simultaneously, and utilized web sockets for intercommunication. The system ran on a computer with Intel Xeon 16 core 3.5Ghz CPU, GTX1080 GPU and 64GB of RAM with monitor at a 3840x2160 resolution. The participants used a mouse and a keyboard to interact with the interface.

Participants

Ten participants (6 female, 4 male; Age: 23-29) took part in our study: 5 graduate students and 5 participants who have full-time jobs not related to animation. Six participants reported having some knowledge about computer animation techniques but no experience with facial animation specifically.

Study protocol

Our study aimed to evaluate three aspects of our interface:
A1. Discoverability: how easy it is for a user to use the interface with minimal to no guidance.
A2. Usability: how effective it is to pose face expressions.
A3. Expressivity: how hard it is to pose desired, diverse and meaningful expressions.

To this effect, we designed two tasks. **Task 1** compares two conditions: the **projection**+**refinement** condition, using our slider based interface with manifold **projection** and the **refinement** interface together, and the **baseline** condition, using the traditional slider interface without manifold projection. **Task 2** was an open-ended exploration task, where we only study the **projection**+**refinement** condition.

Task 1 focused on **A1** and **A2**. Participants were given a reference facial expression (Fig.14(left)), and instructed to produce an accurate replica. Participants were not instructed to perform the task as fast as possible, but were given a time limit

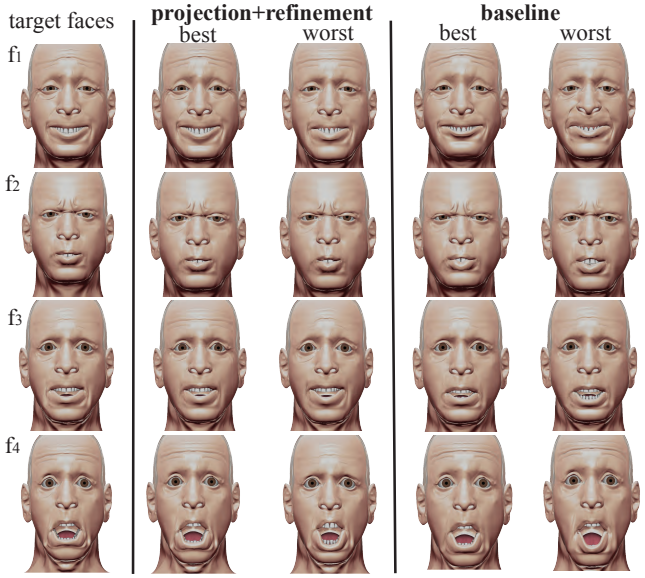

Figure 14. Four faces (f_1, f_2, f_3, f_4) were shown to the participants during the comparative study. For each target face we show best (distance wise) and worst faces created by the participants per condition.

(3min) to perform the task. They could also submit their work before the countdown when satisfied with the result. Four different faces f_1, f_2, f_3, f_4 were shown to each participant as follows: f_1, f_2 were part of the training data set and belong to the manifold; f_3, f_4 were not present in the training set and were a small distance (0.2) off the manifold (Fig. 14). For every pair (f_i, f_j), the distance $d(f_i, f_j)$ was greater than a diversity threshold 1.0. Each participant was randomly assigned the condition order of **projection**+**refinement** $\rightarrow$ **baseline**, or **baseline** $\rightarrow$ **projection**+**refinement**. The order of presentation of the faces (f_i) was generated randomly for each participant. Participants completed the task with the first condition, and then again with the other condition. We chose to use the same order of presentation of faces for each condition to maximize the number of trials between two same facial expressions, to mitigate any learning/memory effects. Each condition was prefaced with a 3-5 min training session for that interface. We administered a questionnaire after each condition using a 5-point Likert scale (1–strongly disagree, 5–strongly agree).

Task2 focused on **A3** using the **projection**+**refinement** condition only. There are 6 universal emotions: happiness, sadness, anger, surprise, fear, and disgust. Participants were each randomly given two emotions, and instructed to generate two different variations of each emotion within 5 min. The participant could generate a new face either from the last recorded face, or starting from a neutral expression. Figure 16 shows some of the user created faces.

Results

Overall, all participants successfully completed the tasks and created facial expressions within the time limit.

Task 1 – Quantitative results

Figure 15 (left) shows the mean distance error from each of the target faces f_i and overall, per condition. Point estimate and 95% confidence intervals were computed using bootstrapping

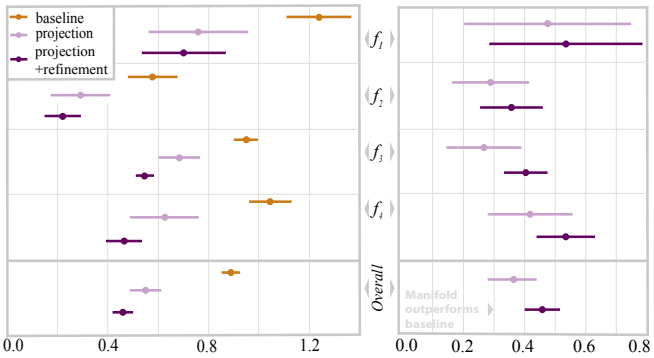

Figure 15. Mean distance error from the target faces (f_1, ..., f_4) (left). Mean distance error difference between the baseline and our method with and without refinement (right). Error bars are 95% CIs.

face	baseline	projection	projection+refinement
f_1	1.24 (std 0.33)	0.76 (std 0.32)	0.70 (std 0.21)
f_2	0.57 (std 0.17)	0.30 (std 0.18)	0.22 (std 0.14)
f_3	0.95 (std 0.11)	0.69 (std 0.14)	0.54 (std 0.12)
f_4	1.04 (std 0.14)	0.63 (std 0.24)	0.50 (std 0.21)

Table 2. Mean distance error in blendshape space.

[28]. The higher values mean higher error, i.e. higher distance from the target face. The right figure contrasts our approach with the baseline: point estimates and intervals are estimates of the difference of mean distance error between the baseline and our method, computed for each participant (this is possible given our within-subjects design). In this figure, positive values mean that our method outperforms the baseline.

On average the participants were able to get closer to the target faces using our **projection** slider interface (Fig. 15, Table 2) with a smaller number of sliders being activated (Table 3). This can be explained by the manifold activating related muscles without the user needing to explicitly drag the sliders and manually activating them. The refinement interface further allowed participants to fine tune expressions closer to the target. Additionally, participants were able to reach the coarse expression resembling the target face (distance to the target $\leq$ 1.0) faster with our **projection** slider interface (Table 4).

Discoverability

Participants found the system easy to use (4 strongly agree, 6 agree) and learn (5 strongly agree, 3 agree, 1 neutral, 1 disagree), and that the various functions in the system were well integrated (3 strongly agree, 5 agree, 2 neutral). As P5 noted, the *"learning curve was not hard"* and interacting with the system *"felt like playing a game"*. P1 mentioned that they can see themselves *"spending a lot of time playing with the system"* and that the *"lack of previous experience in facial animation didn't cause any problems"*. Most participants found the interface *"fun to use"* (P1, P2, P3, P5, P7, P8).

face	baseline	projection
f_1	9.1	7
f_2	6.1	4.9
f_3	7.3	6.1
f_4	8.7	7.6

Table 3. Mean number of sliders activated (value >0.05).

face	baseline	projection	refinement
f_1	110 CI[102 ,119]	92 CI[85,99]	21 CI[19,26]
f_2	104 CI[96,112]	60 CI[54,66]	26 CI[22,30]
f_3	101 CI[92,109]	86 CI[79,92]	25 CI[21,30]
f_4	125 CI[117,132]	92 CI[83,102]	22 CI[19,26]

Table 4. Mean time and 95% CIs until the distance to target reached a threshold of 1.0 and mean time spent refining the face (in sec).

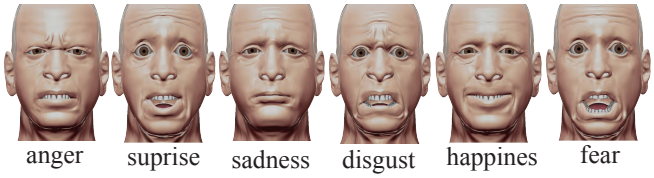

Figure 16. Examples of different emotions that were created by the participants.

Usability and Expressivity

Most participants felt that the interface allowed them to create the expression they wanted (4 strongly agree, 5 agree, 1 neutral). A common theme among users was that the **projection** slider interface allowed them to create a rough expression matching the target quicker and easier than the **baseline** (P1, P2, P3, P4, P5, P7, P10). P5 liked that the interface *"adjusted related muscles for you, so you don't have to think afterwards about additional muscles you need to activate"* while P1 mentioned that there was *"less tweaking involved"* and that *"they didn't have to go through all the sliders to get the result."*

The **refinement** interface proved to be useful for fine tuning the facial expression with 9/10 participants agreeing that it helped them create better expressions. P4 and P5 agreed that the **refinement** interface *"gives you a lot of options that you didn't think of before"* and that those options were *"better than what they originally created"*. The distance to manifold color coding received mixed feedback with 5/10 users saying that they factored the color coding into their final choice and that they found faces closer to manifold more natural. 5/10 users did not use the color coding or used it every other time and preferred just exploring the the whole space and didn't mind choosing faces that were further away from the manifold. Majority of the participants (8/10) found images of neighbouring faces useful and often relied on them to navigate the latent space. Sometimes the images seemed similar and users mentioned that they relied on the label (indicating the main muscle that was changing).

The biggest contrast between the **projection** slider interface and the **baseline** was creation of unnatural faces when changing slider values. Users found that they were often creating unnatural faces with **baseline** interface (1 strongly agree, 6 agree, 2 neutral, 1 disagree) compared to **manifold** interface (2 strongly disagree, 7 disagree, 1 agree). While talking about the posing process, P3 mentioned that manifold **projection** *"makes the slider adjustment feel more natural"*.

Slider responsiveness

Most participants (8/10) found that the **projection** slider interface was more difficult to use once they created a rough expression of the face and they wanted to make small isolated changes. They often found that making a small change to one muscle would change other muscles as well that they didn't necessarily want. P4 mentioned that *"some faces were harder to make than the others"* and it would have been *"helpful to move sliders in isolation"*. This was the most common feedback for faces f_3, f_4 that are not on the manifold.

Suggestions for Improvement

Study participants gave useful feedback on how we can improve our system. Some (P3,P4,P6,P7) mentioned that they wanted to be able to turn off projection for the isolated control of the individual muscles after posing the face. P5 suggested to add a toggle key-bind to the **refinement** interface to quickly switch between the current face and the original face to see how the face changed/improved. Some users (P2,P4,P8) wanted to see a more extensive list of muscles that are changing when moving to certain areas of the latent space instead of only knowing the one muscle that changes the most.

DISCUSSION

Our approach to the authoring and interactive control of facial expressions differs from prior work in two main ways. First, we provide dynamic auto-correction of unnatural faces into similar but meaningful facial expressions through projection onto a face manifold. Second, we support finer control of facial expression by interacting with a dynamically computed manifold that captures the neighborhood of the currently explored face.

Sense of control. Getting a strong sense of control is critical to creative authoring [46]. Constraining the user/artist to be on the space of the learnt manifold does take away an element of control over the output. During our evaluation however, participants commented on only wanting isolated control of the weights after coarse posing was done.

Our goal is to support users in their creative endeavour, but such support comes at the cost of taking away some decisions. There is a sweet spot between providing enough assistance so that the tool is actually helpful, and too much assistance that can take away sense of control and in some cases, sense of ownership of the created artifact. Prior work supporting guided authoring of sketches for novices, such as sketch-sketch revolution [18] and PortraitSketch [56] also discuss these trade offs. In our case, however, the user can always refine the face to lie off the manifold.

Expressiveness. Our approach is limited by the quality and richness of the data used to build the manifold. While it can seen as a constraint (and it certainly is in some cases), this also makes it possible for the user to use a specific dataset as a way to *intentionally* bias the manifold to generate expressions that conform to a particular style.

During an informal evaluation of our approach with a senior professional animator, he pointed out that "*large-scale animation productions commonly impose a character style guide to ensure the integrity of a character authored by different animators*". He referred to this as *animating "on-model" or "off-model"* in industry parlance, and said it is precisely what our concept of "on/off-manifold" captures.

User Demographics. Our tool for interactive exploration on and around a face manifold serves two goals: first, manifold projection allows users to pose faces quicker, i.e. manipulating a few parameters is enough for remaining parameters to automatically conform to produce an expressive face; second, the manifold itself captures the desired range of expression or "on-model" for a character's face. While expert animators certainly benefit from visualizing how their posed faces relate to the given "on-model" character style, they are already adept at posing nuanced and expressive faces quickly, and sometimes prefer to control all face parameters independently. We chose to formally evaluate our approach with novices and animation enthusiasts, who have little or no understanding of facial anatomy, to better measure the impact both on the quality of a posed face, as well the time speed-up that our tool offers.

We informally evaluated our approach with a professional animator. He provided a mix of high-level and detailed feedback. His overall feeling though was very positive. While he felt that our approach might not improve the quality of his work, or help him steer clear of unrealistic faces (he knows his rigs and its realistic subspace), it would certainly make him more efficient (exploring coarse expression rapidly). It would also allow him to work more effectively and uniformly with other animators on the same character, because the manifold would provide them with a common expressive baseline.

Limitations. The biggest limitation of our work is the reliance of our approach on a meaningful facial manifold. In the absence of a good representative data set of example face configurations, manifold projection will constrain the exploration to a small subspace of natural faces. Our definition of a "natural" expression is fully determined by the training data. To our knowledge, no open-source database of a large number of facial animations based only on FACS exists yet. Restricting the dataset to a single stylized character can introduce biases and force the manifold to adhere to the stylistic limitations. Ideally, to learn a general manifold, data should come from multiple characters with different styles. This would enrich a space of "natural" expressions and potentially improve the sense of control. However, there is risk of introducing conflicts between blendshape correlations which could results in the network outputting the average of different styles. This is a subject for future work.

Aside from using pre-existing animation data, the corpus can be built by showing crowd-sourced users random face configurations and asking them to accept or reject them as realistic. Importantly however, manifolds largely generalize across human faces and it may be easier to use existing face manifolds as a baseline for novel characters.

Another limitation is the lack of a guarantee that our refinement technique will precisely embed a desired facial expression. The embedding is however, constructed using a judicious sampling of the neighbourhood of a face including faces that are previously bookmarked, and those that lie both on and off the manifold. Users can further search this neighbourhood by incrementally re-embedding a refined face.

Given the importance of static expressive faces for applications like representative freeze or key-frames for film and animation, advertising billboards, custom emoji's and profiles for avatars in social media and games, the modeling of facial prosthetics, this paper remained focused on static facial expressions. An extension of this work to a dynamic or animated face manifold is subject to future work.

CONCLUSION

We have presented a workflow to address the interactive exploration and refinement of static facial expressions. Our overall contribution is a workflow for face posing that uses a manifold learnt from existing facial expression data to constrain a user's input, and a subsequent 2D interactive embedding of the neighbourhood of a face to refine its expression. This allows users with no previous experience in facial animation to produce natural faces poses with less effort. Our approach integrates seamlessly with a variety of high-dimensional input devices such as cameras, sliders and game controllers. Our evaluation shows the manifold to be a sound representation of the subspace of plausible faces, that can generalize across humanoid face models that are different from the training dataset. Finally, a user study shows our approach to face manifold exploration and refinement to compare favourably to independent slider manipulation. Faces are crucial to human visual communication and we see our approach to provide a novel and compelling workflow to aid users in understanding and authoring expressive 3D digital faces.

ACKNOWLEDGEMENTS

We wish to thank Chris Landreth for providing the data as well as facial rigs, the participants for their time and reviewers for their valuable feedback which helped improve this work.

REFERENCES

[1] Rinat Abdrashitov, Alec Jacobson, and Karan Singh. 2019. A system for efficient 3D printed stop-motion face animation. *ACM Transactions on Graphics (TOG)* 39, 1 (2019), 1–11.

[2] Stephen W Bailey, Dave Otte, Paul Dilorenzo, and James F O'Brien. 2018. Fast and deep deformation approximations. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–12.

[3] Steve Bako, Thijs Vogels, Brian Mcwilliams, Mark Meyer, Jan NováK, Alex Harvill, Pradeep Sen, Tony Derose, and Fabrice Rousselle. 2017. Kernel-predicting Convolutional Networks for Denoising Monte Carlo Renderings. *ACM Trans. Graph.* 36, 4, Article 97 (July 2017), 14 pages.

[4] Tadas Baltrušaitis, Marwa Mahmoud, and Peter Robinson. 2015. Cross-dataset learning and person-specific normalisation for automatic action unit detection. In *Automatic Face and Gesture Recognition (FG), 2015 11th IEEE International Conference and Workshops on*, Vol. 6. IEEE, 1–6.

[5] Tadas Baltrusaitis, Amir Zadeh, Yao Chong Lim, and Louis-Philippe Morency. 2018. Openface 2.0: Facial behavior analysis toolkit. In *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*. IEEE, 59–66.

[6] Kiran S Bhat, Rony Goldenthal, Yuting Ye, Ronald Mallet, and Michael Koperwas. 2013. High fidelity facial animation capture and retargeting with contours. In *Proceedings of the 12th ACM SIGGRAPH/eurographics symposium on computer animation*. ACM, 7–14.

[7] Sofien Bouaziz, Yangang Wang, and Mark Pauly. 2013. Online modeling for realtime facial animation. *ACM Transactions on Graphics (ToG)* 32, 4 (2013), 1–10.

[8] Eric Brochu, Tyson Brochu, and Nando de Freitas. 2010. A Bayesian interactive optimization approach to procedural animation design. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 103–112.

[9] Chen Cao, Yanlin Weng, Stephen Lin, and Kun Zhou. 2013. 3D shape regression for real-time facial animation. *ACM Transactions on Graphics (TOG)* 32, 4 (2013), 41.

[10] Chakravarty R. Alla Chaitanya, Anton S. Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. 2017. Interactive Reconstruction of Monte Carlo Image Sequences Using a Recurrent Denoising Autoencoder. *ACM Trans. Graph.* 36, 4, Article 98 (July 2017), 12 pages.

[11] Ya Chang, Changbo Hu, Rogerio Feris, and Matthew Turk. 2006. Manifold based analysis of facial expression. *Image and Vision Computing* 24, 6 (2006), 605–614.

[12] Ya Chang, Changbo Hu, and Matthew Turk. 2003. Manifold of facial expression.. In *AMFG*. 28–35.

[13] Ya Chang, Changbo Hu, and Matthew Turk. 2004. Probabilistic expression analysis on manifolds. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, Vol. 2. IEEE, II–II.

[14] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16 (2002), 321–357.

[15] Girum G Demisse, Djamila Aouada, and Björn Ottersten. 2018. Deformation-Based 3D Facial Expression Representation. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 14, 1s (2018), 1–22.

[16] Zhigang Deng, Pei-Ying Chiang, Pamela Fox, and Ulrich Neumann. 2006. Animating blendshape faces by cross-mapping motion capture data. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games*. ACM, 43–48.

[17] Rosenberg Ekman. 1997. *What the face reveals: Basic and applied studies of spontaneous expression using the Facial Action Coding System (FACS)*. Oxford University Press, USA.

[18] Jennifer Fernquist, Tovi Grossman, and George Fitzmaurice. 2011. Sketch-sketch revolution: an engaging tutorial system for guided sketching and application learning. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, 373–382.

[19] Pablo Garrido, Levi Valgaerts, Chenglei Wu, and Christian Theobalt. 2013. Reconstructing detailed dynamic face geometry from monocular video. *ACM Trans. Graph.* 32, 6 (2013), 158–1.

[20] Sarah Gibson, Paul Beardsley, Wheeler Ruml, Thomas Kang, Brian Mirtich, Joshua Seims, William Freeman, Jessica Hodgins, Hanspeter Pfister, Joe Marks, and others. 1997. Design galleries: A general approach to setting parameters for computer graphics and animation. (1997).

[21] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

[22] X. Han, C. Gao, and Y. Yu. 2017. DeepSketch2Face: A Deep Learning Based Sketching System for 3D Face and Caricature Modeling. *ACM Transactions on Graphics* 36, 4 (July 2017).

[23] Daniel Holden. 2018. Robust Solving of Optical Motion Capture Data by Denoising. *ACM Trans. Graph.* 37, 4, Article 165 (July 2018), 12 pages.

[24] Daniel Holden, Jun Saito, and Taku Komura. 2016. A deep learning framework for character motion synthesis and editing. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 138.

[25] Daniel Holden, Jun Saito, Taku Komura, and Thomas Joyce. 2015. Learning motion manifolds with convolutional autoencoders. In *SIGGRAPH Asia 2015 Technical Briefs*. ACM, 18.

[26] Liwen Hu, Shunsuke Saito, Lingyu Wei, Koki Nagano, Jaewoo Seo, Jens Fursund, Iman Sadeghi, Carrie Sun, Yen-Chun Chen, and Hao Li. 2017. Avatar digitization from a single image for real-time rendering. *ACM Transactions on Graphics (TOG)* 36, 6 (2017), 195.

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

[28] Kris N Kirby and Daniel Gerlanc. 2013. BootES: An R package for bootstrap confidence intervals on effect sizes. *Behavior research methods* 45, 4 (2013), 905–927.

[29] Yuki Koyama, Daisuke Sakamoto, and Takeo Igarashi. 2014. Crowd-powered parameter analysis for visual design exploration. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*. ACM, 65–74.

[30] Chris Landreth. 2013. Subconscious Password. National Film Board of Canada.

[31] Manfred Lau, Jinxiang Chai, Ying-Qing Xu, and Heung-Yeung Shum. 2009. Face poser: Interactive modeling of 3D facial expressions using facial priors. *ACM Transactions on Graphics (TOG)* 29, 1 (2009), 3.

[32] Neil D Lawrence. 2004. Gaussian process latent variable models for visualisation of high dimensional data. In *Advances in neural information processing systems*. 329–336.

[33] Brian Lee, Savil Srivastava, Ranjitha Kumar, Ronen Brafman, and Scott R Klemmer. 2010. Designing with interactive example galleries. In *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 2257–2266.

[34] John P Lewis, Ken Anjyo, Taehyun Rhee, Mengjie Zhang, Frederic H Pighin, and Zhigang Deng. 2014. Practice and Theory of Blendshape Facial Models. *Eurographics (State of the Art Reports)* 1, 8 (2014), 2.

[35] John P Lewis and Ken-ichi Anjyo. 2010. Direct manipulation blendshapes. *IEEE Computer Graphics and Applications* 30, 4 (2010), 42–50.

[36] John P Lewis, Jonathan Mooser, Zhigang Deng, and Ulrich Neumann. 2005. Reducing blendshape interference by selected motion attenuation. In *Proceedings of the 2005 symposium on Interactive 3D graphics and games*. 25–29.

[37] Hao Li, Laura Trutoiu, Kyle Olszewski, Lingyu Wei, Tristan Trutna, Pei-Lun Hsieh, Aaron Nicholls, and Chongyang Ma. 2015. Facial performance sensing head-mounted display. *ACM Transactions on Graphics (ToG)* 34, 4 (2015), 47.

[38] Hao Li, Jihun Yu, Yuting Ye, and Chris Bregler. 2013. Realtime facial animation with on-the-fly correctives. *ACM Trans. Graph.* 32, 4 (2013), 42–1.

[39] Stuart Lloyd. 1982. Least squares quantization in PCM. *IEEE transactions on information theory* 28, 2 (1982), 129–137.

[40] Karl F. MacDorman, Robert D. Green, Chin-Chang Ho, and Clinton T. Koch. 2009. Too real for comfort? Uncanny responses to computer generated faces. *Computers in Human Behavior* 25, 3 (2009).

[41] Utkarsh Mall, G Roshan Lal, Siddhartha Chaudhuri, and Parag Chaudhuri. 2017. A deep recurrent framework for cleaning motion capture data. *arXiv preprint arXiv:1712.03380* (2017).

[42] Brais Martinez, Michel F Valstar, Bihan Jiang, and Maja Pantic. 2017. Automatic analysis of facial actions: A survey. *IEEE transactions on affective computing* (2017).

[43] José Carlos Miranda, Xenxo Alvarez, João Orvalho, Diego Gutierrez, A Augusto Sousa, and Verónica Orvalho. 2011. Sketch express: facial expressions made easy. In *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling*. ACM, 87–94.

[44] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS-W*.

[45] Caifeng Shan, Shaogang Gong, and Peter W McOwan. 2005. Appearance manifold of facial expression. In *International Workshop on Human-Computer Interaction*. Springer, 221–230.

[46] Ben Shneiderman. 2007. Creativity support tools: Accelerating discovery and innovation. *Commun. ACM* 50, 12 (2007), 20–32.

[47] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[48] Tanasai Sucontphunt, Zhenyao Mo, Ulrich Neumann, and Zhigang Deng. 2008. Interactive 3D facial expression posing through 2D portrait manipulation. In *Proceedings of graphics interface 2008*. Canadian Information Processing Society, 177–184.

[49] Jerry O Talton, Daniel Gibson, Lingfeng Yang, Pat Hanrahan, and Vladlen Koltun. 2009. Exploratory modeling with collaborative design spaces. *ACM Transactions on Graphics-TOG* 28, 5 (2009), 167.

[50] Nobuyuki Umetani. 2017. Exploring generative 3D shapes using autoencoder networks. In *SIGGRAPH Asia 2017 Technical Briefs*. 1–4.

[51] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *J. Mach. Learn. Res.* 11 (Dec. 2010), 3371–3408. http://dl.acm.org/citation.cfm?id=1756006.1953039

[52] Thibaut Weise, Sofien Bouaziz, Hao Li, and Mark Pauly. 2011. Realtime performance-based facial animation. In *ACM transactions on graphics (TOG)*, Vol. 30. ACM, 77.

[53] Lance Williams. 1990. Performance-driven facial animation. In *ACM SIGGRAPH Computer Graphics*, Vol. 24. ACM, 235–242.

[54] Jun Xiao, Yinfu Feng, Mingming Ji, Xiaosong Yang, Jian J. Zhang, and Yueting Zhuang. 2015. Sparse Motion Bases Selection for Human Motion Denoising. *Signal Process.* 110, C (May 2015), 108–122.

[55] Rui Xiao, Qijun Zhao, David Zhang, and Pengfei Shi. 2011. Facial expression recognition on multiple manifolds. *Pattern Recognition* 44, 1 (2011), 107–116.

[56] Jun Xie, Aaron Hertzmann, Wilmot Li, and Holger Winnemöller. 2014. PortraitSketch: face sketching assistance for novices. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*. ACM, 407–417.

[57] Junyuan Xie, Linli Xu, and Enhong Chen. 2012. Image denoising and inpainting with deep neural networks. In *Advances in neural information processing systems*. 341–349.

[58] Raymond A Yeh, Chen Chen, Teck Yian Lim, Alexander G Schwing, Mark Hasegawa-Johnson, and Minh N Do. 2017. Semantic image inpainting with deep generative models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5485–5493.

[59] Eduard Zell, JP Lewis, Junyong Noh, Mario Botsch, and others. 2017. Facial retargeting with automatic range of motion alignment. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 154.

[60] Károly Zsolnai-Fehér, Peter Wonka, and Michael Wimmer. 2018. Gaussian material synthesis. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 76.