User-Guided Facial Animation through an Evolutionary Interface

K. Reed and D. Cosker

Department of Computer Science, University of Bath, Bath, UK
{k.h.p.reed, d.p.cosker}@bath.ac.uk

Abstract
We propose a design framework to assist with user-generated content in facial animation — without requiring any animation experience or ground truth reference. Where conventional prototyping methods rely on handcrafting by experienced animators, our approach looks to encode the role of the animator as an Evolutionary Algorithm acting on animation controls, driven by visual feedback from a user. Presented as a simple interface, users sample control combinations and select favourable results to influence later sampling. Over multiple iterations of disregarding unfavourable control values, parameters converge towards the user’s ideal. We demonstrate our framework through two non-trivial applications: creating highly nuanced expressions by evolving control values of a face rig and non-linear motion through evolving control point positions of animation curves.

Keywords: animation interfaces, facial animation, genetic algorithms, user-generated content

ACM CCS: Animation → Animation Systems; Facial Animation; Methods and Applications → Genetic Algorithms

1. Introduction

In computer graphics, despite the prevalent use of parametric models, there is relatively little work on how to explore these shape spaces towards optimal results. A highly elusive example can be found in facial animation, where the control of a parametric face rig is limited to the expert knowledge of animators. This is a highly manual process, involving an animator hand-tweaking the many controls of a face rig at keyframes over time. Though the use of performance capture is commonly used to help automate this pipeline, many tasks have an ambiguous element that relies on an animator’s judgement - such as ‘tweaking’ correctives and experimental prototyping. However, as facial expressions are highly subjective and difficult to communicate verbally, there might be discrepancies in the interpretation between the animator and other key stakeholders. Hence, the use of a user-guided system could allow creative stakeholders to prototype their vision directly, removing this ambiguity and aiding workflow. Such a prototyping tool that produces animation without any animation experience could also be useful for: researchers from psychology and social sciences and non-technical practitioners in a range of disciplines (inside and outside the creative sector).

Our work presents a design framework that can expedite experimental tasks in facial animation, while being accessible to general users. Our method involves encoding a parametric face rig into an evolutionary algorithm (EA) — driven by user validation of observed samples from the model (as in Figure 1). EAs present a global optimization scheme based on reinforcement learning, suited to non-linear high-dimensional problems, such as facial expression. Of interest to our work is their powerful ability to adopt simple search heuristics, i.e. user selection replacing the need for reference training data making them ideal for experimental prototyping. Despite the simplicity of this heuristic, humans are experts at processing facial expression, allowing our evolutionary scheme to be highly discriminative while generalizing to arbitrary users. Furthermore, as our evolutionary interface treats the model as a black-box function, our design is easily adapted to various parametric models that can be assessed through appearance. To demonstrate our framework, we task unskilled users in animation to produce nuanced expressions and non-linear dynamics, with this paper contributing to the following:

- A novel method for user-guided authoring and editing of static expressions by applying an EA to parameters of a face rig.
- A novel user-guided method for crafting facial dynamics through applying an EA to the sampling of animation curves.
- A design outline to explore parametric spaces and streamline prototyping tasks involving visual feedback.

© 2019 The Authors. Computer Graphics Forum published by Eurographics - The European Association for Computer Graphics and John Wiley & Sons Ltd
This is an open access article under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.
2. Related Work

Facial animation pipeline A traditional facial animation pipeline involves an animator crafting expressions at keyframes through an intuitive face rig, which are interpolated to produce facial dynamics. An industrial standard for rigging the face is the use of blendshapes. Blendshapes are simple vector offsets of a mesh from a neutral source \( \mathbf{b}_0 \) to a target expression \( \mathbf{b}_k \), which can be combined linearly to cover an extensive range of motion for very few parameters (a full review can be found in [LAR*14]). This is controlled via weights \( w_k \), typically ranging between \([0,1]\), such that any face expression \( \mathbf{F} \) can be constructed as follows, for \( K \) blendshapes:

\[
\mathbf{F} = \mathbf{b}_0 + \sum_{k=1}^{K} w_k (\mathbf{b}_k - \mathbf{b}_0). \tag{1}
\]

For the face, a rig consists of many blendshapes of a character at unique expressions, e.g. ‘eyebrows raised’, typically governed by FACS [EF78] and visualized into an interpretable slider interface. Commonly, a set of these controls encodes non-deformable components, such as eyes and jaw, which are modelled separately as rigid joints. Without compromising on interpretability, sophisticated rigs can also facilitate shape correctives and conflicts [LMDN05], physical simulation [KBB*17] and non-linear dynamics [3La].

Traditional animation interfaces Animators typically control an animated sequence via two interfaces: the face rig and the animation curve editor. The face rig offers an extensive and intuitive control of face shape at a single keyframe. It is typically presented as an array of sliders, arranged into the muscle actuators they represent. Though the rig is fairly interpretable, blendshapes are numerous and forming precise expressions relies on animation experience. Direct manipulation methods [LA10, SILN11, ATL12] resolve this ambiguity by offering a pin-and-drag manipulation on the 3D model, by solving an inverse problem of blendshape weights for a segmented input face. Though this offers great precision, setting pins is still a time-consuming task. To refine facial dynamics, an animator can use an animation curve editor, offering a global view of all keyframes in the sequence. For each rig control, how its intensity changes over time can be visualized as a 2D curve. Keyframes produce control points on the curve, which can be easily tuned to control facial dynamics. Though this allows for more extensive control, using many blendshapes can lead to curve combinations that are difficult to interpret for novice users.

User-guided techniques As an alternative approach, user-guided tools provide interfaces that do not rely on animation experience. Sketch-based posing presents a high-level approach to controlling a 3D model by drawing 2D sketches or actuator paths [SMND08, MAO*12, HMC*15] and has been used to author motion directly [CLS*16, CGNS17, DBB*18]. Though far more accessible to new users, it does rely on the user to ‘recall’ the composition of expression which is slower and less reliable than our ability of facial ‘recognition’ [BY86, GS84, SBOR06]. To this end, to take advantage of this ability, an alternative is to display multiple choices for a user to observe. A typical approach, commonly used in procedural generation, involves sampling from a constrained model for experimental content generation. There are several examples of producing constrained models for faces and bodies through data-driven methods, such as dimensionality reduction to cover a wide range of shape and motion as a few latent variables. For example, PCA [BV99, CH05], GPLVMs [GMHP04, WFH07] and autoencoders [DWW15, FLFM15, HSK16]. However, these latent parameters are non-intuitive and there has been little work in enabling users to explore and interpret these shape spaces. Verbal crowd-shaping is a simple example of such, used to generate descriptive body types [SQRH*16], by learning a regression between sampled parameters and descriptive labels from a population. However, this relies on a vast training dataset and refining precise results is restricted to the use of simple descriptions, which might not be suited to the nuanced and subjective nature of faces.

Evolutionary approaches From the limitations discussed with current animation interfaces, we turn to the highly established field of e-fit generation; where adopting an evolutionary interface for the experimental task of refining facial identity [Gib03, FHC04] has shown to be a proficient solution in facial perception studies [FCN*05]. For this example, users guide evolution of PCA parameters for face shape and texture through a face selection interface. Covered in detail in Section 4, EAs are highly adaptable optimization frameworks suited to highly complex search problems. As a reinforcement learning approach, the algorithm relies on mass sampling of parameter combinations and an evaluation scheme to assess the quality of outcomes. As is the case for holistic e-fit systems, this evaluation scheme can be adapted to be user guided. Despite

![Image](52x609 to 549x731)

Figure 1: Prototyping a new expression through our framework by evolving starting expression (left) to desired target (right). This is through several iterations of sampling face rig controls, with favourable results selected by a user (denoted as ticks) to influence later sampling.
their flexibility, EAs have been rarely used in animation: featuring chiefly in procedural generation of offspring characters [Sim94, DiP02, PJ08] and biological structures [HSS17]. In this paper, we propose how an evolutionary interface can be extended to tasks in facial animation. To our knowledge, ours is the first to demonstrate how an EA framework can be used to evolve the shape and dynamics of facial expression.

3. Method Overview

Our system works by encoding an EA into a simple interface which can be used to aid users with prototyping new facial expressions and their dynamics. Detailed in Section 4, the EA works through several iterations of sampling, selection, breeding and mutation to converge — or evolve — parameters towards the user’s ideal.

To adapt this framework for facial animation, we use rig control values as parameters to evolve. By treating the rig as a black box system; we can generalize to arbitrary rig interfaces and leverage underlying rig intelligence without requiring knowledge of how these mechanics work. Given such a face rig (following a logical namespace hierarchy) in Autodesk Maya, our system starts by creating a dictionary of rig control values in the scene.

For prototyping new expressions, we sample new control configurations, with results visualized to the user for review. Favourable faces are selected by the user to influence further sampling until convergence towards the user’s ideal. This concept can be extended to prototyping dynamic expressions, by creating experimental animation curves and sampling control points along the curve, which are analogous to rig control intensities at keyframes. Alongside Figure 2, our pipeline can be summarized as the following:

1. **Initialization:** Each tool is initialized by a shared plugin which parses information about the scene, including the naming hierarchy of the rig and their starting control values. As input, users also supply the current frame number and which rig groups, e.g. brow, mouth and nose, they wish to include in evolution. Finally, the GUI to sample and visualize faces is generated.

2. **Sampling:** For the first generation, new parameter configurations are sampled at random from the selected rig groups and visualized to the user. For later generations, samples are taken from the pool of saved trials from previous generation.

3. **User evaluation and selection:** Users experiment with many samples. When the user sees a favourable result, they can save them to influence later sampling.

4. **Breeding and mutation:** Once the required number of selections has been met, we proceed to the next generation by constraining the current sampling pool with samples selected by user. The parameters of these samples are first bred (parameter swapping) and mutated (uniform resampling) to encourage variation.

5. **Convergence:** The user continues with successive phases of sampling, selection, and evolution until satisfied with the generated facial appearance or animation.

For the remainder of this paper, we introduce the foundations of the EA framework (Section 4) and how it can be applied to facial animation. We then demonstrate several applications of its use (Section 6) including: user-authoring and editing of facial expressions and dynamics. To improve performance and generalization of our framework, we also devise control schemas that govern the use of constraints during evolution (Section 5).

4. Evolutionary Framework

The EA is a framework which aims to resemble natural selection, where the ‘best’ genes in a population propagate through to subsequent generations. Genes are representations of what we are trying to optimize, typically a string of multiple parameter values. Analogous to natural selection, genes are evolved through iterative cycles...
of the following routines: generation, evaluation, selection, breeding and mutation. In this manner, the search space (or gene pool) converges to an optimal set of genes via eliminating genes with a poor fitness over several iterations.

Underpinning the success of an EA transpires how we define genes and these evolution routines. For our framework, we represent genes as a string of rig control values, which is synonymous to the physical appearance of the face (Section 4.1). As summarized in Algorithm 1, the algorithm is initialized by first sampling new control configurations, creating new face expressions to display to the user (Section 4.2). Next, instead of using an error-based fitness function for evaluation and selection, we combine these into a user selection phase of most favourable samples (Section 4.3) and evolve their parameters through recombination (Section 4.4).

4.1. Gene representation

In the traditional literature [Hol92], genes come as binary encodings (relating to actual gene strings), though any numerical encoding works in the same manner. A blendshape rig (as in Equation (1)) is a suitable candidate for encoding face shape, where a face can be represented as a series of control weights \( W \) for \( K \) controls. For keyframe animations, we can represent animation as a sequencing of control values, forming animation curves for each control via interpolation of \( T \) keyframes. To generalize to both shape and dynamics, for our framework we use animation curves as our gene representations \( c \), therefore compacting observed animation \( C \) of multiple curves (and static shape when \( T = 1 \)) as the following:

\[
c = \{w_1, \ldots, w_T\}
\]
\[
C = \{c_1, \ldots, c_K\}. \tag{2}
\]

4.2. Sampling

To initialize the EA, we define two sets. Set \( G \) here represents the current search space (or gene pool) — all possible parameter combinations which sampling can select from. Set \( S \) is a trial face by sampling weights/curves from the gene pool. For our interface, we show multiple trials at one time for faster convergence. Therefore, sets so far follow the following relationship:

\[
w_{ik}^j \in c_k \in C \in S \subset G. \tag{3}
\]

At the first evolution phase, samples will be very varied. Initialization might have specific implementation details per tool (see Section 6), but in general will sample new rig control values between a range of \([0,1]\) such that:

\[
S = \{w_k \mid w_k \sim U([0, 1]), 1 \leq k \leq K\}. \tag{4}
\]

Over the course of evolution, the gene pool is constrained as a set of selected samples. Generating samples can be generalized as sampling gene configurations from the updated gene pool, where \( S \) is a stochastic selection scheme outlined in Section 4.4:

\[
S = \{w_k \mid w_k \sim S(G_k), 1 \leq k \leq K\}. \tag{5}
\]

4.3. Evaluation and selection

In the traditional literature, evaluation occurs by determining a cost function for each sampled face, where selection consists of preferring genes which attribute the lowest error. As our task is experimental, targets are not well defined so there is no cost function to drive selection. Instead, we employ user selection as a gene selection scheme by picking faces with most desired characteristics.

To drive evolution, users are presented with multiple trials and select those with favourable results denoted as \( S' \). The genes of these samples are then added to an updated gene pool \( G \), where \( i \) denotes the evolution phase number and \( M \) is the number of samples selected by the user:

\[
G^{(i+1)} = \{S'_1, \ldots, S'_M\}. \tag{6}
\]

The user is also prompted to choose the elite face \( E \), being the best sample they have witnessed from current evolution phase. The elite face is a unique sample as has much higher influence on subsequent sampling. For easy reference, we put this at the start of the current gene pool such that \( G_i = E \in S' \).

4.4. Evolution

Once the user has selected multiple samples and an elite, they can proceed to the next evolution phase by updating the gene pool such that \( G \leftarrow G^{(i+1)} \). After the first evolution phase, we introduce a new sampling scheme \( S \) for sampling new genes; which generates new samples through breeding, mutation or both.

Breeding of genes \( B \) involves parameter cross-over of selected faces, akin to chromosomes during meiosis. Two parent samples are selected at random \( \{P_1, P_2\} \in G \), where each parent gene has a 50:50 chance of being chosen for the child, as in Equation (7). The elite face is encoded to have a much higher likelihood of being selected as a parent. For our examples, this is akin to swapping animation curves of different regions of the face.

\[
B(P_1, P_2) = \begin{cases} \text{elite} & \text{if } \gamma \sim U([0, 1]) < 0.5 \\ c_k \in P_1 & \text{otherwise} \\ c_k \in P_2 & \text{otherwise} \end{cases} \tag{7}
\]

As with meiosis, there is also a chance of mutation \( M \) within the genes. This is where there is a small possibility \( \gamma \) that a gene can change to a value not seen in either parent. Together with crossing-breeding, this offers a perfect degree of variability seen in the children, needed to explore the search space.

\[
M(S, \gamma) = \begin{cases} w_{ik}^j \in S & \text{if } \gamma \sim U([0, 1]) < \gamma \\ w_{ik}^j \in S & \text{otherwise} \end{cases} \tag{8}
\]

After several iterations of the described schemes, samples from the gene pool should start to converge locally towards the user-driven target; though there is no guarantee that this will be the optimal solution or how many iterations evolution will take.

5. System Optimization

As sampling from face rigs presents a considerably large search space, we devise several constraints that can be used by our system.
includes some other important pa-
in Maya to include. Throughout evolution, we also
in range \([0,1]\) which
\(\gamma\) (Section 5.2) to define the management of
rig
to cull rig controls that have a value of 0 in the scene.
T
controls, the current search complexity is
uniform sampling
selection limit
control schemas
For certain tasks, we might wish to make sure
K
mutation rate
flag,
keyframes) when involving temporal
strongest shape
Advanced Control Panel
Symmetry
value. This is particularly effective for ‘tweaking’ shapes.
\([0,1]\) for each rig control, we sample uniformly around the current

5.1. Constraints
For rigs using \(K\) controls, the current search complexity is \(K!\) with
each value for \(K\) being a float in the range \([0,1]\). This is increased
substantially (by a factor of \(T\) keyframes) when involving temporal
information for expression dynamics. Thus, we can provide additional
constraints to the system to reduce the size of the search space
and adapt a coarse-to-fine convergence. All of these constraints are
formulated by our initialization plugin introduced in Section 3 and
visualized in Figure 3.

Feature selection Being able to select/exclude individual features
of the face can greatly constrain the search space by being able
to evolve granularly: starting with evolving broad face regions, followed
by feature groups and then even single rig controls. To achieve
this, we allow users to disregard certain face regions when initializing
the algorithm by explicitly selecting rig control groups from the
Outliner View in Maya to include. Throughout evolution, we also
provide the option to evolve these groups separately.

Uniform sampling To encourage local variation, another flag we
provide is that of uniform sampling. Instead of sampling in the range
\([0,1]\) for each rig control, we sample uniformly around the current
value. This is particularly effective for ‘tweaking’ shapes.

Symmetry For certain tasks, we might wish to make sure
that faces stay symmetrical so left/right controls have the same
value — greatly reducing the search space. To achieve this, we cre-
ate a dictionary of all control names in the rig and their correspond-

5.2. Control schemas
With the benefits of using constraints, we must devise how we wish
to use them during evolution. An ideal design would be adopt a
course-to-fine approach: where we start with a global search to find a
broad subset of parameter combinations where the optimum is likely
to occur; followed by local convergence akin to gradient descent.
For example, for authoring new expressions, it would be useful to
initiate using strongest shape and symmetry flags over all rig groups.
For later/local convergence, it would be useful to refine a smaller
number of shapes to a smaller degree. To define such behaviour of
these hyper-parameters, we design control schemas which act as
configuration files that the algorithm refers to, as demonstrated in
Figure 4. An important motivation behind the use of these schemas
is to hide these constraints to the user, so we can retain the simple
selection design of the interface. However, for more experienced
users, we also provide the option to explicitly control these hyper-
parameters through an Advanced Control Panel.

The Advanced Control Panel includes some other important pa-
rameters we have not previously mentioned. The selection limit
encodes the number of user selections required before evolving the
user’s saved selections and advancing to next generation. This is
used to enforce variety in early generations to avoid being stuck
in local minima. Another control for reducing gene pool stagnancy
is the mutation rate, which governs how much random variation is
shown in samples. We define this as a scale \(\gamma\) in range \([0,1]\) which
combines the likelihood of mutation, the intensity of mutation and
number of controls to be mutated.

```
Default
{  Active only: 'Off'
  Prioritize most active: 'Off'
  Prioritize strongest: 'On'
  Symmetry: 'On'
  Uniform Sampling: 'Off'
}

Gen 4
{  Selection Lim: 6
  Mutation Rate: 0.4
  Shape groups: 'Brows'
}

Gen 1.3
{  Selection Lim: 12
  Mutation Rate: 0.2
  Shape groups: 'All'
}

Gen 5:end
{  Selection Lim: 5
  Mutation Rate: 0.6
  Shape groups: 'SingleCurve'
}
```

Figure 4: Example control schema for authoring new expressions.
6. Applications

In this section, we discuss the two tools we have developed to demonstrate the utility of EAs and our framework: one for editing facial expression and the other for dynamics. Both applications follow a consistent pipeline, as presented in Algorithm 1. Modifying face expression modifies rig control values at a single keyframe, with dynamics involving multiple keyframes. The use of constraints, however, does greatly depend on the task which we describe here — alongside any other implementation specifics.

6.1. Evolving facial expression

For evolving static facial expression, we cater for two use cases: authoring new expressions and editing existing expressions. Forming new expressions from scratch is intended for non-animators: for example, by researchers to generate content for perception and psychological studies. Editing existing expressions is useful for fine-tuning the appearance at a keyframe to change its emotional response or for correctives, e.g. sticky lips. For these applications, we use the same tool, tailoring the task via control schemes as follows.

1. Authoring new shapes - For authoring new shapes, we start from a neutral blendshape rig (all weights set to 0) so active only and most active flags are made redundant — as expressed in Figure 4. However, the strongest shapes flag becomes very significant as it allows the most varied shapes — such as smiles, frowns and sneers — to be sampled first. We also use a high selection limit for initial generations to increase the chances of the most suitable coarse shape being sampled. Likewise, we do not exclude any rig groups (other than excluded by user explicitly), so shape variation is as broad as possible. In a coarse-to-fine manner for later generations, we lower the degree of variability to converge towards local targets. This includes evolving facial features, e.g. Brows separately, by sampling only from specific rig groups for certain generations.

2. Editing shape at existing keyframes - For this task, as we already have a coarse shape in place, mutation is of greater importance to fine-tune the shape, so we use a relatively high mutation rate (γ = 0.7). The tool is started by the user providing the keyframe they wish to edit and rig groups to exclude - likely to be more specified. The use of active only flag is useful for initial generations to investigate whether the desired shape can be created by permuting the current shapes. We can do this in a coarse-to-fine manner over generations by applying the most active shapes flag. At later generations, to add shapes that might be missing, we turn off active only and sample from individual rig controls that were not activated in the original shape.

6.2. Evolving facial dynamics

For evolving the dynamics of facial expression, we present similar applications to Section 6.1: authoring new dynamic expressions for a target shape and editing existing animation curves for a given animation. The former could be an alternative way to add personalization and non-linearity to idiosyncratic blendshapes, such as the smile. Editing existing curves could be useful when retargeting a performance to change its emotional context, e.g. duration of a smile or systematically imposing corrective shapes. Though the underlying EA is consistent with evolving static shape, the implementation differs in that we focus on animation curves. To sample animation curves, the user provides keyframes for the start and end frames for the sequence they wish to edit and animation curves in this time frame are stored. To trial different facial dynamics, we sample new XY locations for control points at keyframes directing the shape of the 2D curve C. Where Y represents control value and X the time/frame, this is analogous to sampling new facial dynamics.

1. Authoring non-linear dynamics - For this, we rely on the user selecting one frame as the neutral $t_N$ and a subsequent to be the target $t_T$ expression, e.g. frames $t_0$ and $t_{60}$, respectively in Figure 5. This target frame could be a result from using the tools from the previous section. By keyframing both these frames, we gain a linear interpolation for all blendshapes from neutral to the target — as in row 1. To add broad non-linear dynamics for each trialled animation, we sample two new control points (Figure 5, Gen. 1). To achieve this, we select random inbetween times $t_P, t_Q$, such that $t_Q < t_P < t_T < t_T$. As a simple constraint, we enforce that $t_P$ and $t_Q$ are more than $0.1(t_T - t_N)$ frames apart to avoid unrealistic exponential curves. We set rig control intensity values $C_P, C_Q$ along a curve $C$ by sampling two random scalars $[s_P, s_Q]$ to apply to the value at $C_T$ — maintaining uphull curves for all blendshapes:

$$\{C_P, C_Q\} = [s_P, C_T, s_Q, C_T] \mid [s_P, s_Q] \sim U([0, 1]), s_T < s_Q.$$  

(9)

The process of evolution now occurs by breeding various animation curves from various user selected animations. This creates more complex dynamics as in Gen 2, where different facial regions, i.e. C2, experience different onset/offset phases. For this tool, instead of relying on mutation, we opt for many coarse samples by setting high values for selection limits (about 10). We also modify our mutation function, so it gives the option of adding a third (or more) control point $C_R$ to a curve, bounded by neighbour control

---

© 2019 The Authors. Computer Graphics Forum published by Eurographics - The European Association for Computer Graphics and John Wiley & Sons Ltd
points, and provide a limit for this in the control schema (default being 4).

2. Retargeting existing animation curves - We build this tool with the results from our editing shape tool (Section 6.1): where for an original performance \( A \), as in Figure 13, we edit frame \( A_{88} \) to target expression \( B_{88} \). As it stands, we have no technique for introducing these edits back to the original animated sequence. In a naive approach, as demonstrated in row B, we can calculate the delta of each rig control for target \( B \) and original shape \( A \) and apply this as a static offset to each entire animation curve \( C \), respectively:

\[
B = \{A^c_0, \ldots, A^c_T\} + (B^c_T - A^c_T).
\] (10)

However, as this might greatly affect the appearance of the rest of the performance, we design a tool to allow users to correct the dynamics of poor curves. For this, we take a very similar approach to the methods addressed in the previous subsection, but applied to a subset of curves with the highest control value delta, i.e. curves that have changed the most. As before, for evolution, we utilize strongest and most active flags — though for the latter, this is now ordered by the size of the control delta.

Algorithm 1. Evolutionary framework

1: function User Interface
2: \( G^{(t=1)} \leftarrow \emptyset \), gen = 1
3: while !user Satisfaction do
4: \( S \leftarrow \text{SAMPLE}(G, \text{gen}) \)
5: visualise Samples()
6: if user Selection || elite Selection then
7: \( S^* \leftarrow \{S^*\} + S \) || \( E \leftarrow S \)
8: if numSamples >= required then
9: \( G^{(t+1)} \leftarrow \{E, S^*\}, \text{gen}++ \) \( \triangleright \) Proceed to next gen.
10: function SampleG, gen
11: \( S \leftarrow \emptyset \)
12: \( \{\gamma, \ldots\} \leftarrow \text{infer Constraints} (\text{gen}) \)
13: if gen == 1 then
14: \( S \leftarrow \text{randomSample}([0, 1]) \) \( \triangleright \) As in Eq. 4
15: else
16: elite Parent Flag \( \leftarrow \text{rand} (\text{True, False}) \)
17: curve Options \( \leftarrow \text{rand} (\text{All, Curve Group, Single Curve}) \)
18: mutate Flag \( \leftarrow (\text{rand} (0, 1) < \gamma) \)
19: \( [P_1, P_2] \leftarrow \text{rand} (G) \)
20: if elite Parent Flag then
21: \( P_1 \leftarrow E \)
22: \( S \leftarrow \text{breed Curves} (P_1, P_2, \text{curve Options}) \) \( \triangleright \) As in Eq. 7
23: if mutate Flag then
24: \( S \leftarrow \text{mutate Curves} (S, \text{curve Options}) \) \( \triangleright \) As in Eq. 8
25: return \( S \)

7. Results

To validate our framework, alongside demonstrations of its use for prototyping tasks, we also provide a user perception study to assess results and devise a quantitative task to measure system performance. To demonstrate that our framework is rig agnostic, we provide examples using various rigs including: a linear blendshape rig (procedurally generated using deformation transfer [SP04]) and a sophisticated industrial-grade rig (courtesy of 3Lateral [3La]).

7.1. Prototyping expression shape

To test the system’s capabilities as a prototyping tool for facial expression, we tasked 31 users — with no animation experience — with creating happy, sad, angry and fearful expressions. Some

![Example results from multiple users evolving towards different target expressions.](image1)

![Example results for various nuanced smiles.](image2)

Table 1: Results for relabelling coarse generated shapes.

| Expression to pick | Happy | Sad | Angry | Fearful |
|--------------------|-------|-----|-------|---------|
| Relabelling success rate (%) | 83.9 | 73.9 | 82.2 | 61.6 |

© 2019 The Authors. Computer Graphics Forum published by Eurographics - The European Association for Computer Graphics and John Wiley & Sons Ltd
Table 2: Results for labelling a variety of nuanced smiles, for images of real people and results generated from our system.

| Smile to pick | Sarcastic | Confused | Confident | Sinister | 'Fake' smile |
|--------------|-----------|----------|-----------|----------|-------------|
| Image (%)    | 48.3      | 58.3     | 49.4      | 54.1     | 69.2        |
| Ours (%)     | 37.2      | 45.6     | 38.3      | 68.3     | 43.3        |

Figure 8: Example results for generating 'genuine' and 'fake' smiles. Images property of Ekman International [Ekm].

results are visualized in Figure 6, all of which were given a limit of six generations. It is noticeable how varied results are — expected with the stochastic nature of the EA and differences in user perception — yet all are very much representative of their intended expression.

Similarly, to test the role of the system as an expression editing tool, we show how a generic smile can be refined towards various abstract smile styles: 'sarcastic', 'confused', 'confident' and 'sinister'. This task is designed to be challenging as such expressions are very subjective and rely on high-frequency detail. Some results are demonstrated in Figure 7, which reflect the subjective nature of the task as results are noisier and the boundary between expressions is less defined. For example ‘confident’ and ‘sinister’ smiles use many of the same blendshapes, though have very different semantics. These examples were generated by an experienced user of the system and on average took 10.4 generations to produce.

To evaluate how suitable the system was for expression generation by non-expert users, we tasked a separate group of 45 participants via a survey with labelling these generated expressions, with the results highlighted in Tables 1 and 2. As expected, coarse expressions were easily identified (Table 1), though sad and fearful expressions were occasionally confused. For the harder task of labelling nuanced smiles, we repeat this task for results from our system and also with photographs of people performing the same expression. Our hypothesis is that if a similar recognition error is witnessed between the two tasks, then this suggests the error is due to subjectivity rather than performance of our system. To push the limits of our system, we also task users with spotting 'fake' smiles among 'genuine' smiles. Genuine (or ‘Duchenne’ smiles) can be distinguished through three high-frequency shapes in FACS: cheekRaiser, lipCornerPull and eyeCompress — making formation/recognition a very challenging task [EDF90]. Results from this can be seen in Table 2, where it is shown that error rates follow a similar trend to using photographs, though with lower success rate. This is not a surprising result due to the high intraclass and low interclass variation as displayed in Figures 7 and 8. In future research, it would be interesting to determine whether the use of texture in our experiments would improve user performance in this task.

7.2. Algorithm performance

To test the performance of the algorithm itself, we rely on a quantitative measure, i.e. the error between a ground truth. To this end, we devise an experiment where we trial our EA to evolve towards a ground truth target with known values and record the Sum of Squared Error (SSE) of global vertex errors.

Figure 9: Results and error for first 10 generations of user refining towards an observed target for following tasks (a) Local refinement with known active rig controls, (b) local refinement with unknown active rig controls and (c) global refinement with unknown active rig controls.
Figure 9 visualizes the results and error of the first 10 generations for a user evolving a given shape to a highly nuanced observed target. To test the strengths of our constraints in Section 5.1, we also impose various starting scenarios. For task (a), the target shapes are known in advance so the search space is relatively small. As we can see, this provides results with the lowest error. When shapes are not known, as in (b) this poses a far greater search space. Convergence is slower and results lack high-frequency detail around the lips, as there are an abundance of shapes for the feature. Lastly in (c), we demonstrate the task of authoring expression from a neutral. Coarse shapes converge relatively quickly though detailed shapes, similar to (b), are difficult to optimize.

Through use of the ‘lowest SSE’ as an evaluation/selection criterion, we can fully automate the process of static facial expression generation using the EA. Figure 10 presents the convergence for the same task repeated for different selection methods over 25 generations. The results present numerous findings; chiefly, as the overall SSE is lower for using the EA than random sampling, it demonstrates that the EA works as an optimization framework for this task. It also shows that people are a viable method for recognizing facial error as users commonly pick samples that offer the lowest SSE. For all methods, convergence slows as high-frequency shapes are more numerous and have less effect visually and on the SSE, so are not selected by EA. Thus, over many generations, the performance of the EA converges towards that of random sampling. This is most apparent for user validation where convergence decelerates between 5 and 10 generations. We believe this is around the point where user fatigue from seeing many samples becomes significant — therefore we stop at 10 generations.

7.3. Prototyping expression dynamics

To demonstrate the systems capabilities at evolving motion, here we provide several examples of how it can be used, using various face rigs. More results can be seen in the Supplementary Video.

Figure 11: Example of adding non-linear dynamics (bottom) to generic linear ‘smileOpen’ combination blendshape (top) using our framework, via the following blendshapes: mouthCornerPull (blue), jawOpen (red), dimpler (magenta), browRaise (green) and eyeSquint (yellow).

Figure 12: For an observed dynamic expression (top), users are tasked with editing given curves to resemble the original animation. Curves used being: mouthCornerPull (blue), browRaise (green), browLateral (red), upperLipRaise (magenta) and chinRaise (black).
Authoring non-linearity. In Figure 11, we present a simple demonstration of the results from the application outlined in Section 6.2.1, with the aim to create a new smileOpen blendshape. The first row shows how this can be done through a combined linear blendshape of smile and openJaw controls. To create a more personalized smile, we first use our expression authoring tool (Section 6.1.1) to add more personalized shapes such as dimples and create a combined linear blendshape. By adding non-linearity through evolution of animation curves, we change the dynamics of the expression. This animation can now be baked into a rig; useful for adding additional control and personalization to generic rigs.

Editing curves. Similar in nature to the task in Figure 9, for the task demonstrated in Figure 12, we present users with an observed animation for a dynamic smile expression and task them with recreating the dynamics starting with a linear interpolation. Overall, curves present similar activations and are consistent for separate features, though timings are not exact and curves are simplified. Common feedback for this task is that selection of samples was challenging as they tended to look very similar, despite very different animation curves. This suggests that this task is best suited for strongest blendshapes, as less significant shapes are hard to identify and refine. This claim is supported in this example by how the chin shape looks to be ignored by the user.

Retargeting a performance. As a demonstration of how we can combine tools for editing shape and dynamics, in Figure 13, we provide an example of how we can retarget a neutral performance into a happy performance. As outlined in Section 6.2.2, row B presents the static ‘smile’ offset as seen in $B_{88}$ applied to each respective animation curve for the length of the sequence. However, such a simple offset might form static animation curves or over-exaggerated expressions that look unnatural. In this example, we use our curve editing tool to edit the influence of the brows so that it is not constantly activated — as in sequence B. Sequence C shows our results where brow curves are now dynamic and are most activated when the smile is at its fullest.

8. Discussion

Our experiments demonstrate that our system performs well with subjective tasks, such as prototyping new expressions and non-linear animation curves. The main benefit of our system is that it has very few requirements — requiring no ground truth reference or animation experience — so can be adapted to a high variety of tasks where there might be very few other optimization options. Through our positive results, we have shown that user validation is a viable search method for refining facial expression; with users routinely selecting samples that attribute the lowest error (as seen in Figure 10). A limitation of user validation is that it restricts the extent of sampling, due to user fatigue. Though this can reduce the precision, through use of our coarse-to-fine schemas, we can effectively reduce the majority of the error for around five generations. To this end, for adapting our framework, there should be consideration of the precision required, as animator quality is not guaranteed.

A key contributing factor to positive results has come from the design of constraints and their application to facilitate a coarse-to-fine search. For example, using higher sampling and mutation rates greatly reduces the chance of getting stuck in local minima. From user trials, we have discovered that convergence is...
faster when the user is experienced with controlling constraints explicitly during evolution via the Advanced Control Panel. This is in contrast to the use of pre-defined control schemas and a simple selection interface, which is much more accessible to novice users. To this end, it would be beneficial to further investigate different interface designs; including the incorporation of other user-based techniques, e.g. sketch based and label based.

Furthermore, we can improve our system in future through use of additional well-defined constraints. For example, though a blendshape rig is an effective prior for facial expression, this does not directly extend to facial dynamics; where instead our constraints consist of enforcing well-behaved uphill curves. As this does not guarantee plausible dynamics, our procedure of sampling animation curves can often require many trials. Moreover, we could greatly improve the robustness of creating plausible animations through use of data-driven or anatomical priors for facial dynamics. An interesting experiment would be to extend our EA framework to evolve latent variables of a shape/motion manifold directly — also exemplifying the adaptability of our framework.

9. Conclusion

In this work, we have proposed an approach to facial expression creation through use of an EA driven by a selection interface. We have demonstrated this through applying our framework to prototyping tools that evolve rig parameters at keyframes followed by the animation curves they act upon — resulting in animations with nuanced expressions and non-linear dynamics. From our experiments, we conclude that our approach can be useful for non-expert users in creating facial expressions, and for animators for tasks of an experimental nature, such as rig tweaking, especially when experienced with the additional constraints provided. We believe our framework is useful in several applications: for prototyping new facial expressions in perception studies, as a communication tool for creative stakeholders and to ‘block’ out basic animations, such as adding emotion to text-to-animation systems. Due to the flexibility of the framework, applications could even generalize to non-animation tasks; such as finding optimal settings of a tool based on visual perception, e.g. render settings.

References

[3La] Character modelling and rigging (2018). http://www.3lateral.com/.

[ATL12] Aniyo K., Todd H., Lewis J.: A practical approach to direct manipulation blendshapes. Journal of Graphics Tools 16, 3 (2012), 160–176.

[BV99] Blanz V., Vetter T.: A morphable model for the synthesis of 3D faces. In Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (1999), ACM Press/Addison-Wesley Publishing Co., pp. 187–194.

[BY86] Bruce V., Young A.: Understanding face recognition. British Journal of Psychology 77, 3 (1986), 305–327.

[CGNS17] Ciccone L., Guay M., Nitti M., Summer R. W.: Authoring motion cycles. In Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (New York, NY, USA, 2017), SCA ’17, ACM, pp. 8:1–8:9.

[CH05] Chai J., Hodges J. K.: Performance animation from low-dimensional control signals. ACM Transactions on Graphics 24 (2005), 686–696.

[CLS*16] Choi B., Blanca1 Ribera R., Lewis J. P., Seol Y., Hong S., Eom H., Jung S., Noh J. Sketchimo: Sketch-based motion editing for articulated characters. ACM Transactions on Graphics 35, 4 (2016), 146.

[DBB18] Denev D., Beeler T., Bradley D., Bächer M., Xu H., Kavan L.: User-guided lip correction for facial performance capture. Computer Graphics Forum 37 (2018), 93–101.

[DiP02] DiPaola S.: Facespace: A facial spatial-domain toolkit. In IV (2002).

[DWW15] Du Y., Wang W., Wang L.: Hierarchical recurrent neural network for skeleton based action recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2015), pp. 1110–1118.

[EDF90] Ekman P., Davidson R. J., Friesen W. V.: The Duchenne smile: Emotional expression and brain physiology: II. Journal of Personality and Social Psychology 58, 2 (1990), 342.

[EDF90] Ekman P., Davidson R. J., Friesen W. V.: The Duchenne smile: Emotional expression and brain physiology: II. Journal of Personality and Social Psychology 58, 2 (1990), 342.

[EF78] Ekman P., Friesen: Facial Action Coding System: A Technique for the Measurement of Facial Movement. Palo Alto, CA: Consulting Psychologists Press, 1978.

[Ekn] Ekman P.: Spot the fake smiles (2018). http://www. ekmaninternational.com/paul-ekman-international-plc- home/news/can-you-spot-a-fake-smile.aspx.

[FCN*05] Frowd C. D., Carson D., Ness H., Richardson J., Morrison L., McLanaghan S., Hancock P.: A forensically valid comparison of facial composite systems. Psychology, Crime & Law 11, 1 (2005), 33–52.

[FHC04] Frowd C. D., Hancock P. J., Carson D.: Evofit: A holistic, evolutionary facial imaging technique for creating composites. ACM Transactions on Applied Perception 1, 1 (2004), 19–39.

[FLFM15] Fragkiadaki K., Levine S., Felsen P., Malik J.: Recurrent network models for human dynamics. In 2015 IEEE International Conference on Computer Vision (ICCV) (2015), IEEE, pp. 4346–4354.

[Gib03] Gibson S. J., Solomon, C. J. and Pallares Béjarano, A., Synthesis of photographic quality facial composites using evolutionary algorithms. In: Richard Harvey, J., Bangham, A. (Eds.), British Machine Vision Conference (2003), vol. 1, pp. 221–230.

[GMHP04] Grochow K., Martin S. L., Hertzmann A., Popović Z.: Style-based inverse kinematics. ACM Transactions on Graphics 23 (2004), 522–531.

[GS84] Gillund G., Shiffrin R. M.: A retrieval model for both recognition and recall. Psychological Review 91, 1 (1984), 1.
[HMC*15] Hahn F., Mutzel F., Coros S., Thomaszewski B., Nitti M., Gross M., Sumner R. W.: Sketch abstractions for character posing. In Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation (2015), ACM, pp. 185–191.

[Hol92] Holland J. H.: Genetic algorithms. Scientific American 267, 1 (1992), 66–73.

[HSK16] Holden D., Saito J., Komura T.: A deep learning framework for character motion synthesis and editing. ACM Transactions on Graphics 35, 4 (2016), 138.

[HSS17] Haubenwallner K., Seidel H.-P., Steinberger M.: Shape-genetics: Using genetic algorithms for procedural modeling. Computer Graphics Forum 36 (2017), 213–223.

[KBB*17] Kozlov Y., Bradley D., Bächer M., Thomaszewski B., Beeler T., Gross M.: Enriching facial blendshape rigs with physical simulation. Computer Graphics Forum 36 (2017), 75–84.

[LA10] Lewis J., Anjyo K.-I.: Direct manipulation blendshapes. IEEE Computer Graphics and Applications 30, 4 (2010), 42–50.

[LAR*14] Lewis J. P., Anjyo K., Rhee T., Zhang M., Pighin F. H., Deng Z.: Practice and Theory of Blendshape Facial Models. Eurographics, 2014.

[LMDN05] Lewis J. P., Mooser J., Deng Z., Neumann U.: Reducing blendshape interference by selected motion attenuation. In Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games (2005), ACM, pp. 25–29.

[MAO*12] Miranda J. C., Alvarez X., Orvalho J., Gutiérrez D., Sousa A. A., Orvalho V.: Sketch express: A sketching interface for facial animation. Computers & Graphics 36, 6 (2012), 585–595.

[PJ08] Pilat M. L., Jacob C.: Creature academy: A system for virtual creature evolution. In IEEE Congress on Evolutionary Computation, 2008. CEC 2008, (IEEE World Congress on Computational Intelligence). (2008), IEEE, pp. 3289–3297.

[SBOR06] Sinha P., Balas B., Ostrovsky Y., Russell R.: Face recognition by humans: Nineteen results all computer vision researchers should know about. Proceedings of the IEEE 94, 11 (2006), 1948–1962.

[SILN11] Seo J., Irving G., Lewis J., Noh J.: Compression and direct manipulation of complex blendshape models. ACM Transactions on Graphics 30 (2011), 164.

[Sim94] Sims K.: Evolving virtual creatures. In Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques (1994), ACM, pp. 15–22.

[SMND08] Sountphunt T., Mo Z., Neumann U., Deng Z.: Interactive 3D facial expression posing through 2D portrait manipulation. In Proceedings of Graphics Interface 2008 (2008), Canadian Information Processing Society, pp. 177–184.

[SP04] Sumner R. W., Popović J.: Deformation transfer for triangle meshes. ACM Transactions on Graphics 23 (2004), 399–405.

[SQRH*16] Streuber S., Quiros-Ramirez M. A., Hill M. Q., Hahn C. A., Zuffi S., O’Toole A., Black M. J.: Body talk: Crowd-shaping realistic 3D avatars with words. ACM Transactions on Graphics 35, 4 (2016), 54.

[WFH07] Wang J. M., Fleet D. J., Hertzmann A.: Multifactor Gaussian process models for style-content separation. In Proceedings of the 24th International Conference on Machine Learning (2007), ACM, pp. 975–982.

Supporting Information

Additional supporting information may be found online in the Supporting Information section at the end of the article.