Evaluation of a Facial Animation Authoring Pipeline Seamlessly Supporting Performance Capture and Manual Key-pose Editing

Fabrizio Nunnari, Alexis Heloir

EAI Endorsed Transactions
on Creative Technologies

Research Article ICST.ORG

Evaluation of a Facial Animation Authoring Pipeline Seamlessly Supporting Performance Capture and Manual Key-pose Editing

Fabrizio Nunnari[1,*], Alexis Heloir[1,2]

[1]Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI), CAMPUS D3.2, 66123, Saarbrücken, Germany
[2]LAMIH-UMR CNRS 8201, Le Mont Houy, Univ. Valenciennes, France

Abstract

In this paper, we present an architecture following a novel animation authoring pipeline seamlessly supporting performance capture and manual editing of key-frames animation. This pipeline allows novice users to record and author sophisticated facial animations in a fraction of the time that would be required using traditional animation tools. This approach paves the way towards novel animation pipelines which

seamlessly merge the roles of the animator and the actor. The second contribution is a method assessing a facial retargeting system, we conducted a user study where participants assessed the emotions conveyed by the facial expression displayed in the control and the authored animation. Contrary to existing evaluation methods, it factors out possible misinterpretations of the intended emotion and focuses on assessing the retargeting quality.

Keywords: facial animation, performance capture, retargeting, evaluation, animation authoring.
Received on 23 May 2014, accepted on 2 January 2015, published on 2 June 2015

doi: 10.4108/ct.2.3.e4

1. Introduction

The face is a very expressive and complex non-verbal communication channel for humans. It is the preferred mean of expressing emotions and share feelings. The importance of facial expression is not only the privileged location for emotion expression [1], it is also a paramount articulator for signed language where lack of facial animation prevents the content to be understood [2]. Depending on how researchers agree on categorizing them, the human face counts between 42 and 52 facial muscles. These muscles combines to activate and move the above skin to produce a surfacing expression.

In computer graphics, A facial *mesh*, is defined by set of triangles, stitched to each other in a three dimensional space, small enough to approximate the smoothness of the curvatures of the face. The underlying facial muscles are, in most cases not simulated. Instead, their surfacing influence is reenacted using a combination of so called blendshapes [3]. A blendshape is a vector of offsets to be applied to the 3D vertices describing the topology and the morphology of the face in a neutral configuration. Each blendshape is configured to move a small subset of the face triangles, as a single facial muscle would do. An animator can configure complex facial expressions by modulating the influence of a blendshape (i.e., by applying a scale to the vector of offsets) and by using multiple blendshapes at once.

The most straightforward metaphor depicting the craft of animation would be the puppetry metaphor: one could think of an animated virtual character as a digital marionette that might be actuated by an animator. In both cases, the limbs and other body parts are not usually directly manipulated, there is an intermediate interface offering higher-levels handles [4] that give an abstracted control to the puppeteer. This higher set of controls is called *control rig* [5] and its purpose is to ease the work of the animator. Professional animators manually author key-framed animations using elaborated control rigs, which often present more than two hundreds controllers. Twenty or more of these controllers are dedicated to the animation of the face. Despite of the presence of the control rig, animating the face of a 3D avatar is an activity which requires a significant amount of time, passion, and dedication. Fortunately, recent consumer-range technology [6] has proved to be capable of enabling users authoring animation of human-like bodies or interactively controlling physical or digital puppets.

Past research already proposed solutions to increase the productivity of animation through the performance

[*]Corresponding Author. Email: fabrizio.nunnari@dfki.de

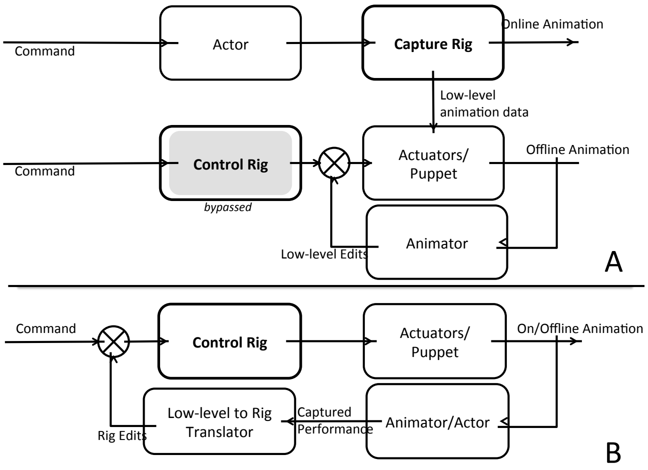


Figure 1. A: In existing workflows, the captured motions are recorded only once by an actor, then edited using low level controls.
B: In our approach, the animator is also the actor, and he is able to seamlessly switch between motion capture and manual edit during the animation authoring.

capture of facial expressions [7, 8]. In performance capture the animator can use his own face to drive the animation of an arbitrary avatar. So far, no performance capture system is able to guarantee the same expressive quality attained by manual animation. Hence, for real productions, the animation captured via performance capture must be furtherly post-processed and edited, by the animator. However, the output of the performance capture is expressed as a low-level animation data, which drive the low-level mesh deformers, i.e. the blendshapes, that are available on the animated figure. As already stated, it is not a convenient method for manual animation. The animator needs a control rig in order to refine captured performances.

In existing animation production workflows, the performance capture produces low-level animation data which are then transferred to the manual animation pipeline bypassing the control rig. Further edit operations will be a hard task for the human operator, and are for this reason relegated to data cleanup operations. If a major editing is needed, a new capture is required, as depicted in Figure 1A.

We propose to map the performance of the actor onto the control rig. As such, the animator can later refine the animation using the sliders of the control rig. In this use-case, the roles of actor and animator thus overlap. Experienced animators can seamlessly switch between traditional space-time constraint edit and interactive performance capture recording, as shown in Figure 1B.

In this paper, we introduce a pipeline driving the animation of arbitrary faces using performance capture and manual animation in an integrated approach. The animation pipeline includes an innovative stage which maps a captured facial performance onto a control rig. In so doing, the animator, after the capture phase, is able to perform further edits using the same control rig used in pure manual animation.

A facial retargeting method must be evaluated in order to prove its effectiveness. The evaluation can be either qualitative or quantitative. A quantitative evaluation based on inter-surface comparison can be applied only when the source and the target faces match, which happens when the target face is a 3D reconstruction of a real face. Since the source (perfomer's facial morphology) and the target faces (animated character's morphology) differ, a qualitative method must be used. A qualitative evaluation involves a user study. So far, there are no de facto standards to conduct the qualitative evaluation of a facial retargeting method. The second contribution of this paper is a qualitative evaluation procedure based on the recognition of emotions conveyed through facial expressions.

The assessment of a retargeting system is generally based on the judgment of facial expressions conveying specific "pure" emotions (e.g., happiness, anger, ...). But emotions are subject to misinterpretation; even when performed by professional actors. If the evaluation includes only correct interpretations, important cues about how subjects perceive the intended emotion are lost. Our procedure takes into account misinterpretations and allows for the evaluation of mixed facial emotions. This method is thus also suited to casual users without acting background.

The evaluation procedure starts with the selection of the users who will perform a set of reference facial animations. Each animation intends to convey a pure emotion. The performances of the users are later retargeted to the face of a virtual character and rendered as animated video clips. Another set of users, called respondents, will judge the reference videos, the original live performances, and the rendered 3D face. For each video, the judgment consists of expressing a level of perception for all the emotions involved in the study.

The judgment is implemented as an online survey asking to fill a form. The output of the online survey is a set of contingency tables. Contingency tables are a data structure commonly used in statistics to describe the frequency distribution of variables [9]. We measure the quality of the mapping method by comparing the contingency tables of the live performances and of the rendered videos.We show, in a user study, that the retargeting system proposed in this paper allows for users with no experience in computer animation and computer graphics to effectively use our tool

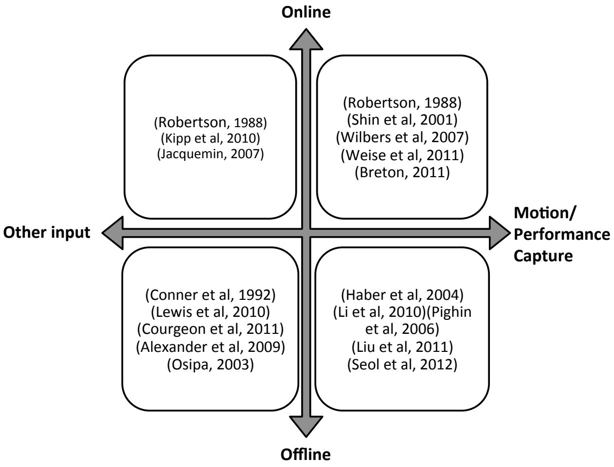


Figure 2. A comprehensive classification of the related work.

to produce convincing animations. Our evaluation method allows for the measurement of the quality of the mapping method even if the involved performers are not professional actors.

This paper is organized as follows. Section 2 presents related work in the field of facial performance capture. Section 3 describes in detail the architecture proposed in this paper, which is depicted in Figure 1B. Section 4 describes the method which maps the low-level capture information to the control-rig. Section 5 presents the user study that we conducted to evalutate the proposed workflow. Finally, Section 6 concludes the paper.

2. Related Work

In computer animation, the animation control interface and the way it is connected to the underlying body elements is called *control rig* [5]. As we will see in the following, rigs provide the interface between the manual inputs of an animator who authors an animation on a key-frame basis [3, 5]. Alternatively, the interface between an interactive input method and a virtual or real puppet generates low-level animation data [8, 10]. In fact, most animation authoring pipelines can be classified according to two dimensions: input type (manual edit vs captured motion) and usage (online puppetry vs offline animation). Figure 2 gives an overview of the related work we are going to present and discuss.

2.1. Offline Animation

In traditional animation, rigging is the process of setting up a group of controls to operate a 3D model, analogous to the strings of a puppet. It plays a fundamental role in the animation process as it eases the manipulation and editing of expressions. Control rigs are created to facilitate and speed up the task of animation. Their purpose is to reduce the degrees of freedom (DOFs) needed to control the character. They provide the animator with a simple but convenient layer of abstraction [4]. It is essential for the animator to have a rig that allows a right expression palette, that is convenient to use, and at the same time compact (not too many handles) and expressive (the handles provide a good factorization of the controllers). In the industry, rigs are created by a dedicated professional, the rigger, who sets it up in tight collaboration with the animator. A rig is therefore a complex mixture of science, art, and craft. As a consequence, rigs are very often unique, tailored for a single character, and tuned for a particular animator.

Following the conventions introduced in the early 90's [11], a manual control rig is a set of 3D objects, known as *controllers*, or *rig-handles*, which are used to manipulate a character's body or facial configuration. For example, a sliding controller (one degree of freedom) for the *smile* expression is generally associated to the movement of four or more blend shapes [3] which operate at the level of facial muscles. Another example, at skeletal level [12], is the rig-handle associated to the *hand opening and closing* action, where a single degree of freedom is associated to the rotation of 14 hand joints. Finally, the *wrist* rig-handle, very commonly used in full body animation, allows the animator to position and rotate a hand in space (6 DOFs) without having to manually edit the rotation of the joints of the arm, the clavicle and the spine (potentially more than 15 DOFs). This last handle generally invokes *Inverse Kinematic* (IK) algorithms [13], an approach inherited from robotics to calculate the rotations of a chain of joints given the desired position and rotation in space of a terminal end-effector.

Traditionally, the handles that compose the control rig are exposed to the animator in two ways: either in a different window than the viewport window displaying a 3D view of the character being edited, in which case handles are represented by simple 2D controllers constrained to squares [5, 14, 15], or more complex anthropomorphic controllers [16].

Sketching techniques have also been used as UI control systems [17]. Kipp et Nguyen [18] proposed a multi-touch interface allowing an animator to interactively control an IK-driven human arm and record poses or captured motions, whereas Sanna et al. [6] achieved a similar task for the whole body using the Kinect[1] depth-camera. Like Sanna et al., the proposed animation workflow uses a depth sensor, but for recording facial animation instead of the body. Like

[1] http://www.xbox.com/en-US/kinect/ - 7 Jan. 2015

Kipp et al., our system is also capable of storing single poses or recorded performances.

2.2. Online Puppetry and Performance Capture

When performing online puppetry, the puppeteer interactively manipulates a dedicated interface that allows the live control of a *puppet*. This puppet might be physical, made out of rag and wood, in which case the animation rig is a wooden frame actuating the puppet using strings. The physical puppet might as well be a way more complex Geminoid [10], with an elaborate control rig capable of converting real time live 3D data collected from a set of facial markers glued onto a performer's face into dynamic commands sent to the geminoid's actuators. In 1988, Robertson demoed the first interactive anthropomorphic computer puppet: Mike [19]. Mike was capable of performing simple facial expressions and moving the lips according to the puppeteer's speech. Mike's animation rig consisted of data glove and a speech recognition system.

Mike was later followed by systems capable of capturing in real time the animation of an actor's face without markers. For example, see [8] or the Dynamixyz[2] systems. When it includes face, Motion Capture [20] is often referred to as *performance capture*. In performance capture, the puppeteer's face usually does not match the puppet's face: they often have different topology and morphology. The mapping of the puppeteer's facial motion on the puppet face is called *retargeting* [7, 21–23].

All the surveyed retargeting approaches are conceived to map the performance to low-level animation controls. None of them maps directly to a control rig, ignoring the need of a post-production editing phase. Seol et al. [24] specifically address the problem of providing room for additional editing by an animator. But again, the target of their retargeting method is a set of blendshapes, which, for animators, is harder to use with respect to a control rig.

The animation workflow presented in this paper is based on the technology developed by Weise et al. [8], where a facial expression is described through a weights vector of a set of blendshapes. Section 4 describes the mapping method which translates the weights vector into a configuration of the control rig. The translation is computed via a linear cross-mapping [25]. This translation stage is a form of *facial motion retargeting* which, following the terms introduced by Pighin [7], is based on a *scattered data interpolation* approach (the mapping is driven by a matrix multiplication function) fed by an *art directed* input (the authors manually edited the correspondence values – matrix elements – between each source blend shape to a target control rig configuration.)

The effectiveness of a facial retargeting method must be evaluated. The evaluation can be qualitative or quantitative. A qualitative evaluation involves a user study. A quantitative evaluation is based on the computation of a numerical difference between the source and the target expressions. Previous results, whether qualitative [23] or quantitative [26], tell us that a perfect match between the source and the target is never achieved. The animation pipeline presented in this work is evaluated, qualitatively, by measuring the recognizability of the six basic facial emotions defined by Ekman [1]. In a previous work, Song et al. [26] used facial emotions like *expressing surprise*, *crying*, and *laughing* to train the retargeting routine. In our work, we exploit emotion transfer to evaluate the retargeting. Also Battocchi [27] and Costantini et al. [28] used emotion judgment as mean of evaluation. However, as shown in [27], even if a professional actor perform the emotions, subjects falls into misinterpretation. As great as an actor can be in conveying a certain emotion through a facial expression, there are no guarantees that the emotion is correctly perceived. The correctness of the perception decreases when the same emotion is conveyed by the target animated face.

In the above mentioned studies, subjects are asked to guess the emotion conveyed by a short video, or a static picture, which shows a facial expression. Subjects rate a video by selecting the perceived expression from a closed list. The goodness of a facial expression is evaluated as ratio between correct vs. incorrect classifications. In this way, information about the incorrect interpretation is lost. When the facial expression of a human is transferred, to a virtual character, this misinterpretation is expected to be found also in the target virtual face.

The evaluation method proposed in this paper retain misinterpretations and includes them in measuring the quality of the retargeting. This is accomplished by asking subjects to rate each video across all the emotions. Subjects rate a video (showing an expression) through several Likert scales, one for each emotion. The result of the evaluation of a video is a contingency table describing the distribution of the votes.

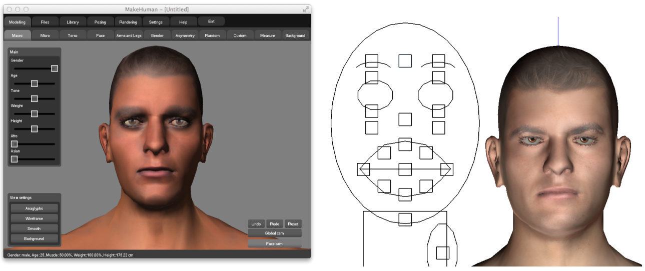
The quality of the facial retargeting is then measured by comparing the contingency tables associated to the source and the target expressions. In statistics, the similarity between contingency tables is generally computed using a Chi-square test for goodness of fit[3] [9]. However, the chi-square test is not applicable to the proposed method because the votes distribution is likely to leave many cells at 0. This breaks one of the

[2] http://www.dynamixyz.com/ - 7 Jan. 2015

[3] http://en.wikipedia.org/wiki/Chi-square_test - 7 Jan. 2015

pre-conditions for the applicability of the Chi-square test, which requires that all levels have a minimum value of 5.

Hence, our method relies on a different metrics to compute the similarity between contingency tables, namely, the generalized Jaccard similarity coefficient[4] [9]. The Jaccard similarity coefficient expresses the similarity between two vectors of real numbers by performing a pairwise comparison on the elements of the vectors. Section 5.3 describes in the detail how the Jaccard coefficient is computed and used.



Gender:male, Age:25, Muscle:50.00%, Weight:100.00%, Height:175.22 cm

Figure 4. Left: the MakeHuman GUI. Right: the head mesh and the control rig imported in Blender

3. Architecture

This section describes the authoring pipeline proposed in this paper. The description gives details of all the stages composing the pipeline, the software therein used, and how the stages are connect with each other.

Four stages compose the authoring pipeline, as depicted in Figure 3: 1) Creation of the 3D character, 2) Animation of the character's face using a Natural User Interface, 3) Refinement of the animation using traditional mouse+keyboard approach, and 4) Exportation of the resulting animation as movie or as animation data.

Stage 1 of the proposed pipeline is based on MakeHuman[5]: an open-source tool for making 3D characters [15]. MakeHuman allows non-expert users for easily authoring a digital character in all its details, without the need to accomplish the complex actions required by full-fledged 3D animation tool (like Autodesk's 3D Studio and Maya, or the open-source Blender), such as vertex-based modelling, vertex weighting, UV mapping, and control rig construction. It offers users an easy-to-use Graphical User Interface (GUI) to edit a human model. The GUI of MakeHuman (Figure 4, left) is mainly composed of sliders. Each slider controls the deformation of an attribute of the human body: age, gender, height, weight, arms and legs length and size, distance between eyes and ears, torso depth and width, are only a few among the more than hundred controls available. Starting from a default androgen 3D character, the user operates the sliders to achieve the desired result.

The edited human can be exported as static 3D mesh in the OBJ and Collada formats. For the proposed pipeline, an editable version of the character can be imported in Blender through the dedicated MHX (Make Human Exchange) exchange format. Once imported in Blender, the character presents both a skeletal structure and an external 3D mesh. The 3D mesh includes a rich set of blend shapes for facial animation (70 in total). Finally, the character offers both a facial and a full-body control rig. The character is thus ready-to-use for a full featured animation work. The MakeHuman face animation rig is composed of 20 controllers. Each controller is a 3D box, sliding on the xz plane, which controls a particular element of the face plus the tongue. The latter has not been used in our work because FaceShift doesn't provide motion capture data of the tongue. Three additional anthropomorphic controllers allows for the rotation of the neck and the eyes. Figure 4, on the right, shows a screenshot of the control rig together with the associated mesh of the head and the face.

Stage 2 of the pipeline consists of animating the character via a performance capture, within the Blender software, using a Natural User Interface. The performance capture is based on the use of a depth sensor (like the Microsoft Kinect, or any equivalent on the market) together with the FaceShift[6] application.

FaceShift is an application able to use a 3D depth sensor to reconstruct the 3D mesh of a face an animate it in real-time according to the actual facial expression of the performer. The details of its underlying implementation can be seen in [8]. Faceshift is based on the OpenNI[7] middleware, which guarantees compatibility with a wide range of Kinect-like hardware available on the consumer market. FaceShift requires a calibration phase for each user. The calibration, lasting less than 10 minutes, consists of mimicking a set of facial expressions. While keeping an expression on his/her face, the user turns his/her head of about 30 degrees on both sides, for a couple of times, to let the system collect data about the shape of the face. The result of the calibration is a 3D reconstruction of the user head and face. The 3D face can be animated via blendshape deformations. A vector with the weights of the blendshapes represent the current state of the user facial expression. Each weight is a real number normalized in the range $[0, 1]$. After the calibration,

[4] http://en.wikipedia.org/wiki/Jaccard_index - 7 Jan. 2015
[5] http://www.makehuman.org/ - 7 Jan. 2015
[6] http://www.faceshift.com/ - 7 Jan. 2015
[7] http://structure.io/openni - 7 Jan. 2015

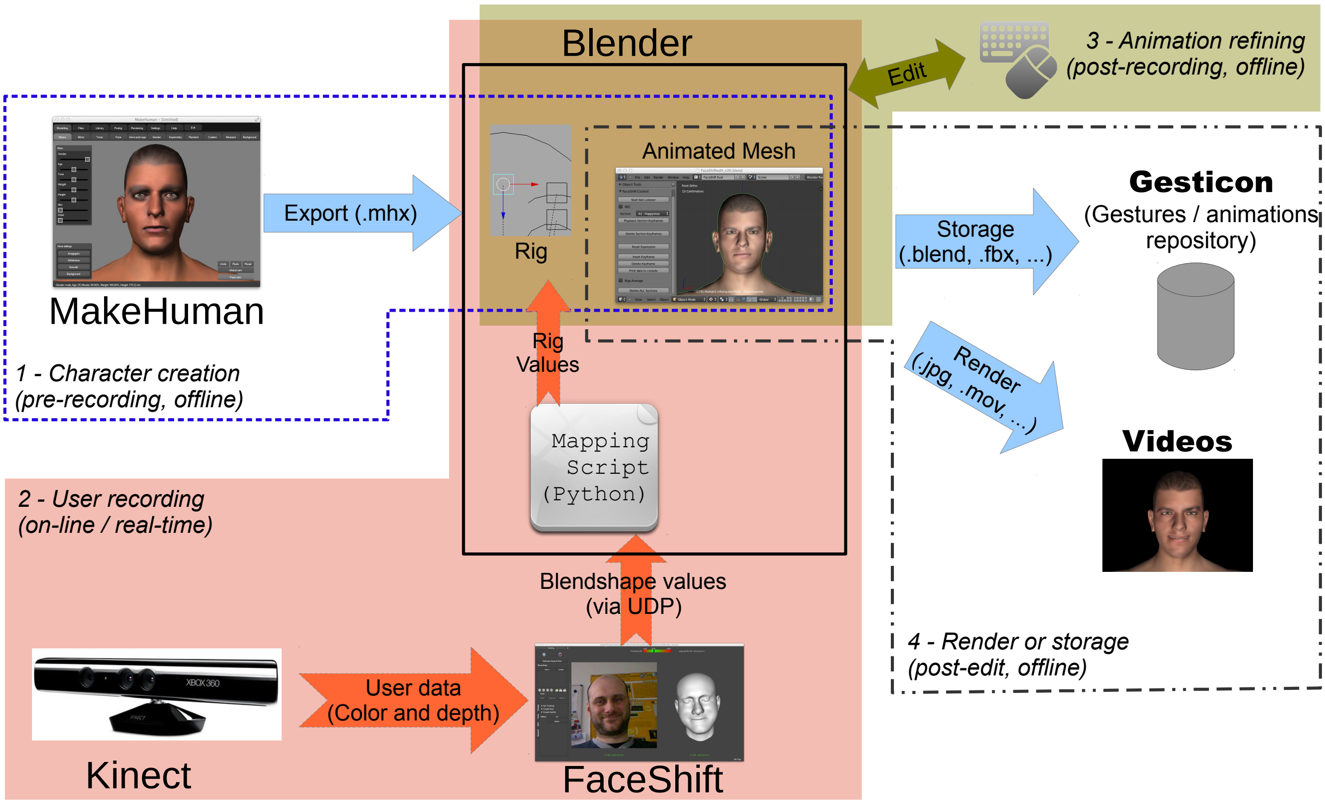


Figure 3. The architecture of the proposed workflow.

FaceShift can be used in real-time tracking mode. Here, the user face expression framed by the Kinect is continuously analyzed and used to pilot the 3D reconstruction of the user face.

The integration with Blender is performed thanks to the capability of FaceShift to output its tracking information to TCP or UDP network sockets. The captured data, which include the vector with the blendshape weights, is processed by a Blender addon, written in Python, developed by the authors. The Blender addon, described in Section 4, decodes the FaceShift binary network communication protocol and allows for the real-time recording of the facial expression and the head movement. More importantly, the addon maps the FaceShift blend shapes values to the control rig of the face. In so doing, the Blender user can have access to the control rig animation keyframes to further edit the facial animation.

Stage 3 of the pipeline allows for the user to manually operate on the control rig to refine facial animations. Obviously, the refinement of the facial and head animations can be associated to a synchronised full body animation for a more comprehensive production. The recording session generates animation keyframes at a rate of about 30 frames per seconds. As a result, animations present a very high data density which makes them hardly editable by hand. Thus, in our pipeline we suggest to perform a curve simplification through the *simplify curve* addon included by default in Blender.

Finally, in **Stage 4**, the edited animations can be rendered as videos, or image sequences, or stored in a separate Gesticon (gestures and animations repository). A Gesticon is intended to be a repository of classified animation data that can be reused across different scenes of the same project or across different projects. Common actions such as smiling, walking, running, or a full set of signs used in sign languages are all perfect candidates for a Gesticon.

The pipeline was developed and tested with the following software versions: MakeHuman v1.0 alpha7, FaceShift 1.1.05, Blender 2.66a. MakeHuman and Blender are both open-source projects and available for free. FaceShift is a commercial application, but facial performance capture is a feature of increasing popularity which might be integrated in some competing open-source project in the near future.

The Python scripts developed during this study are published online[8].

4. Real-time Mapping of Tracked Blendshapes to Control-rig

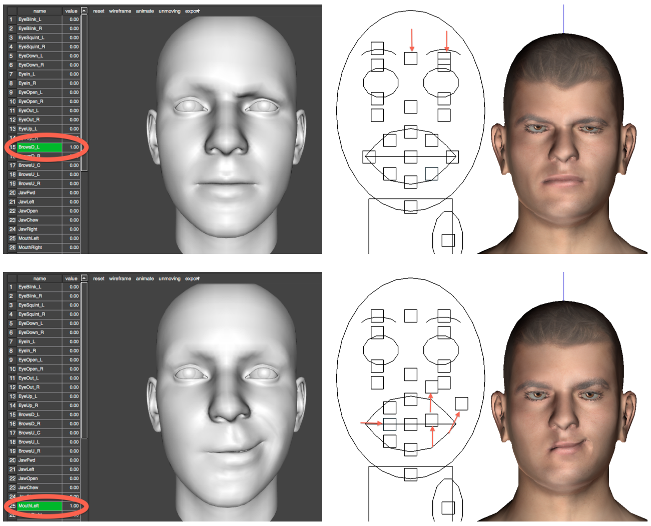
This section describes the retargeting method which converts the tracking information received by FaceShift into the MakeHuman control rig. The method has been developed as an addon of the Blender 3D authoring tool and is implemented in the Python language. The tracking information sent by FaceShift are received by the addon via UDP packets. The addon uses the following data:

1. a *track_ok* flag indicating whether the user face is being correctly tracked or not;
2. head rotation (yaw/pitch/roll angles as quaternion);
3. eyes rotation (yaw and pitch angles for each eye);
4. and a set of 48 float values, in the range [0,1], representing the weights of the 48 blend shapes animating the 3D model of the face.

The mapping script works as follows. If the *track_ok* flag is false, the received data are skipped and the MakeHuman rig isn't moved. Otherwise. The rotation of the head rotation is directly applied to controller *Neck*. The *theta* and *pi* rotation angles of the eyes are mapped as *-theta*, *0*, *phi* euler angles (in XYZ order) to the controllers *Eye_L* and *Eye_R*. Finally, the blend shape values are mapped to offsets of the facial control rig, as described in the rest of this section.

Each FaceShift blend shape value consist of a float number in the range $[0, 1]$. When all the blend shape values are at 0 the FaceShift face is in its neutral position. In Blender, each controller of the facial rig is associated to a 3D vector which acts as offset of the controller from its default position. The face is neutral when all offsets are at $< 0, 0, 0 >$. The main mapping procedure consist of applying the 48 FaceShift blend shape values (EyeBlink_L, EyeBlink_R, EyeSquint_L, ..., CheekSquint_R) to 3D vector offsets for the 20 controllers of the MakeHuman facial rig (PBrows, PBrow_R, PBrow_L, ..., PJaw).

The mapping is computed as a single multiplication between a matrix M and the blend shapes vector. Given the vector V_{bs} of 48 blend shape values, the resulting facial rig configuration vector V_r (20 3D vectors) is computed as $V_r = V_{bs} * M$, where M is a 48 lines by 20 columns matrix. Each column of the matrix is associated to a controller of the rig, while each line of the matrix is associated to a blendshape. Each line of the matrix represent how much the 20 controllers have to be offsetted when the corresponding blend shape is a maximum value (1.0). In other words, during the multiplication, the 48 offset vectors associated to a rig controller are scaled according to the actual blend shape values. The 48 scaled offsets are then accumulated (added) into a final offset. That's repeated for each controller.



	name	value
1	EyeBlink_L	0.00
2	EyeBlink_R	0.00
3	EyeSquint_L	0.00
4	EyeSquint_R	0.00
5	EyeDown_L	0.00
6	EyeDown_R	0.00
7	EyeIn_L	0.00
8	EyeIn_R	0.00
9	EyeOpen_L	0.00
10	EyeOpen_R	0.00
11	EyeOut_L	0.00
12	EyeOut_R	0.00
13	EyeUp_L	0.00
14	EyeUp_R	[illegible]
15	BrowsD_L	1.00
16	BrowsD_R	[illegible]
17	BrowsU_C	0.00
18	BrowsU_L	0.00
19	BrowsU_R	0.00
20	JawFwd	0.00
21	JawLeft	0.00
22	JawOpen	0.00
23	JawChew	0.00
24	JawRight	0.00
25	MouthLeft	0.00
26	MouthRight	0.00

	name	value
1	EyeBlink_L	0.00
2	EyeBlink_R	0.00
3	EyeSquint_L	0.00
4	EyeSquint_R	0.00
5	EyeDown_L	0.00
6	EyeDown_R	0.00
7	EyeIn_L	0.00
8	EyeIn_R	0.00
9	EyeOpen_L	0.00
10	EyeOpen_R	0.00
11	EyeOut_L	0.00
12	EyeOut_R	0.00
13	EyeUp_L	0.00
14	EyeUp_R	0.00
15	BrowsD_L	0.00
16	BrowsD_R	0.00
17	BrowsU_C	0.00
18	BrowsU_L	0.00
19	BrowsU_R	0.00
20	JawFwd	0.00
21	JawLeft	0.00
22	JawOpen	0.00
23	JawChew	0.00
24	[illegible]	0.00
25	MouthLeft	1.00
26	[illegible]	[illegible]

Figure 5. Examples from the calibration procedure of the matrix M. The picture shows the configuration for two of the 48 blend shapes: *BrowsD_L* (top) and *MouthLeft* (bottom). The blend shape is shown in the FaceShift configuration panel (left, the active blend shape is highlighted). The resulting expression is manually reproduced (as close as possible) in Blender using the control rig (right, the arrows indicate the moved controllers and the direction of the movement).

The matrix M has been manually configured by the authors, following an artistic-directed approach [7]. The FaceShift 3D model and the MakeHuman character in Blender have been put side-by-side. One by one, each of the 48 FaceShift blend shapes has been visualized at full weight. For each blend shape, the MakeHuman control rig has been configured to match (as much as possible) the Faceshift expressions. Each resulting control rig configuration is a list of 20 3D offsets. Each list has been inserted as line of the matrix M. Figure 5 shows examples of the side-by-side authoring process. Table 1 reports an extract of the resulting matrix.

After the multiplication, a logarithmic function is applied to each separate component (x, y, z) of the vectors in V_r. The logarithm limits the saturation resulting from the additive approach while leaving sensibility for low values. Given the control rig components offset range $[-0.25, 0.25]$, a log base of 3.5

[8] http://slsi.dfki.de - 7 Jan. 2015

Table 1. An extract of the matrix M mapping the blendshapes to the facial control rig.

		rig offsets (20)			
		PBrows	PBrow_R	...	PJaw
blend shapes (48)	EyeBlink_L	$0, 0, 0$	$0, 0, 0$	...	$0, 0, 0$
	EyeBlink_R	$0, 0, 0$	$0, 0, 0$	...	$0, 0, 0$
	...	...	...	...	...
	BrowsD_L	$0, 0, 0.12$	$0, 0, 0$	...	$0, 0, 0$
	BrowsD_R	$0, 0, 0.12$	$0, 0, 0.12$	...	$0, 0, 0$
	...	...	...	...	...
	CheekSquint_R	$0, 0, 0$	$0, 0, 0$	...	$0, 0, 0$

was empirically found to be satisfactory.

Actually, the rig controllers are constrained to move along 1 or 2 dimensions, yielding to a total of ca. 30 controlled degrees of freedom out of the 60 possible (20 vectors * 3 axes). Additionally, for most of the blend shapes, only one or two offsets out of 20 are used. As a consequence, most of the matrix elements are set to 0. Hence, there is room for optimizations. But this specific case is peculiar to the MakeHuman rig, while the algorithm is open to support more complex mapping configurations.

Two modification were applied to the original MaheHuman rig in order to increase its expressivity level. First, the range of movement of the Eyebrows was quite limited in the default MakeHuman rig. Hence, the range of movement of the PBrows, PBrow_L and PBrow_R controllers was doubled before calibrating the matrix M. Second, the corners of the mouth did not retract and widen enough. This is solved by controlling the value of the *mouth retraction* Expression Unit. Among the rich set of controllers in the MakeHuman rig, Expression Units are a set of facial expressions achievable through a set of dedicated sliders (range $[0, 1]$). Such expressions cannot in many cases be matched using the 3D controllers. The level of the mouth retraction control w_{mr} is calculated through an average of the offset applied to the mouth corners:

$$w_{mr} = \frac{PMouth\_L_z + PMouth\_R_z}{2} * 1.8$$

The Constant 1.8 has been again artistically determined.

Consider that the two above described modifications can be implemented by editing the control rig in Blender, rather than via explicit code, thus leaving the whole retargeting to a pure matrix multiplication. However, such option would have required a significant amount of 3D authoring which goes beyond the expertise of the authors.

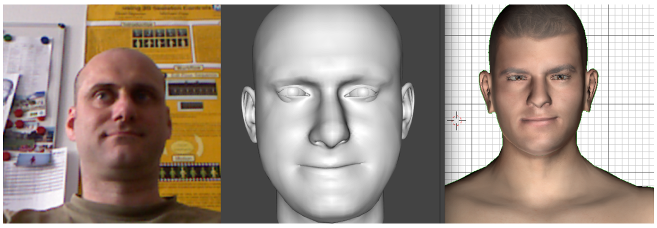


Figure 6. From left to right, screenshots of: the *live* Kinect color camera, the preview in FaceShift, the *rendered* character.

It is worth noticing that the resulting matrix M can be re-used for any FaceShift user and any additional character exported by MakeHuman. This is possible because both the blendshape values and the control rig offset spaces are normalized, respectively, in the ranges $[0, 1]$ and $[-0.25, 0.25]$.

5. Evaluation

This section describes the subjective user study which evaluates the animation workflow presented in the previous sections. The purpose of the user study was to give an estimation of the efficiency of the facial retargeting method and consisted of two stages. Firstly, the subjects used the system at the lab and recorded a set of dynamic facial expressions. The facial expressions were inspired by a set of *reference* videos performed by a professional actor. Each video showed one of the six Ekman [1] basic emotions: happiness, anger, disgust, surprise, fear, sadness. The recording stage produced two new set of videos: the *live* recordings of the subjects and the *renders* of the virtual character. Figure 6 shows an example. Secondly, another set of subjects (respondents, in the following) assessed the reference, the live, and the rendered videos via an online questionnaire.

The remaining of this section gives the details of the two evaluation stages and reports a discussion of the results.

5.1. Recording of Dynamic Expressions

Sixteen students from the campus participated to the recording stage. All of them were studying Computer Science or Communication Technologies. The call has been advertised in mailing lists, social networks, fliers, and posts on public billboards. Participants were rewarded with a 3D model of their head. The 3D model was acquired during the calibration of FaceShift. Two participants were female and two were male. They were informed of the purpose of the study, were instructed to watch the reference videos, and were invited to reproduce the emotions through their facial expression.

The performance capture has been conducted on a laptop and an additional external 22-inches monitor.

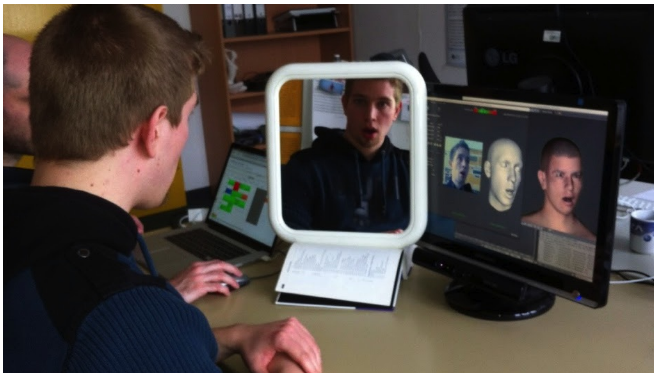


Figure 7. The user study setup.

The subjects were sitting in front of the monitor, without keyboard nor mouse. No written instruction was given to them since they were directed by an operator sitting next to them.

The Kinect camera was positioned in front of the monitor, under the screen (See Figure 7). Subjects's face distance from the Kinect was kept around 65cm, which is the optimal distance suggested by the FaceShift software. A mirror was provided to help users during the task.

The operator, using the laptop, assisted the subject during the calibration and provided guidance during the task. The operator was able to monitor the behaviour and the facial expression of the subject by looking the Kinect color camera preview on his screen.

Before starting the capture, subjects were assisted by the operator in configuring the FaceShift software. During the configuration, the FaceShift control window was shown on the screen of the subjects. The configuration includes a FaceShift training phase, which consists of holding a set of facial expressions while the Kinect capture depth and color information. After the training, users could play for one to two minutes with the FaceShift tracking interface, which showed the Kinect color camera output next to the 3D rendering of their head reconstruction.

After the configuration, the performance capture started. The FaceShift GUI was removed from the subjects' monitor, leaving space to the Blender GUI and the window of a video player. The Blender GUI was showing the virtual character exported by MakeHuman, which was moving its head and face, in real time, according to the subject's performance. Subjects were given one to two minutes to accommodate with the animated character. The task for the subject consisted of watching a reference movie of an actor performing a dynamic facial expression and reproducing the same facial expression on the virtual character. Starting from a neutral expression, after a short countdown, subjects had 15 seconds to reproduce the facial expression and go back to a neutral expression. The mirror was helping the subjects to fix the desired expression on their own face before applying it to the virtual character. Professional animators do the same: they explore different stances in front of a small mirror before authoring a facial pose using their traditional manual animation rig.

The six reference videos, displaying the six Ekman's emotions [1], belong to the DaFEx database of dynamic facial expressions [27] (Actor 1, high intensity). The videos show a professional female actress. High intensity can be considered as mild. For each subject the experiment lasted between 30 and 40 minutes: 8 to 10 minutes were needed for FaceShift configuration, the remaining time for the recordings.

5.2. Online Survey

The second part of the study was conducted as a public online questionnaire written in English, and advertised in mailing lists as well as social networks. The questionnaire was anonymous and consisted of 57 pages. The first and the third pages introduced the questionnaire and presented the context of the study. The second page gathered anonymous information about the respondents (age, experience with computers and video games). The following pages (54) displayed, in random order, the expression videos described below.

Four subjects ($U5$; $U6$; $U8$; $U10$) were selected among the 16 who participated to the recording stage. These subject were selected according to their "acting talent". The aim was to select users able to perform convincing facial expressions. The acting talent was first determined by the authors via a screening of the performances and later verified according to the rates of the online survey. The verification consisted in comparing the rates of the users to the ones of the professional actor.

As such, the 54 videos submitted for rating were composed as follows. Six *reference* videos, showing the six facial expressions performed by the professional actor; 24 *live* videos ($= 4 * 6$), showing the six facial expressions performed by the four selected subjects (these videos were recorder during the performance capture); another 24 *rendered* videos showing the virtual character animated by the performance capture.

Respondents of the online questionnaire were asked two questions per video. The first request was to rate each video along the six Ekman dimensions: happy, surprised, scared, sad, angry, disgusted. Each rating was done on a 6 point scale from 0 to 5.

The second question was: *"If you had to choose one word that qualifies the expression, what would it be?"*. Users had to select one among: happiness, surprise, fear, sadness, anger, disgust. This question was mainly used to detect possible inconsistent answers and, if necessary,

Value	Range
Happy	$[0\ldots5]$
Surprised	$[0\ldots5]$
Scared	$[0\ldots5]$
Sad	$[0\ldots5]$
Angry	$[0\ldots5]$
Disgusted	$[0\ldots5]$

discard respondents. Circa 30 minutes were needed to complete the on-line survey.

In total 96 participants answered the questionnaire. Among them, 30 completed it. The questionnaire was driven by the Limesurvey[9] software.

5.3. Similarity between contingency tables

This section describes the formula to calculate the similarity coefficient between two set of ratings (reference vs live, live vs rendered, ...).

Table 2 shows an example of the votes obtained from the video displaying a happy expression in the reference, live, rendered, and "ideal" version. The *ideal* table is an hypothetical distribution of votes where all the subjects give the maximum score to the intended emotion and zero to all the other emotions. The rates for the live and rendered videos are normalized to have the same total per line as the reference videos. The normalization is required to compute the similarity between types.

The similarity between two types of videos is computed using the generalized Jaccard similarity coefficient [9]. The coefficient J is defines as:

$$J(\mathbf{x}, \mathbf{y}) = \frac{\sum_i \min(x_i, y_i)}{\sum_i \max(x_i, y_i)}$$

where $\mathbf{x}$ and $\mathbf{y}$ are two vectors of the same length. The coefficient lies in the range $[0, 1]$. Comparing a vector with itself results in a coefficient value 1.0: a perfect match.

The coefficient introduced above can be used to measure the similarity between any video type. The tables are converted into vectors by serializing in the same order, line by line, the voting counts. Table 3 shows the results of the comparisons between some video types, divided by expression. The similarity between the *ideal* and the *reference* tables measures the performance of the actor. The similarity between the *ideal* and the *live* tables measures the performance of the subjects. The similarity between the *ideal* and the *rendered* tables assess the performance of the virtual character. Finally, the similarity between the *live* and the *rendered* tables assesses the goodness of the retargeting method.

The Jaccard coefficient is always positive.Hence, to evaluate which of the two compared type performs better, or worse, the coefficient must be compared with the averages of the input vectors. The coefficients shown in Figure 3 are discussed in detail in the next section.

5.4. Results and Discussion

Complete vs incomplete questionnaires. In total, 96 respondents participated to the survey. Only 30 respondents completely filled the questionnaire, probably due to the excessive time required (ca. 30 minutes). Before including the ratings of the incomplete questionnaires in the overall statistics, ANOVA tests between the ratings obtained in a dimension (0 to 5) vs. the questionnaire status (complete, incomplete) have been conducted. An ANOVA test has been conducted separately for each dimension (happiness, anger, ...). For all of the dimensions the p-value was > 0.1, meaning that, with at least 10% confidence, there is not significant difference between the scores obtained in the complete questionnaire vs. the scores obtained in the incomplete questionnaires. We therefore included the partially filled questionnaires into the statistical analysis.

Reliability of the respondents. The respondents of the questionnaires rated the videos similarly to previous published results. Figure 8 shows the ratings obtained by the six *reference* videos along each dimension.

These results can be compared with the first row of Table 3, showing the coefficients of the comparison between the ideal rating and the reference videos. Our subjects recognized pretty wrongly the *disgust* dimension (J=0.438). All the other results match with the results claimed in [27], who evaluated their reference videos on 80 subjects. Quoting [27]: "anger seems to be the emotion which is best recognized, [...]. On the other hand, fear seems to be the emotion which is worse recognized [...]. Happiness seems to be the most stable emotion".

The average of the coefficient (average J=0.656) can be taken as measurement of the talent of the actor. The standard deviation (0.122) given an estimation of the consistency of the acting across all the dimensions.

"Acting talent" of the performers. Figure 9 shows the results of the rates obtained by the four subjects selected as best performers. This graphs can be compared with the second row of Table 3, showing the coefficients of the comparison between the ideal condition and the live videos. The coefficients confirm that, as average, the subjects performed worse than the professional actors (average J=0.536 vs. 0.656). There is an exception is the disgust dimension, where subjects

[9] http://www.limesurvey.org/ - 7 Jan. 2015

Table 2. The rates distribution for the happiness expression: i) the rates obtained by the reference videos, ii) the rates for the live videos, iii) the rates for the rendered videos, and iv) the ideal distribution. The live and rendered rates have been scaled to match the same total per line of the reference rates. The ideal table is generated ad-hoc.

Reference	0	1	2	3	4	5	Tot
Hap	0	0	1	4	12	33	50
Ang	45	5	0	0	0	0	50
Dis	46	4	0	0	0	0	50
Sur	40	6	2	2	0	0	50
Fea	45	4	0	1	0	0	50
Sad	43	5	1	1	0	0	50

Live	0	1	2	3	4	5
Hap	0.00	0.63	2.22	7.91	15.82	23.42
Ang	47.78	2.22	0.00	0.00	0.00	0.00
Dis	47.15	2.85	0.00	0.00	0.00	0.00
Sur	36.71	8.54	3.48	1.27	0.00	0.00
Fea	48.10	1.90	0.00	0.00	0.00	0.00
Sad	45.25	3.48	0.32	0.95	0.00	0.00

Rendered	0	1	2	3	4	5
Hap	8.72	2.33	5.52	10.17	12.21	11.05
Ang	38.95	4.36	1.74	1.45	1.45	2.03
Dis	42.73	4.36	1.45	1.16	0.29	0.00
Sur	41.28	6.98	0.58	0.58	0.58	0.00
Fea	41.86	4.07	0.87	1.74	0.00	1.45
Sad	45.06	2.91	1.45	0.29	0.29	0.00

Ideal	0	1	2	3	4	5
Hap	0	0	0	0	0	50
Ang	50	0	0	0	0	0
Dis	50	0	0	0	0	0
Sur	50	0	0	0	0	0
Fea	50	0	0	0	0	0
Sad	50	0	0	0	0	0

Table 3. The Jaccard coefficient between selected couples of video types.

Groups	Meaning	Hap	Ang	Dis	Sur	Fea	Sad	Average	Std dev
ideal vs. reference	Actor performance	0.724	0.769	0.438	0.661	0.560	0.781	0.656	0.122
ideal vs. live	Subjects' acting perf.	0.707	0.475	0.599	0.540	0.445	0.449	0.536	0.094
ideal vs. rendered	Virtual character perf.	0.583	0.533	0.459	0.480	0.380	0.387	0.470	0.073
live vs. rendered	Retargeting quality	0.754	0.784	0.738	0.842	0.834	0.793	0.791	0.038

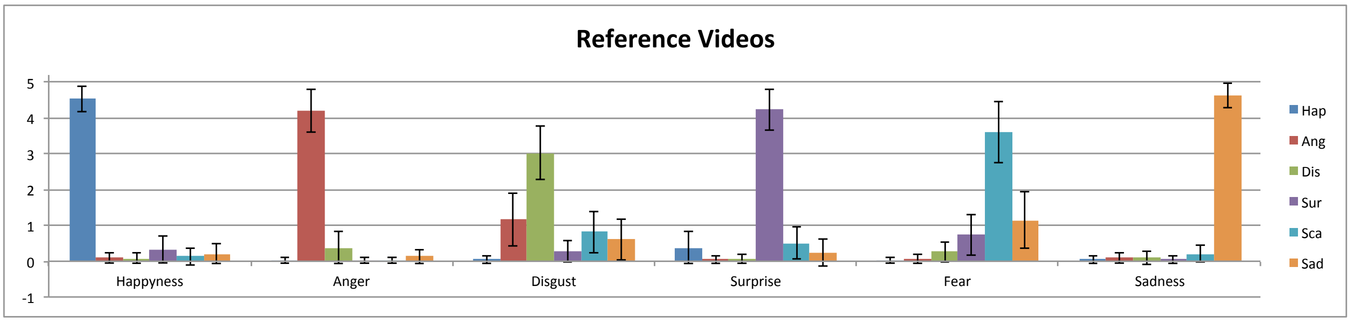


Figure 8. Rating obtained for the reference videos (actor) for each expression.

performed better than the actor (J=0.599 vs. 0.438). The fear dimension performed really bad (J=0.445), with the subjects misinterpreting the expression as *surprised* rather than *scared*.

Emotions conveyed by the virtual character. Figure 10 shows the rates obtained by the rendered videos. This graphs can be compared with the third row of Table 3, showing the coefficients of the comparison between the ideal condition and the rendered videos. These values indicate that that the 3D rendered videos have been rated worse than the live videos (average J=0.470 vs. 0.536), with the exception on anger (J=0.533 vs. 0.475).

Efficiency of the mapping method. Lastly, a comparison between the rates obtained by the live vs. the rendered videos gives an estimation of the quality of the facial retargeting method. The rates obtained by the live and the rendered videos can be visually compared by looking at Figures 9 and 10. It is possible to notice a similarity between the two performances. The similarity is numerically expressed by the coefficients reported in the fourth row of Table 3. The overall quality of the retargeting method is expressed by its average coefficient (average J=0.791). The low standard deviation value (0.038) indicates that the retargeting is consistent across all the six dimensions.

It is worth noticing that this evaluation metric allows for an estimation of the quality of the retargeting even if the performers do not present high acting talent. In fact, as can be seen in Figure 9, in the live videos the recognition of Anger, Fear and Sadness is pretty low, with the rates scattered through all other dimensions (except happiness). Also, Fear is misinterpreted as Surprise. Nevertheless, the same phenomenon can be observed for the rendered videos (Figure 10). The high values of the coefficients of the comparison (Respectively J=0.784, 0.834, and 0.793 for the anger, fear, and sadness dimensions) confirm this similarity.

6. Conclusion

To sum up, we presented an animation authoring workflow enabling unexperienced users to intuitively and quickly author facial animations. The workflow maps the facial performance to a control rig, allowing for convenient editing at any point of the animation pipeline. We suggest that this system might also be used by professional animators to increase their productivity and foster their creativity by allowing them to explore and test multiple alternative performance captured animations while maintaining the possibility to seamlessly refine and edit the captured animation using traditional animation rigs.

Allowing users to manually edit an animation after it has been recorded requires a retargeting method that maps the captured data – recorded as vectors of blend shapes values – onto the animation rig – described as a combination of sliders – that is eventually edited by the user.

We assessed the accuracy of our facial retargeting method by conducting a user study where participants assessed the emotions conveyed by the facial expression displayed in the control and the authored animation. Contrary to existing evaluation methods, the method we used accounts for possible misinterpretations of the intended emotion.

In the future, we would like to perform further study involving both professional actors, casual users, and experienced animators. This would give better evidence of the efficiency of the evaluation method. The evaluation will be performed on an updated version of the facial retargeting method, which will map to a bone-based facial control rig rather than to the slider-based one that we used in this work.

References

[1] Ekman, P., Friesen, W.V. and Ellsworth, P. (1972) *Emotion in the human face: guide-lines for research and an integration of findings*, no. PGPS-11 in Pergamon general psychology series (New York: Pergamon Press).

[2] Kipp, M., Heloir, A. and Nguyen, Q. (2011) Sign language avatars: animation and comprehensibility. In *Proceedings of the 10th international conference on Intelligent virtual agents*, IVA'11 (Berlin, Heidelberg: Springer-Verlag): 113–126.

[3] Lewis, J.P. and Anjyo, K.i. (2010) Direct manipulation blendshapes. *IEEE Computer Graphics and Applications* **30**(4): 42–50. doi:10.1109/MCG.2010.41.

[4] Witkin, A. and Kass, M. (1988) Spacetime constraints (ACM Press): 159–168. doi:10.1145/54852.378507.

[5] Osipa, J. (2003) *Stop staring: facial modeling and animation done right* (San Francisco: SYBEX).

[6] Sanna, A., Lamberti, F., Paravati, G. and Rocha, F.D. (2012) A kinect-based interface to animate virtual characters. *Journal on Multimodal User Interfaces* doi:10.1007/s12193-012-0113-9.

[7] Pighin, F. and Lewis, J.P. (2006) Facial motion retargeting. In *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06 (New York, NY, USA: ACM). doi:10.1145/1185657.1185842.

[8] Weise, T., Bouaziz, S., Li, H. and Pauly, M. (2011) Realtime performance-based facial animation. In *ACM SIGGRAPH 2011 papers*, SIGGRAPH '11 (New York, NY, USA: ACM): 77:1–77:10. doi:10.1145/1964921.1964972.

[9] Tan, P., Steinbach, M. and Kumar, V. (2006) *Introduction to Data Mining*, Pearson International Edition (Pearson Addison Wesley).

[10] Wilbers, F., Ishi, C. and Ishiguro, H. (2007) A blendshape model for mapping facial motions to an android. In *IEEE/RSJ International Conference on Intelligent Robots and Systems IROS* (IEEE): 542–547. doi:10.1109/IROS.2007.4399394.

[11] Conner, B.D., Snibbe, S.S., Herndon, K.P., Robbins, D.C., Zeleznik, R.C. and van Dam, A. (1992) Three-dimensional widgets (ACM Press): 183–188. doi:10.1145/147156.147199.

[12] Magnenat-Thalmann, N. and Thalmann, D. (2004) *Handbook of virtual humans* (Chichester, England ; Hoboken, NJ: Wiley).

[13] Tolani, D., Goswami, A. and Badler, N.I. (2000) Real-time inverse kinematics techniques for anthropomorphic limbs. *Graphical Models* **62**(5): 353–388. doi:10.1006/gmod.2000.0528.

[14] Villagrasa, S. and Susin, A. (2009) Face! 3d facial animation system based on facs. In *Proceedings of the IV Iberoamerican Symposium in Computer Graphics*.

[15] Bastioni, M., Re, S. and Misra, S. (2008) Ideas and methods for modeling 3D human figures: the principal algorithms used by MakeHuman and their implementation in a new approach to parametric modeling. In *Proceedings of the 1st Bangalore Annual Compute Conference*, COMPUTE '08 (New York, NY, USA: ACM): 10:1–10:6.

[16] Alexander, O., Rogers, M., Lambeth, W., Chi-Ang, M. and Debevec, P. (2009), Creating a photoreal digital

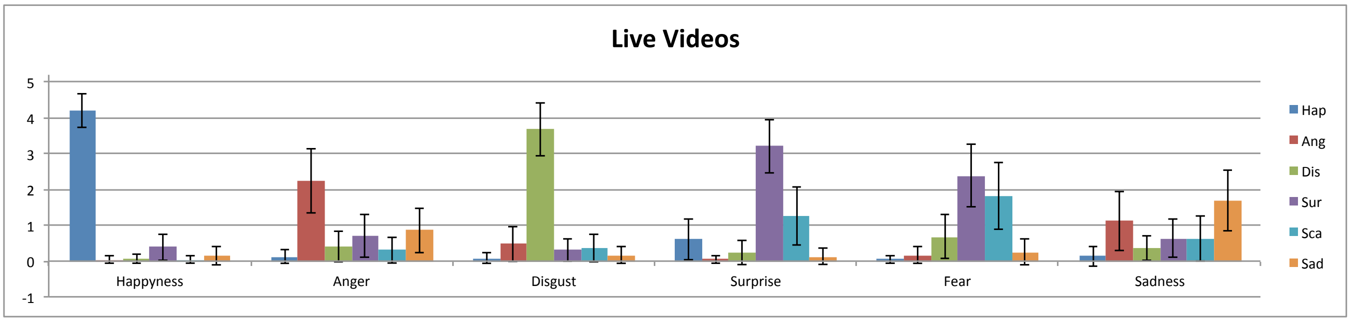


Figure 9. Rating obtained for the live videos (subjects) for each expression.

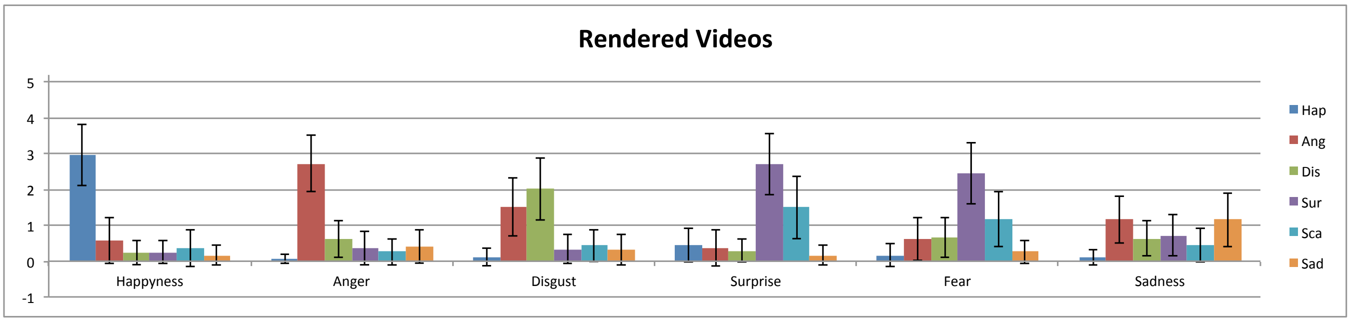


Figure 10. Rating obtained for the rendered videos (avatar) for each expression.

actor: The digital emily project.

[17] Chang, E. and Jenkins, O.C. (2008) Sketching articulation and pose for facial animation. In Deng, Z. and Neumann, U. [eds.] *Data-Driven 3D Facial Animation* (London: Springer London), 145–161.

[18] Kipp, M. and Nguyen, Q. (2010) Multitouch puppetry: creating coordinated 3D motion for an articulated arm (ACM Press): 147. doi:10.1145/1936652.1936682.

[19] Robertson, B. (1988) Mike the talking head. *Computer Graphics World* **11**(7).

[20] Menache, A. (1999) *Understanding Motion Capture for Computer Animation and Video Games* (San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.).

[21] Shin, H.J., Lee, J., Shin, S.Y. and Gleicher, M. (2001) Computer puppetry: An importance-based approach. *ACM Transactions on Graphics* **20**(2): 67–94. doi:10.1145/502122.502123.

[22] Liu, K.Y., Ma, W.C., Chang, C.F., Wang, C.C. and Debevec, P. (2011) A framework for locally retargeting and rendering facial performance. *Computer Animation and Virtual Worlds* **22**(2-3): 159–167. doi:10.1002/cav.404.

[23] Curio, C., Breidt, M., Kleiner, M., Vuong, Q.C., Giese, M.A. and BÃijlthoff, H.H. (2006) Semantic 3D motion retargeting for facial animation. In *Proceedings of the 3rd Symposium on Applied Perception in Graphics and Visualization* (ACM Press): 77. doi:10.1145/1140491.1140508.

[24] Seol, Y., Lewis, J., Seo, J., Choi, B., Anjyo, K. and Noh, J. (2012) Spacetime expression cloning for blendshapes. *ACM Transactions on Graphics* **31**(2): 1–12. doi:10.1145/2159516.2159519.

[25] Lewis, J.P. and Pighin, F. (2005) Cross-mapping. In *ACM SIGGRAPH 2005 Courses* (ACM Press): 3. doi:10.1145/1198555.1198583.

[26] Song, J., Choi, B., Seol, Y. and Noh, J. (2011) Characteristic facial retargeting. *Computer Animation and Virtual Worlds* **22**(2-3): 187–194. doi:10.1002/cav.414.

[27] Battocchi, A., Pianesi, F. and Goren-Bar, D. (2005) A first evaluation study of a database of kinetic facial expressions (DaFEx). In *Proceedings of the 7th international conference on Multimodal interfaces*, ICMI '05 (New York, NY, USA: ACM): 214–221.

[28] Costantini, E., Pianesi, F. and Prete, M. (2005) Recognising emotions in human and synthetic faces: The role of the upper and lower parts of the face. In *Proceedings of the 10th International Conference on Intelligent User Interfaces*, IUI '05 (New York, NY, USA: ACM): 20–27. doi:10.1145/1040830.1040846.