Universiteit Utrecht

User studies on real-time facial motion retargeting in online multiplayer games

Author: Zhi Kang Shao
Project supervisor: dr. ir. Arjan Egges
Daily supervisor: dr. ir. Ronald Poppe

Master thesis
ICA-3943372
October 16, 2014

Abstract

In most online games, communication between players is limited to text messages, voice chat and predefined animations. In this work we look at the utility of real-time face tracking to animate the faces of player avatars, i.e. facial motion retargeting, in online games. We study the effects of tracking accuracy and timing via user evaluations of facial animations with simulated errors. From the results, we derive a scheme for balancing performance parameters to maximize perceived animation quality. We present an implementation of facial motion retargeting in the form of an add-on for the multiplayer game Garry's Mod. Our data gathered from private and public game sessions with the add-on indicates that directly controllable facial expressions encourage player interaction and are useful in both competitive and social gameplay.

Contents

1 Introduction **4**

1.1 Goal . . . 4

1.2 Experiments . . . 4

1.3 Structure . . . 5

2 Related work **6**

2.1 Facial expressions . . . 6

2.2 Face detection and tracking . . . 6

2.3 Facial feature detection . . . 7

2.4 Facial motion retargeting . . . 8

3 Real-time facial motion retargeting in Garry's Mod **9**

3.1 Requirements and assumptions . . . 9

3.2 Method overview . . . 9

3.3 Facial expression and head rotation tracking . . . 10

3.4 Face tracking verification . . . 11

3.5 Facial feature extraction . . . 12

3.6 AU normalization and user calibration . . . 14

3.6.1 User calibration process . . . 14

3.6.2 Alternatives to user calibration . . . 16

3.7 Network synchronization . . . 17

4 Experiment 1: Effects of tracking timing and accuracy **18**

4.1 Experimental setup . . . 18

4.1.1 Participants . . . 18

4.1.2 Materials . . . 19

4.1.3 Design . . . 21

4.1.4 Procedure . . . 22

4.2 Results . . . 23

4.2.1 Correlation . . . 23
4.2.2 Comparison to expectation . . . 23
4.2.3 Linear regression . . . 25
4.2.4 Example: Optimization using linear trends . . . 26

5 Experiment 2: Evaluation of utility through local play tests 27
5.1 Experimental setup . . . 27
5.1.1 Participants . . . 27
5.1.2 Materials . . . 27
5.1.3 Design . . . 27
5.1.4 Procedure . . . 28
5.2 Results . . . 28

6 Experiment 3: Evaluation of public online use 30
6.1 Experimental setup . . . 30
6.1.1 Integrated statistics . . . 30
6.1.2 Questionnaire . . . 31
6.2 Results . . . 31
6.2.1 Overall statistics . . . 31
6.2.2 Overall statistics per game mode . . . 33
6.2.3 Facial expression activity per game mode . . . 35
6.2.4 Questionnaire results . . . 37

7 Discussion and conclusion 41

Appendix A LAN session questionnaire 45

Appendix B Public in-game questionnaire 46

1 Introduction

Although facial expressions have been shown to play an important role in face-to-face conversation [7], in most online multiplayer games communication between players is limited to text messages, voice chat and predefined animations. Allowing the player to directly control his avatar's facial expression through face tracking has not yet found widespread application in modern games, although one notable exception exists.[1]

1.1 Goal

In this work we look at the benefits of using face tracking to control facial animation in real-time in multiplayer games. We refer to the act of tracking facial motions to drive facial animation as facial motion retargeting. We are interested in the following:

- How do tracking accuracy and timing affect the perceived quality of the facial animation?
- To what extent does facial motion retargeting enhance the multiplayer experience?
- In what types of games are gamers more likely to use facial motion retargeting?

To answer these questions we performed a series of experiments that involve subjects evaluating facial animations with simulated errors in accuracy and timing, and subjects using facial motion retargeting in private and public sessions of the multiplayer game Garry's Mod. Through the combined results of the experiments, we attempt to answer whether the feature provides lasting benefits to modern multiplayer games.

1.2 Experiments

Our first experiment is designed to study effects of tracking accuracy and timing on the perceived quality of the resulting facial animation. We processed videos with facial motion into high quality facial animation and then modify the animation to simulate errors that can arise due to general limitations of face tracking algorithms, user hardware and network performance. The videos were included in an online survey in which participants were asked to rate the animations displayed in the videos. A total of 136 usable response sets were gathered, from which we derive the effects of the simulated performance variables on perceived animation quality.

In light of the remaining experiments we developed FlexPoser: a real-time facial motion retargeting add-on for the online game Garry's Mod (GMod). GMod allows players to implement their own game rules via custom game modes and add additional features in the form of add-ons. With the FlexPoser add-on, players in a multiplayer match can connect their webcams, after which their facial expressions and head rotation are tracked and applied to their in-game avatar for all players to see. For the real-time face tracking task, FlexPoser makes use of the FaceTracker library presented by Saragih et al. [20].

The second experiment is aimed at determining whether motion retargeting adds any value to the multiplayer gaming experience. Findings by Ekman et al. [7] suggest that facial expressions play a crucial role in the formation of even casual relationships. Thus, adding a feature for facial expressiveness may prove beneficial to multiplayer experiences which are inherently social albeit with varying degrees. To look into this we organized a number of local GMod game sessions in a LAN setup with the FlexPoser add-on enabled and every player's facial expression

[1] EverQuest II by Sony Online Entertainment

simultaneously being tracked and applied to their character. At the end of each session we asked players to fill in a questionnaire to describe their use and experience.

The third experiment is aimed at identifying types of games in which players are more likely to make use of facial motion retargeting, as well as attempting to answer whether the feature has lasting potential. To this end we distributed the add-on via the official forums of Garry's Mod[2] and Steam Workshop[3], and gathered use statistics (with consent) from public game servers during a period of 55 days. At the end of each game session, a server with FlexPoser and statistics enabled would send session and tracking statistics to our web server. Furthermore, players submitted information about their experience via an in-game questionnaire.

1.3 Structure

The remainder of this work is structured as follows. In Section 2 we discuss related work in the fields of facial expressions, tracking and motion retargeting. Section 3 gives a detailed description of FlexPoser, the facial motion retargeting add-on we implemented for Garry's Mod. Sections 4-6 present per experiment the setup and the results. Section 7 concludes this work and gives suggestions for future research.

[2]http://www.facepunch.com/
[3]http://steamcommunity.com/sharedfiles/filedetails/?id=282498239

2 Related work

This section presents related work in the fields of facial expressions, tracking and motion retargeting.

2.1 Facial expressions

Studies by Ekman [8] suggest that facial expressions play a large role in the formation of interpersonal attachments. For example, in one of his studies [7] he interviewed people with congenital facial paralysis who report to have great difficulty in developing and maintaining even casual relationships due to having no capability for facial expressiveness. We derive from Ekman's findings the idea that giving players control over the facial expression of their virtual avatar may be beneficial to online social experiences. Currently, faces of player avatars are motionless in most multiplayer games.

Researchers in the field of facial expression recognition commonly represent the state of a face as a list of action units (AUs), where each action unit represents an individual facial motion such as the raising of an eyebrow. Ekman and Friesen proposed a standard coding system, Facial Action Coding System (FACS) [9], which represents a face's state in terms of 46 AUs which roughly correspond to facial muscles. FACS has since become a widely used model in facial expression recognition research [2, 17, 22, 15, 12]. In our implementation, we use a subset of FACS to represent the state of a face.

Note that, in the context of facial expression recognition, a facial expression can refer to a named expression such as 'surprise', but it can also refer to a set of parameters that represent the physical state of a face. We will use 'facial configuration' to refer to the latter, to distinguish between the two.

2.2 Face detection and tracking

For our purpose of real-time facial motion retargeting, we need to automatically detect facial configurations in real-time from a live camera feed. A required step for this is face detection: the automated detection of the presence and location of faces in an image. This step is required for any set of images in which faces have not been labeled beforehand, such as in our case. Face tracking refers to continuous face detection in image sequences, such as recorded videos and live camera feeds. Both face detection and tracking are open problems and techniques for these tasks face many challenges including bad lighting conditions, the wide spectrum of personal appearances, partially occluded faces, head rotation, etc. Zhang and Zhang [25] present a survey about recent advances in face detection and tracking algorithms.

Viola and Jones [23] developed a fast face detection algorithm with high detection accuracy. They use a cascade of classifiers to quickly discard background regions of an image, leaving only regions that potentially contain key features. The key features that they look for are Haar-like features, i.e. they resemble Haar basis functions, selected using the AdaBoost learning algorithm. The output of the algorithm is a list of 2D bounding boxes, which are the rectangular regions of the image that contain a frontal face. Furthermore, they present an image repesentation that allows fast computation of Haar-like features on multiple scales. Their algorithm has achieved high detection accuracy when tested with the MIT + CMU face database [19].

Although face detection techniques can also be applied to individual frames in an image sequence, face tracking techniques generally retain a state based on previously processed frames

and use the extra information to gain an advantage over single image-based techniques, mainly speed and correctness in the presence of temporary occlusion and motion blur. In the following we will discuss face tracking algorithms that simultaneously track the global position of the face and the positions of facial features.

2.3 Facial feature detection

After the position of a face has been found in an image or image sequence, information about facial features must be extracted to capture the expression that the person is making. Different approaches have been suggested. Holistic approaches estimate facial features while representing the face as a whole unit, such as a deformable 3D mesh, while analytic approaches represent the face as a set of points or templates mapped to parts of the face such as the eyes and mouth.

Dornaika and Davoine [6] present a holistic approach. They developed a real-time face tracking algorithm, which requires a specific face texture as input. The face texture depicts a cutout of the subject's face without a background, and is used both for determining the initial position of the face and for updating the mesh. This is demonstrated in Figure 1. The technique uses a particle filtering method [18] of which the objective is to project a 3D deformable facial mesh onto the current image such that the appearance of the mesh in texture space resembles that of the prepared facial texture.

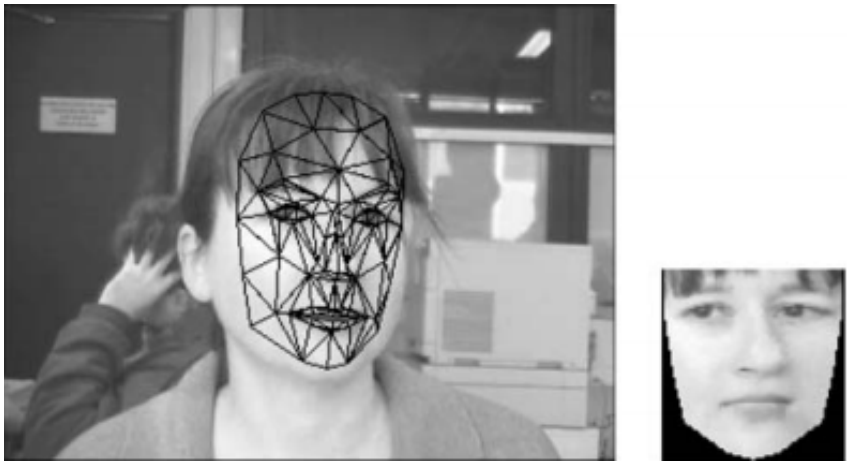

Figure 1: Mesh fitting approach by Dornaika and Davoine. Image taken from [5].

This is an example of an active appearance model (AAM), where a known statistical model of the user is matched to a new image. In this case, the face texture represents the appearance of individual facial features, but also their structure with respect to each other. The structure places restrictions on where facial features can be located with respect to each other. These restrictions are utilized by holistic approaches by not considering estimations that have an invalid structure. In contrast, by detecting facial features independently, analytic approaches can produce output with an invalid structure. In these cases, the output must be modified at a later stage to produce correct results.

Bartlett et al. [2] present an analytic approach: they experimented with different classifiers for automatically detecting 20 AUs from the FACS standard. Specifically, they focused on detecting AUs from video recordings of spontaneous expressions, which have been shown to differ from posed expressions in both the intensity and timing of facial motions [3, 10]. During

experimentation, they compared their AU output to manual AU labeling by five certified FACS coders. They obtained their best detection performance using AdaBoost with Gabor filters, and training support vector machines on the outputs of the filters selected by AdaBoost.

Saragih et al. [20, 21] present a hybrid approach: they search for a set of facial features, but then use the tracked positions of the features to fit a mesh to the tracked user's face. They update the positions of the features using a method coined subspace constrained mean-shifts (SCMS). This is a two-phase method, in which first response maps are computed individually for their tracked features, in a patch around their current tracked position. This is demonstrated in Figure 2. Then, the feature responses are jointly iteratively maximized using a mean-shift procedure [4].

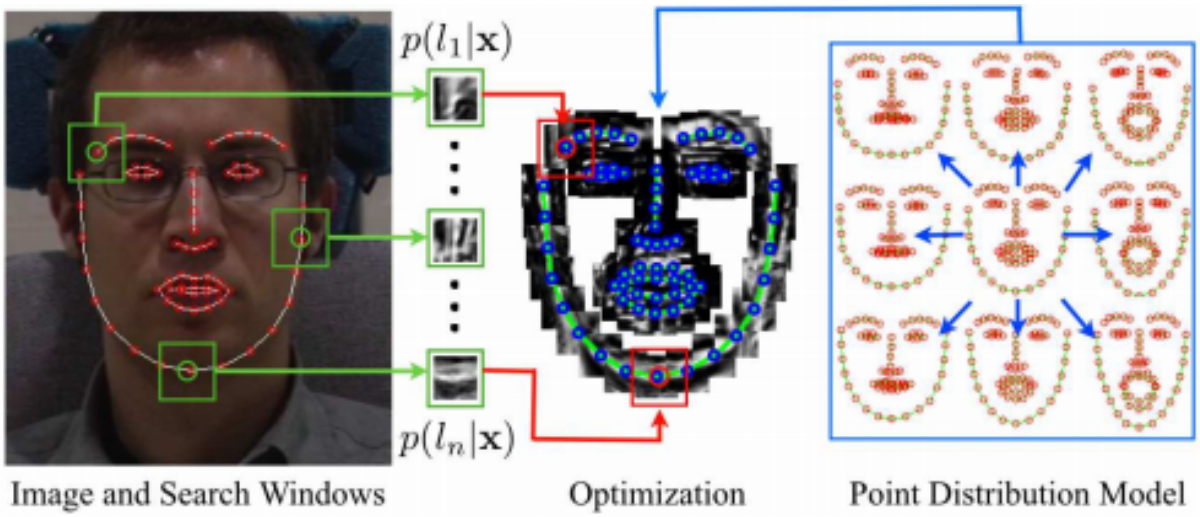

Figure 2: Mesh fitting approach by Saragih et al. Image taken from [20].

2.4 Facial motion retargeting

Ennis et al. [11] played back motion captured facial and body motions on virtual characters, where the motions each displayed an emotion such as anger or happiness. They showed instances of only facial motion, only body motion, and facial and body motion to participants and measured their success in recognizing the displayed emotions. They found that while best results are obtained when both body and facial motion are displayed, either type of motion is sufficient for steady recognition of emotions. We derive from their findings that facial motion retargeting by itself can convey emotions, though we must keep in mind that real-time tracked facial motions using only consumer-grade webcams will not look as accurate as motions tracked using motion capture technology.

In the remainder of this work, we use facial expression to refer to a set of parameters that represent the physical state of a face. In our approach to facial motion retargeting we are only interested in replicating the motion of a user's face. We leave classification of facial expressions to the observers: the other players in a multiplayer game.

3 Real-time facial motion retargeting in Garry's Mod

This section presents FlexPoser: a real-time facial motion retargeting pipeline implemented as an add-on for the game Garry's Mod. This section is not meant to advocate a specific approach to real-time facial motion retargeting, but rather to clarify how the facial motion retargeting was achieved in the multiplayer experiments discussed in Sections 5 and 6.

3.1 Requirements and assumptions

We will first present the requirements that we composed for a real-time face tracking algorithm to be useful in a practical gaming context, and the assumptions that must be met in order for our final implementation to work as intended.

For the purposes of FlexPoser a face tracking component is required that would work for many users at home while they are playing a game. The users are not required to be knowledgeable about face tracking technology. Taking this into account, we composed the following list of requirements for the face tracking component:

- must run real-time ($>$ 10 fps), regardless of webcam image conditions
- must work in the presence of glasses, facial hair, headset microphones
- must work for a wide spectrum of people of all ages starting from young adults (13+ yrs)
- must be portable, i.e. easy to run without external dependencies

Furthermore the availability of source code would be a large plus. We selected the library FaceTracker by Saragih et al. [20], because it satisfied all our requirements and its source code was available online. With the integration of FaceTracker, the facial motion retargeting is expected to work robustly under the following conditions:

- the face is present in the image and entirely visible
- the face is not rotated more than 30 degrees away from frontal view, on any axis
- the face is well and uniformly lit (avoiding high contrast and dark shadows)
- there is no over-exposure to daylight from the back or side of the person

When any of the conditions is not met, accurate motion retargeting is not guaranteed. However, it will also not noticeably affect computation time per frame. Furthermore, we assume that when multiple faces are present in the image, the one that appears largest is intended to be tracked. FaceTracker also makes this assumption.

3.2 Method overview

FlexPoser's facial motion retargeting pipeline consists of four steps, which are displayed in Figure 3. We use FaceTracker to track the user's facial movements and head rotation. FaceTracker maintains a 3D deformed facial mesh that accurately tracks the user's eyelid, eyebrow and lip movements, as well as the global parameters scale, position and rotation. Distances between specific pairs of vertices are extracted from the deformed mesh which are then transformed to 15 AU activation levels using user calibrated values. The tracked AUs and global parameters are listed in Section 3.5, Table 1.

A second opinion style verification task is employed using Viola-Jones' face detection algorithm [23]. Our test runs show that Viola-Jones' algorithm is more robust than FaceTracker for detecting faces in the presence of difficult image conditions such as bad lighting, visually

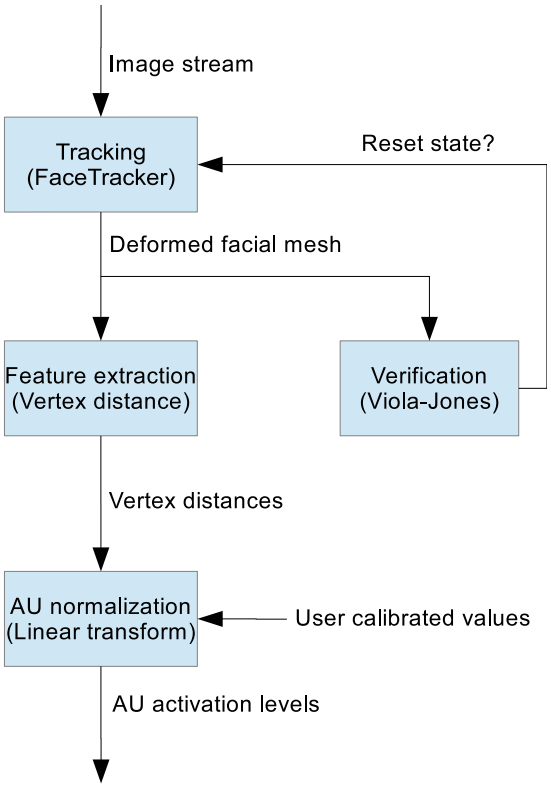

Figure 3: Method overview

convoluted backgrounds and background motion. Recall that in the presence of multiple faces, we assume the largest face is the one to be tracked. Thus, when FaceTracker's mesh does not match Viola-Jones' largest detected face, we conclude that FaceTracker's state is invalid and reset the tracking process. Upon a reset FaceTracker redetects the face in the next frame, using no information from previous frames. Details on the verification are presented in Section 3.4.

The verification step effectively reduces the duration of drift errors that sometimes arise when the conditions listed in Section 3.1 are not met. Because the add-on is intended for use at home and by users not knowledgeable about face tracking technology, we must assume that often image conditions are far from perfect despite presenting guidelines. In the following sections, we will present design motivations and implementation details for the individual tasks.

3.3 Facial expression and head rotation tracking

We use the FaceTracker library by Saragih et al [20] for tracking inner-facial and head movements. FaceTracker's approach to face tracking, coined subspace constrained mean-shifts by the authors, is a two-phase approach in which first response maps for each feature individually in a patch around their current tracked position, and then the feature responses are jointly iteratively maximized using mean-shifts. For details please refer to Saragih et al. [20, 21].

Its output is a 3D deformed facial mesh, which is defined by a locally deformed mesh and global parameters. The locally deformed mesh is scale, rotation and translation invariant, allowing us to inspect facial deformation without worrying about the face's position or rotation. The

global parameters describe the face's position and rotation in the image. A visualization of FaceTracker's output can be seen in Figure 4.

Figure 4: FaceTracker's output is a 3D deformed facial mesh.

3.4 Face tracking verification

Since face tracking techniques retain a state based on previous frames, such techniques must be wary of errors accumulated over multiple frames (drift). FaceTracker attempts per-frame fault estimation in an attempt to detect when face tracking has gone wrong and resets the state when it has estimated that the current state is invalid. The subject's face is redetected the next frame.

Despite this mechanism we noticed during test sessions that FaceTracker's state could be invalid for longer periods of time without being detected by the fault recovery mechanism. This was especially the case when a face is erronously detected in the background and that particular part of the image is not moving. We noticed that often in these cases a state reset would result in FaceTracker correctly finding the user's face again the next frame, assuming the required conditions are met (see 3.1). To improve robustness we added another fault detection mechanism to FlexPoser, which verifies FaceTracker's output using an alternative face detection method and resets its state if it does not pass the verification.

We selected Viola and Jones' algorithm [23] as alternative face detection method, because of its high accuracy in detecting faces. To be able to compare FaceTracker's and Viola-Jones' output, we project all the vertices of FaceTracker's deformed mesh to image space and construct the tightest 2D bounding box that contains all these points. We then compare this bounding box with the 2D bounding box of the largest face detected by Viola-Jones in the following way. Let $area(F)$ be the area of the bounding box containing FaceTracker's result, $area(V)$ be the area of the bounding box produced by Viola-Jones, and let $area(F \cap V)$ be the area of their intersection. FaceTracker's state is regarded as valid if and only if:

$$\frac{area(F \cap V)}{area(F)} > 0.5 \quad \wedge \quad \frac{area(F \cap V)}{area(V)} > 0.7.$$

That is, we compare two measures of rectangle overlap to constant thresholds. If the rectangles do not sufficiently overlap, FaceTracker's state is regarded as invalid and is then reset. We

selected these values using the June 1, 2013 implementation of FaceTracker[4] and Viola-Jones implementation in OpenCV 2.4.8[5] with the included frontal face training values.

The thresholds were picked with preference to false positives (falsely regarded as valid) over false negatives. We prefer false positives because resetting FaceTracker when face tracking is currently accurate (i.e. a false negative) will likely result in the face being detected in the same position the next frame, resulting in repeated, ineffective resets. In practice we have found that the verification task is effective in detecting a large number of invalid cases. One such case is demonstrated in Figure 5.

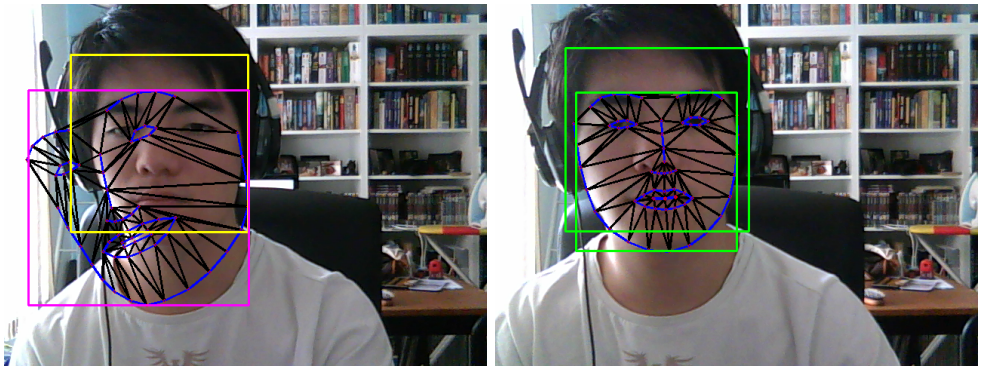

Figure 5: An invalid state is detected with Viola-Jones (left), and solved by resetting FaceTracker (right).

Finally, we chose to make the verification asynchronous, i.e. run on another computation thread, in such a way that it will be executed at a lower priority when not enough computational power is available to perform it for every frame. Viola and Jones [23] reported processing rates of 30 images per second, but this performance cannot be expected from consumer hardware when the same machine is also running a resource-heavy computer game and FlexPoser's other components. With this approach, FaceTracker processes frames at a steady, user specifiable rate, but a frame and FaceTracker's corresponding output are only verified when the verification thread is not busy verifying an earlier frame. On our machine with a CPU with four cores clocked at 2.44GHz (note: only two cores are used by FlexPoser), errors such as in Figure 5 are generally detected within 300ms.

3.5 Facial feature extraction

Facial feature extraction for motion retargeting is a problem that depends on the animation system of the virtual environment in which the facial animation is to be applied. In the case of Garry's Mod, which is built on the Source engine, facial animation is achieved using **linear vertex animation**: a set of animation channels called flexes directly control vertex displacement. Depending on the value of a flex and a weight assignment for each (flex, vertex) pair, the vertices are moved along a linear path which can also differ for each pair. The task of assigning a weight to each (flex, vertex) pair is called rigging.

Garry's Mod exposes the Source engine's facial animation system by allowing add-ons to directly set any model's flex weights, i.e. the values of the animation channels (not to be confused with flex-vertex weight assignments). Interpolation of flex weights is handled by the engine. The default character models in Garry's Mod were originally designed for Half-Life 2 and their list of

[4]https://github.com/kylemcdonald/FaceTracker
[5]http://opencv.org/

Figure 6: Example of vertex animation using the flexes involving the right eyebrow.

flexes is based on FACS [9]. Thus the problem of facial feature extraction in this case becomes a problem of extracting a set of AU values that are used by Garry's Mod player models, which is a subset of FACS AUs.

Table 1 lists all facial parameters that are tracked and applied by FlexPoser. All parameters are represented by 32-bit floating point values. While Garry's Mod player models generally have more AUs, some have been omitted, either due to difficulty tracking (for example, nose dilation) or due to the corresponding flex not being consistently rigged for all default player models.

Parameter name	FACS index
AUs	
Upper eyelid raiser (L+R)	5
Inner brow raiser (L+R)	1
Outer brow raiser (L+R)	2
Brow lowerer (L+R)	4
Lip corner puller (L+R)	12
Lip pucker (L+R)	18
Lip part (L+R)	25
Jaw drop	26
Globals	
Pitch	N/A
Yaw	N/A
Roll	N/A

Table 1: List of tracked facial parameters. L+R denotes a left and a right AU.

FaceTracker provides two representations for the deformed facial mesh that it maintains: a local space and a global space representation. The global space representation holds information about the position and rotation of the user's face. Thus the pitch, yaw and roll are directly provided by FaceTracker, where the full frontal view corresponds to (pitch, yaw, roll) = (0, 0, 0).

On the other hand, the local space representation is scale, rotation and translation invariant. This enables us to extract AU-like features by computing distances between specific pairs of vertices in the 3D local space representation. For each AU we handpicked a pair of vertices that

moved almost linearly towards and away from each other while performing that action. For example, for the AU 'left eye open' we compute the distance between a vertex at the top and bottom of the left eye. Figure 7 demonstrates the selected pairs of vertices for some AUs.

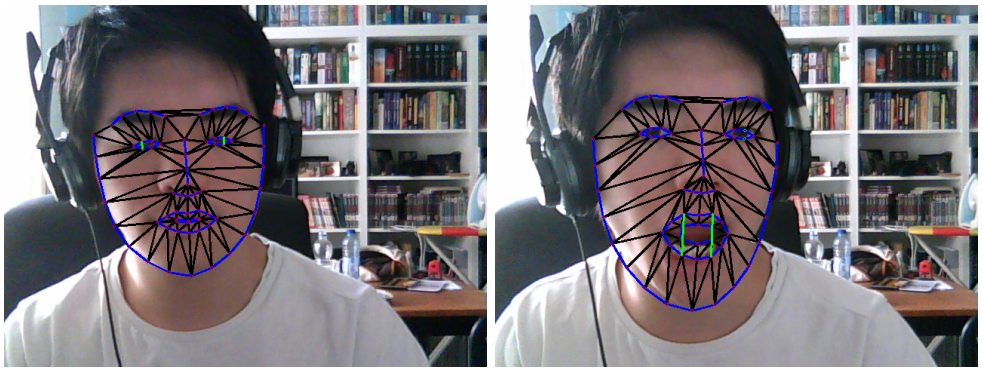

Figure 7: Selected vertex pairs (green) for the AUs 'eye open' (left) and 'mouth part' (right).

3.6 AU normalization and user calibration

The Source engine's facial animation system requires AU activation levels $0 \leq f_i \leq 1$ as input, as opposed to vertex distances. The purpose of AU normalization is to transform the vertex distances so that they represent AU activation levels. An in-game calibration process is present to obtain necessary information about the user's facial structure for this normalization step. In general, AU activation levels are also more useful than vertex distances, because they represent a user's expression independent of the user's facial structure.

The Source engine refers to AU activation levels as flex weights, in relation to its animation system. The calibration process exists to determine for each AU $i = 1..15$ the vertex distances $xmin_i$ and $xmax_i$, that respectively correspond to an inactive AU (flex weight = 0) and a fully activated AU (flex weight = 1). Once these vertex distances are known, each AU's current vertex distance x_i can be normalized to a flex weight f_i using:

$$f_i = clamp\left(\frac{x_i - xmin_i}{xmax_i - xmin_i}, 0, 1\right),$$

where $clamp(x, 0, 1)$ clamps the value x to the interval $(0, 1)$. We will first discuss the calibration process for determining $\{xmin_i, xmax_i\}_{i=1..15}$ and afterwards some of the alternative methods we considered and tested.

3.6.1 User calibration process

The in-game calibration process is a 7-step procress for determining $xmin_i$ and $xmax_i$ for the 15 tracked AUs. The user is asked to mimic poses presented by images and short descriptive text. The images are shown in Figure 8. The accompanying text is presented in Table 2, together with which AU values are recorded during which pose. During the neutral pose the user's rotation (pitch, yaw, roll) is stored. This rotation is used for reference when retargeting head rotation and corresponds in-game to a head turned straight ahead.

We experimented with several methods of recording the values and advancing to the next step. Our first attempt required the user to click a "Next" button while making each pose, but during initial testing with 15 subjects we received feedback from several subjects that it was

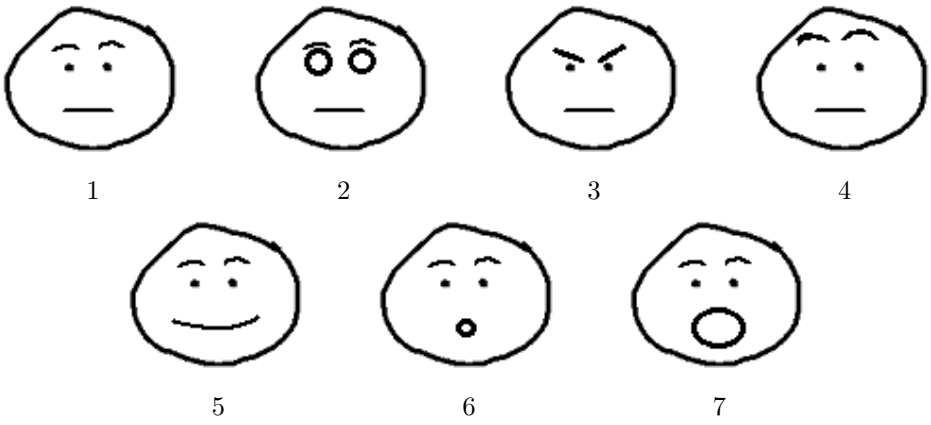

Figure 8: The seven poses the user has to mimic.

Pose #	Accompanying text	Recorded values
1	Make a neutral expression	All (min), pitch, yaw, roll
2	Open your eyes widely	Eye open (max)
3	Look angry	Brow lowerer (max)
4	Raise your eyebrows	Inner, outer brow raiser (max)
5	Smile with mouth closed	Mouth pull (max)
6	Make a small o with your mouth	Mouth pucker (max)
7	Make a large O with your mouth	Mouth part (max)

Table 2: Accompanying text per pose.

inconvenient to click the button while making a pose. This was especially the case for the poses involving the eyes and eyebrows. Based on this we experimented with several ideas to make the process more user friendly.

Our final approach requires the user to click "Next" manually only once, while making the neutral pose. At this instant all minimal values $xmin_i$ are recorded. During each of the remaining poses we look at the current offsets from the recorded minima, $x_i - xmin_i$, to determine when a user is performing a pose correctly and automatically advance when a pose is held long enough.

To estimate whether the user is currently performing the requested pose, we compare the offsets $x_i - xmin_i$ to adaptive thresholds T_i for all relevant AUs. For example, pose 2 in Figure 8 is considered correctly mimicked if the offsets $x_i - xmin_i$ for both 'eye open' AUs are greater than their corresponding adaptive thresholds T_i. The thresholds T_i are adaptive in the sense that they are lowered over time to ensure that everyone can complete the calibration process. For this, we use the following scheme:

$$T_i(t) = \begin{cases} T_i(0) & t < 2 \\ T_i(0) \cdot (1 - \frac{1}{3}(t-2)) & t \geq 2, \end{cases}$$

where t is the time in seconds passed in the current calibration step. $T_i(0)$ refers to handpicked initial values per AU, which represent offsets from minima that are generally not passed unless intentionally performing that AU. After 2 seconds the initial threshold is lowered at a steady rate of a third per second, until after 5 seconds the threshold has become trivially passable. This ensures that everyone can finish the calibration process, even when some AUs cannot properly

be performed by the subject or tracked in current image conditions (though obviously motion retargeting will then fail for those AUs).

We assume that when a user makes a pose and stops moving, then he is performing that pose to his full, comfortable extent. Then, when the user performs a pose for one second without interruption, we store for all relevant AUs the highest measured values during the entire time spent in this calibration step as $xmax_i$ and automatically advance to the next step. Note that the poses are ordered in such a way that there is no overlap between consecutive poses, so that the user is not unintentionally performing the pose already when arriving at the next step.

Finally, to give the user some visual feedback during calibration, some indicators are shown while performing the poses: a 'pose' indicator shows whether the user's current pose passes the thresholds and a 'face' indicator shows whether any face is being detected at all. Both are visible in Figure 9 (left). After all 7 poses have been mimicked a preview screen allows the user to test his current calibration, after which he can either accept the result or redo the mimicking steps.

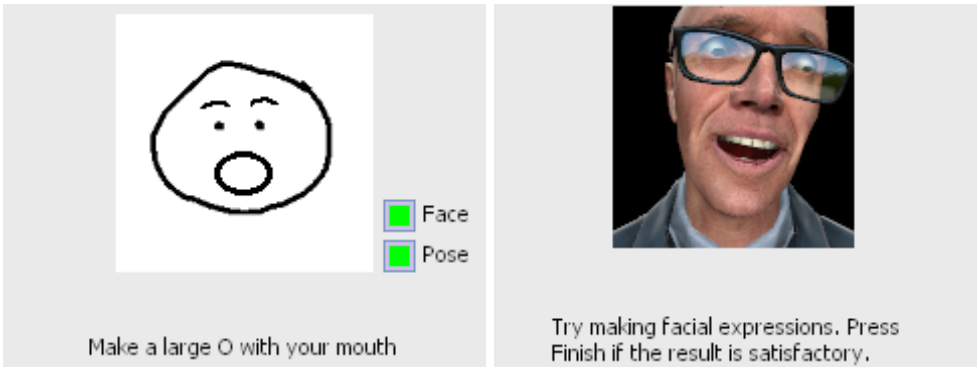

Figure 9: Pose mimicking step (left) and final result verification (right).

3.6.2 Alternatives to user calibration

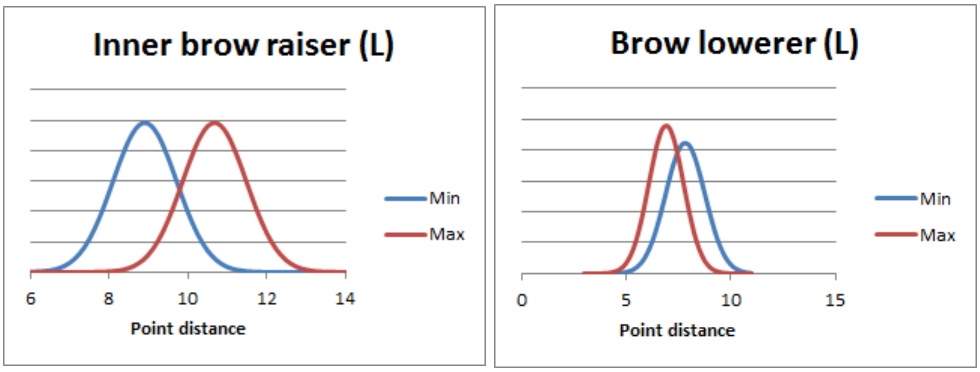

Figure 10: Normal distributions for 2 AUs constructed from 15 calibration data sets.

We considered and tested two alternative methods that could possibly remove the calibration process altogether, but failed to find a solution that would work robustly for a large variety of users and lighting conditions. Firstly, we considered the naive option of using the same set of min-max values for everyone. To study whether this would be possible we invited 15 people to

perform the calibration process and we analyzed their data. They performed the process under guidance to ensure that lighting conditions were perfect and the tracking was working correctly.

The analysis confirmed our expectation that facial structures differ too greatly from person to person to allow for static calibration data. Figure 10 shows reconstructed normal distributions of the recorded min-max values for two AUs. It becomes clear that the recorded min-range and max-range overlap too much. This implies that for any two constants $cmin_i, cmax_i$ where $cmax_i > cmin_i$, there will always be insensitivity or oversensitivity issues for a large group of users.

Another option we considered was to do online calibration by tracking and updating the minimally and maximally measured values while the user is using the motion retargeting. For each AU, the current vertex distance is then normalized using the lowest and highest measured distances for that AU until then. This introduces two problems, which we call the start-up issue and the insensitivity issue.

The **start-up** issue is deciding the initial min-max values. Using the first measured value as min and max introduces an effect that for the first few moments, any motion 'pushes' the min and max boundaries away from each other. For example, when raising an eyebrow for the first time, during the entire motion the corresponding vertex distance will be the highest measured distance until then. This will result in invalid, twitching facial animation until each AU is at least performed fully once. Other initial values bring other problems.

The **insensitivity** issue is the opposite of the start-up issue: the longer the face tracking runs, the more insensitive the motion retargeting becomes, because we only ever expand the min-max range. Noise will contribute to this, while tracking errors (the deformed mesh being completely off target) will cause this method to fail completely even if such an error appears for only one frame. While still other options exist (min-max tracking with a range shrinking mechanism), we ultimately decided on using the manual calibration process due to its simplicity and trustworthiness.

3.7 Network synchronization

Synchronization of facial animation between game clients is achieved by synchronizing the facial parameters from Table 1. When FlexPoser has processed a new frame client-side it will send the data to the game host, w will then broadcast it to all other clients (sender excluded). One frame of facial data consists of 18 32-bit floating point values, thus is 72 bytes in size. To give an example of the data traffic generated by real-time facial motion retargeting, a player tracking and sending his facial data at a rate of 15 frames per second would result in the server receiving and broadcasting 1.05 KiB/s. This amount of server traffic is negligible compared to server traffic due to real-time positional synchronization in games such as Garry's Mod.

4 Experiment 1: Effects of tracking timing and accuracy

This section describes the setup and results of the first experiment. In this experiment we study the effects of various performance parameters, such as face tracking accuracy and computation time, on perceived animation quality. An online video survey is executed in which we asked participants to rate animations, most of which are altered to simulate different types of errors in varying degrees. The goal is to construct a scheme for balancing performance parameters against each other when designing and optimizing a facial motion retargeting pipeline.

4.1 Experimental setup

When using facial motion retargeting in a multiplayer game, factors such as network performance, face tracking computation time and face tracking accuracy all affect the appearance and timing of the final animation. To study their effects on perceived animation quality, we processed four videos offline using the method described in Section 3, while simulating various types of degradation in the output animation. We then asked participants in an online survey to rate the resulting animations. Their ratings are used to derive relationships between the simulated variables and the perceived animation quality.

4.1.1 Participants

We distributed the online survey to three groups of people. We posted the survey on the official forums of the Garry's Mod community[6], frequented by players of the game. Additionally, we distributed the survey to two mailing lists: a mailing list for students and staff of the Game and Media Technology (GMT) group at Utrecht University, and a mailing list for computer vision researchers. We received 202 response sets in total. We combine all results indiscriminately, regardless of personal background.

Based on date of submission, we estimate roughly 60% of the responses to originate from the Garry's Mod forums, 20% from the GMT mailing list and 20% from the computer vision mailing list. Participants reported their age, sex, hours spent weekly on gaming related activities, and hours spent weekly using video chat such as Skype. Figure 11 displays the age distribution of participants. The average age is 26. 21% of the participants is female. Figure 12 shows a distribution of how many hours weekly participants spent on gaming related activities and Figure 13 shows their video chat activity.

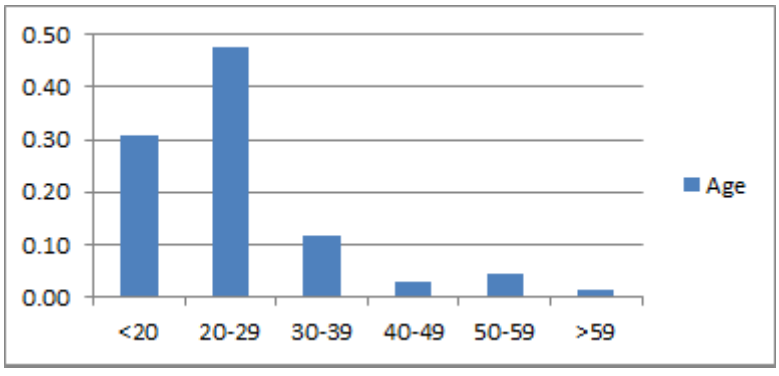

Figure 11: Distribution of participants' age

[6] http://facepunch.com/showthread.php?t=1402521

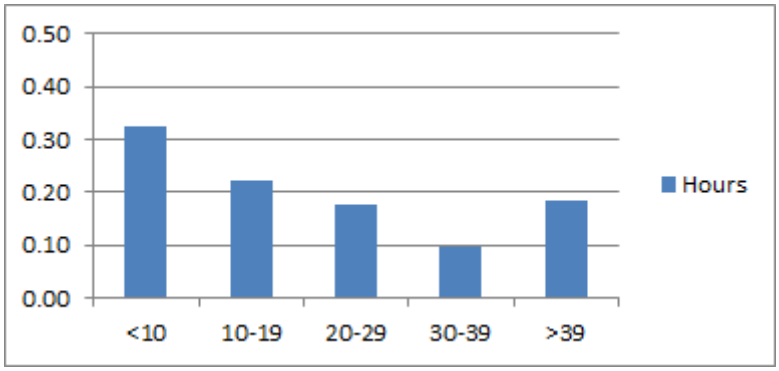

Figure 12: Distribution of participants' weekly hours spent on gaming

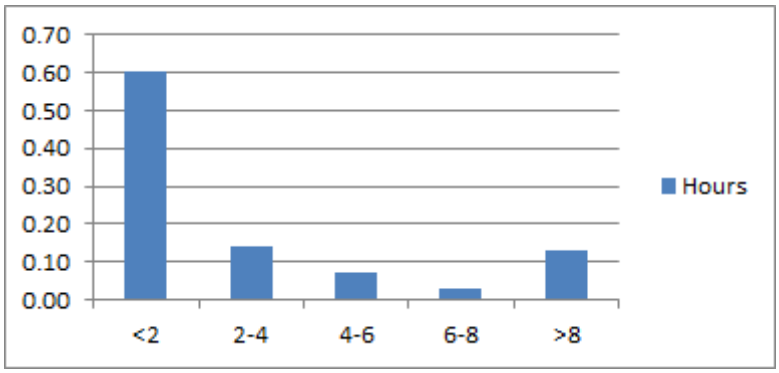

Figure 13: Distribution of participants' weekly hours spent on video chat

4.1.2 Materials

For the online survey we processed four videos depicting a speaking person into facial animation. The videos were downloaded from YouTube. We will refer to the videos as expat[7], day9[8], rhaea[9] and walken[10]. The selected segments are from movies, interviews and video blogs. Two videos depict a male speaker and the other two a female speaker. The videos were selected based on ease of tracking, for which we preferred a clearly visible frontal face at all times, good lighting and a sharp image. Manual verification was done for each video to see whether facial expressions were accurately being tracked.

The videos were processed offline (frame-by-frame) into high quality facial animation, which we will refer to as the clean animation. We used the motion retargeting pipeline from Section 3 to obtain the facial animation, the same pipeline as used in FlexPoser. However, due to lack of calibration data of the original actors, we manually tweaked calibration values per video for the AU normalization step described in Section 3.6. This is not ideal and results in some quality loss when retargeting the facial motion.

After obtaining a clean animation per video, we modified it in various ways simulating different types of degradation. Table 3 states the variables we simulated and their effect. We simulated

[7]https://www.youtube.com/watch?v=swOZn4h3DmU
[8]https://www.youtube.com/watch?v=CdLnuGAPNUg
[9]https://www.youtube.com/watch?v=r0HpbASP7AM
[10]https://www.youtube.com/watch?v=YFtHjV4c4uw

one variable at a time and used four different intensities for each variable. While the clean animation also includes head rotation tracked from the original video, with the exception of the 'freeze' variable we do not modify head rotation when simulating the variables.

Variable	Effect
Noise	Noise in all action unit channels.
Delay	Delay between animation and accompanying audio.
Frequency	Frames processed per second, as opposed to skipped.
Freeze	Animation freezing at periodic intervals.

Table 3: Simulated variables

The **noise** variable simulates inaccuracy in tracking the individual action units. It represents the case in which the face tracking algorithm produces noisy output but the facial features are generally estimated in the right place. To add noise to the clean animation we first computed the standard deviation σ_i of each AU channel in the clean animation. We then added Gaussian noise to each frame, with $\mu = 0$ and $\sigma = c\sigma_i$. Thus, for each channel in the original animation the σ of the probability density function used to generate the Gaussian noise on that channel is a scalar times the channel's measured standard deviation. The noise samples are independent between frames. We selected $c = 0.1, 0.2, 0.3$ and 0.4 for the different intensities.

The **delay** variable simulates delay between the user performing a motion and the retargeted animation appearing on screen. Each step in the motion retargeting pipeline, such as camera access and filtering during animation playback, can cause a visible effect of delay, but in general the face tracking task will contribute the most to it. We are interested in how delay affects perceived animation quality, because this allows us to balance face tracking quality versus computation time. This in turn helps us in selecting a face tracking algorithm, or tweaking a selected one. To simulate delay we simply delay playback of the animation by a few frames, which is apparent due to desynchronization with the accompanying audio. We used delays of $0.125, 0.250, 0.375$ and 0.500 seconds for the different intensities.

The **frequency** variable simulates the number of frames per second (fps) that are processed as opposed to skipped. The original videos featured 30 fps and we also processed this offline into 30 fps animation, but real-time face tracking at a rate of 30 fps while playing a resource-heavy game is currently too much to ask from general consumer hardware. Via the frequency variable we study how various degrees of frame skip affect the perceived animation quality. We simulated frame skip by only processing one out of every $2, 3, 4$ and 5 frames, resulting in a tracking rate of $15, 10, 7.5$ and 6 fps. During skipped frames, we filled in animation by linear extrapolation on each animation channel using the previous two tracked frames so that the animation appears smooth despite gaps between inputs.

The **freeze** variable simulates periodic freezing of the animation, which is meant to simulate two situations that can happen in practice: when the face tracking algorithm loses track of the face momentarily and when during online use network instability causes facial data from other clients to be delayed (a lag spike). The freeze variable is used to study how either case affects perceived animation quality. Note that the appearance of network delay differs from the simulated delay variable, as network delay in general causes both facial data and audio to be delayed equally. We simulated freezing by pausing the entire animation (including head rotation) for 0.5 seconds in periodic intervals of $4, 3, 2$ and 1 seconds.

We combined each of the four input videos with each variable and intensity, resulting in 64 modified animations. After computing the clean and modified facial animations, we played them back on characters in Garry's Mod that fit the original speaker's voice and recorded the in-game animations using screen capture software. This resulted in 68 videos: 4 showing clean animations and 64 showing modified animations. Finally, the original audio was merged with every video.

4.1.3 Design

As it is infeasible to ask a survey participant to rate 68 videos, we divided them into four sets. Each set contains all 4 clean animations and 16 modified animations in such a way that every combination of (variable, intensity) is simulated exactly once and the same input video is used exactly once per variable (rows), and once per intensity (columns). Table 4 displays an example assignment of input video to (variable, intensity) pair. In this example, input video 'expat' is selected for simulating noise with intensity 1, delay with intensity 2, and so on. The input video assignments for the other three sets were generated by rotating the input videos in Table 4. For example, a second set is obtained by substituting 'day9' for 'expat', 'rhaea' for 'day9', and so on. All clean animations are appended to all sets.

Variable	Intensity 1	Intensity 2	Intensity 3	Intensity 4
Noise	expat	day9	rhaea	walken
Delay	walken	expat	day9	rhaea
Frequency	rhaea	walken	expat	day9
Freeze	day9	rhaea	walken	expat

Table 4: Assignment of input video to (variable, intensity) pair for one of the four video sets

To prevent the participant from recognizing patterns in the ordering of videos, which can introduce response bias, we impose a fixed random ordering on the videos. The random order is shown in Table 5. We use the same random ordering in all sets, so that the (variable, intensity) pairs appear in the same order in all four versions of the survey. Our motivation for this is that, in this case, any effects due to the ordering of (variable, intensity) pairs are consistent between different versions of the survey. Finally, half of the participants are shown the video sequences in reversed order, to average the effects on rating due to a video appearing earlier or later in the presented sequence.

#	Video	#	Video	#	Video	#	Video
1	frequency, 1	6	frequency, 4	11	delay, 3	16	delay, 2
2	freeze, 1	7	noise, 4	12	frequency, 3	17	frequency, 2
3	clean	8	noise, 3	13	clean	18	delay, 4
4	noise, 2	9	freeze, 4	14	freeze, 2	19	freeze, 3
5	noise, 1	10	clean	15	delay, 1	20	clean

Table 5: Order of (variable, intensity) pairs being shown in the non-mirrored survey

We merge the results of all the versions of the survey in the following way. Ratings of videos that show a simulated error are merged based on the variable and intensity. For example, all ratings of videos showing the 'frequency' variable with intensity 1 are combined. Ratings of clean videos are combined based on the input video, i.e. they are ratings of the exact same video. We standardize each participant's ratings such that every participants complete response set

has mean $\mu = 0$ and standard deviation $\sigma = 1$. This compensates for the fact that participants make different use of the Likert scale, which can be caused by different experience in computer animation/vision, or different tendencies in general when rating videos.

Incomplete response sets are discarded. Our motivation for this is that incomplete sets can skew the overall ratings, such as when a person who fills in high ratings in general only answers the first few questions. Our preference is that in the event that a person demonstrates a strong personal bias when rating, it is either applied to all questions or none.

Besides discarding incomplete response sets, we also filter out sets which we thought were not filled in seriously, specifically participants who exclusively or almost exclusively filled in ratings of 1 on the 7-level Likert scale from 1 to 7. We used the following criterion to filter them out: response sets in which the average rating was lower than 1.5 were removed. From the 202 response sets, 154 remained after removing incomplete entries. After filtering sets with average rating lower than 1.5, 136 response sets remained.

Recall that we first generated four video sequences and then mirrored those sequences to obtain eight versions of the survey. We used a third party survey tool which included a built-in mirroring feature for parts of the survey. We made use of this feature, but unfortunately we discovered too late that the survey tool did not store per participant whether he was presented the base or the mirrored version of one of the four surveys. This makes us unable to study whether mirroring was effective in reducing order bias. The consequences are discussed in the results.

4.1.4 Procedure

The four unmirrored versions of the survey were created using the online in-browser survey tool SurveyMonkey[11]. A link to a php script was distributed to all participants. The script redirected every participant to one of the four versions of the survey, such that every version would receive an equal amount of visitors. SurveyMonkey ensured that half of the participants viewed the video sequence in reversed order.

In the survey, we first ask participants to fill in their age, gender, hours per week spent on gaming related activities, and hours per week spent using video chat. Then, we show the 20 videos depicting clean or modified facial animation, one video per page, either in the order displayed in Table 5, or its reverse. On each page we ask the participant to rate the animation, formulated as *What do you think of the quality of the facial animation?* Participants choose a value on a 7-level Likert scale with the ends annotated as *Terrible* and *Excellent*. The final page allowed participants to leave comments.

[11] http://www.surveymonkey.com/

4.2 Results

We expect the clean animations to get the highest ratings, since they mimic the original facial motion with highest precision in timing and facial expression. Furthermore, since for each variable its intensity represents the amount of change applied to a clean animation, we expect each variable's average rating to decrease, as the variable's intensity increases.

We will first show that gathered ratings illustrate a strong correlation between variable intensity and rating, for all variables. Then, we will discuss the results with respect to our expectation. Finally, we will motivate and apply linear regression to each variable, to estimate the rate at which each simulated variable decreases perceived animation quality.

4.2.1 Correlation

We analyzed the correlation between each variable's intensity and the corresponding ratings, which were standardized per person. We treated all ratings of clean videos as variable intensity = 0 for each variable, and included these in the analysis. Thus, per participant we have 4 clean ratings and 4 ratings at different intensities. Recall that we used ratings of 136 participants. This results in a sample size of $N = 136 \cdot (4 + 4) = 1088$ per variable.

The computed correlation coefficients and p-values are shown in Table 6. For all variables, we find a low negative correlation between intensity and rating. The negative correlation confirms that ratings decrease as the variable intensity increases. The correlation being low tells us that there is high variance in the ratings per intensity, despite having standardized the ratings per person.

For every simulated variable, we find that the correlation between variable intensity and ratings is significant at the 0.01 level. This indicates strongly that the observed effect, of ratings decreasing as intensity increases, is due to the relationship between intensity and perceived animation quality, and not due to sampling error.

Variable	Correlation coefficient	p-value
Noise	-.308	2.2351E-25
Delay	-.256	4.5953E-18
Freeze	-.368	2.6812E-36
Frequency	-.239	1.2026E-15

Table 6: Correlation (N = 1088) between variable intensity and standardized ratings. Ratings of clean videos were used as intensity 0 for each variable.

4.2.2 Comparison to expectation

Figure 14 shows the averages of the standardized ratings for each (variable, intensity) pair. The average rating of all clean videos is appended to each graph, as intensity = 0. For each (variable, intensity) pair, one standard deviation is plotted above and below the average. Linear trend lines are also plotted.

In general, most of the data points in Figure 14 confirm our expectation that ratings decrease as variable intensity increases. However, the average ratings of both the frequency and the freeze variable at intensity 1 conflict with our expectation. Note that these two (variable, intensity) pairs appear first in the video sequence for participants who are presented an unmirrored version of the survey, see Table 5.

We suspect that participants, who are shown these two animations first, rate the animations extra low due to not knowing yet what to expect in terms of high quality and low quality animations. Unfortunately, we cannot verify whether the other participants, who were shown the mirrored sequence, rated these animations higher, due to not possessing data on whether a participant was presented an unmirrored or a mirrored sequence. We think that this start-up issue is less of a problem for the mirrored sequences, because they start with a clean video, which quickly sets an expectation for what to expect in terms of high quality animation.

If the issue is indeed caused by video presentation order, it would occur less likely if participants were shown examples of what to expect in terms of high quality and low quality animations. However, this would introduce a different form of bias: that participants do not give their raw opinion. We preferred participants to give their raw opinion during the design of this experiment, but we did not expect such strong start-up effects as with the (freeze, 1) and (frequency, 1) ratings. In the remainder, we still assume that ratings decrease as variable intensity increases, and we regard (freeze, 1) and (frequency, 1) as exceptions caused by the video ordering. However, their ratings will be included in linear regression.

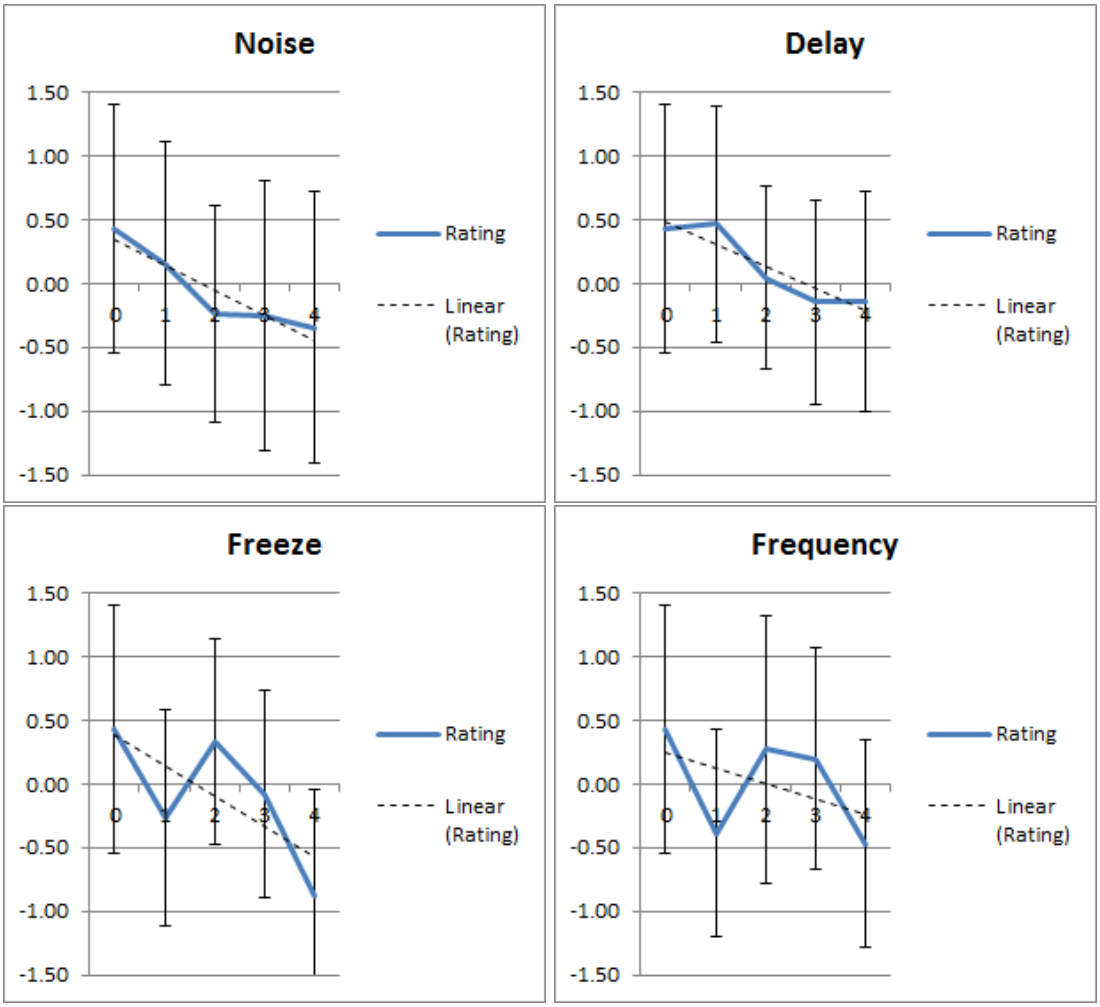

Figure 14: Average rating per (variable, intensity) pair after standardizing. Average rating of clean videos is used in each graph as intensity 0. One standard deviation is displayed above and below each average. Linear trend is shown.

4.2.3 Linear regression

We use regression to determine relationships between facial motion retargeting performance parameters and perceived animation quality. It allows us to estimate, given a measured change in one of the performance parameters, how that would affect the rating observers would give the resulting animation on average. We use linear regression, because we expect there to be an almost linear relationship between variable intensities and ratings, since for each variable there is a linear relationship between intensity level and amount of change applied to the clean animation.

Notice that the four variables we simulated are often to be balanced together. For example, a face tracking algorithm may produce more accurate results (a decrease in noise) if it is given more time (an increase in delay). The goal of linear regression is to help us decide, for example, whether x amount of noise reduction is worth y amount of extra time, by estimating for both changes their effect on perceived animation quality.

We first introduce a change in independent variables, replacing variable intensity levels with more practical measurement quantities. Table 7 shows the new independent variables and the corresponding new quantities that were previously represented by intensities 0 to 4. We will briefly explain the new variables. The variable c is the ratio between noise standard deviation and signal standard deviation $c = \sigma_N : \sigma_S$. The variable t is the delay in seconds between real-life motion and virtual animation. The variable d represents the interval in seconds at which short freezes occur. The variable f represents the number of skipped frames each time after processing a frame, assuming a base frame rate of 30.

		New intensity values				
Variable	Independent variable	0	1	2	3	4
Noise	c	0.0	0.1	0.2	0.3	0.4
Delay	t	0.000	0.125	0.250	0.375	0.500
Freeze	d	-	4	3	2	1
Frequency	f	0	1	2	3	4

Table 7: Variables introduced to replace intensity levels with practical measurement quantities.

Some may prefer to express noise using signal-to-noise ratio σ_S^2/σ_N^2. Conversion between c and signal-to-noise ratio can be done using the following relationship: $\sigma_S^2/\sigma_N^2 = c^{-2}$. Similarly, conversion between frame skip f and frame rate F can be done using $F = 30/(1 + f)$.

We perform linear regression on the new independent variables and corresponding ratings. Table 8 shows the computed linear trend lines. With exception of freeze, the clean ratings were also included in the linear regression as values corresponding previously to intensity = 0. For freeze, this was not done, because conceptually a clean animation can be viewed as having a freeze interval of ∞, but a data point with $d = \infty$ is not suited for linear regression.

Note that since we apply linear regression to standardized values, the found trends cannot be used to estimate absolute ratings. For example, an estimated rating > 0.0 does not represent that participants would give an average rating of above 4 on the 1-to-7 Likert scale. Instead, it represents that participants of this survey individually would rate the animation quality above average within their own set of responses to this survey. The results are best applied to estimate effects of changes in variables, while the values of the variables remain between the lowest and highest values used in Table 7.

Variable	Linear trend (change of variable)
Noise	$.401 - 2.154c$
Delay	$.451 - 1.314t$
Freeze	$-.780 + .223d$
Frequency	$.369 - .161f$

Table 8: Linear trend lines for each variable.

4.2.4 Example: Optimization using linear trends

Finally, we will demonstrate how the linear trends in Table 8 can be used to optimize an already functional facial motion retargeting pipeline, using predictions of effect on perceived animation quality as measure. It is important to note that for simplicity we assume that all independent effects of noise, delay, freeze and frequency can be summed and still provide an accurate estimate of perceived animation quality.

To start, we execute the method on a pre-recorded video, without skipping frames, to produce facial animation signals ϕ for all AUs. During computation, we measure the average time in seconds t it takes to process a frame. Notice that t corresponds to one of the variables in Table 8: it estimates the delay between real-life motion and virtual animation.

Furthermore, we run a non-interactive face tracking algorithm on the video to determine the "ground truth" signals ϕ_S for all AUs. Then, for each AU we extract the noise signal ϕ_N, assuming $\phi = \phi_S + \phi_N$. We then compute the standard deviations σ of ϕ_S and ϕ_N to obtain a value for our independent variable for noise: $c = \frac{\sigma_N}{\sigma_S}$. Since there is a different noise factor c for every AU, we must decide a method for selecting a single, representative noise factor, for example the average c over all channels.

Now suppose we find $c_1 = 0.150$ and $t_1 = 0.100$ using one configuration of our real-time method. This can be interpreted as: our method takes 100ms on average to process a frame and produces signals with noise with standard deviation equal to 15% the standard deviation of the true signal. After tweaking the method a bit, we obtain slightly different performance which reduces the noise but introduces more delay: $c_2 = 0.085$ and $t_2 = 0.180$. We now want to know whether the change is worth it: does it result in perceptually better looking animation?

For both cases, we estimate the total effect R on perceived animation quality due to both noise and delay, using the linear trends in Table 8. We find that:

$$R_1 = -2.154c_1 - 1.314t_1 = -0.455$$
$$R_2 = -2.154c_2 - 1.314t_2 = -0.420$$

We find that since $R_2 > R_1$, the second case will likely produce better looking animation.

5 Experiment 2: Evaluation of utility through local play tests

The second experiment is a user test in which we test facial motion retargeting in a controlled environment. We invited players to use the add-on discussed in Section 3 in a series of LAN matches. We record their experiences and use them to gain insights on how facial motion retargeting can enhance multiplayer experiences.

5.1 Experimental setup

We prepared machines and webcams for every participant. In the series of matches we alternated between sessions in which either everyone used facial motion retargeting or no one did, and we tried different types of gameplay. The players' experiences were recorded using questionnaires and an open discussion after the sessions.

5.1.1 Participants

Participants in the online survey from the first experiment were given the option to leave their contact information if they were interested in participating in future experiments. For this experiment, we invited eight people to take part in the LAN matches. All eight participants were present at the experiment day, but due to in-game graphical issues on one machine we tested with seven participants. The players were all male, aged 21 to 28. The participants were all bachelor and master students of game technology, and were avid gamers and game programmers with an average of 28 hours weekly spent on gaming related activities. Two players had prior experience with the game.

5.1.2 Materials

We prepared a room at Utrecht University and with eight desktop PCs, each with a Logitech C210 webcam connected. Garry's Mod and FlexPoser client files were installed on all PCs. The PCs were connected to a switch, so that multiplayer games could be played in a LAN setup with minimum latency.

5.1.3 Design

We installed the custom Garry's Mod game modes Trouble in Terrorist Town[12] (TTT) and Murder[13] on all machines. We selected these two game modes because they assign players specific roles, which we presume may make it easier for players to 'act' out a character.

In TTT, players are assigned a role: either 'traitor' or 'innocent'. Traitors must conspire together to eliminate the innocents. The innocents are in the majority, but do not know with certainty which players are the traitors. A lot of fun from this game mode comes from attempting to mislead other players as a traitor, or defending your innocence to other (presumably) innocents. Note that, while this game mode enforces player models with ski-masks that have no facial rigging, we altered the game mode to use unmasked, facially rigged player models instead.

Murder is a game mode with a similar premise, in that there is a murderer among the players and people do not know who it is. A major difference is that in Murder, the murderer is by himself and has to stealthily kill all other players without being found out. In this game mode

[12] http://ttt.badking.net/
[13] http://gmod.wikia.com/wiki/Murder

the other players, playing the role of civilians, are more likely to work together because there is a smaller chance that the person they are talking to is actually the murderer.

We perform sessions with both game modes, because their different premises may affect how the players communicate with each other, and thus affect how facial motion retargeting is used. Players are asked to fill in a questionnaire after every session. The contents of the questionnaire is shown in Appendix A. Answers are given in the Likert scale of 1 to 7. We average the questionnaire answers and study the averages to see how facial motion retargeting affected the players' communication.

5.1.4 Procedure

We start the experiment by giving a brief demonstration of the FlexPoser add-on for Garry's Mod. Players are also explained the rules of the two game modes, Murder and TTT, and shown a demonstration of both game modes. After the demonstrations, players are asked to take a seat and perform the user-specific calibration required for the add-on. After they are done calibration, they are given a few moments to try out the facial motion retargeting. They are then asked to join the game server and prepare to play. We host four sessions in total. The first two sessions feature the Murder game mode and the last two feature the TTT game mode. Each session lasts 12 minutes and is followed by a moment of break.

In the Murder sessions, the first session features people playing normally without facial motion retargeting, thus players can only communicate through typing and talking (in this case, shouting across the room). During the second session, facial motion retargeting is enabled for every player, thus everyone's facial expression is being tracked and displayed in real-time on their game character. In the TTT sessions, the first session features tracking enabled, and the second session has it disabled.

After each two sessions, players are asked to fill in the questionnaire for one game mode. At the end of the experiment, an open discussion is held to talk about what they thought of the experiment, the game modes and the facial motion retargeting feature.

5.2 Results

Table 9 shows the averaged questionnaire answers. Comparing the sessions with and without face tracking within each game mode, we see that players' outward verbal and non-verbal communication does not change much, although adding facial expressions seemingly reduced the use of other forms of communication a bit for both game modes. For both game modes, we see that enabling facial motion retargeting has almost no influence on how much attention players pay to each other's verbal and non-verbal actions.

We see that for Murder, players spent reasonable time making facial expressions at each other. For TTT, this was less the case. In both game modes, participants apparently did not notice each other's facial expressions much, nor did they feel their facial expressions were being noticed. We conclude that for this test group, enabling facial motion retargeting did not effectively encourage players to interact more with each other.

An open discussion was held to obtain feedback and useful insights for applications of facial motion retargeting. Opinions about the tracking accuracy varied initially, but after exploring the feature for a few minutes players felt that the tracking was adequate once they grasped the limitations. All players had a tendency to immediately explore the limitations of the face

	Murder		TTT	
Tracking status:	Disabled	Enabled	Disabled	Enabled
Your actions - How often did you...				
– speak verbally?	4.0	4.0	5.4	4.7
– make non-verbal gestures to others?	3.6	2.7	3.4	3.4
– make facial expressions to others?	N/A	3.4	N/A	2.3
– feel your facial expressions were noticed?	N/A	1.6	N/A	1.9
Your perception - How often did you...				
– pay attention to others speaking verbally?	5.1	4.7	6.0	5.7
– notice others making non-verbal gestures?	2.7	2.9	3.0	2.6
– notice others making facial expressions?	N/A	1.4	N/A	1.0

Table 9: Averaged questionnaire answers organized by game mode and whether facial motion retargeting was enabled. Questions are displayed in shorter notation. See Appendix A for full questionnaire.

tracking algorithm, and did not settle down to use the feature 'properly' until they had a feel for the limitations. With the exception of one participant, who had tracking issues caused by a combination of light eyebrows and glasses, everyone could finish the user calibration process without much trouble.

Some participants mentioned that they felt that both game modes were unsuited for facial motion retargeting. They reason that because players are unsure of each other's role in the game, they prefer to stay at range from each other, which drastically reduces the usefulness of facial motion retargeting. We think that this may be player preference, and that other players would be more willing to put their score at risk in return for fun using facial expressions. Participants also mentioned that the feature did seem fun, but suited more for non-violent games or cooperative (players versus AI) games such as MMORPGs.

Some suggestions were made to improve the impact of the feature. Firstly, someone suggested a panel as part of the game's user interface, which could show the in-game faces of nearby players that are talking. Secondly, game modes specifically designed for this feature would make a difference. An idea that was thrown around was a "big heads" feature or game mode, that increases the size of players' heads, for example based on score (this feature exists in the Unreal Tournament games).

Finally, a popular idea among the participants was to combine facial motion retargeting with the "kill cam" feature that is present in many modern shooters. The kill cam refers to a recently defeated player being shown a live visual of his attacker, allowing the attacker to taunt his victim. A game that features both kill cams and facial motion retargeting would enable players to taunt their victims using facial expressions.

Although results of this user test seem discouraging about the utility of facial motion retargeting in Garry's Mod, we think that utilization of the feature may differ greatly depending on player group and game mode. In the next experiment, we release the add-on publicly to gather usage statistics from practical multiplayer sessions that can feature any game mode.

6 Experiment 3: Evaluation of public online use

In the third experiment we attempt to identify factors that influence how players utilize facial motion retargeting. We examine usage data after releasing FlexPoser, the add-on for Garry's Mod presented in Section 3, online via Steam Workshop.[14] The add-on was released on July 7th, 2014. We gathered data about how the add-on was used during a period of 55 days.

Upon joining a game server with the add-on installed, players were made aware of the add-on via a chat message. Each player could decide individually whether to enable or disable facial motion retargeting. It was disabled by default.

6.1 Experimental setup

We study use of the add-on in Garry's Mod through data gathered from two sources: the first being anonymized statistics integrated in the add-on and the second being a questionnaire that can be opened in-game at any time while playing Garry's Mod. We will describe the setup of both.

6.1.1 Integrated statistics

The integrated statistics gathered information about **servers**, **sessions** and **players**. Their relationship is as follows. A session represents a single online match of Garry's Mod, which starts as soon as the game host has finished loading a map and ends when a map change occurs or the server closes. A server is a physical machine that hosts sessions. Players are the participants in sessions.

All servers that had the add-on installed automatically submitted all players' usage data, unless the server admin had disabled statistics. Several measures have been taken to respect the privacy of both server owners and players. Firstly, the add-on page on Steam Workshop was transparent about usage statistics being tracked, detailing also what information was being sent. Secondly, all data that could be used to identify a server or player, such as server IPs and player IDs, was salted with an arbitrary string and then encrypted before ever being sent from game server to our data listening server. Thirdly, the add-on's download page listed an in-game option for server admins to turn off statistics if desired.

For a session, we firstly store the encrypted IP address of the server that hosted it, and timestamps of the session's start and end. This allows us to link multiple sessions to the same anonymous server. Secondly, we store the game mode and map name. From this information we can derive in broad terms the type of gameplay featured in that session. For example, some Garry's Mod game modes are known to feature action-packed gameplay, while other game modes function more as hide and seek, or social hangouts. Thirdly, we store some information that enable us to quickly filter sessions based on facial motion retargeting activity. We store:

- the peak number of concurrent players
- the peak number of concurrent players with facial motion retargeting enabled
- the peak ratio between facial motion retargeting enabled and disabled players

For players, we compute summaries of their facial activity over two overlapping intervals: for an entire session and for 3-minute intervals within a session. Each summary contains two parts. The first part contains the average value and variance for each of the 15 AUs, as well as for

[14] http://steamcommunity.com/sharedfiles/filedetails/?id=282498239

the 6 global parameters representing position and rotation. We use the information per AU to represent how specific parts of the face, such as eyes and mouth, were used. The second part is a simple metric: during tracking frames are flagged as "expressive" when one or more of the AUs listed in Table 1 are activated at least 75%. The value 75% was chosen for being unlikely to be exceeded when making a neutral expression. The number of expressive frames is divided by the number of total frames, resulting in a percentage representing average expressiveness over time. We use the metric to represent during how large a part of the session the player was making an expression.

Since participation with facial motion retargeting is optional per player, we assume that a player only enables it on moments when he wishes to share his facial expressions. We then assume that, in general, the less the player is making a neutral expression, the more useful facial motion retargeting is in bringing enjoyment and communicative value to the players. We consider subtle and exaggerated expressions as equally valuable. In the discussion of the results we use session, server and player data to examine how different game types affect players' face tracking activity.

6.1.2 Questionnaire

We included a questionnaire with FlexPoser, that could be opened in-game from the add-on's settings menu. Players could open and fill in the questionnaire at any time during play. They could resubmit as often as they wanted; only the last submission per player was stored in our database.

The questionnaire was designed with two goals in mind. Firstly, we want to learn about the player's intent when making facial expressions. Secondly, we want to know the effects of using facial motion retargeting on the player's experience. Appendix B shows the contents of the questionnaire. Players could select multiple checkboxes for questions 5 and 8. Questions 6 to 10 only appeared if the player selected in question 4 that he had used FlexPoser before.

6.2 Results

During the statistics gathering period, we gathered data from 13,927 unique servers. Note that, since Garry's Mod's appeal heavily relies on custom user content, many server admins installed the add-on as a service to their players without knowing whether players would use it or not. As a result, many servers and sessions were logged, in which no player actually made use of the add-on. Still, we gathered data from 1,987 online sessions where facial motion retargeting was actively used. We will first present an overview of the quantity of gathered data, before studying AU activity of players.

6.2.1 Overall statistics

Table 10 lists quantities of the gathered data. For servers, we firstly see that although many servers had the add-on installed, on only 5% of the servers did players actually use it during one of the logged sessions for that server. The table also presents counts for servers for which we have logged larger numbers of sessions, indicating that these are persistently running servers. For these counts we also included sessions in which no facial motion retargeting was used. For sessions, we see that even when facial motion retargeting is used, the number of tracked players does not often pass two.

Figure 15 shows the number of sessions logged per day, in which at least one player had face tracking enabled. We see that, after an initial spike of popularity, its daily use drops to around 30 sessions per day spread across multiple servers.

	Count
Servers	13927
– with tracking activity	721 (5%)
– with 20-49 sessions	697
– with 50-99 sessions	149
– with 100-199 sessions	57
– with 200 or more sessions	13
Sessions	92637
– with tracking activity	1987 (2%)
— 1 tracked player	1852
— 2 tracked players	127
— 3 tracked players	7
— 4 or more tracked players	1
Players	36378
– with tracking activity	827 (2%)

Table 10: Overview of database entries, where tracking activity denotes use of facial motion retargeting.

Table 11 lists the session and distinct player count per game mode, as well as the number of sessions that featured face tracking and the total number of players that used face tracking in that game mode. We only considered game modes for which we logged at least 10 sessions with face tracking activity. We see that 'Sandbox' is the most logged game mode by far. This can be explained by the fact that Sandbox is the default installed game mode, and is generally used by players who wish to try out a new add-on. The occurrence of the other game modes roughly corresponds with their current popularity. We will give a brief description of each game mode:

- **Sandbox**: Players can manipulate a virtual world by spawning and positioning static and dynamic entities.
- **Prop Hunt**: Players from one team can possess static objects, and must avoid being detected by the other team.
- **Murder**: One player is secretly a murderer, and must kill all other players while being sneaky about it.
- **Trouble in Terrorist Town (TTT)**: Players battle each other as innocents and traitors, but innocents initially don't know who is a traitor and who is not.
- **DarkRP**: Players role-play as inhabitants of an urban district, taking on the roles of shopkeepers, police officers and gangsters.
- **Cinema**: Virtual hangout in which players can watch online videos together.
- **Horror Story**: Features creepy locations for players to explore together.

In general, we see that for every game mode in Table 11, in 3-4% of the logged sessions there was face tracking activity. Prop Hunt is a notable exception to this, with only 0.4% of logged sessions containing face tracking activity. We presume this to be caused by the premise of the game mode: half of the players control environment objects, and thus do not have a humanoid avatar whose face can be manipulated. On the other hand, DarkRP has the largest percentage of logged sessions with tracking data with 8.4%.

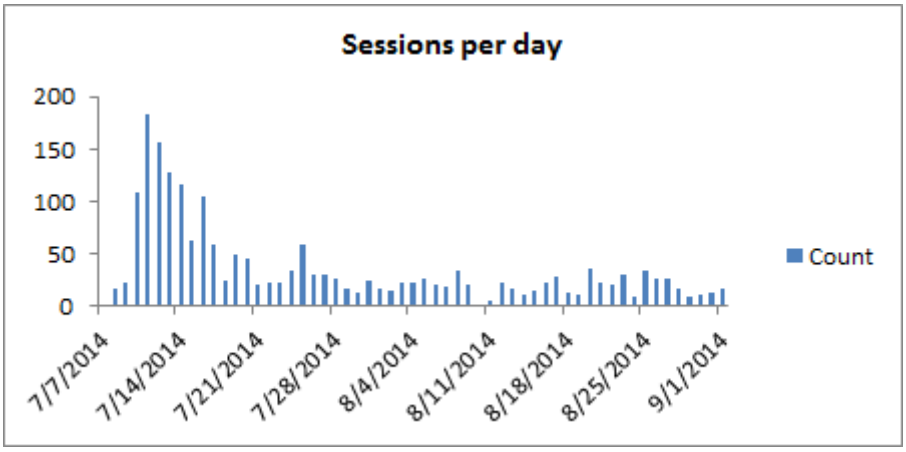

Figure 15: Sessions with tracking activity logged per day.

Game mode	Session count	with tracking activity	Player count	with tracking
Sandbox (default)	72557	2179 (3.0%)	22734	850 (3.7%)
Prop Hunt	3275	12 (0.4%)	1582	9 (0.6%)
Murder	2934	80 (2.7%)	11472	20 (0.2%)
TTT	2898	106 (3.6%)	2095	21 (1.0%)
DarkRP	2101	177 (8.4%)	13126	55 (0.4%)
Cinema	854	33 (3.9%)	2006	21 (1.0%)
Horror Story	323	14 (4.3%)	214	4 (1.9%)

Table 11: Session and player count per game mode.

In the remaining discussion of the results, in which we study AU activity, we will only consider game modes for which we have logged face tracking activity for at least 20 distinct players. We also choose to ignore the Sandbox game mode, because logged sessions with this mode are likely to feature players trying out the add-on for the first time; we prefer data gathered during actual multiplayer gameplay. Thus, in the remaining discussion we will only consider the game modes Murder, Trouble in Terrorist Town, DarkRP and Cinema. We study both session summaries and periodic summaries. For session summaries, we only consider summaries in which the player has activated face tracking for at least 60 seconds.

6.2.2 Overall statistics per game mode

	Session summaries			Periodic summaries		
Game mode	Summaries	Players	Avg. use	Summaries	Players	Avg. use
Cinema	33	18	51 min.	298	12	75 min.
DarkRP	61	13	60 min.	1760	48	110 min.
Murder	68	17	43 min.	280	15	56 min.
TTT	94	18	51 min.	257	13	60 min.

Table 12: Summary count, player count and average use duration, per summary type and game mode.

Table 12 shows the number of collected summaries of both types, and the number of distinct players they were collected from, per game mode. The average use duration per player is also

shown for both summary types. For session summaries we summed each individual player's tracking duration per session, in order to compute the average use duration, whereas for periodic summaries we use the knowledge that they are sent at 3 minute intervals (interruptions allowed within a session).

Notice that the number of distinct players differs greatly between the summary types, despite involving the same player base. This is due to the way the integrated statistics were implemented: session summaries are only sent when a map changes or a server closes properly, and periodic summaries are only sent after at least 3 minutes of facial expression activity. Thus, players that use the add-on for less than 3 minutes do not generate periodic summaries, whereas players in unfinished and crashed sessions do not have a session summary for that session. (Note: Table 11 displays correct statistics even for crashed sessions.)

Since we create a session entry at the start of a session, and modify the entry when the session ends properly, we can recognize a crashed session if the same server starts a new session while the previous one's database entry is incomplete. Table 13 shows the average number of unclosed sessions per server, where a higher number of unclosed sessions implies a higher crash rate. We see that DarkRP servers have the highest crash rate, which explains why the number of distinct players in session summaries is so low compared to the number of distinct players in periodic summaries.

Game mode	Servers	Avg. unclosed sessions
Cinema	68	2.53
DarkRP	169	11.46
Murder	89	4.25
TTT	113	2.35

Table 13: Average number of unclosed sessions per server. Any number > 1 implies crashed sessions.

Looking at the average use duration computed for both summary types, we see that DarkRP players used facial motion retargeting the most on average. This is especially interesting since looking at the periodic summaries, 48 distinct players contributed to the measured average use time per player of 110 minutes, lowering the possibility that a small number of die-hard players skew this result. Players of the game mode Cinema come second with an average use time of 75 minutes.

Next, we compare the average session duration of sessions with and without active use of face tracking. Table 14 shows these per game mode along with the sample size. Most notably, we see that DarkRP and Murder sessions that feature active use of face tracking last considerably longer than sessions without. For Cinema and TTT, the average session duration hardly changes. In the case of TTT, this may be explained by the fact that communicative behavior during TTT does not change much with or without facial motion retargeting, as players indicated during the offline experiment. In the case of Cinema, we believe that the average session duration does not change much, because the original average duration of 97 minutes is already quite long. Although the cause of differences in average session times cannot be extracted from the data, we interpret the increased session times of DarkRP and Murder as supporting that facial motion retargeting is best compatible with long, uninterrupted sessions.

Game mode	Avg. duration	N	with tracking	N
Cinema	97 min.	854	108 min.	33
DarkRP	108 min.	2101	531 min.	177
Murder	59 min.	2934	99 min.	80
TTT	26 min.	2898	33 min.	106

Table 14: Average session duration with and without tracking. Only uncrashed sessions are included.

6.2.3 Facial expression activity per game mode

We will now look at the contents of both summary types. Recall that an instance of either summary type contains a value for the expressiveness metric, and a mean and variance for every AU. Each summary belongs to a player. We will process the sets of summaries in the following way. In both cases, we will first average all summaries that belong to the same game mode and player. Then, per game mode we will average the results of all players, resulting in a single summary per game mode.

Using this method, every player contributes equally to the final summary, no matter his total number of session or periodic summaries. However, there is a subtle difference between how both final summaries should be interpreted. In the first case, we give each session equal weight within a player's summary, implicitly giving equal weight to any external factors within that session that may influence a player's behavior. In the second case, we give each fixed interval equal weight within a player's summary, thus ignoring external influences.

Rather than discussing each AU individually, we have categorized them into groups that correspond to parts of the face. Table 15 shows the categorization using FACS names and indices.

Group	FACS index
Eyelids	
–Upper lid raiser (L+R)	5
Eyebrows	
–Inner brow raiser (L+R)	1
–Outer brow raiser (L+R)	2
–Brow lowerer (L+R)	4
Mouth	
–Lip corner puller (L+R)	12
–Lip pucker (L+R)	18
–Lip part (L+R)	25
–Jaw drop	26

Table 15: Categorization of AUs into groups. L+R denotes a left and a right AU.

We will now look at the final summaries per game mode, using the AU groups in Table 15. Tables 16 and 17 show the averaged data of all session summaries and periodic summaries, respectively, per game mode. For quick reference, the number of distinct players and summaries are included in both tables in the columns P and S, respectively.

				Eyelids		Eyebrows		Mouth	
Game mode	P	S	Expr. metric	Mean	Var	Mean	Var	Mean	Var
Cinema	18	33	0.78	0.24	0.11	0.31	0.10	0.27	0.10
DarkRP	13	61	0.87	0.20	0.10	0.28	0.12	0.30	0.10
Murder	17	68	0.91	0.22	0.08	0.36	0.10	0.35	0.10
TTT	18	94	0.83	0.27	0.08	0.33	0.08	0.31	0.10

Table 16: Averaged data of all session summaries per game mode.

				Eyelids		Eyebrows		Mouth	
Game mode	P	S	Expr. metric	Mean	Var	Mean	Var	Mean	Var
Cinema	12	298	0.88	0.27	0.11	0.30	0.10	0.26	0.11
DarkRP	48	1760	0.74	0.25	0.08	0.29	0.09	0.32	0.09
Murder	15	280	0.83	0.21	0.07	0.34	0.10	0.35	0.10
TTT	13	257	0.76	0.26	0.09	0.33	0.08	0.33	0.09

Table 17: Averaged data of all periodic summaries per game mode.

Looking at the expressiveness metric, which indicates the percentage of frames in which players did **not** make a neutral expression, we see that the metric is quite high for all game modes. This indicates that, no matter the game mode, people who enable facial motion retargeting rarely keep a neutral face. There are no differences in the metric value between game modes that are consistent between both summary types.

Looking at the means of the eyebrows and mouth in both tables, we see that Murder and TTT feature the highest in both AU groups. This indicates that players of these two game modes make more exaggerated motions than players of other game modes. We can think of two reasons for this, both related to the competitive nature of the game modes. Firstly, players are often at a distance from each other, thus exaggerated facial expressions are necessary for the expressions to be noticed at all. Secondly, the violent action in both game modes may stimulate players to react more strongly to each other, for example by shouting or making taunting expressions.

For the eyelids, TTT and Cinema players have the highest means. In both cases, we think this is related to attention. TTT is a fast paced game mode, thus requires players to react fast and to be at attention at all times. Murder and DarkRP are less fast paced game modes, which would explain their lower means for eyelid activation. In the case of Cinema, the purpose of the game mode being that players watch videos explains why they keep their eyes more open in general.

An interesting result is that the variance of the eyelids is highest in Cinema, indicating that players have a higher blink rate in this game mode. This cannot be attributed to the players watching videos: studies [16, 13] have shown that humans tend to blink less during activities that require visual attention. We think the blink rate can be explained by the type of communication between players. Blink rate has been shown to be inversely proportional to cognitive load [14, 1, 24]. As such, we expect players who are acting out a role to blink less than players who are being themselves. Thus, we believe that the higher blink rate in Cinema indicates that players in Cinema are more often being themselves, whereas players in other game modes are more often acting out their gameplay role.

Finally, DarkRP players do not show extreme behavior in any AU group, though perhaps it is to be expected that the results of this game mode fall between the results of Cinema (subtle expressions), and TTT and Murder (exaggerated expressions): DarkRP is a non-violent game mode, thus players can safely move within range of each other to communicate using subtle expressions. On the other hand, players are moving around a lot, in contrast to Cinema, thus exaggerated motions are useful to get the attention of players at a distance.

6.2.4 Questionnaire results

We will now look at the questionnaires, which provide insight on the intent of players to make facial expressions. We received 97 submissions for the in-game questionnaire. The questionnaire was filled in by players aged 10 to 31. The average age was 17. 93% of submissions were sent in by male players. We will first discuss the questions that required a Yes/No answer, before discussing the open questions.

Purpose and experience questions
Table 18 shows results to Yes/No questions. Firstly, we see that only 65% of submissions was by players who had actually used FlexPoser. Others may have lacked a compatible webcam or experienced other technical issues. Players who had not used FlexPoser were not presented the remaining Yes/No questions, although they were presented the open question about what games/genres they would like to see facial motion retargeting in, and were allowed to leave comments and suggestions.

Of the people that indicated they had used FlexPoser to track their facial expressions, we see that 63% of them used the feature for natural communication, and also 63% of them used the feature for non-serious role-play. We see clearly that these two options are more popular than serious role-play, although the lack of interest in serious role-play is likely a characteristic of Garry's Mod's player base: serious role-play is more common in games that have detailed background lore such as most fantasy games. Garry's Mod does not have background lore. It is good to see that many players who used FlexPoser noted that they made more use of voice chat and made more facial expressions. This indicates that the feature motivates communication between players.

	Count
Submissions	97
– used FlexPoser	62 (65%)
Purpose	
– screenshots	22 (35%)
– machinima	10 (16%)
– natural communication	39 (63%)
– serious role-play	16 (26%)
– non-serious role-play	39 (63%)
Experience	
– more use of text chat	12 (19%)
– more use of voice chat	34 (55%)
– more facial expressions	45 (73%)

Table 18: Questionnaire results of yes/no questions.

To study the relationships between the three communicative purposes, we also counted how many people checked combinations of the three. The results are listed in Table 19. Apparently, no player used facial motion retargeting solely for serious role-play, or serious role-play in combination with natural communication. 13 of the 16 people that did use the add-on for serious role-play, also used it for non-serious role-play and natural communication. We conclude that natural communication and non-serious role-play, whether individually or combined, have been the main use of the add-on in Garry's Mod.

C	S	N	Count	
X			14	(22%)
	X		0	(0%)
		X	11	(17%)
X	X		0	(0%)
X		X	12	(18%)
	X	X	3	(5%))
X	X	X	13	(20%)

Table 19: Number of people, out of the ones that used FlexPoser, that checked combinations of natural communication (C), serious role-play (S) and non-serious role-play (N).

Continuing, we will discuss the open questions. Despite some of the open questions being targeted specfically towards FlexPoser's implementation and user interface, we will discuss all questions, seeing as the answers may provide useful general insights. Roughly half of the participants filled in answers to the open questions, though the exact number varies per question.

What do you think of the tracking and animation quality of FlexPoser?
Participants generally thought the add-on worked well, with roughly 30% giving answers like "great' and "amazing", and the remaining answers saying "not bad", "reasonable but glitchy", and "it's a good start". We received one unspecific negative comment.

The remark that the tracking felt glitchy occurred often, with some people saying "sometimes it works well but other times it doesn't" and "it depends on the player model". From this we gather that some of the issues users were experiencing stem from lighting conditions, but in general the face tracking requires improvement or better indication of its limits. The notion that the animation quality varies per model is to be expected as, with exception of models that were taken from the game Half-Life 2, Garry's Mod player models do not have facial rigging of consistent quality.

Another recurring comment is "it needs more poses", but since FlexPoser isn't pose recognition based, this can be interpreted as "more action units should be tracked". This is certainly a valid point as using only 15 AUs puts limits on what expressions can be captured, but we must also take into consideration that Garry's Mod's player models were not rigged for complex, realistic facial animations. As a result, some motions that are accurately tracked with FlexPoser cannot be represented in-game on default models, such as the mouth opening widely. We think that rigging player models with facial motion retargeting in mind, and thus using animation controls that correspond directly to the tracked AUs, should be a priority over adding more AUs to track.

What do you think of the ease-of-use of FlexPoser?
While this question was added mainly with in-game setup in mind, some participants also gave comments about the installation of the add-on. We will only discuss the comments about the

in-game profile setup procedure and user interface. In general people thought the profile setup to be very easy, with positive comments about the automatically advancing pose mimicking steps and people liking the option to enable a preview of the player model's face. The only suggestion that was made, is that some players would have liked to see the actual footage of the webcam with tracking information overlaid. This feature was originally planned, but not implemented due to difficulty transferring images between FlexPoser's C++ library and Garry's Mod's LUA API at a real-time rate.

What games/genres would you like to see a feature like FlexPoser in?
Answers to this question varied widely, with some being as unspecific as saying "all games ever". We compiled a list of all mentioned games and genres, presented in Table 20. For genres, we see that role-playing games (RPGs) and massively multiplayer online role-playing games (MMORPGs) are mentioned the most. RPGs are a broad term that can refer to both singleplayer and multiplayer games. Since FlexPoser is a social feature, we assume that players who mentioned RPGs actually refer to multiplayer RPGs or MMORPGs. Either way, it is clear that online RPGs gathered the most of mentions, followed by first person shooters (FPS).

	Count
Genres	
RPG	19
MMORPGs	8
All games	8
FPS	6
Co-op	6
Horror	4
Sandbox	3
Games	
Garry's Mod (Sandbox)	7
Team Fortress 2 (FPS)	7
Source Filmmaker (Machinima)	4
Call of Duty (FPS)	4
Battlefield (FPS)	4
Minecraft (Sandbox)	3
Grand Theft Auto (Open World)	3
Arma (FPS)	3
Star Citizen (MMORPG)	1

Table 20: Number of people that checked combinations of natural communication, serious role-play and non-serious role-play.

When viewing the list of mentioned games we must keep in mind that, since we interview players of Garry's Mod, many of them also like similar games. For example, Team Fortress 2 and Source Film Maker are related to Garry's Mod in that they all use the Source Engine. Furthermore, most of the mentioned games feature first-person camera views just like Garry's Mod.

Interestingly, although RPGs and MMORPGs were mentioned a lot as genres, with the exception of Star Citizen, none of the mentioned games are typical RPGs although Minecraft and GTA feature role-play elements. Most of the mentioned games are FPSs, but we have noticed during the offline experiment how competitive games that feature violence do not lend itself

for face-to-face communication during gameplay. Thus, we think that many players suggested FPSs with the currently popular "kill cam" feature in mind, which we also discussed in Section 5.2.

Finally, it is interesting to see Source Filmmaker being mentioned, since it is not a game but rather an animation tool that uses game assets. We learn from this that there are people interested in using real-time facial motion retargeting for creating machinima: animation videos using game engines.

Do you have any comments or suggestions?
Only two recurring suggestions were given. The suggestions match those given in previous questions: that it would be better if the player could see the actual webcam footage with tracking data overlaid, and that more poses would be nice. One unique suggestion stands out: one user remarks that tracked eye movement would be a nice addition, since currently the eyes are always turned towards the player's aiming direction.

From the gathered statistics, we see that players make facial expressions with different average intensity between game modes. The questionnaire revealed that natural communication and non-serious role-play were the main purposes for which players used the add-on. Combining these results, we believe that facial motion retargeting is mostly used for conversation in non-violent game modes, such as Cinema and to some extent DarkRP, while violent game modes such as Murder and TTT mostly feature players taunting each other. The questionnaire also revealed that players who use facial motion retargeting generally make more use of voice chat and make more facial expressions. We view these as positive effects. We conclude that while facial motion retargeting is generally not for everyone, we see much potential in its application in both competitive gameplay and social gameplay.

7 Discussion and conclusion

In this work, we presented an implementation of a real-time facial motion retargeting framework and we studied utility of the feature via a series of user studies. Via an online survey, we showed facial animations with simulated errors in accuracy and timing to 136 participants and asked them to rate the animation quality. From the responses we derived a scheme that can be used to tweak performance parameters of a working facial motion retargeting framework, to maximize perceived animation quality.

Our facial motion retargeting implementation comes in the form of an add-on for the multiplayer game Garry's Mod. We tested the add-on in a controlled environment and in public online sessions. Players in the controlled environment indicated that they did not communicate differently whether facial motion retargeting was enabled or disabled. On the other hand, players in public online sessions reported that they generally made more use of voice chat and facial expressions. This leads us to believe that factors such as whether the players know each other, whether they are frequent visitors of the game server, and how frequently they normally use voice chat affect whether they would use facial motion retargeting together. More investigation into the social conditions that encourage the use of facial motion retargeting is needed to maximally utilize the feature.

Garry's Mod players self-reportedly made more use of voice chat and made more facial expressions when face tracking was enabled. Furthermore, since online session data shows that the add-on has been actively used in both competitive and social game modes, we believe that facial motion retargeting has its uses in a wide variety of games. Anonymized face tracking data indicate that players made more exaggerated expressions during competitive game modes. Social game modes generally featured more subtle expressions. Combining this with player feedback, we believe that players are more likely to use facial motion retargeting for conversation in non-violent games, role-playing games and cooperative games. On the other hand, in violent games players would more likely use the feature for taunting each other and having fun using exaggerated expressions. Both we consider beneficial to the users' social experience.

Our experiments had a number of shortcomings. In the online survey, the clean animations that were used as basis were not of decidingly high quality, which we think is one of the causes for the high variance in the ratings per video. Repeating the experiment with more acccurate facial animations as ground truth, for example using an offline tracking algorithm or manual animation by an animator, would make the results more suited for predicting not only effects, but also absolute ratings. In the controlled game sessions, players indicated that they had trouble seeing the expressions of other players, partly due to distance between players and violent gameplay. As a result, we only got to inquire for feedback based on an experience in which facial motion retargeting was not utilized much. We believe it would be useful to perform more controlled tests with facial motion retargeting, after alleviating the visibility issue.

A number of factors can be improved upon to increase the impact of facial motion retargeting with respect to our results. A fair amount of users reported the face tracking to feel glitchy. Improving the quality of tracking, most notably consistent tracking results from different angles, will increase the trust that users place in the accuracy of the face tracking. We believe that trust in the face tracking plays an important role in the utility of the feature: ideally users should not be distracted at any moment by worries of whether the tracking is working correctly. Furthermore, we only tested the add-on in game modes that were not designed with facial motion retargeting in mind. Gameplay designed specifically with facial motion retargeting in

mind may encourage more players to use the feature at the same time. An example would be giving players a power boost when they are performing an angry expression and a defensive boost when making a happy expression.

Upcoming virtual reality head-mounted displays, such as the Oculus Rift and Sony's Project Morpheus, will significantly increase the realism with which we experience virtual worlds. This may in turn require an update to the ways players can interact, facial motion retargeting being one of the options. Since head-mounted displays occlude the upper face, new approaches to face tracking must be developed. For example, an external camera could be used to track the lower, unoccluded part of the face, while a camera inside the head gear is used to track eye, eyelid and eyebrow movement. Finally, some head-mounted displays offer positional tracking data, which can be used to speed up and improve the accuracy of face tracking.

References

[1] Bagley, J., and Manelis, L. Effect of awareness on an indicator of cognitive load. *Perceptual and Motor Skills 49*, 2 (1979), 591–594.

[2] Bartlett, M. S., Littlewort, G. C., Frank, M. G., Lainscsek, C., Fasel, I. R., and Movellan, J. R. Automatic recognition of facial actions in spontaneous expressions. *Journal of Multimedia 1*, 6 (2006), 22–35.

[3] Cohn, J. F., and Schmidt, K. L. The timing of facial motion in posed and spontaneous smiles. *International Journal of Wavelets, Multiresolution and Information Processing 2* (2004), 121–132.

[4] Comaniciu, D., and Meer, P. Mean shift: A robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 24*, 5 (2002), 603–619.

[5] Dornaika, F., and Davoine, F. On appearance based face and facial action tracking. *Circuits and Systems for Video Technology, IEEE Transactions on 16*, 9 (2006), 1107–1124.

[6] Dornaika, F., and Orozco, J. Real time 3d face and facial feature tracking. *Journal of real-time image processing 2*, 1 (2007), 35–44.

[7] Ekman, P. Facial expressions of emotion: New findings, new questions. *Psychological science 3*, 1 (1992), 34–38.

[8] Ekman, P. Facial expressions. *Handbook of cognition and emotion 16* (1999), 301–320.

[9] Ekman, P., and Friesen, W. V. *Manual for the facial action coding system.* Consulting Psychologists Press, 1978.

[10] Ekman, P., Friesen, W. V., and O'Sullivan, M. Smiles when lying. *Journal of personality and social psychology 54*, 3 (1988), 414.

[11] Ennis, C., Hoyet, L., Egges, A., and McDonnell, R. Emotion capture: Emotionally expressive characters for games. In *Proceedings of Motion on Games* (2013), ACM, pp. 53–60.

[12] Essa, I. A., and Pentland, A. P. Coding, analysis, interpretation, and recognition of facial expressions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 19*, 7 (1997), 757–763.

[13] Fogarty, C., and Stern, J. A. Eye movements and blinks: their relationship to higher cognitive processes. *International Journal of Psychophysiology 8*, 1 (1989), 35–42.

[14] Holland, M. K., and Tarlow, G. Blinking and thinking. *Perceptual and motor skills 41*, 2 (1975), 403–406.

[15] Kanade, T., Cohn, J. F., and Tian, Y. Comprehensive database for facial expression analysis. In *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on* (2000), IEEE, pp. 46–53.

[16] Karson, C. N. Physiology of normal and abnormal blinking. *Advances in neurology 49* (1987), 25–37.

[17] Lien, J. J., Kanade, T., Cohn, J. F., and Li, C.-C. Automated facial expression recognition based on facs action units. In *Automatic Face and Gesture Recognition, 1998. Proceedings. Third IEEE International Conference on* (1998), IEEE, pp. 390–395.

[18] Pitt, M. K., and Shephard, N. Filtering via simulation: Auxiliary particle filters. *Journal of the American statistical association 94*, 446 (1999), 590–599.

[19] Rowley, H. A., Baluja, S., and Kanade, T. Neural network-based face detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 20*, 1 (1998), 23–38.

[20] Saragih, J. M., Lucey, S., and Cohn, J. F. Face alignment through subspace constrained mean-shifts. In *Computer Vision, 2009 IEEE 12th International Conference on* (2009), IEEE, pp. 1034–1041.

[21] Saragih, J. M., Lucey, S., and Cohn, J. F. Deformable model fitting by regularized landmark mean-shift. *International Journal of Computer Vision 91*, 2 (2011), 200–215.

[22] Tian, Y.-l., Kanade, T., and Cohn, J. F. Recognizing action units for facial expression analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 23*, 2 (2001), 97–115.

[23] Viola, P., and Jones, M. J. Robust real-time face detection. *International journal of computer vision 57*, 2 (2004), 137–154.

[24] Wallbott, H. G., and Scherer, K. R. Stress specificities: differential effects of coping style, gender, and type of stressor on autonomic arousal, facial expression, and subjective feeling. *Journal of Personality and Social Psychology 61*, 1 (1991), 147.

[25] Zhang, C., and Zhang, Z. A survey of recent advances in face detection. Tech. rep., Tech. rep., Microsoft Research, 2010.

Appendix A LAN session questionnaire

Participant information	
What is your gender?	_____
What is your age?	_____
How many hours per week do you spend on gaming related activities?	_____

The following tables appeared twice, once for each game mode. Participants were explained that on the scale $1-7$, the extremes represent 'never' and 'all the time'.

[Murder/TTT] without facial motion retargeting	
Your actions	**Use scale 1-7**
How often did you speak verbally? (examples: typing, voice chat, shouting across room)	
How often did you make non-verbal gestures to others, excluding facial expressions? (examples: jumping, staring)	
Your perception	**Use scale 1-7**
How often did you pay attention to others speaking verbally?	
How often did you notice others making non-verbal gestures?	

[Murder/TTT] with facial motion retargeting	
Your actions	**Use scale 1-7**
How often did you speak verbally? (examples: typing, voice chat, shouting across room)	
How often did you make non-verbal gestures to others, excluding facial expressions? (examples: jumping, staring)	
How often did you make facial expressions towards others?	
When making facial expressions, how often did you feel they were noticed?	
Your perception	**Use scale 1-7**
How often did you pay attention to others speaking verbally?	
How often did you notice others making non-verbal gestures?	
How often did you notice others making facial expressions?	

Appendix B Public in-game questionnaire

Participant information
1. What is your gender? ☐ Male ☐ Female 2. What is your age? _____ 3. How many hours per week do you spend on gaming related activities? _____ 4. Have you used FlexPoser to track your facial expression? ☐ Yes ☐ No
Experience
5. For what purposes have you used or will you use FlexPoser? ☐ Screenshots ☐ Machina ☐ Natural communication (non-RP) ☐ Serious role-play ☐ Non-serious role-play 6. On a scale from 1 to 5, how often do you use text chat in multiplayer? _____ 7. On a scale from 1 to 5, how often do you use voice chat in multiplayer? _____ 8. With FlexPoser enabled, which of the following applies to you? ☐ I make more use of text chat than usual. ☐ I make more use of voice chat than usual. ☐ I make more facial expressions than usual.
Opinion
9. (Open question) What do you think of the tracking and animation quality of FlexPoser? _____ 10. (Open question) What do you think of the ease-of-use of FlexPoser? _____ 11. (Open question) What games/genres would you like to see a feature like FlexPoser in? _____ 12. (Open question) Do you have any comments or suggestions? _____