Interactive Gaze Driven Animation of the Eye Region [*]

Debanga R. Neog João L. Cardoso Anurag Ranjan Dinesh K. Pai
Sensorimotor Systems Laboratory, Department of Computer Science, University of British Columbia

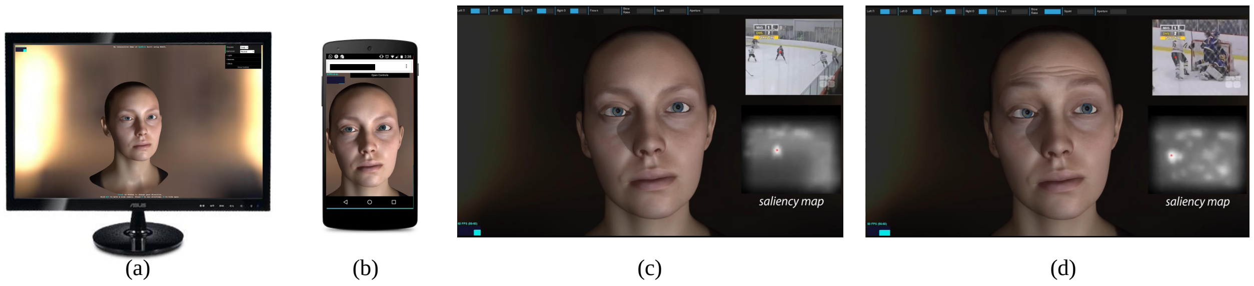


Figure 1: *We describe a complete pipeline to model skin deformation around the eyes as a function of gaze and expression parameters. The face is rendered with our WebGL application in real-time and runs in most modern browsers (a) and mobile devices (b). An example of automatic facial animation generation while watching a hockey video is shown (c-d). The character starts watching the match with a neutral expression (c) and gets concerned when a goal is scored (d). Eye movements were generated automatically using salient points computed from the hockey video.*

Abstract

We propose a system for real-time animation of eyes that can be interactively controlled in a WebGL enabled device using a small number of animation parameters, including gaze. These animation parameters can be obtained using traditional keyframed animation curves, measured from an actor's performance using off-the-shelf eye tracking methods, or estimated from the scene observed by the character, using behavioral models of human vision. We present a model of eye movement, that includes not only movement of the globes, but also of the eyelids and other soft tissues in the eye region. The model includes formation of expression wrinkles in soft tissues. To our knowledge this is the first system for real-time animation of soft tissue movement around the eyes based on gaze input.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation I.3.8 [Computer Graphics]: Three-Dimensional Graphics and Realism—Applications

Keywords: WebGL, human animation, eyes, soft tissue, gaze.

1 Introduction

Whether or not the Bard actually said "The eyes are the window to your soul," the importance of eyes is well recognized, even by the general public. Alfred Yarbus's influential work in the 1950s and 1960s quantified this intuition; using an eye tracker, he noted that observers spend a surprisingly large fraction of time fixated on the eyes in a picture. The eyes of others are important to humans because they convey subtle information about a person's mental state (e.g., attention, intention, emotion) and physical state (e.g., age, health, fatigue). Consequently, eyes are carefully scrutinized by humans and other social animals. Creating realistic computer generated animations of eyes is, therefore, a very important problem in computer graphics.

But what is it about eyes that conveys this important information to observers? To discuss this, we need to introduce some terms. The colloquial term "eye" is not sufficiently precise. It includes the *globe*, the approximately spherical optical apparatus of the eye that includes the colorful *iris*. The globe sits in a bony socket in the skull called the *orbit*. The term "eye" also usually includes the upper and lower *eyelids* and *periorbital soft tissues* that surround the orbit, including the margins of the *eyebrows*. When we refer to the eye we mean all of these tissues, and we will use the more specific term where appropriate.

The most obvious property of eyes is *gaze*, that is, what the eyes are looking at. This is entirely determined by the position and orientation of each globe relative to the scene. Almost all previous work in animation of eyes has been on animating gaze, with some recent attention paid to the appearance of the globe and iris, and the kinematics of blinks (see Sec. 2 for a review of the related work). For instance, an excellent recent survey of eye modeling in animation [Ruhland et al. 2014] does not even mention the soft tissues or wrinkles surrounding the eyes.

Gaze is clearly important, but the soft tissues of the eye also convey a lot of information. For instance, the well known "Reading the Mind in the Eyes," test developed by Autism researcher Baron-Cohen [2001] used still images of people's eyes, without any context about what they were looking at, and hence little gaze information[1]. Yet, normal observers are able to read emotion and other attributes from these images alone. We hypothesize that the state of soft tissues surrounding the eyes is a major source of informa-

[*]http://www.cs.ubc.ca/research/eyemoveweb3d16/

Web3D '16,, July 22 - 24, 2016, Anaheim , CA, USA
ISBN: 978-1-4503-4428-9/16/07 $15.00
DOI: http://dx.doi.org/10.1145/2945292.2945298

[1]You can try it yourself at http://well.blogs.nytimes.com/2013/10/03/well-quiz-the-mind-behind-the-eyes/

tion and interest to human observers. Animating these tissues is extremely important for computer graphics, but has largely been ignored so far.

In this paper we present a system for real-time animation of all the soft tissues of the eye, driven by gaze and a small number of additional parameters. A client application can download our motion models from a server and render eye animation interactively using WebGL, a cross platform specification supported by modern web browsers. Our work is complementary to the important previous work in gaze animation and on modeling the geometry and appearance of the globe and eyelid. To our knowledge, our work is the first to specifically address modeling the dynamic motion of eyes along with details of the surrounding skin deformation.

2 Related Work

As seen in [Ruhland et al. 2014], most of the research attention on eyes has been on tracking and modeling gaze, especially the gaze behavior of a character in a virtual environment. In industry, geometric procedural methods are used to model the eye region (e.g., [Pinskiy and Miller 2009]). Eye blinks have also been modeled [Trutoiu et al. 2011]. Eye tracking is now a relatively mature field, and a wide variety of off-the-shelf eye trackers of varying quality are available.

Considerable research in the past decade has been focused around facial simulation and performance capture. Physically based deformable models for facial modeling and reconstruction include the seminal work of Terzopoulos and Waters [1990]. Synthesis of high definition textures using a generative Bayesian model has been discussed in [Tsiminaki et al. 2014]. High quality capture of the appearance of the globe has been described in [Bérard et al. 2014], and of the eyelid fold in [Bermano et al. 2015]. Modeling the globe and the eyelids by using active contours has been addressed in [Moriyama et al. 2006].

Most of the recent work has been on data driven methods. Some state-of-the-art methods focus on obtaining realism based on multi-view stereo [Wu et al. 2011; Ghosh et al. 2011; Beeler et al. 2011; Furukawa and Ponce 2010; Bickel et al. 2007]; this data can be used to drive blendshapes [Fyffe et al. 2013]. Some of the work is based on binocular [Valgaerts et al. 2012] and monocular [Garrido et al. 2013; Shi et al. 2014] videos. Recent work by Li et al. [2013b] described a system for real-time and calibration-free performance-driven animation. [Weise et al. 2011] presented a system for performance-based character animation by using commodity RGB-D cameras that can be used to generate facial expressions of an avatar in real-time.

With the widespread use of WebGL and the hardware commonly found in small devices, it is now possible to render high quality human characters over the web. Both eye diffraction effects [Vill 2014] and skin subsurface scattering proprieties [AlteredQualia 2011] can be rendered at interactive rates. We make use of these advances to generate facial animations using our model. Our system provides novel parametrized facial motion models that require minimal space and computation to simulate complex tissue movements in the eye region.

3 Skin Movement in Reduced Coordinates

Our goal is to model the movements of the skin around the eyes. This is the most important part of the face to convey expression, and therefore it is worthwhile to design a model specifically for this region. Other parts of the face may be modeled by more traditional methods. Our model has two motivations: First, since there are no articulating bones in the eye region, the skin slides on the skull almost everywhere (except at the margins where it slides over the globes; these margins are discussed below). Therefore, we would like to efficiently model this skin sliding. Second, we would like the model to be easily learned using videos of real human subjects. We have developed such a system for tracking and learning skin movement using a single video camera; due to space limitations that is described in a forthcoming paper, with a brief outline provided in Appendix A. The details are not important here, but it is useful to keep this motivation in mind to understand our choices.

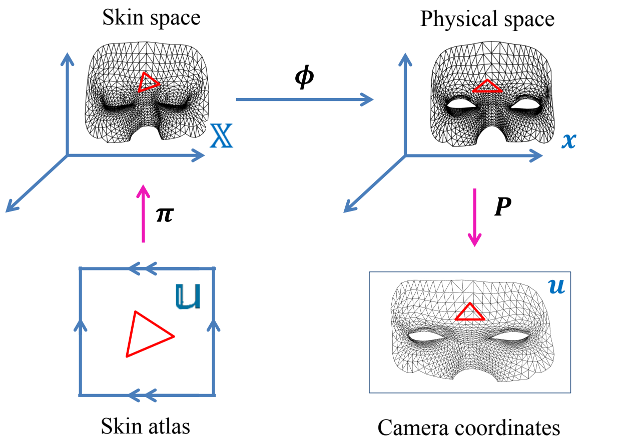


Figure 2: *Overview of the spaces used for modeling skin movement.*

To represent the motion of skin, we use the reduced coordinate representation of skin introduced by [Li et al. 2013a]. This representation constrains the synthesized skin movement to always slide tangentially on the face, even after arbitrary interpolation between different skin poses. This avoids cartoonish bulging and shrinking and other interpolation artifacts. We will see in Sec. 6 that deformation perpendicular to the face can be achieved where needed, for example in the movement of the eyelid. This representation also reduces the data size in most computations.

Skin is represented by its 3D shape in a reference space called *skin space*; this space is typically called “modeling space” in graphics and “material space” in solid mechanics. The key idea is that since skin is a thin structure, we can also represent it using a 2D parameterization π, using an atlas of coordinate charts. See Fig. 2. In our case, a single chart, denoted *skin atlas*, is sufficient; it can be thought of as the skin’s texture space.

Skin is discretized as a mesh $S = (V, E)$, a graph of n_v vertices V and n_e edges E. In contrast to [Li et al. 2013a], this is a Lagrangian mesh, i.e., a point associated with a vertex is fixed in skin space. Since most face models use a single chart to provide texture coordinates, these coordinates form a convenient parameterization π. For instance, that is the case for face models obtained from FaceShift [Weise et al. 2011], which we use for the examples shown in this paper. In a slight departure from the notation of [Li et al. 2013a], a skin material point corresponding to vertex i is denoted $\mathbb{X}_i$ in 3D and $\mathbb{u}_i$ in 2D coordinates. We denote the corresponding stacked arrays of points corresponding to all vertices of the mesh as $\mathbb{X}$ in 3D skin space and $\mathbb{u}$ in the 2D skin atlas.

The skin moves on a fixed 3D *body* corresponding to the shape of the head around the eyes. Instead of modeling the body as an arbitrary deformable object as in [Li et al. 2013a], we account for the specific structure of the hard parts of the eye region. We model the body as the union of two rigid parts, the *skull* (a closed mesh corresponding to the anatomical skull with the eye sockets closed by a smooth surface) and the mobile *globe* that is not spherical, with a prominent cornea that displaces the eyelid skin. See Fig. 3. This allows us to efficiently parameterize changes in shape of the body using the rotation angles of the globe and is useful for modeling gaze drive motion synthesis and described in detail in Sec. 6.

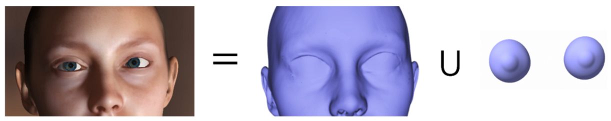


Figure 3: *Body is the union of "skull" and globes.*

The skin and body move in *physical space*, which is the familiar space in which we can observe the movements of the face, for instance, with a camera. For modeling, we assume there is a head-fixed camera with projective transformation matrix P that projects a 3D point corresponding to vertex i (denoted $\boldsymbol{x}_i$) into 2D camera coordinates $\boldsymbol{u}_i$, plus a depth value d_i. This *modeling camera* could be a real camera which is used to acquire video for learning the model (as outlined in Appendix A) or could be a virtual camera, simply used as a convenient way to parameterize the skin in physical space. We note that P is invertible, since P is a full projective transformation, and not a projection. We denote the stacked arrays of points corresponding to all vertices of the mesh as $\boldsymbol{x}$ in 3D physical space and $\boldsymbol{u}$ in the 2D camera coordinates.

4 Gaze Parameterized Model of Skin Movement

During movements of the eyes, the skin in the eye region slides over the body. It is this sliding we are looking to model. Following the standard notation in solid mechanics, the motion of the skin from 3D skin space to 3D physical space is denoted ϕ (see Fig. 2). Therefore, we can write $\boldsymbol{x} = \phi(\mathbb{X})$. However, directly modeling ϕ is not desirable, as it does not take into account the constraint that skin can only slide over the body, and not move arbitrarily in 3D. Instead, the key to our reduced coordinate model is that we represent skin movement in modeling camera coordinates, i.e., we model the 2D transformation

$$\boldsymbol{u} = P(\phi(\pi(\mathtt{u}))) \overset{\text{def}}{=} f_g(\mathtt{u}).$$

Our goal is to directly model the function f_g as a function of input parameters, g, such as gaze and other factors that affect skin movement around the eyes. This representation has the dual advantage of enforcing the sliding constraint and being easy to acquire video data from which to learn how the skin moves.

4.1 Factors affecting skin movement in the eye region

We now examine the different input variables g that determine skin movement in the eye region. The most important and dynamic cause is eye movements that change gaze, i.e., that change what the character is looking at. However, other parameters, like eyelid aperture and expressions, also affect the skin; we combine these into the "generalized" gaze vector g.

Gaze. Humans make an average of three rapid gaze shifts (called saccades) every second, and slow movements called "smooth pursuit" to track small moving targets, and other slow movements called vestibulo-ocular reflex and opto-kinetic reflex to stabilize the image on the retina. Eye movements change the orientation of the globe relative to the head. It is not commonly appreciated that the eyes have full 3 degrees of freedom, even though to point the optical axis at a target only requires 2 degrees of freedom; rotations about the optical axis, called "torsion," have been known and modeled at least since the 19th century. Any parameterization of 3D rotations could be used. We use Fick coordinates [Fick 1854], which are widely used in the eye movement literature to describe the 3D rotation of the eye, since it factors the torsion in a convenient form. These are a sequence of rotations – first horizontal (g_1), then vertical (g_2), finally torsion (g_3). See Fig. 4.

Eyelid Aperture. Eyelid movements are affected by both gaze and other factors. When our gaze shifts, eyelids, especially the upper eyelids, move to avoid occluding vision. We also move our eyelids to blink, and when expressing mental state such as arousal, surprise, fatigue, and skepticism. The upper and lower eyelids move in subtly different ways. Therefore, we use two additional input parameters to define aperture. One is the displacement of the midpoint of the upper eyelid above a reference horizontal plane with respect to the head (g_4); the plane is chosen to correspond to the position of the eyelid when closed. The other input is the displacement of the midpoint of the lower eyelid below this plane (g_5).

Expressions. The skin in the eye region is also affected by facial expressions, such as surprise, anger, and squint. We can optionally extend the input parameters g to include additional parameters to control complex facial expressions. Expressions may be constructed using Action Units (AUs), defined by the Facial Action Coding System (FACS) [Ekman and Friesen 1977]. In our implementation, AUs are used in a similar way as blend shapes; they may be learned from using 'sample poses' that a subject is asked to perform or could also be specified by an artist. See Table 1 (left). The strength of the i^{th} AU used in the model contributes an additional input parameter, $g_{i+10} \in [0, 1]$. Note that we defined five parameters per eye (3 gaze and 2 aperture), which together contribute the first 10 inputs.

Sample poses		Expression generation	
AU	FACS name	Sample poses	AUs
1	Inner brow raiser	Surprise	1, 2, 5
2	Outer brow raiser	Anger	4, 5
4	Brow lowerer	Fear	1, 2, 4, 5
5	Upper lid raiser	Sadness	1, 4
43	Eyes closed	Squint	44
44	Squint	Blink	5, 43

Table 1: *FACS AU used in our experiments (left) and expressions that can be produced using these AUs (right).*

4.2 Generative Model of Skin Movement

Following these observations, we factor the generative model into three parts – *baseline aperture model*, *eyelid shape model*, and *skin motion model*. Each of these models can be constructed or learned separately. A schematic diagram of the implementation is shown in Fig. 4.

In the following, we will assume g is a $n_i \times 1$ column matrix, where n_i is the total number of possible inputs. Submatrices are extracted using Matlab-style indexing, e.g., $g_{1:3}$ is the submatrix comprised of rows 1 to 3, and $g_{[4,5]}$ is a submatrix with just the fourth and fifth elements. To achieve real-time performance at low computational costs, and also to exploit GPU computation using WebGL, we choose linear models for this application. We have explored using non-linear neural network models but these are more expensive to evaluate.

Eyelid Aperture Model. As discussed above, the aperture depends

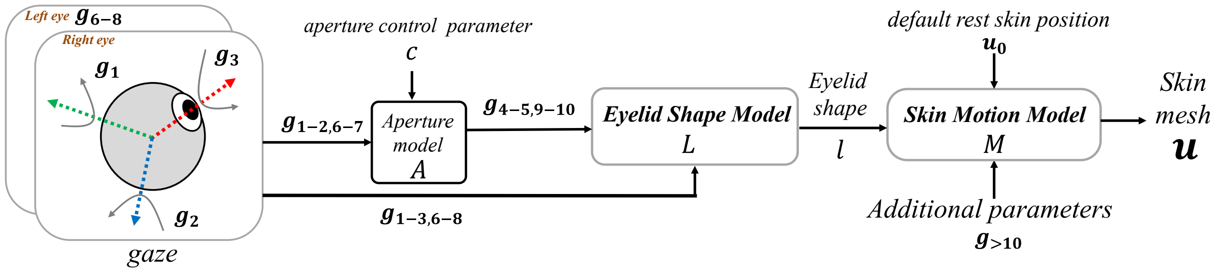


Figure 4: *Gaze parametrized skin movement: We predict eyelid aperture from gaze using an aperture model, and we use gaze and aperture together to predict skin deformation in the eye region. If desired, expression parameters can also be included to produce facial expressions.*

on gaze, but it is modulated by other voluntary actions such as blinking. We use a linear baseline aperture model, A, for predicting the aperture due to gaze; since the torsion angle of the globe does not have a significant effect on aperture, we only use the first two components of gaze. The baseline aperture is then scaled by the *eye closing* factor $c \geq 0$ to simulate blinks, arousal, etc. The resulting model for the left eye is:

$$g_{[4,5]} = c\ A\ g_{[1,2]}. \tag{1}$$

Eyelid Shape Model. We observed that the deformation of skin in the eye region is well correlated with the shape of the eyelid margin. This makes biomechanical sense, since the soft tissues around the eye move primarily due to the activation of muscles surrounding the eyelids, namely *orbicularis oculi* and *levator palpebrae* muscles. We define eyelid shape for each eyelid as piecewise cubic spline curves. We found that using between 17 and 22 control points for each spline faithfully captures the shape of the eyelids. The eyelid shape depends on both gaze and aperture. The general form of the model is

$$l = L\ g, \tag{2}$$

where l is the column matrix of coordinates of all control points for the eyelid.

Skin Motion Model. The skin of the eye region is modeled at high resolution (using about a thousand vertices in our examples). It is deformed based on the movement of the eyelids. We use a linear model for this, since the skin movement can then be efficiently evaluated using vertex shaders on the GPU, and also learned quickly. Note that the skin motion depends on all four eyelids; the stacked vector of coordinates of all four eyelids is denoted $\boldsymbol{l}$. The resulting model is

$$\boldsymbol{u} = \boldsymbol{u}_0 + M\ \boldsymbol{l}, \tag{3}$$

where $\boldsymbol{u}_0$ is the default rest position of the skin.

5 Transferring Animations

The skin motion model of Sec. 4.2 is constructed for a specific subject. The skin model may not include some parts, such as the inner eyelid margins, which are difficult see and track in video. Here we discuss how the information in the generative model can be transferred to other target characters and to untracked parts of eye region.

Target Transfer. Given a new target character mesh with topology different from the *captured subject mesh* (3D face mesh of the subject for whom the model was constructed), we have to map the model output $\boldsymbol{u}$ to new image coordinates $\tilde{\boldsymbol{u}}$ representing the motion of the new mesh in image coordinates. The map is computed as follows: we first use a non-rigid ICP method [Li et al. 2008] to register the target mesh to the captured subject mesh in 3D. The resulting mesh is called the *registered mesh*. The vertices of the registered mesh are then snapped to the nearest faces of the captured subject mesh. We compute the barycentric weights of the registered mesh vertices with respect to the captured subject mesh, and construct a sparse matrix B of barycentric coordinates that can transform $\boldsymbol{u}$ to $\tilde{\boldsymbol{u}}$ as $\tilde{\boldsymbol{u}} = B\boldsymbol{u}$ (see Fig. 5).

Movement Extrapolation. Some vertices of the target mesh, particularly those of the inner eyelid margin, are not included in the skin movement model computed in Eq. 3. This is because such vertices cannot be tracked in video and it is difficult to build a data-driven model for them. Instead the skin coordinates of those vertices are computed as a weighted sum of nearby tracked vertices. The normalized weights are proportional to the inverse distance to the neighboring points (we used 10) in the starting frame.

6 Client Applications

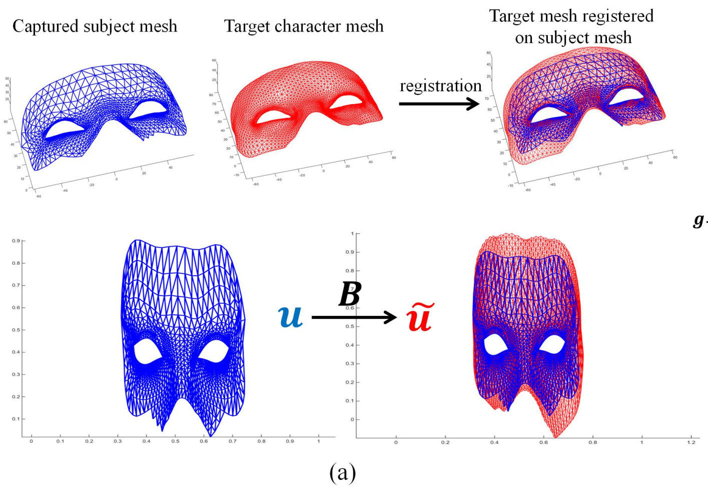
We implemented two different client applications using WebGL and JavaScript. Both synthesize animations of the eye region in real time using our model. We used the *Digital Emily* data provided by the WikiHuman project [Ghosh et al. 2011; WikiHuman]. The applications start by downloading all the required mesh, texture, and model data. Then, they run offline and perform all computations without communicating with a server. Blink input is handled by a stochastic blink model [Trutoiu et al. 2011]. Wrinkles are synthesized using an additional wrinkle map texture.

The two applications differ only on the model input sources: one is a fully interactive user controlled application, while the other plays a cut-scene defined using keyframed animation curves (which are also displayed on screen).

We now discuss some important details of these applications.

Optimized Motion Model Since all our operations are linear, we can premultiply the eyelid shape models L and the skin motion model M to construct one large matrix. However, using principal components analysis (PCA) we found that the results are well approximated using only 4 principal components. The PCA is precomputed on the server, and can be computed at model construction time. This results in a significantly smaller matrices, U (tall and skinny) and W (small), such that

$$M\ \boldsymbol{l} \approx U\ Wg. \tag{4}$$



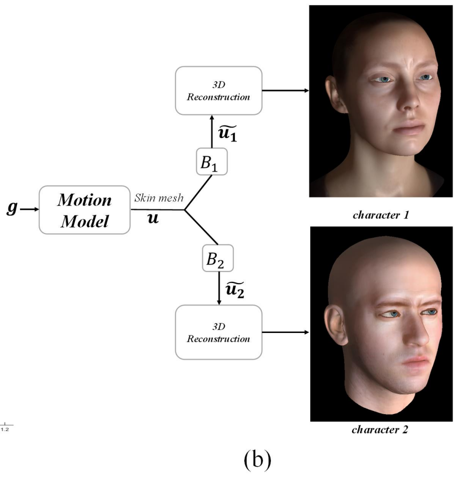


Figure 5: *An overview of target transfer. (a) The target character mesh (red) is registered non-rigidly on the capture subject mesh (blue) shown in the top row. Image coordinates of target mesh are computed from the image coordinates of the model output using barycentric mapping computed during registration. (b) The model trained on one subject can be used to generate animation of a character mesh of any topology uploaded by an user.*

Only the smaller matrices U and W need to be downloaded from the server. We compute $\boldsymbol{e} \overset{\text{def}}{=} W\boldsymbol{g}$ on the CPU, and compute $U\boldsymbol{e}$ using per-vertex computations on the GPU's vertex shader as described below.

Reconstructing 3D Geometry Recall that we represent the 3D coordinates $\boldsymbol{x}$ of the original skull mesh using the 2D skin coordinates $\boldsymbol{u}$ corresponding to the neutral pose. To determine 3D vertex positions $\boldsymbol{x}$ from $\boldsymbol{u}$ we pre-render a depth texture of $\boldsymbol{x}$ in 2D skin space using natural neighbor interpolation on the vertices values. This texture, which we name the *skull map*, can be sampled to obtain $\boldsymbol{x}$ from $\boldsymbol{u}$. We also render a *skull normal map* using the same procedure to sample normals given $\boldsymbol{u}$.

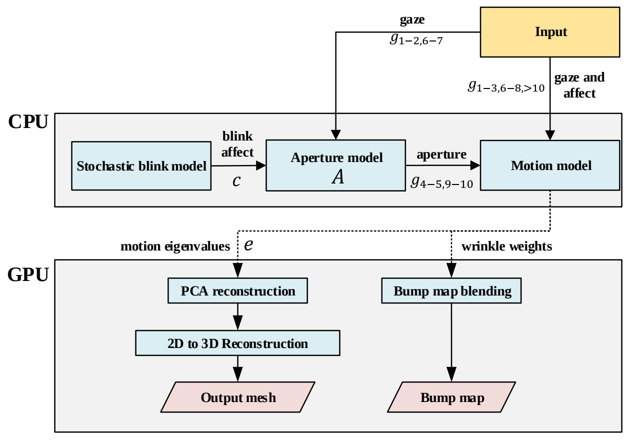


Figure 6: *Flow chart of how inputs are handled in our applications by the animation model.*

GPU Computations Our framework is well suited for real-time synthesis using GPUs. While the aperture model and the PCA-reduced motion model W are applied on the CPU, the PCA reconstruction using U is performed on the GPU, per vertex, as a matrix-vector multiplication in the vertex shader. Expression wrinkles are generated by blending the wrinkle bump maps using the expression parameters as weights in the fragment shader.

Our model predicts motion of the skin in 2D modeling camera coordinates. The vertices are then lifted to their final 3D positions by sampling the *skull map*. However, since the globe is not spherical and has a prominent cornea, rotations of the globe should produce subsurface deformations of the skin. To address this challenge, we first represent the shape of the globe in polar coordinates in the globe's reference frame, and precompute a depth map $D(g_1, g_2)$ that represents the globe shape. For each frame, vertices of the eyelid are radially displaced from the cornea surface at the current globe orientation, using the depth map and the skin thickness.

Additionally, some of the extrapolated vertices (such as those in the canthus and eyelid margins) should not slide directly on the skull or globe geometry. Hence we handle these vertices differently. For each extrapolated vertex, we precompute the radial distance o between the original vertex position of the target mesh and the globe. We then offset the skin vertex at its current position on the skull by o. This allows us to reconstruct the eyelid margin thickness and canthus depressions. The skin normal, especially for an extrapolated vertex in the eyelid margin, is computed by parallel transport from the normal at its original position to its current position.

WebGL Considerations One of the main limitations of current WebGL enabled mobile devices is that there is no support for multi-pass rendering. This feature is extremely important for less traditional rendering methods such as deferred shading, post-processing effects or, more importantly for our case, subsurface scattering methods.

To achieve realistic real-time skin rendering it is standard to apply an approximation of the subsurface scattering proprieties of the skin. We implement the method proposed by Jimenez et al. [2015]. It requires 3 rendering passes: the first projects the geometry to generate screen space diffuse, specular and depth maps of the skin. This is followed by two screen space passes that perform a Gaussian blur on the diffuse illumination map taking into account the depth map. The last of these passes will also sum the diffuse and

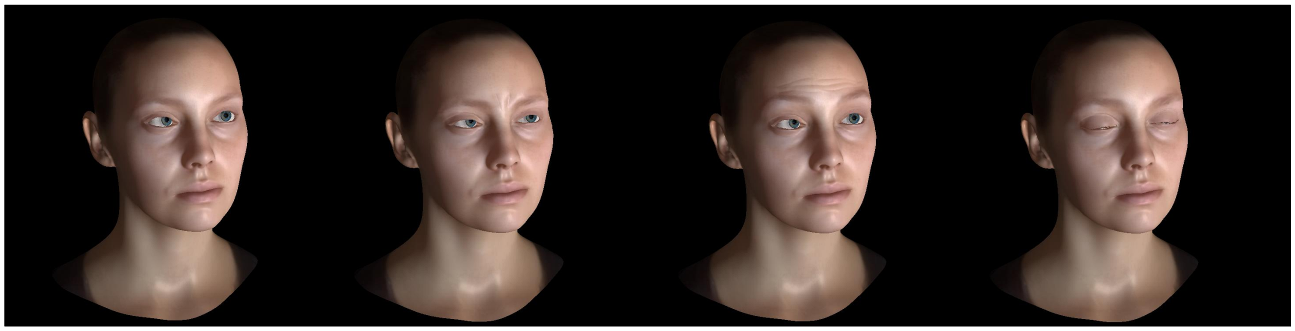


Figure 7: *We can generate facial expressions interactively. From left to right: normal, frown, surprise, and eye closure expressions are shown.*

specular illuminations to achieve the final result.

In our implementation, while we still only perform 3 passes, we need to perform two geometric passes (instead of one): the first rendering pass generates the diffuse illumination and depth maps (depth is stored in the alpha channel). The second one is a screen space Gaussian blur. The third, although it performs the screen space blur, also computes specular illumination and sums it to the diffuse for the final result. Hence, our optimized SSSS system for WebGL requires our vertex shader program to run twice per frame. This is taken into account in Table 2.

7 Results

The applications run in any modern browser, at 60fps on a desktop with an Intel Core i7 processor and an NVIDIA GeForce GTX 780 graphics card, and at 24fps on an ultraportable laptop with an Intel Core i7 processor and integrated Intel HD 5500 Graphics, and at 6fps on a Nexus 5 android phone with Adreno 330 GPU. The majority of workload is for rendering; the model itself is very inexpensive (see Table 2).

Readers can try the WebGL application on the Internet, at `http://www.cs.ubc.ca/research/eyemoveweb3d16/`. A video of the results is provided as supplementary material. The accompanying video shows a clip captured live from the application running in a browser. In the video, we also included head movements of a user tracked using FaceShift and a statistical blink model [Trutoiu et al. 2011] for realism. Different expressions, such as surprise and anger, are added at key moments. Please note changes in wrinkles and eyes during these expressions.

Saliency Map Controlled Movement. When we observe object motion in real life or in a video, our eyes produce characteristic saccades. We computed saliency maps, a representation of visual attention, using the method proposed in [Itti et al. 1998]. Points in the image that are most salient are used as gaze targets to produce skin movements around the eyes. In Fig. 8 and also in the accompanying video, we show an example of skin movement controlled by gaze, using salient points detected in a video of a hockey match.

Static Scene Observation. Our generative gaze model can be controlled by gaze data obtained from any eye tracking system. We used gaze data of a subject observing a painting to drive our system. This produces very realistic movements of eyelid and skin around the eyes as can be seen in Fig. 9.

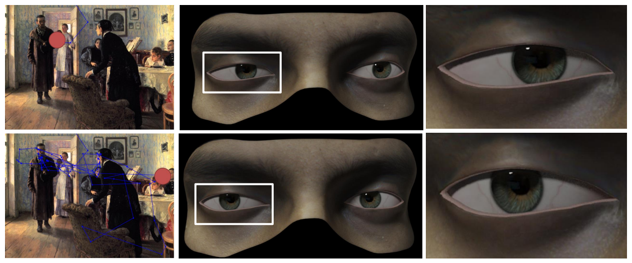


Figure 9: *Skin movement driven by gaze during static scene observation. The red circle in the left represents the image point subject is looking at.*

Eyelid Deformation during Blink. We can generate skin deformation in a blink sequence using a statistical blink model that controls aperture based on the experimental data reported in [Trutoiu et al. 2011]. A blink sequence is shown in Fig. 10.

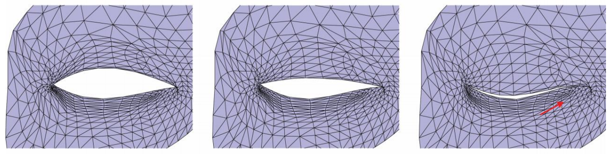


Figure 10: *Skin deformation in eye closing during a blink. The characteristic medial motion of the lower eyelid during blink is generated by the model (shown using red arrow).*

Vestibulo-ocular Reflex. Our model can also generate eye region movement in novel scenarios in which the head moves. For example, when a character fixates on an object and rotates its head, the eyes counter-rotate. This phenomenon is known as the 'vestibulo-ocular reflex'. Our model predicts the skin deformation on the eyelid and around the eyes during such movements. Please see the video.

8 Conclusion

Despite an enormous amount of previous work on modeling eyes for computer animation, the motion of soft tissues around the eyes when a character's eyes move has largely been neglected. Our work

Animation Type	File download (MB)	Memory Usage (MB)	GPU memory (MB)	Runtime per frame (ms)
Static	3.7	240	390	0.5450 ± 0.1553
Animated Emily	5.3	370	417	0.6717 ± 0.1564
Animation overhead	**1.6**	**130**	**27**	$\mathbf{0.1267 \pm 0.0011}$

Table 2: *Overview of the model size, memory usage, and performance of the animation. The experiments are run in Chrome web browser on a desktop with an Intel Core i7 processor and an NVIDIA GeForce GTX 780 graphics card at 60 fps.*

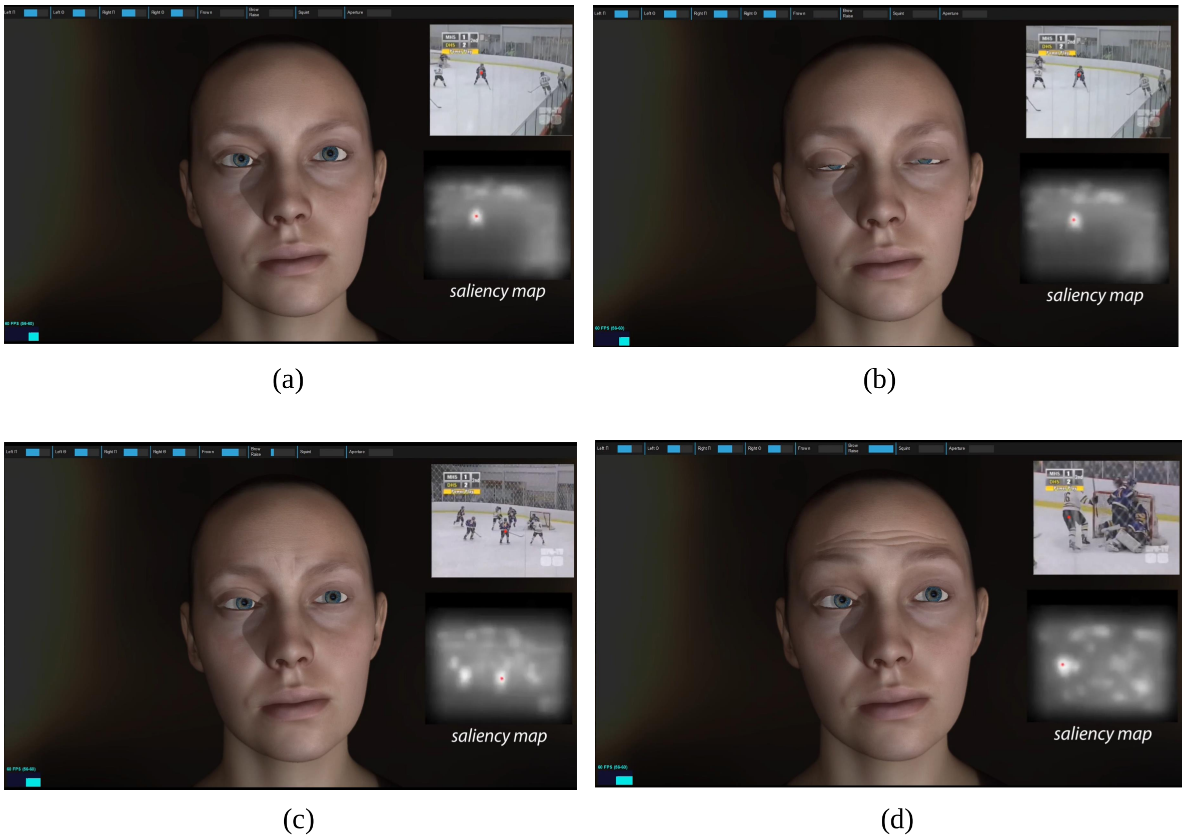


Figure 8: *Facial animation automatically generated from a hockey video. The character starts watching the match with a neutral expression (a), our stochastic model generates natural blinks (b), gets anxious on anticipating a goal (c), and excited when a goal is scored (d). The salient points are computed from the video and used as input to our system.*

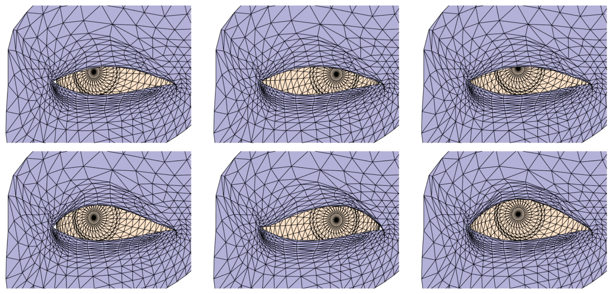


Figure 11: *Comparison with no skin deformations(top), and with deformation (bottom) using our model with different gaze positions.*

is the first to specifically address the problem of soft tissue movements in the entire region of the eye. The model is also inexpensive in terms of run times and memory, as seen in Table 2. Our model currently has a few limitations that we hope to address in the near future. These include the linearity of the generative model required for inexpensive simulation in the browser using WebGL, and the lack of eyelashes. Nevertheless, the system is able to generate realistic eye movements using a variety of input stimuli.

Acknowledgements The authors would like to thank Paul Debevec and the WikiHuman Project for providing the Digital Emily model used in our examples. This work was funded in part by grants from NSERC, Peter Wall Institute for Advanced Studies, Canada Foundation for Innovation, and the Canada Research Chairs Program.

References

ALTEREDQUALIA, 2011. WebGL skin. https://www.chromeexperiments.com/experiment/webgl-skin, Sep.

BARON-COHEN, S., WHEELWRIGHT, S., HILL, J., RASTE, Y., AND PLUMB, I. 2001. The “reading the mind in the eyes” test revised version: A study with normal adults, and adults with asperger syndrome or high-functioning autism. *Journal of child psychology and psychiatry 42*, 2, 241–251.

BEELER, T., HAHN, F., BRADLEY, D., BICKEL, B., BEARDSLEY, P., GOTSMAN, C., SUMNER, R., AND GROSS, M. 2011. High-quality passive facial performance capture using anchor frames. *ACM Trans. Graph. 30*, 4 (July), 75:1–75:10.

BÉRARD, P., BRADLEY, D., NITTI, M., BEELER, T., AND GROSS, M. 2014. Highquality capture of eyes. *ACM Transactions on Graphics 33*, 6, 223.

BERMANO, A., BEELER, T., KOZLOV, Y., BRADLEY, D., BICKEL, B., AND GROSS, M. 2015. Detailed spatio-temporal reconstruction of eyelids. *ACM Transactions on Graphics (TOG) 34*, 4, 44.

BICKEL, B., BOTSCH, M., ANGST, R., MATUSIK, W., OTADUY, M., PFISTER, H., AND GROSS, M. 2007. Multi-scale capture of facial geometry and motion. *ACM Transactions on Graphics (TOG) 26*, 3, 33.

EKMAN, P., AND FRIESEN, W. V. 1977. *Facial action coding system*. Consulting Psychologists Press, Stanford University, Palo Alto.

FICK, A. 1854. Die bewegung des menschlichen augapfels. *Z. Rationelle Med.*, 4, 101–128.

FURUKAWA, Y., AND PONCE, J. 2010. Dense 3d motion capture from synchronized video streams. In *Image and Geometry Processing for 3-D Cinematography*. Springer, 193–211.

FYFFE, G., JONES, A., ALEXANDER, O., ICHIKARI, R., GRAHAM, P., NAGANO, K., BUSCH, J., AND DEBEVEC, P. 2013. Driving high-resolution facial blendshapes with video performance capture. In *ACM SIGGRAPH 2013 Talks*, ACM, 33.

GARRIDO, P., VALGAERTS, L., WU, C., AND THEOBALT, C. 2013. Reconstructing detailed dynamic face geometry from monocular video. *ACM Trans. Graph. 32*, 6, 158.

GHOSH, A., FYFFE, G., TUNWATTANAPONG, B., BUSCH, J., YU, X., AND DEBEVEC, P. 2011. Multiview face capture using polarized spherical gradient illumination. *ACM Transactions on Graphics (TOG) 30*, 6, 129.

ITTI, L., KOCH, C., AND NIEBUR, E. 1998. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence 20*, 11, 1254–1259.

JIMENEZ, J., ZSOLNAI, K., JARABO, A., FREUDE, C., AUZINGER, T., WU, X.-C., VON DER PAHLEN, J., WIMMER, M., AND GUTIERREZ, D. 2015. Separable subsurface scattering. *Computer Graphics Forum*.

LI, H., SUMNER, R. W., AND PAULY, M. 2008. Global correspondence optimization for non-rigid registration of depth scans. *Computer graphics forum 27*, 5, 1421–1430.

LI, D., SUEDA, S., NEOG, D. R., AND PAI, D. K. 2013. Thin skin elastodynamics. *ACM Transactions on Graphics (TOG) 32*, 4, 49.

LI, H., YU, J., YE, Y., AND BREGLER, C. 2013. Realtime facial animation with on-the-fly correctives. *ACM Transactions on Graphics (Proceedings SIGGRAPH 2013) 32*, 4 (July).

MORIYAMA, T., KANADE, T., XIAO, J., AND COHN, J. F. 2006. Meticulously detailed eye region model and its application to analysis of facial images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 28*, 5, 738–752.

PINSKIY, D., AND MILLER, E. 2009. Realistic eye motion using procedural geometric methods. *SIGGRAPH 2009: Talks*, 75.

RUHLAND, K., ANDRIST, S., BADLER, J., PETERS, C., BADLER, N., GLEICHER, M., MUTLU, B., AND MCDONNELL, R. 2014. Look me in the eyes: A survey of eye and gaze animation for virtual agents and artificial systems. In *Eurographics 2014-State of the Art Reports*, The Eurographics Association, 69–91.

SHI, F., WU, H.-T., TONG, X., AND CHAI, J. 2014. Automatic acquisition of high-fidelity facial performances using monocular videos. *ACM Trans. Graph. 33*, 6 (Nov.), 222:1–222:13.

TERZOPOULOS, D., AND WATERS, K. 1990. Physically-based facial modelling, analysis, and animation. *The Journal of Visualization and Computer Animation 1*, 2 (Dec.), 73–80.

TRUTOIU, L. C., CARTER, E. J., MATTHEWS, I., AND HODGINS, J. K. 2011. Modeling and animating eye blinks. *ACM Transactions on Applied Perception (TAP) 8*, 3, 17.

TSIMINAKI, V., FRANCO, J.-S., BOYER, E., ET AL. 2014. High resolution 3d shape texture from multiple videos. In *International Conference on Computer Vision and Pattern Recognition*.

VALGAERTS, L., WU, C., BRUHN, A., SEIDEL, H.-P., AND THEOBALT, C. 2012. Lightweight binocular facial performance capture under uncontrolled lighting. *ACM Trans. Graph. 31*, 6, 187.

VILL, A., 2014. Eye texture raytracer. https://www.chromeexperiments.com/experiment/eye-texture-raytracer, Feb.

WEISE, T., BOUAZIZ, S., LI, H., AND PAULY, M. 2011. Realtime performance-based facial animation. *ACM Transactions on Graphics (Proceedings SIGGRAPH 2011) 30*, 4 (July).

WIKIHUMAN. `http://gl.ict.usc.edu/Research/DigitalEmily2/`.

WU, C., WILBURN, B., MATSUSHITA, Y., AND THEOBALT, C. 2011. High-quality shape from multi-view stereo and shading under general illumination. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, IEEE, 969–976.

Appendices

A Measurement for Data-driven Model Construction

Here we briefly outline the measurement process that was used to construct the model in our system. Offline capture is conducted in a studio setup. To measure the motion around the eye region, we used a single Grasshopper3 [2] camera that can capture up to 120 fps with image resolution of 1960×1200 pixels. The actor sits in a chair and faces the camera with the head rested on a chin rest. The scene is lit by a DC powered LED light source [3] to overcome the flickering due to the ambient AC light source on the high frame rate capture. We use polarizing filters with the cameras to reduce specularity. We

[2]Point Grey Research, Vancouver, Canada

[3]https://www.superbrightleds.com

ask the subject to look at targets generated on a screen, and also to move their eyes as instructed. These data are used to measure skin motion for different gaze positions. We also ask the subject to display different facial expressions (e.g., brow raise, eyebrow gathering, etc., listed as action units in the Facial Action Coding System (FACS) [Ekman and Friesen 1977]). They are used to model additional parameters. Camera calibration is performed using Matlab's Computer Vision System Toolbox to compute the camera projection matrix P. Our skin tracking algorithm requires a subject specific 3D mesh called *skull*; to acquire this we use FaceShift [Weise et al. 2011] technology with a Kinect RGB/D camera. This process takes less than 15 minutes per subject.