Lightweight Eye Capture Using a Parametric Model

Pascal Bérard[1,2] Derek Bradley[2] Markus Gross[1,2] Thabo Beeler[2]
1) ETH Zurich 2) Disney Research

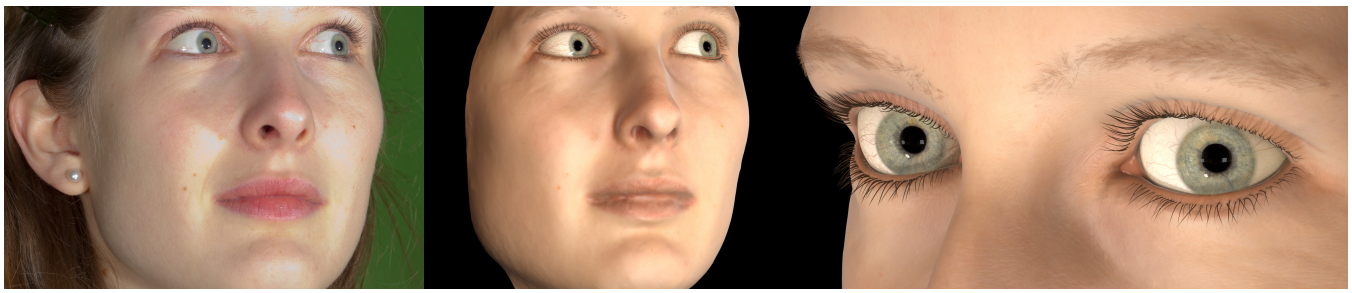

Figure 1: *We present a new parametric eye model and image-based fitting method that allows for lightweight eye capture at very high quality. Our eye capture method can be integrated with traditional multi-view face scanners (as seen here), or can operate even on a single image.*

Abstract

Facial scanning has become ubiquitous in digital media, but so far most efforts have focused on reconstructing the skin. Eye reconstruction, on the other hand, has received only little attention, and the current state-of-the-art method is cumbersome for the actor, time-consuming, and requires carefully setup and calibrated hardware. These constraints currently make eye capture impractical for general use. We present the first approach for high-quality *lightweight* eye capture, which leverages a database of pre-captured eyes to guide the reconstruction of new eyes from much less constrained inputs, such as traditional single-shot face scanners or even a single photo from the internet. This is accomplished with a new parametric model of the eye built from the database, and a novel image-based model fitting algorithm. Our method provides both automatic reconstructions of real eyes, as well as artistic control over the parameters to generate user-specific eyes.

Keywords: Eye capture, Eye modelling, Face reconstruction

Concepts: •**Computing methodologies** → **Reconstruction; Computer graphics; Shape modeling;**

1 Introduction

Capturing faces through 3D scanning techniques has become the industry-standard approach to build face models for video games, visual effects in films, medical applications and personalized figurines. Several decades of research have pushed facial capture technology to an incredible level of quality, where it is becoming difficult to distinguish the difference between digital faces and real ones. On the other hand, most research has focused on the facial skin, ignoring other important characteristics like the eyes. The eyes are arguably the most important part of the face, as this is where humans tend to focus when looking at someone. Eyes can convey emotions and foretell the actions of a person and subtle inaccuracies in the eyes of a character can make the difference between realistic and uncanny.

Despite its importance, capturing the eyes has received far less attention than capturing the rest of the face. The most notable exception is the recent work of Bérard et al. [2014], who devised a technique to scan human eyes in very high quality using a custom capture setup and a trio of algorithms, tailored to handle the three visible parts of the eye; the white sclera, the transparent cornea and the deforming iris. While the results are compelling, the acquisition process is both time consuming and uncomfortable for the actors, as they must lie horizontally with a constraining neck brace while manually holding their eye open for dozens of photos over a 20 minute period for each eye. The physical burden of that approach is quite far from the single shot face scanners that exist today, which are as easy as taking a single photo in a comfortable setting, and thus the applicability of their method is largely limited.

In this work, we present a new lightweight approach to eye capture that achieves a comparable level of quality as Bérard et al. but from input data that can be obtained using traditional single-shot face scanning methods or even just from a single image. Our key idea is to build a parametric model of the eye, given a training database of high-quality scans provided by Bérard et al. Our model succinctly captures the unique variations present across the different components of the eye labeled in Fig. 2, including 1 - the overall size and shape of the eyeball and cornea, 2 - the detailed shape and color of the iris and its deformation under pupil dilation, and 3 - the detailed vein structure of the sclera which contributes to both its color and fine-scale surface details.

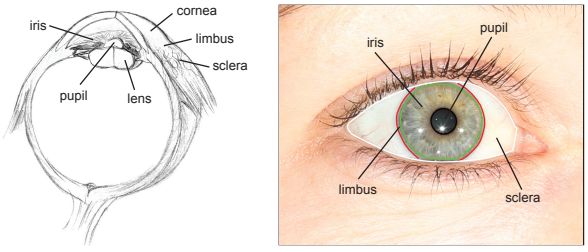

Figure 2: *The visually salient parts of a human eye include the black pupil, the colored iris, and the limbus that demarcates the transition from the white sclera to the transparent cornea. Eye schematic courtesy of Bérard et al. [2014].*

SIGGRAPH '16 Technical Paper, July 24 - 28, 2016, Anaheim, CA,
ISBN: 978-1-4503-4279-7/16/07
DOI: http://dx.doi.org/10.1145/2897824.2925962

Given our model, new and unique human eyes can be created. Aspects like the shape or the color can be controlled without in-depth knowledge of the subtleties of real eyes. Furthermore, we propose a novel fitting algorithm to reconstruct eyes from sparse input data, namely multi-view images, i.e. from a single-shot multi-view face scanner. The results are very plausible eye reconstructions with realistic details from a simple capture setup, which can be combined with the face scan to provide a more complete digital face model. In this work we demonstrate results using the face scanner of Beeler et al. [2010], however our fitting approach is flexible and can be applied to any traditional face capture setup. Furthermore, by reducing the complexity to a few intuitive parameters, we show that our model can be fit to just single images of eyes or even artistic renditions, providing an invaluable tool for fast eye modeling or reconstruction from internet photos. We demonstrate the versatility of our model and fitting approach by reconstructing several different eyes ranging in size, shape, iris color and vein structure.

2 Related Work

Eye reconstruction has so far received only very little attention as a recent survey of Ruhland et al. [2014] shows. One of the first to model the eye were Sagar et al. [1994], who procedurally created eyes for surgery simulation. Their method however does not include any fitting to real data. Lefohn et al. [2003] mimic an ocularists workflow, where different layers of paint are applied to reproduce the look of an iris from a photograph. Their method is tailored to manufacture eye prosthetics, and only considers the synthesis of the iris color, neglecting its shape. François et al. [2009] synthesize the iris geometry from an input photograph using a dark-is-deep approach. More recently, Bérard et al. [2014] presented the first approach to capture the complete visible eye at high fidelity. They demonstrate that every eye is unique – not only in its texture but also in its shape. However, their approach requires a very involved, uncomfortable, and time consuming acquisition process, which limits its applicability. In this work we propose a method to achieve similar reconstruction quality as Bérard et al. [2014] but using a lightweight setup, which is both comfortable for the actor and can be readily integrated with traditional facial scanning pipelines, or even operate on a single image of the face.

Our work is related to facial capture methods, so we will start with a brief overview of these techniques, followed by a description of other methods that are related to our approach at a lower level. Specifically, the algorithms presented in this paper touch on various fields including non-rigid alignment to find correspondences between eye meshes, data driven fitting to adjust an eye model to a given eye mesh, as well as constrained texture and geometry synthesis to create missing details not present in the acquired images.

Facial Capture. Unlike eye reconstruction, the area of facial performance capture has received a lot of attention over the past decades, with a clear trend towards more lightweight and less constrained acquisition setups. The use of passive multi-view stereo [Beeler et al. 2010; Bradley et al. 2010; Beeler et al. 2011] has greatly reduced the hardware complexity and acquisition time required by active systems [Ma et al. 2007; Ghosh et al. 2011; Fyffe et al. 2011]. The amount of cameras employed was subsequently further reduced to binocular [Valgaerts et al. 2012] and finally monocular acquisition [Blanz and Vetter 1999; Garrido et al. 2013; Cao et al. 2014; Suwajanakorn et al. 2014; Fyffe et al. 2014]. To overcome the inherent ill-posedness of these lightweight acquisition devices, people usually employ a strong parametric prior to regularize the problem. Following this trend to more lightweight acquisition using strong parametric priors, we propose to leverage data provided by a high-resolution capture technique such as the one of Bérard et al. [2014] and build up a parametric eye-model, which can then be fit to input images acquired from more lightweight setups, such as face scanners, monocular cameras or even from artistically created images.

Non-Rigid Alignment. A vast amount of work has been performed in the area of non-rigid alignment, ranging from alignment of rigid object scans with low-frequency warps, noise, and incomplete data [Ikemoto et al. 2003; Haehnel et al. 2003; Brown and Rusinkiewicz 2004; Amberg et al. 2007; Li et al. 2008] to algorithms that find shape matches in a database [Kazhdan et al. 2004; Funkhouser et al. 2004]. Another class of algorithms registers a set of different meshes that all have the same overall structure, like a face or a human body, with a template-based approach [Blanz and Vetter 1999; Allen et al. 2003; Anguelov et al. 2005; Vlasic et al. 2005]. In this work we use a variant of the non-rigid registration algorithm of Li et al. [2008] in order to align multiple reconstructed eyes and build a deformable eye model [Blanz and Vetter 1999]. Although Li et al.'s method is designed for aligning a mesh to depth scans, we will show how to re-formulate the problem in the context of eyes, operating in a spherical domain rather than the 2D domain of depth scans.

Texture and Geometry Synthesis. In this work, texture synthesis is used to generate realistic and detailed iris textures and also geometry from low-resolution input images. A very broad overview of related work on texture synthesis is presented in the survey of Wei et al [2009]. Specific topics relevant for our work include constrained texture synthesis [Ramanarayanan and Bala 2007] and example-based image super resolution [Tai et al. 2010], which both aim to produce a higher resolution output of an input image given exemplars. With patch-based synthesis methods [Praun et al. 2000; Guo et al. 2001; Efros and Freeman 2001], controlled upscaling can be achieved easily by constraining each output patch to a smaller patch from the low-resolution input. These algorithms sequentially copy patches from the exemplars to the output texture. They were further refined with graph cuts, blending, deformation, and optimization for improved patch-boundaries [Kwatra et al. 2003; Mohammed et al. 2009; Chen et al. 2013]. Dedicated geometry synthesis algorithms also exist [Wei et al. 2009], however geometry can often be expressed as a texture and conventional texture synthesis algorithms can be applied. In our work we take inspiration from Li et al. [2015], who propose to use gradient texture and height map pairs as exemplars where in their work the height map encodes facial wrinkles. We expand on their method and propose to encode color, geometry and also shape deformation in a planar parameterization, allowing us to jointly synthesize texture, shape and deformation to produce realistic irises that allow dynamic pupil dilation.

3 Method Overview

Our main goal is to generate a parametric eye model that can be fit to sparse image data, leveraging a database of high-resolution eye reconstructions. Since eyes are composed of several different components and contain interesting variations at multiple scales, a single all-encompassing parametric model is not practical. For this reason we compose a model built from three separate components, namely an *eyeball model* (Section 5) that represents the low-frequency variability of the entire eyeball shape, an *iris model* (Section 6) that represents the high-resolution shape, color and pupillary deformation of the iris, and a *sclera vein model* (Section 7) that represents the detailed vein structure in the sclera, including the vein network and the width and depth of individual veins, as well as fine-scale geometric surface details. In Section 8 we show how the model parameters can be estimated by fitting the model to 3D face scans, single images, or even artistic portraits, drastically simplifying the process of creating 3D high-quality eyes.

4 Input Data

An eye database is provided by Bérard et al. [2014] who captured a set of 30 high-quality eyes. This database provides high-resolution meshes and textures for the white sclera and the colored iris (please refer to the schematic in Fig. 2). The iris geometry is provided as a deformation model making it possible to create meshes for an entire range of pupil dilations. The database contains eyes of different iris colors ranging from brown to green-brown to blue, and the high resolution geometry captures intricate eye-specific surface details. A subset of the database eyes are shown in Fig. 3. We assume that right and left eyes are anti-symmetric and we thus mirror the left eyes when building the model for the right eye. Similarly, a mirrored version of the model can be used to represent the left eye. The data provided contains also a limbus opacity mask defining the transparency transition from sclera to cornea, from which the position of the limbus can be extracted by mapping the 50 percent opacity level to the mesh.

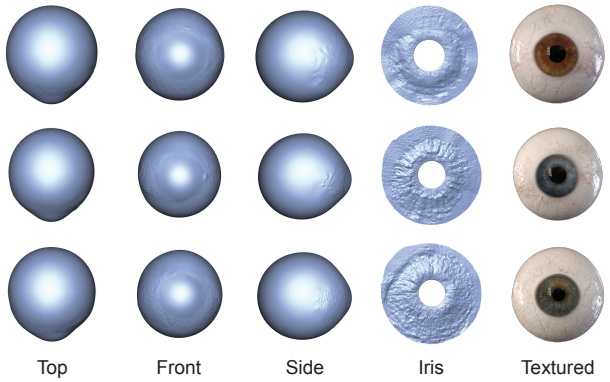

Figure 3: *We create a database of eyes captured by Bérard et al. [2014], which contains high-resolution meshes and textures for eyeball and iris. Notice how the geometric structure of the iris (4th column) is linked to its color (5th column), in that browner irises are smoother while bluer ones are more fibrous.*

5 Eyeball Model

The eyeball is represented by a morphable model [Blanz and Vetter 1999], which has been demonstrated to be a good representation to capture low-frequency variation. A morphable model is a linear combination of a set of samples. To avoid overfitting to the samples, the dimensionality is oftentimes reduced using methods such as principal component analysis (PCA). PCA computes the mean shape plus a set of mutually orthogonal basis vectors from the samples, ordered according to their variance. Truncating the dimensions with lower variance leads to a subspace that captures the major variation in the samples and is resilient to noise. In addition to the shape variation, our model also includes a rigid transformation for the eyeball as well as a uniform scale factor.

A morphable model requires all samples to be in perfect correspondence, which is unfortunately not the case for our database of eyes. In our case, eyeballs exhibit only few semantic features that can be used to establish correspondence. The most important one is the limbus, the boundary between the white sclera and the transparent cornea (Fig. 2). Other features are less salient, such as the overall asymmetry of the eye, but have to be encoded as well. These features, however, are not well defined and thus the traditional two step approach to build a morphable model by first establishing correspondences between all samples and then computing the model does not lead to satisfactory results.

Instead, we perform an iterative scheme that alternates between establishing correspondences and computing the model. The algorithm iteratively refines the model in three steps, first by fitting the previous guess of the model to the sample shapes, second by deforming this fit outside of the model in order to more closely fit the samples, and third by recomputing the model from these fits. Next we will discuss these three steps in more detail.

Step 1: Within-Model Fit. The eyeball model $\mathcal{M}$ is fit to a sample shape $\mathcal{S}$ by finding the model parameters $\mathbf{p}$ that minimize the energy

$$E_{model} = \lambda_{shape} E_{shape} + \lambda_{limbus} E_{limbus} + \lambda_{coeff} E_{coeff} \quad (1)$$

where the shape term

$$E_{shape} = \frac{1}{|\mathcal{M}|} \sum_{\mathbf{x}_i \in \mathcal{M}|_{\mathbf{p}}} \|\mathbf{x}_i - \chi(\mathbf{x}_i, \mathcal{S})\|^2 \quad (2)$$

penalizes the distance between points $\mathbf{x}_i$ on the model $\mathcal{M}$ evaluated at $\mathbf{p}$ and their closest points $\chi(\mathbf{x}_i, \mathcal{S})$ on the sample shape $\mathcal{S}$, and the limbus term

$$E_{limbus} = \frac{1}{|\mathcal{L}^{\mathcal{M}}|} \sum_{\mathbf{y}_i \in \mathcal{L}^{\mathcal{M}}|_{\mathbf{p}}} \left\| \mathbf{y}_i - \phi(\mathbf{y}_i, \mathcal{L}^{\mathcal{S}}) \right\|^2 \quad (3)$$

penalizes the distance between points $\mathbf{y}_i$ on the model limbus $\mathcal{L}^{\mathcal{M}}$ evaluated at $\mathbf{p}$ and their closest points $\phi(\mathbf{y}_i, \mathcal{L}^{\mathcal{S}})$ on the limbus of the sample shape $\mathcal{L}^{\mathcal{S}}$. The shape coefficients term

$$E_{coeff} = \frac{1}{k} \sum_{i=1}^{k} \left(\frac{c_i - \mu_i}{\sigma_i} \right)^2 \quad (4)$$

penalizes shape coefficients c_i far away from the mean coefficients of the current model $\mathcal{M}|_{\mathbf{p}}$, where μ_i and σ_i are the mean and the standard deviation of the i-th shape coefficient. The number of shape coefficients is k. We set the constants to $\lambda_{shape} = 1$, $\lambda_{limbus} = 1$, and $\lambda_{coeff} = 0.1$.

The parameter vector $\mathbf{p}$ consists of a rigid transformation, uniform scale, as well as an increasing number of shape coefficients as discussed in step 3.

Step 2: Out-Of-Model Fit. The morphable model $\mathcal{M}|_{\mathbf{p}}$ fit in the previous step will not match the sample $\mathcal{S}$ perfectly since it is constrained to lie within the model space, which has only limited degrees of freedom. In order to establish better correspondences for the next step, we therefore need to further deform the mesh non-rigidly to bring it out-of-model. We use a variant of the non-rigid deformation method proposed by Li et al. [2008], which was designed for aligning a mesh to depth scans using a deformation graph and a continuous approximation of the target shape. In our context, we wish to align the fitted model mesh to the database samples. We modify the method of Li et al. to operate in the spherical domain rather than the 2D depth map domain, and we add additional constraints to match both the limbus boundary and the mesh normals during deformation. We use two spherical coordinate parameterizations which are wrapped like the patches of a tennis ball so that the distortion in the domains is minimal. Closely following Li et al., the energy that is minimized by the algorithm can be expressed as the sum of the following terms:

$$E_{nonrigid} = \lambda_r E_r + \lambda_s E_s + \lambda_f E_f + \lambda_n E_n + \lambda_l E_l, \quad (5)$$

where E_r and E_s correspond to the $rigid$ and $smooth$ energies as defined in the original paper of Li et al. [2008]. We set the constants to $\lambda_{r,s} = 0.01$ and $\lambda_{f,n,l} = 1$. The shape and limbus energies E_s and E_l correspond to the ones used in the previous step as defined in Eq. 2 and Eq. 3, respectively. The normal energy E_n is defined analogously to the shape energy as the Euclidean difference between the normals of the model and the normals of the respective closest points on the sample shapes. The non-rigid deformation produces meshes $\{\tilde{\mathcal{M}}\}$ which closely resemble the database samples $\{\mathcal{S}\}$ but have the same topology as the eyeball model.

Step 3: Update Eyeball Model. From the non-rigidly aligned shapes $\{\tilde{\mathcal{M}}\}$ an updated version of the model is computed using PCA and keeping only the mean shape plus the k most significant dimensions. In order to be robust towards initial misalignment, the algorithm starts with a very constrained model that consists of the mean shape only ($k = 0$).

The proposed algorithm iterates these three steps and increases the dimensionality k of the model every 10 iterations by including the next most significant PCA vector. Increasing the dimensionality allows the model to better explain the data and by doing so gradually provides robustness. We use a fixed amount of iterations because the error is not comparable from one iteration to the other since the model has been updated at the end of each iteration. After expanding the model three times ($k = 3$), we found that the first mode of the deformable model accounts for 92 percent of the variance, the first two for 96 percent, and the first three for 98 percent of the variation, which covers the low-frequency variation we are targeting with this model. The final eyeball model thus contains 10 dimensions, six of which account for rigid transformation, one for uniform scale, and three for shape variation. Fig. 4 shows the deformation modes of our eyeball model.

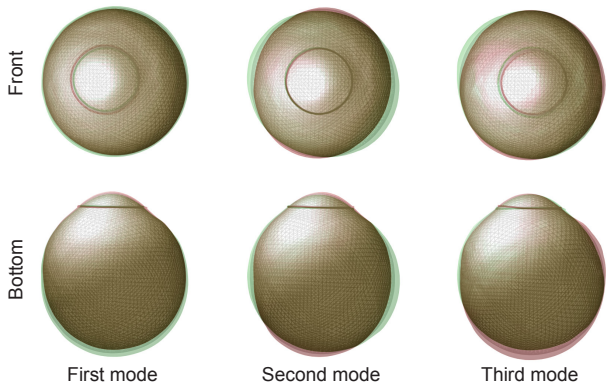

Figure 4: *This figure visualizes the three modes of our eyeball prior. For visualization purposes we show the normalized dimensions, scaled by a factor of 50. As the eyeball does not contain much variation, we found three modes to be sufficient as shape prior.*

6 Iris Model

We now turn our attention to the iris and describe our model for parameterizing the texture and geometry of an iris given the database of captured eyes. The iris is arguably the most salient component of the eye, and much of the individuality of an eye can be found in the iris. A large variety of irises exist in the human population, and the dominant hues are brown, green and blue. In addition to the hue, irises also vary greatly in the number and distribution of smaller features like spots, craters, banding, and other fibrous structures. Interestingly, iris color and geometry are related, as the iris color is a direct function of the amount of melanin present in the iris. Irises with little melanin have a blueish hue, where irises that contain more melanin become more brown. As reported by Bérard et al. [2014], this accumulation of melanin changes the geometric structure of the iris, with blueish irises being more fibrous and brown irises being smoother overall as shown in Fig. 3. We exploit the relationship between color and structure in our iris model and propose a single parameterization that will account for both.

Since irises have such a wide range of variation, it would be impractical to parameterize them using a Guassian-distributed PCA space as we do for the eyeball. Instead, we account for the variability by parameterizing the iris using a low-resolution control map, which represents the spatially varying hue and the approximate distribution of finer-scale features (see Fig. 5.b for an example control map). The control map will guide the creation of a detailed high-resolution iris through constrained texture synthesis, using the irises in the database as exemplars. The use of a control map is a very intuitive and convenient way to describe an iris, as it can be extracted from a photograph when reconstructing the eye of a specific person, or it can be sketched by an artist when creating fictional eyes. Based on the low-resolution control map, we propose a constrained synthesis algorithm to generate a detailed color texture in high resolution (Section 6.1), and then extend the synthesis to additionally create the geometric iris structure (Section 6.2).

6.1 Iris Texture Synthesis

Guided by the low-resolution control map our goal is to synthesize a high-resolution texture for the iris based on our eye database, similar to example-based super resolution [Tai et al. 2010] and constrained texture synthesis [Ramanarayanan and Bala 2007]. We achieve this by composing the high-resolution texture from exemplar patches from the database, following the well-studied image quilting approach introduced by Efros and Freeman [2001]. In our application the process is guided by the control map and selects suitable patches from the database ensuring they conform both with the control map and the already synthesized parts of the texture. Once the patches have been selected, they are stitched together using graph cuts and combined to a seamless texture using Poisson blending. Finally, this texture is merged with the intial control map in order to augment the low-res control map with high-res detail. Fig. 5 shows the individual steps, which we will describe in more detail in the following.

Patch Layout. The structure of an iris is arranged radially around the pupil. Operating in polar coordinates (angle/radius) unwraps the radial structure (Fig. 5) and presents itself well for synthesis with rectangular patches. We synthesize textures of resolution 1024x256 pixels with patch sizes of 64x64 pixels that overlap each other by 31 pixels in both dimensions. While iris features are distributed without angular dependency, they do exhibit a radial dependency since the structure close to the pupil (pupillary zone) differs substantially from the one closer to the limbus (ciliary zone). To synthesize a specific patch in the output, the algorithm can thus consider sample patches at any angle with similar radii ($\pm 10\%$). The only drawback of the polar coordinate representation is that the synthesized texture must wrap smoothly in the angular dimension (i.e. across the left and right image boundaries), which is handledby temporarily extending the texture by one block in the angular direction. To guarantee a consistent wrapping the first and last block are updated as pairs.

Output Patch Selection. The iris is synthesized by iteratively placing patches from the database iris textures. In each iteration, we first need to decide where to synthesize the next patch in the output texture. Many synthesis algorithms employ a sequential

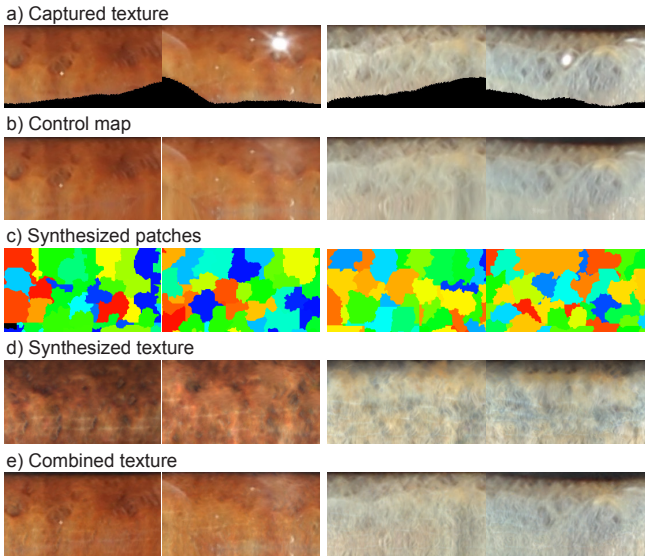

Figure 5: *Synthesizing an iris consists of capturing initial textures (a), from which control maps are generated by removing specular highlights (b). This control map is input to a constrained texture synthesis that combines irregular patches (c) from the database to a single texture (d), which is then filtered and recombined with the control map to augment the low-res control map with high-frequency detail (e). The figure shows close-ups from a brown iris on the left and from a blueish iris on the right.*

order, typically left to right and top to bottom. We found that this leads to unsatisfactory results since important features, such as spots or freckles, can easily be missed because neighboring patches in the output may provide a stronger constraint than the control map. Instead we select the next patch based on control map saliency, which synthesizes patches in visually important areas first, thus allowing them to be more faithful to the control map and spreading the control map residual error into less salient areas. Saliency is computed via steerable filters as proposed by [Jacob and Unser 2004].

Exemplar Selection. Once the location for the next patch to be synthesized has been determined, a suitable patch exemplar has to be retrieved from the database. This exemplar should be faithful to both the control map and any neighboring patches that have already been chosen. Similarity to the control map, denoted e_c, is computed as the mean squared error between a downscaled version of the exemplar and the patch of the control map. To gain invariance over differences in exposure and because the most important quantity at this stage is faithful color reproduction, the error is computed over the RGB channels, but the mean intensity of the exemplar is scaled globally to match the mean intensity of the control patch. Similarity to the already synthesized texture, denoted e_n, is computed as mean squared error over the overlapping pixels. The two similarity measures are linearly combined into a single quantity

$$e = \alpha e_n + (1 - \alpha) e_c, \tag{6}$$

where we use $\alpha = 0.25$ for all examples in this paper. The exemplar patch with the smallest error is chosen.

Patch Merging. The end result of the above steps is a set of overlapping patches that cover the entire texture. Even though the patches have been carefully selected they will still exhibit seams. To alleviate this we follow Kwatra et al [2003] and employ a graph cut to find seams between patches that better respect the underlying image structure, i.e. finding a seam that minimizes the color difference across the cut. For each patch, pixels at the boundary of the patch that overlap neighboring patches are labeled as sinks and the pixel at the center of the patch as a source. A graph for the current block is constructed with horizontal and vertical edges. The capacity of each edge is set to be the difference of the two connected pixels. We use the max-flow/min-cut algorithm of Boykov et al. [2004] to solve for the cut.

Patch Blending. Once merged, the texture has a unique color per pixel with minimal, yet still visible seams between patches. To completely remove the seams we employ Poisson blending [Pérez et al. 2003], setting the desired color gradients across patch seams to be zero while preserving color detail within patches.

Texture Blending. By definition, the synthesized texture T should well-match the control map C and contain more high frequency detail. However, the control map itself already contains a lot of structural information that we wish to preserve. Therefore we propose to blend the synthesized texture with the control map, however we need to take care not to superimpose the same frequencies. Assuming the captured image is in focus, the frequency content of the control map is determined by the resolution at which it was acquired. If we base our synthesis on a low resolution image that captures the complete face, then we want to add a larger range of spatial frequencies than if the original input image was focused onto the eye and hence of high resolution. To avoid superimposing the same frequency bands we thus need to bandpass filter the synthesized texture before blending with the control map. We model the bandpass filter as a Gaussian $\mathcal{G}$ with the standard deviation computed from the ratio in width of the synthesized texture T_{width} and the control map C_{width} as

$$\sigma = \frac{T_{width}}{C_{width}} \sigma', \tag{7}$$

where σ' is the standard deviation of the Gaussian at the resolution of the control map, typically set to 1px. In some cases it makes sense to pick a larger σ' to account for noise or defocus of the control map. The high-pass filtered texture and low-pass filtered control map are then combined as

$$T \leftarrow (T - \mathcal{G} * T) + \mathcal{G} * C, \tag{8}$$

and then re-converted from polar coordinates to create the final texture of the iris.

6.2 Iris Geometry Synthesis

As mentioned above, there is an inherent coupling between iris texture and geometric structure. The idea is thus to exploit this coupling and synthesize geometric details alongside the iris texture. The database provided by Bérard et al. [2014] contains both high-res iris textures and iris deformation models, which encode iris geometry as a function of the pupil dilation. Since the iris structure changes substantially under deformation, we aim to synthesize the geometry at the observed pupil dilation. In addition, the algorithm will also provide extrapolations to other pupil dilations, allowing us to control the iris virtually after reconstructing the eye.

Geometry Representation. The iris geometries in the database are encoded in cylindrical coordinates (angle/radius/height), which renders them compatible to the domain used for texture synthesis. Spatially, the iris deformation model is discretized such that it has one vertex per pixel of the corresponding texture, with full connectivity to its 8 neighbors. Temporally, the deformation model is discretized at four different pupil dilations, spaced equally to span

the maximum dilation range common to all exemplars. One of the pupil dilations is picked to match the dilation of the input image.

Synthesizing geometry cannot be performed using absolute spatial coordinates since patches are physically copied from one spatial location in the exemplar to another in the output texture. For this reason, we find it convenient to encode the geometry using differential coordinates that encode the difference in angle, radius and height between neighboring vertices, and then the synthesized geometry can be reconstructed. Specifically, for every vertex, the iris geometry is encoded by the eight differential vectors to its spatial neighbors (in practice, to avoid redundant storage we only need the four vectors corresponding to top, top-right, right, and bottom-right), plus three differential vectors forward in time, which we refer to as trajectories in the following. See Fig. 6 for a schematic. These seven differential vectors result in 21 additional dimensions that are added to the three color channels for synthesis.

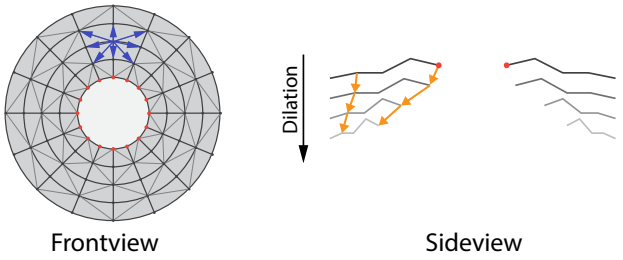

Figure 6: *The iris geometry is tesselated uniformly in the polar domain, yielding eight neighbors per vertex spatially (blue, left) as well as the forward neighbors in time (orange, right), which describe the trajectory a vertex follows during dilation. The trajectory is longest at the pupil and has to be properly scaled during synthesis.*

Trajectory Scaling. The synthesis algorithm can place patches at different radii than they were taken from in the exemplar. Even though this radius difference is limited to $\pm 10\%$, we still need to adjust for the fact that deformation trajectories closer to the pupil are longer than trajectories closer to the limbus (see Fig. 6 for a schematic explanation). Therefore, we scale the difference vectors of each trajectory by

$$\rho = \frac{r_l - r_{to}}{r_l - r_{from}}, \tag{9}$$

where r_{from} is the radius at which the patch is extracted and r_{to} the radius where it is placed. r_l is the limbus radius at which we assume no deformation.

Reconstruction. The synthesized differential vectors in the final iris texture are assembled to a linear Laplacian system for generating the final iris geometry similarly to Sorkine et. al [2004]. Since all vectors are relative, the system is under-constrained and we need to provide some additional constraints. The most natural choice is to constrain the positions of the pupil, which ensures a faithful fit to the observed pupil. Since the geometry is encoded in cylindrical coordinates, we need to scale the angular dimension (radians) to render it compatible with the radial (mm) and height (mm) dimensions. Thus, the angular dimension is multiplied by 5 mm, which corresponds to the average iris radius present in the database.

The result of this section is an iris model parameterized by a low-resolution control map, which allows high-resolution geometric and texture reconstructions using constrained synthesis given the database of eyes as exemplars.

7 Sclera Vein Model

Finally, we complete the eye by presenting a model for synthesizing the sclera. By far the most dominant features of the sclera are the veins, which contribute substantially to the visual appearance of the eye. Depending on the physical and emotional state, the appearance of these veins changes. For example they swell when the eye is irritated or when we are tired, causing the infamous "red eye" effect. Veins also travel under the surface at varying depths, and deeper veins appear thicker and softer, while veins at the surface appear more pronounced.

We propose to model the different states of veins with a parametric vein model. Such a model allows us to continuously change parameters and blend between different states. Also, besides modeling changes we can create additional detail not visible in the input data. Our model grows veins from seed points following a parameter configuration. In the following, we first describe our vein model and how the vein network is synthesized, and then describe how the synthesized veins are rendered to create the sclera texture, including a synthesized normal map to provide fine-scale surface details.

7.1 Vein Model

When scrutinizing the veins of a real sclera, one can see that they exhibit an enormous visual richness and complexity (see Fig. 7.d), caused by the superposition of a large number of veins with varying properties such as color, thickness, scale, shape, and sharpness. To resemble this complex structure, we model the sclera vein network as a forest, where an individual tree corresponds the the vein network generated from a single large vein. The large veins are the most salient structures when looking from afar, and will be referred to as *primary level* veins. These veins branches off into smaller second, third and lower level veins. Similar to L-Systems [Rozenberg and Salomaa 1976] and tree growing methods [Palubicki et al. 2009; Sagar et al. 1994] we create the vein network based on a set of rules, which control the vein properties described next.

Vein Properties. A single vein is represented by a series of control points, which are interpolated with a spline to provide a smooth and continuous curve in the texture domain. These positional control points govern the shape of the vein. Similarly, other spatially varying properties can also be discretized along this spline and interpolated when required. The properties we synthesize include position offsets along the curve normals, the vein thickness, the vein depth, which relates to its visibility, and vein branching points. Note that the discretization is independent per property, as some properties vary more quickly when traversing a vein network. To account for the irregularity present in nature, we define the properties with a certain amount of randomness. We employ two types of random functions, where one follows a Gaussian distribution $\mathcal{N}$, parameterized by the mean and standard deviation. The second one is a colored noise function $\mathcal{C}$, which is parameterized by the amplitude controlling the amount of perturbation and the spectral power density, which is controlled by the exponent in $1/f^x$ and specifies the noise color.

The position offsets are defined by the colored noise function $\mathcal{C}_{offset}$ in the range of pink ($x = 1$) and red ($x = 2$) noise. Thickness is specified at the starting point $\rho_{thickSeed}$, along with a decay factor $\rho_{thickDecay}$, again perturbed with a colored noise function ($\mathcal{C}_{thick}$). Depth is computed as an offset to a given average depth ρ_{depth}, created by adding colored noise ($\mathcal{C}_{depth}$). Finally, the locations of the branching points and the corresponding branching angles are determined by two Gaussian distributions, $\mathcal{N}_{branchPos}$ and $\mathcal{N}_{branchAngle}$, respectively.

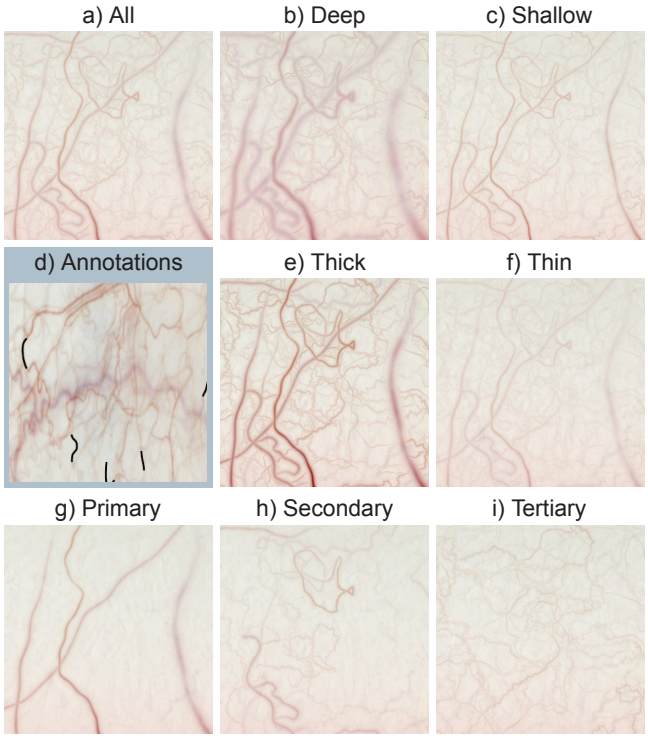

Figure 7: *Our parametric vein model allows the manipulation of the appearance of the veins (a) using a parameter for thickness and one for depth. The vein appearance is computed from an annotated exemplar texture (black segments in d), and our parametric vein model allows to independently manipulate depth (b,c) and thickness (e,f) to control the appearance. Veins are defined by different vein recipes for the three different level (g,h,i).*

Vein Recipes. Our model allows us to generate veins given a set of parameters (a *vein recipe*) describing the properties of the vein (width, depth, crippling, noise, etc.). Multiple veins can be synthesized with the same recipe. However, a single recipe does not account for the variation observed in nature. Therefore, we empirically create multiple vein recipes that describe veins belonging to different branching levels (primary, secondary, tertiary). . We found that a set of 24 recipes (10 primary, 6 secondary, and 12 tertiary) can produce vein networks of adequate visual complexity. In addition to the parameters described above, the recipes will also prescribe the parameters used for vein growing described below.

Vein Growing. Vein synthesis takes place in an unwrapped texture domain, with the limbus at the top and the back of the eyeball at the bottom. Veins on the sclera grow from the back of the eyeball to the front, and hence we grow them from bottom to top.

Growth is governed by a step size ρ_{step} and a direction $\mathbf{d}$ at every point. The step size is attenuated during growth by a decay factor $\rho_{stepDecay}$. The growing direction is influenced by three factors, 1 - a Gaussian distribution $\mathcal{N}_\beta$ that provides a general growth bias towards the top of the domain, 2 - a second Gaussian distribution $\mathcal{N}_\gamma$ that controls how much the vein meanders, and 3 - a repulsion term that discourages veins from growing over each other. The repulsion term stems from a repulsion map $\mathcal{R}$ that is computed while growing the veins, by rendering the veins into an image, indicating that a particular area has become occupied. The best growing angle can be computed with the distributions defined above as

$$\alpha = \max_{\alpha}(\mathcal{N}_\beta(\alpha)+\varepsilon)(\mathcal{N}_\gamma(\alpha)+\varepsilon)(1-\mathcal{R}(\mathbf{x}+\mathbf{d})+\varepsilon). \quad (10)$$

The direction $\mathbf{d}$ is computed from the angle α and current step size, and $\mathbf{x}$ denotes the current position. Also, $\mathcal{N}_\gamma$ is evaluated relative to the last growing angle. Since the terms could fully deactivate each other in pathological cases, we add a ε to the terms ($\varepsilon = 0.001$).

Vein Seeds. Veins start growing at seed points at the bottom of the texture for primary veins, or at branching points for higher levels, and growing is terminated if they reach a predescribed length or grow past the limbus. The primary vein seeds are generated at random positions at the bottom of the texture. We use 10 seeds in all our results. The seeds are generated sequentially. To prevent that two seeds are too close to each other we reject the seeds that are closer than 300 pixels. The final texture is 4096 by 2048 pixels.

7.2 Vein Rendering

Given a synthesized network of veins with spatially varying properties, we now render the veins into the texture using an appearance model learned from the database of eyes.

The appearance of a vein is influenced by many different factors, such as its diameter, how shallow it grows, its oxygenation, and others. The most important factor is the depth, which influences the color since the sclera has a higher absorption coefficient in the red wavelengths and as a consequence deeper veins appear blueish. The depth also influences the sharpness of the vein, since more subsurface scattering blurs out the cross-profile. Next to depth, thickness plays a central role since thin and thick veins are visually quite different. Note that the two parameters are not independent, since thin veins for example can only appear close to the surface as they would be washed out further in, and consequently have to be of reddish color. We use a data-driven approach to map depth and thickness to appearance, determined from exemplary textures in the eye database, as described in the following.

Cross-Section Model. We manually label 60 short vein segments in exemplary textures, which span the vein appearance. From these segments we sample cross-section profiles of the RGB space by fitting an exponential along the profile:

$$\mathbf{c}(r) = \mathbf{c}_{bkgnd} - \delta \exp\left(\frac{-\|r\|_1}{2\psi}\right), \quad (11)$$

where r is the distance from the labeled vein along the profile, in pixels. The fitting estimates thickness ψ, depth δ and background color $\mathbf{c}_{bkgnd}$ of these cross-sections. Subtracting the background from the cross-section will allow us to add the model to any background.

Given the synthesized thickness ψ and depth δ, we retrieve all samples with similar depth and thicknesses from the labeled veins, where similarity is computed as Euclidean $distances$ on normalized tickness and depth values. A similarity threshold th is set to 1.1 times the distance to the third nearest neighbour. The retrieved cross-profiles are scaled to match to the query parameters, and the final cross-profile used for rendering is computed as their weighted average, where the weights are set to $1 - \frac{distance}{th}$.

This model allows us to compute a cross-section for any pair of thickness and depth parameters. Finally, the cross-section model is evaluated for each pixel in the neighborhood of a vein with the local width and depth, and added to the backplate.

Backplate. Our vein model describes the vein network but not the background into which the veins are to be rendered. This background contains two components: the low frequency variation and the high-frequency structure of the sclera texture. The mid-frequency features are provided by the vein model.

The high-frequency component accounts for visual noise and imperfections. This high-frequency texture is created manually by copying sclera patches that contain no veins from the database textures. Since it does not contain any recognizeable structures we can employ the same high-frequency components for every eye.

The low-frequency component is extracted from the smoothed input images with the intent to match the perceived overall color variation. Since only parts of the sclera texture can be computed from the images, we extrapolate the low-frequency component of the sclera to the entire eyeball by fitting a smooth spline surface to the visible parts of the texture. The spline surface is cyclic in the horizontal direction so that the left and right border match seamlessly. We also constrain the bottom of the texture to a reddish hue since there is no data present at the back of the eyeball and visually eyes appear more red near the back.

The high- and low-frequency components are combined into a single backplate image, into which the veins are rendered. An example of a final vein texture is shown in Fig. 7 (a), which additionally shows the impact of the depth (b,c) and thickness (e,f) parameters.

Geometric Surface Details. The geometric surface details of the sclera are important for the visual appearance of the eye since these little bumps affect the shape of specular highlights. The bumps consist of a mix of random bumps and displacements that correlate with the positions of big veins. Thus, we create a normal map based on a combination of procedural noise and displacements that follow the thick veins to render all our results.

This concludes the parametric model, which is able to synthesize all visible parts of the eye, including the eyeball, the iris, and the veins.

8 Model Fitting

The model described in the previous sections allows us to create a wide range of realistic eyes based on a few parameters and an iris control map. In this section we describe how these parameters can be estimated automatically and how the required iris control map is extracted from various sources. We focus on two different use-case scenarios. In a first use-case, we demonstrate how the proposed method may be used to complement existing photogrammetric face scanners to augment the facial geometry that is inaccurate for the eye itself with high-quality eye reconstructions, and in a second use-case we show how our method can be used to compute eye geometry and textures from single, uncalibrated input images.

8.1 Multi-View Fitting

In the multi-view scenario we fit our eye model to a 3D face scan provided by a multi-view stereo (MVS) reconstruction algorithm. In this work we leverage the system of Beeler et al. [2010], but any other system that provides calibrated cameras and 3D geometry would also work. The MVS algorithm reconstructs the white sclera reasonably well since its surface is mostly diffuse, albeit at lower quality than skin due to strong specular reflections which result in a noisier surface. Here, our model will serve as a regularizer to get rid of the noise. Most other parts of the eye, such as the cornea or the iris, pose greater challenge to the system, as they are either invisible or heavily distorted. Here, our model will fully replace any existing 3D data and rely solely on the imagery to reconstruct geometry and texture. In the following we will describe fitting of the model to a single or even multiple face scans with different eye gazes simultaneously.

Eyeball Fitting. The input images are annotated by labelling the limbus (red), the pupil (black), the sclera (white), and the iris (green) as shown in Fig. 2. Manually labelling these features is quick and could potentially be automated with existing eye detection techniques. Based on the input mesh and the labels we estimate the parameters for each eye. Specifically, we estimate the rigid transformation, the scale, the coefficients of the deformable model, as well as the radius and position of the pupil, yielding a total of 14 unknowns for a single eye. The orientation of the pupil is constrained by our model to the average pupil orientation of the database. Fitting is based on four weighted energy terms, which form the total energy E_{total} to be minimized:

$$E_{total} = \lambda_s E_{\text{sclera}} + \lambda_l E_{limbus} + \lambda_p E_{pupil} + \lambda_c E_{coeff}. \quad (12)$$

The sclera energy term (E_{sclera}) penalizes the distance between the model mesh $\mathcal{M}$ and the sclera mesh $\mathcal{Z}$ from the face scan, and is defined as

$$E_{sclera} = \frac{1}{|\mathcal{Z}|} \sum_{\mathbf{x}_i, \mathbf{n}_i \in \mathcal{Z}} \left\| \langle (\mathbf{x}_i - \chi(\mathbf{x}_i, \mathcal{M})), \mathbf{n}_i \rangle \right\|^2, \quad (13)$$

where $\mathbf{x}_i$ are the sclera mesh points and their closest points on the model are $\chi(\mathbf{x}_i, \mathcal{M})$. Distance is only constrained along the normal $\mathbf{n}_i$, which allows tangential motion. The sclera mesh is segmented from the full face mesh using the sclera and limbus annotations.

The limbus energy term (E_{limbus}) penalizes the distance between the projection of the model limbus into the viewpoint and the limbus:

$$E_{limbus} = \frac{1}{|\mathcal{L}^{\mathcal{S}}|} \sum_{\mathbf{y}_i \in \mathcal{L}^{\mathcal{S}}} \left\| \mathbf{y}_i - \phi(\mathbf{y}_i, \mathcal{L}^{\mathcal{M}}) \right\|^2, \quad (14)$$

where $\mathbf{y}_i$ are the limbus annotations and their closest points to the projected model limbus are $\phi(\mathbf{y}_i, \mathcal{L}^{\mathcal{M}})$.

Similarly, the pupil energy term (E_{pupil}) penalizes deviation of the projected model pupil from the pupil annotations. Unlike the limbus energy, this energy has to take into account the refraction taking place at the cornea interface when projecting the pupil into the camera. For the refraction computation we use a continuous spline approximation of the cornea surface.

The last term corresponds to the coefficient term defined in equation 4. All terms are weighted equally, i.e. all lambdas are set to 1.

Since this is a highly non-linear energy, we optimize it iteratively following an Expectation-Maximization (EM) schema. In the E-step we recompute all the correspondences based on the current estimate of the model, and in the M-step we fix the correspondences and optimize for the parameters using the Levenberg-Marquart algorithm. Typically, the optimization converges in about 5 iterations.

Iris Control Map. The optimization above yields the eyeball geometry and a disk centered at the fitted pupil, which will serve as proxy to compute the iris control map. As this disk only approximately corresponds to the real iris geometry, each view will produce a slightly different iris texture. Since the cameras of the MVS system frame the full head and lack resolution in the eye area, we employ two zoomed in cameras to compute the iris texture. From the two, we manually select the one producing the sharpest texture as our primary view. The other view is used to inpaint the highlights only.

The algorithm computes a highlight probability using the method of Shen et al. [2009] for each view and combines the iris texture maps according to

$$C = \frac{C_p w_p + C_s(1 - w_p)w_s}{w_p + (1 - w_p)w_s}, \tag{15}$$

where C_p and C_s are the colors of the primary and secondary textures, and w_p and w_s are the highlight confidence maps. As discussed in Section 6, the resolution of the control map depends on the resolution of the input images. In our particular setup, the resolution of the control map in polar coordinates is 256x64 pixels.

Eye Pair Fitting. The properties of a pair of eyes are typically highly correlated, as was also shown by Bérard et al. [2014]. This correlation can be leveraged to reduce the dimensionality of the fitting task from naïvely 28 dimensions to 21. Since it is reasonable to assume that the eyes have a similar (but antisymmetric) shape we can use the same shape coefficients and scale for the second eye. Furthermore, the rigid transformation of the second eye is linked to the first and can be reduced from 6 to 3 degrees of freedom, one for the vergence angle and two for the inter-ocular vector. The remaining parameters are then pupil radius and position, which may differ between the two eyes.

Multi-Pose Fitting. If we assume that the shape of the eyeball is rigid, we can leverage multiple eye poses to better constrain the optimization. The shape coefficients and global scale, as well as the inter-ocular vector are then shared amongst all poses, as are the pupil positions. Fig. 8 shows an example of multi-pose fitting, where we jointly optimize the parameters based on three poses.

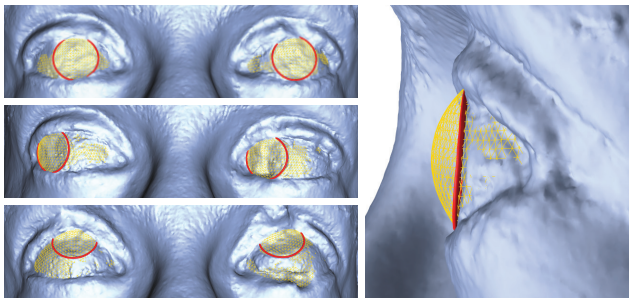

Figure 8: *We can leverage multiple eye poses to better constrain the fitting optimization. Here we fit simultaneously to three poses.*

8.2 Single Image Fitting

Fitting our eye model to a single image is much less constrained than the multi-view scan fitting since less data is available. Neither can we rely on depth information nor do we have multiple views to constrain the optimization. Still, by making some assumptions we are able to extract plausible model parameters for a given image.

The optimization for single image fitting is based on the same energy formulation as the multi-view case, described in Eq. 12, but since we do not have 3D information, the sclera term is removed. Thus the proposed method requires just limbus and pupil annotations, and relies stronger on the model prior. For example, we fix the scale of the eye to 1 due to the inherent depth/scale ambiguity in the monocular case. Furthermore, we rely on the position of the model pupil and optimize for pupil radius only. To project the limbus and pupil into the image, the method requires a rough guess of the camera parameters (focal length, and sensor size), which can be provided manually or extracted from meta-data (EXIF).

9 Results

In this section we will demonstrate the performance of the proposed method on a variety of input modalities, ranging from constrained multi-view scenarios to lightweight reconstruction from single images. Before showing fitting results, we will demonstrate the benefits of the parametric eye model for manipulation.

The appearance of the vein network in the sclera varies as a function of the physiological state of the person, leading to effects such as red eyes caused by fatigue. The proposed parametric vein model can account for such effects (and others) as shown in Fig. 9, where we globally change the depth at which veins grow from shallow (a) to deep (b), which influences their visibility, as well as the overall vein thickness from thick (c) to thin (d).

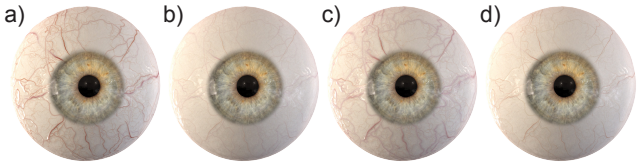

Figure 9: *Since we can parametrically control the sclera vein network and appearance, we can simulate physiological effects such as red eyes due to fatigue. Here we globally change the depth at which the veins grow from shallow (a) to deep (b), as well as their thickness from thick (c) to thin (d).*

Since we do reconstruct the complete dilation stack of an iris, the pupil size can be manipulated to account for virtual illumination conditions or to simulate some physiological effects, such as hippus, which is an oscillation of the pupil diameter. Fig. 10 shows three different irises at four different dilation stages. Our method nicely models the geometric detail that varies as the the pupil dilates (left to right). The three irises differ in color, ranging from brown (top) to blue (middle) to dichromatic (bottom). One can clearly see the different surface structure, which is inherently linked to the color, with brown irises being smoother and blueish more fibrous. Since our method generates the structure as a function of the iris color, one can indirectly control the structure by changing the color of the iris. In the special case of the dichromatic iris (bottom), the method produces structural details that vary spatially and match the color. The dichromatic iris corresponds to the right iris in Fig. 11.

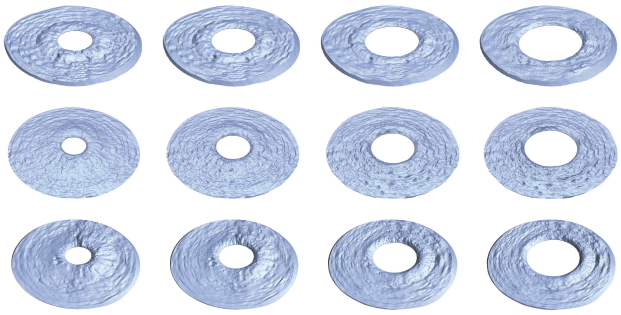

Figure 10: *Since geometric detail is inherently linked to the color of an iris, we can synthesize realistic microstructure, ranging from smooth for brown (top) to fibrous for blueish irisis (center). The bottom row shows a dichromatic iris that mixes gray-green and red-brown colors, which is clearly visible in the synthesized structure.*

Fig. 12 shows reconstruction results on a variety of different eyes, all captured in the multi-view setup and reconstructed using multi-view

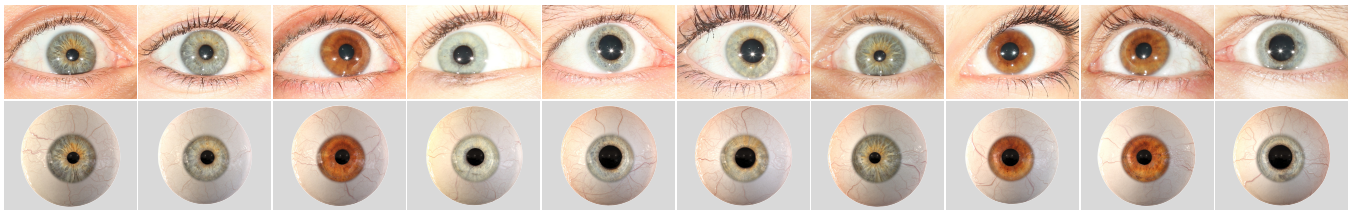

Figure 12: *Our method can reconstruct a variety of different eyes with varying eyeball shape, iris structure and color, and synthesize realistic scleras with vein textures and surface details.*

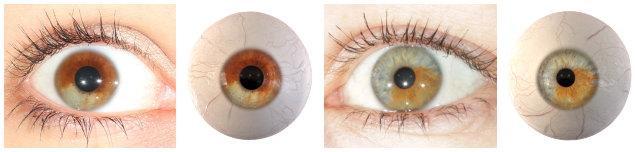

Figure 11: *Our method is able to reconstruct complex dichromatic irises by combining different color exemplars from the database.*

fitting. The eyes exhibit varying iris color and structure, eyeball shape and sclera vein networks. Since we operate on the same input data as multi-view face reconstruction algorithms, namely a set of calibrated images, our method seamlessly integrates with existing facial capture pipelines and augments the face by adding eyes, one of the most critical components, as can be seen in Fig. 1.

Fig. 13 demonstrates the robustness of our method. It shows the result of a single image fit and the effect of reducing the resolution of the input image by a factor of 35. Credible high-frequency detail missing in the low-resolution image is synthesized by our method to produce similar quality outputs.

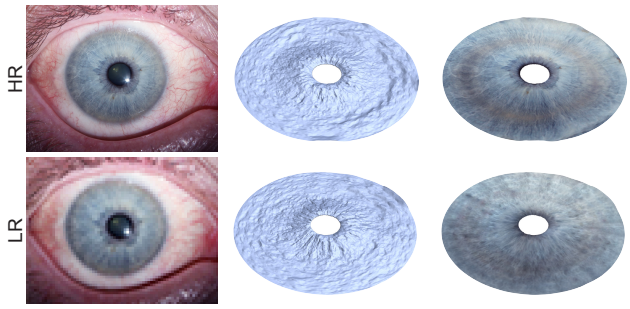

Figure 13: *We demonstrate the robustness of our method by fitting the eye model to a single high-resolution (HR) image and a low-resolution image (LR) obtained by down-sampling the first by a factor of 35. The figure shows the reference images (left), the reconstructed iris geometries (center), and the textured iris geometries (right).*

Fig. 14 shows a comparison of the method of Bérard et al. [2014] with our lightweight fitting approach for an eye not contained in the eye database. The results are generated from the same multi-view data from which also the image for the comparison in Fig. 13 stems. Despite fitting the model to just a single pose our approach produces results which are very close to the more laborious method of Bérard et al. [2014]. The mismatch of the back parts of the eyeballs is of little significance since neither of the methods produces an accurate reconstruction of these hidden parts.

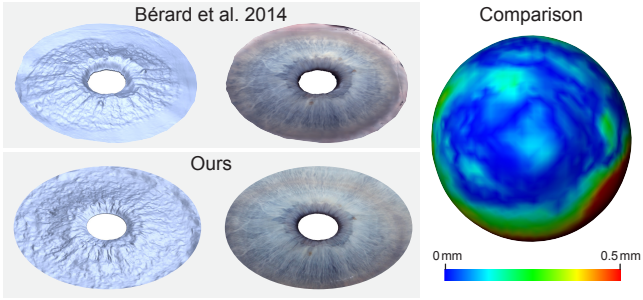

Figure 14: *Reconstruction comparison between Bérard et al. [2014] and our method. The figure shows the iris geometries (left) and textures (center) generated from the same multi-view data from which also the image for the comparison in Fig. 13 stems. The right side shows a comparison of the reconstructed eyeball meshes. The color map visualizes the error between the eyeball meshes of two methods.*

The computational effort required to reconstruct an eye is about 20 minutes. The most time intense parts are the iris synthesis and the reconstruction of the Laplacian system formed by the iris stack. Labelling a single image takes about 3 minutes, which is the only user input required.

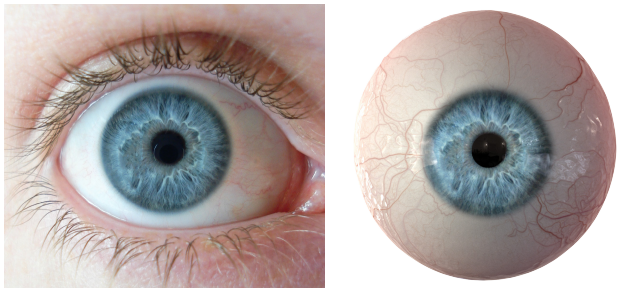

Figure 15: *The proposed method can fit eyes even to single images such as this one, opening up applications for eye reconstruction from internet photos. Source: [Wikimedia Commons 2006].*

Being able to reconstruct eyes from single images as shown in Fig. 15 provides a truly lightweight method to create high-quality CG eyes, not only from photographs but also from artistic renditions, such as sketches or paintings and even extending beyond human eyes, as shown in Fig. 16.

10 Conclusion

In this work we present a new parametric model of 3D eyes built from a database of high-resolution scans with both geometry and texture. Our model contains a shape subspace for the eyeball, a

Figure 16: *We show the robustness of our method by fitting eyes even to artistic paintings and single images of animals. Sources: [Wikimedia Commons 1887; Wikimedia Commons 1485; Flickr 2006].*

coupled shape and color synthesis method for the iris parameterized by a low-resolution control map, and a sclera vein synthesis approach also with tunable parameters to generate a variety of realistic vein networks. We also present an image-based fitting algorithm that allows our parametric model to be fit to lightweight inputs, such as common facial scanners, or even single images that can be found on the internet. Our parametric model and fitting approach allow simple and efficient eye reconstructions, making eye capture a more viable approach for industry and home use. Furthermore, the model allows to manipulate the captured data as it is fully parametric, such as changing the amount and appearance of sclera veins to simulate physiological effects or controlling the pupil size to have the eye react to synthetic illumination.

Our method is not without limitations. As can be seen in some of the single-image reconstructions, reflections off the cornea are difficult to identify and ignore and thus can become baked-in to the iris texture (see the bird example in Fig. 16). Additionally, our sclera vein synthesis does not guarantee to produce vein networks that match any partial veins that might be visible in the input images, which we plan to address in the future. Finally, our model is naturally limited by the size and variation of the input database, and since only a limited number of scanned high-quality real eyes are currently available, our results may not optimally match the inputs, but this will be alleviated as more database eyes become available. Nevertheless, even with these limitations our method provides a great starting point for CG artists to create realistic eyes from images.

Acknowledgements

We wish to thank Maurizio Nitti for the amazing eye renders and his endless patience. We would also like to thank all of our eye models, who made this work possible.

References

ALLEN, B., CURLESS, B., AND POPOVIĆ, Z. 2003. The space of human body shapes: reconstruction and parameterization from range scans. In *ACM Transactions on Graphics (TOG)*, vol. 22, ACM, 587–594.

AMBERG, B., ROMDHANI, S., AND VETTER, T. 2007. Optimal step nonrigid icp algorithms for surface registration. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, IEEE, 1–8.

ANGUELOV, D., SRINIVASAN, P., KOLLER, D., THRUN, S., RODGERS, J., AND DAVIS, J. 2005. Scape: shape completion and animation of people. In *ACM Transactions on Graphics (TOG)*, vol. 24, ACM, 408–416.

BEELER, T., BICKEL, B., SUMNER, R., BEARDSLEY, P., AND GROSS, M. 2010. High-quality single-shot capture of facial geometry. *ACM Trans. Graphics (Proc. SIGGRAPH)*.

BEELER, T., HAHN, F., BRADLEY, D., BICKEL, B., BEARDSLEY, P., GOTSMAN, C., SUMNER, R. W., AND GROSS, M. 2011. High-quality passive facial performance capture using anchor frames. *ACM Trans. Graphics (Proc. SIGGRAPH) 30*, 4, 75.

BÉRARD, P., BRADLEY, D., NITTI, M., BEELER, T., AND GROSS, M. 2014. High-quality capture of eyes. *ACM Trans. Graphics (Proc. SIGGRAPH Asia) 33*, 6, 223:1–223:12.

BLANZ, V., AND VETTER, T. 1999. A morphable model for the synthesis of 3d faces. In *Proc. of the 26th annual conference on Computer graphics and interactive techniques*, 187–194.

BOYKOV, Y., AND KOLMOGOROV, V. 2004. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 26*, 9, 1124–1137.

BRADLEY, D., HEIDRICH, W., POPA, T., AND SHEFFER, A. 2010. High resolution passive facial performance capture. *ACM Trans. Graphics (Proc. SIGGRAPH) 29*, 4, 41.

BROWN, B. J., AND RUSINKIEWICZ, S. 2004. Non-rigid range-scan alignment using thin-plate splines. In *3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004. Proceedings. 2nd International Symposium on*, IEEE, 759–765.

CAO, C., HOU, Q., AND ZHOU, K. 2014. Displaced dynamic expression regression for real-time facial tracking and animation. *ACM Transactions on Graphics (TOG) 33*, 4, 43.

CHEN, K., JOHAN, H., AND MUELLER-WITTIG, W. 2013. Simple and efficient example-based texture synthesis using tiling and deformation. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, ACM, 145–152.

EFROS, A. A., AND FREEMAN, W. T. 2001. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM, 341–346.

FLICKR. 2006. *Mohammed Alnaser - Mr. Falcon*. Creative Commons - Attribution 2.0. https://www.flickr.com/photos/69er/324313066.

FRANÇOIS, G., GAUTRON, P., BRETON, G., AND BOUATOUCH, K. 2009. Image-based modeling of the human eye. *IEEE TVCG 15*, 5, 815–827.

FUNKHOUSER, T., KAZHDAN, M., SHILANE, P., MIN, P., KIEFER, W., TAL, A., RUSINKIEWICZ, S., AND DOBKIN, D. 2004. Modeling by example. In *ACM Transactions on Graphics (TOG)*, vol. 23, ACM, 652–663.

FYFFE, G., HAWKINS, T., WATTS, C., MA, W.-C., AND DEBEVEC, P. 2011. Comprehensive facial performance capture. In *Eurographics*.

FYFFE, G., JONES, A., ALEXANDER, O., ICHIKARI, R., AND DEBEVEC, P. 2014. Driving high-resolution facial scans with video performance capture. *ACM Trans. Graphics 34*, 1, 8:1–8:14.

GARRIDO, P., VALGAERTS, L., WU, C., AND THEOBALT, C. 2013. Reconstructing detailed dynamic face geometry from monocular video. *ACM Trans. Graphics (Proc. SIGGRAPH Asia) 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 Trans. Graphics (Proc. SIGGRAPH Asia) 30*, 6, 129.

GUO, B., LIANG, L., LIU, C., SHUM, H.-Y., AND XU, Y. 2001. Real-time texture synthesis by patch-based sampling.

HAEHNEL, D., THRUN, S., AND BURGARD, W. 2003. An extension of the icp algorithm for modeling nonrigid objects with mobile robots. In *IJCAI*, vol. 3, 915–920.

IKEMOTO, L., GELFAND, N., AND LEVOY, M. 2003. A hierarchical method for aligning warped meshes. In *3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings. Fourth International Conference on*, IEEE, 434–441.

JACOB, M., AND UNSER, M. 2004. Design of steerable filters for feature detection using canny-like criteria. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 26*, 8, 1007–1019.

KAZHDAN, M., FUNKHOUSER, T., AND RUSINKIEWICZ, S. 2004. Shape matching and anisotropy. In *ACM Transactions on Graphics (TOG)*, vol. 23, ACM, 623–629.

KWATRA, V., SCHÖDL, A., ESSA, I., TURK, G., AND BOBICK, A. 2003. Graphcut textures: image and video synthesis using graph cuts. In *ACM Transactions on Graphics (ToG)*, vol. 22, ACM, 277–286.

LEFOHN, A., BUDGE, B., SHIRLEY, P., CARUSO, R., AND REINHARD, E. 2003. An ocularist's approach to human iris synthesis. *IEEE CG&A 23*, 6, 70–75.

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

LI, J., XU, W., CHENG, Z., XU, K., AND KLEIN, R. 2015. Lightweight wrinkle synthesis for 3d facial modeling and animation. *Computer-Aided Design 58*, 117–122.

MA, W.-C., HAWKINS, T., PEERS, P., CHABERT, C.-F., WEISS, M., AND DEBEVEC, P. 2007. Rapid acquisition of specular and diffuse normal maps from polarized spherical gradient illumination. In *Proc. Rendering Techniques*, 183–194.

MOHAMMED, U., PRINCE, S. J., AND KAUTZ, J. 2009. Visiolization: generating novel facial images. *ACM Transactions on Graphics (TOG) 28*, 3, 57.

PALUBICKI, W., HOREL, K., LONGAY, S., RUNIONS, A., LANE, B., MĚCH, R., AND PRUSINKIEWICZ, P. 2009. Self-organizing tree models for image synthesis. In *ACM Transactions on Graphics (TOG)*, vol. 28, ACM, 58.

PÉREZ, P., GANGNET, M., AND BLAKE, A. 2003. Poisson image editing. In *ACM Transactions on Graphics (TOG)*, vol. 22, ACM, 313–318.

PRAUN, E., FINKELSTEIN, A., AND HOPPE, H. 2000. Lapped textures. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 465–470.

RAMANARAYANAN, G., AND BALA, K. 2007. Constrained texture synthesis via energy minimization. *IEEE TVCG 13*, 1.

ROZENBERG, G., AND SALOMAA, A. 1976. *The mathematical theory of L systems*. Springer.

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 State of the Art Reports*, 69–91.

SAGAR, M. A., BULLIVANT, D., MALLINSON, G. D., AND HUNTER, P. J. 1994. A virtual environment and model of the eye for surgical simulation. In *Proceedings of Computer Graphics and Interactive Techniques*, 205–212.

SHEN, H.-L., AND CAI, Q.-Y. 2009. Simple and efficient method for specularity removal in an image. *Applied optics 48*, 14, 2711–2719.

SORKINE, O., COHEN-OR, D., LIPMAN, Y., ALEXA, M., RÖSSL, C., AND SEIDEL, H.-P. 2004. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, ACM, 175–184.

SUWAJANAKORN, S., KEMELMACHER-SHLIZERMAN, I., AND SEITZ, S. M. 2014. Total moving face reconstruction. In *Computer Vision–ECCV 2014*. Springer, 796–812.

TAI, Y.-W., LIU, S., BROWN, M. S., AND LIN, S. 2010. Super resolution using edge prior and single image detail synthesis. In *CVPR*.

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.

VLASIC, D., BRAND, M., PFISTER, H., AND POPOVIĆ, J. 2005. Face transfer with multilinear models. In *ACM Transactions on Graphics (TOG)*, vol. 24, ACM, 426–433.

WEI, L.-Y., LEFEBVRE, S., KWATRA, V., AND TURK, G. 2009. State of the art in example-based texture synthesis. In *Eurographics 2009, State of the Art Report, EG-STAR*, Eurographics Association, 93–117.

WIKIMEDIA COMMONS. 1485. *Sandro Botticelli - The Birth of the Venus*. Public Domain. https://commons.wikimedia.org/wiki/File:Venus_botticelli_detail.jpg.

WIKIMEDIA COMMONS. 1887. *Vincent Van Gogh - Self-Portrait*. Public Domain. https://commons.wikimedia.org/wiki/File:VanGogh_1887_Selbstbildnis.jpg.

WIKIMEDIA COMMONS. 2006. *Blue Eye Image*. GNU Free Documentation License Version 1.2. https://commons.wikimedia.org/wiki/File:Blueye.JPG.