Learning an appearance-based gaze estimator from one million synthesised images

Erroll Wood[1] Tadas Baltrušaitis[2] Louis-Philippe Morency[2] Peter Robinson[1] Andreas Bulling[3]

[1]University of Cambridge, United Kingdom {erroll.wood,peter.robinson}@cam.ac.uk
[2]Carnegie Mellon University, United States {tbaltrus,morency}@cs.cmu.edu
[3]Max Planck Institute for Informatics, Germany bulling@mpi-inf.mpg.de

Abstract

Learning-based methods for appearance-based gaze estimation achieve state-of-the-art performance in challenging real-world settings but require large amounts of labelled training data. Learning-by-synthesis was proposed as a promising solution to this problem but current methods are limited with respect to speed, the appearance variability as well as the head pose and gaze angle distribution they can synthesize. We present *UnityEyes*, a novel method to rapidly synthesize large amounts of variable eye region images as training data. Our method combines a novel generative 3D model of the human eye region with a real-time rendering framework. The model is based on high-resolution 3D face scans and uses real-time approximations for complex eyeball materials and structures as well as novel anatomically inspired procedural geometry methods for eyelid animation. We show that these synthesized images can be used to estimate gaze in difficult *in-the-wild* scenarios, even for extreme gaze angles or in cases in which the pupil is fully occluded. We also demonstrate competitive gaze estimation results on a benchmark in-the-wild dataset, despite only using a light-weight nearest-neighbor algorithm. We are making our UnityEyes synthesis framework freely available online for the benefit of the research community.

Keywords: appearance-based gaze estimation, learning-by-synthesis, 3D morphable model, real-time rendering

Concepts: •**Computing methodologies** → *Real-time simulation; Rasterization;* Learning settings;

1 Introduction

Appearance-based methods have significant potential to address the limitations of model and feature-based gaze estimation in unconstrained daily-life settings, particularly with respect to robustness and speed. Early work aimed to learn appearance-based gaze estimators from few samples [Lu et al. 2011] but this approach does not generalize well to *in-the-wild* situations. Such situations are characterised by unconstrained lighting conditions and significant variability in head poses and eye region appearances. State-of-the-art appearance-based methods therefore use large amounts of training data. For example, the TabletGaze dataset contains 100,000 images of people looking at a tablet screen in a lab setting [Huang et al. 2015] while MPIIGaze contains nearly 214,000 images collected over several months during everyday laptop use [Zhang et al. 2015]. However, despite their size, current datasets are still limited in the head pose and eye region appearance variability that they cover. In addition, collecting such large amounts of ground truth annotated training data is time-consuming and costly.

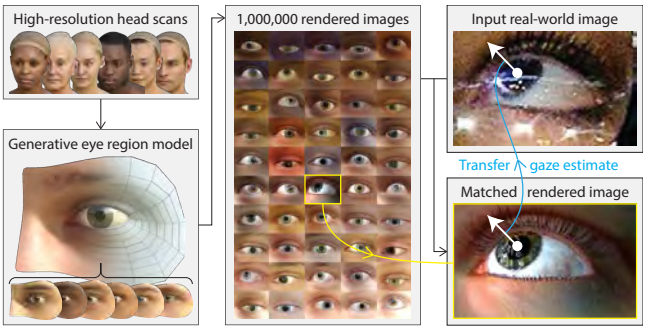

Figure 1: *We rendered one million realistic images of eyes using our generative eye region model. These are matched to an input image using a nearest-neighbor approach for gaze estimation. Our model manages to find good matches even with extreme gaze angles and glare from glasses.*

Learning-by-synthesis addresses this problem by replacing the manual collection of training data with an automatic procedure for synthesizing eye region images using 3D models generated using a calibrated multi-camera system [Sugano et al. 2014]. Zhang et al. used synthesised data to pre-train a multimodal convolutional neural network and fine-tune it using real images for appearance-based gaze estimation in the wild. This approach has demonstrated state-of-the-art performance for the most challenging and practically relevant task – device and person-independent gaze estimation [Zhang et al. 2015]. Recent work explored a fully synthetic approach by rendering perfectly-labelled eye images with illumination variation to pre-train the network [Wood et al. 2015]. No manual collection of images for defined gaze and head pose ranges was necessary in this case. However, their method had limited appearance variability from only ten participants and limited scale with rendering time of 5.26s/image. At this rate it would take two months of rendering to generate a dataset of a million images – too long for a practical development cycle.

We present UnityEyes, a novel method for rapidly synthesizing large numbers of variable eye region images for training data (see Figure 1). Our method combines a novel generative 3D model of the human eye region with a real-time rendering framework. The eye region model is derived from high-resolution 3D face scans and uses real-time approximations for complex eyeball materials and structures, as well as anatomically inspired procedural geometric methods for eyelid animation. We synthesize images using a rasterizing renderer and image-based lighting for realistic and varied illumination conditions. These synthesized images can be matched

ETRA '16, March 14 - 17, 2016, Charlston, SC, USA
ISBN: 978-1-4503-4125-7/16/03
DOI: http://dx.doi.org/10.1145/2857491.2857492

to real-world input images using nearest-neighbor approaches to estimate gaze. The UnityEyes framework will be made publicly available for the benefit of the research community.[1] The specific contributions of this work are three-fold: First, we introduce a novel statistically-derived generative 3D model of the eye region for increased appearance variation. Though several full-face morphable models already exist, this is the first detailed eye region morphable model of its kind. Second, we describe a rendering framework for rapidly synthesizing eye images $200\times$ faster than previous work. This allows us to rapidly generate a large number of images and thereby densely sample from the gaze direction and corresponding eye region appearance distributions. Third, we demonstrate the importance of covering this variability in the training data by showing competitive performance for device and person-independent appearance-based gaze estimation, despite only using a light-weight k-Nearest-Neighbor classifier.

2 Related work

Our work is primarily related to two types of previous work: 1) statistically derived 3D morphable face models, and 2) learning-by-synthesis for gaze estimation. Realistic eye-region rendering is also very important for the entertainment industry, and recent work shows that their complex structure and movements can be captured in high detail [Bermano et al. 2015; Bérard et al. 2014]. These methods however are focussed on re-creating individual eye-regions, not the generation of models with variety.

3D morphable face models

Morphable face models are used for a wide range of computer vision problems because they provide pose and illumination invariance – two yet-outstanding challenges for gaze estimation. This is done by modelling the shape and texture of 3D faces, as well as the process of image formation itself. Blanz and Vetter built the first such *morphable model* using a set of 200 laser-scanned 3D faces. Morphable face models are statistically derived 3D models that represent a face as a linear combination of 3D basis faces. They showed how it could be used to reconstruct a 3D face by matching the morphable model to an image, following manual initialization [Blanz and Vetter 1999]. More recently, Paysan et al. published the Basel Face Model – an improved open source generative 3D face model, and showed how it could be used for illumination invariant face recognition [Paysan et al. 2009]. It has since been used in a number of systems, including learning-by-synthesis for head pose estimation [Fanelli et al. 2013], real-time facial performance capture [Li et al. 2013], and learning a person-specific face model for gaze estimation [Mora and Odobez 2012].

A limitation of these models is that they only represent neutral face-shape so do not capture facial motion, e.g. eyelid movement during vertical saccades. To capture facial motion, Vlasic et al. developed a multilinear model that separately parameterizes differences in identity, expression, and viseme [Vlasic et al. 2005], and showed how it could be used for face transfer – mapping expressions from one person onto another. This system however does not handle geometry for the most complex parts of the face, including the eyes and eyelids – these were transferred by blending textures only. Cao et al. published a similar open source multilinear morphable model, but used smoothed data from a commodity depth camera, so could not accurately capture eye region detail.

These morphable models exclude eye region details because scanning equipment cannot accurately capture the surface of the eyeball due to its transparent material. Therefore they model the face and eyes as a single surface. This is a critical limitation as in reality the eyeballs are separate organs and can move independently, and should thus be modelled separately.

[1] http://www.cl.cam.ac.uk/research/rainbow/projects/unityeyes/

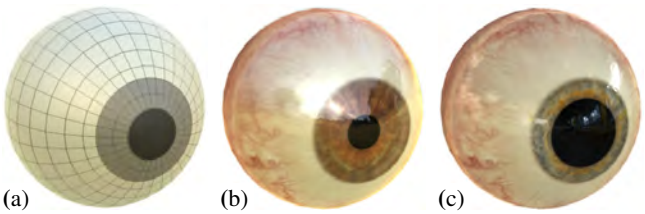

Figure 2: *Our eyeball mesh (a) shown rendered with physically-based materials and refraction effects. We model pupillary contraction (b) and dilation (c) as part of the refraction shader.*

Learning-by-synthesis for gaze estimation

The success of learning-based approaches depends on how well the training data covers the test data. Ideally, we would capture enough training images of eyes to completely cover all possible test cases, but this is impractical. To collect eye images, participants must be called into a laboratory or hired over a period of time, limiting environment and participant variability. Additionally, the range of gaze directions is then limited by practical matters, e.g. the size and placement of the screen used for gaze markers.

Instead, researchers have synthesized training data for eye tracking problems. Lu et al. synthesized eye images for head pose-free gaze estimation. Rather than use 3D graphics techniques, they synthesized eye images by deforming captured images with 1D pixel displacements [2012]. While applicable to head motion, this simple synthesis approach cannot be extended to improve person or environment variation. Sugano et al. published the UT Multi-view Gaze dataset that used reconstructed 3D eye region models captured using eight synchronized webcams [2014]. They showed improved results from synthesizing dense training data, but used only simplified graphics techniques ignoring illumination or material effects. Additionally, their 3D models were low resolution and failed to accurately reconstruct the eyeball due to its complex material. Recently, Wood et al. built a collection of dynamic eye-region models from high quality 3D head scans which could be controlled to determine eye gaze [2015]. They rendered realistic eye images using a path-tracer with physically-based materials and varying illumination. They trained a deep neural network [Zhang et al. 2015] with these images and showed state-of-the-art results for cross-dataset appearance-based gaze estimation in-the-wild. However, their training data had limited appearance variation from only ten separate participant models, and their preparation method required manual animation for each participant – a time-consuming task. In addition, their renderer was not real-time, limiting the number of images they could synthesize in a reasonable time.

3 Approximate eyeball model

The orientation of the eyeball determines gaze, so it must be present in training data for gaze estimation. Our aim was to synthesize large amounts of varied training data, so in this section we describe techniques for efficiently rendering realistic eyeballs, and modelling shape and texture variation. Though it is one of the most complex organs in the body, we modelled it using a single 3D mesh. This mesh corresponds to the eye's external surface (Figure 2a), and its shape is defined by two spheres representing the sclera ($r = 12$mm) and the cornea ($r = 8$mm) [Ruhland et al. 2014].

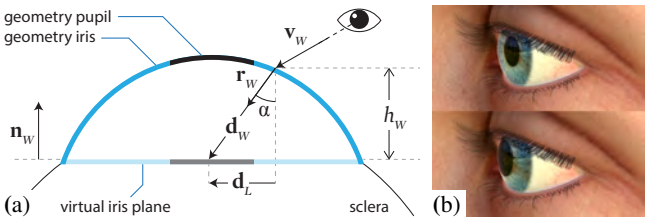

Figure 3: *We model iris refraction by altering texture look-ups. In (a), a viewed pixel is refracted correctly to show black (pupil) instead of blue (geometry surface). Example renders with (top) and without (bottom) refraction are shown in (b).*

Physically based refraction

In reality, the iris is a flat disk of muscle that appears distorted through the refractive corneal bulge. This phenomenon is particularly apparent when the eye is viewed at an angle (Figure 3b), so it was important to model it. Refraction is easy to simulate using ray-tracing, but is not part of the rasterization pipeline. We therefore developed physically correct refractive effects using a *fragment shader* – a GPU program that processes each pixel in a graphics pipeline [Shirley et al. 2009]. We followed previous work [Jimenez et al. 2012], and altered each texture look-up with a calculated texture-space offset $\mathbf{d}_L$. For each pixel on the surface of the cornea:

$$\text{refracted pixel color} = \text{EyeTexture}(\mathbf{uv} + \mathbf{d}_L) \quad \text{where} \tag{1}$$

$$\mathbf{d}_L = \left(\mathbf{M}^{-1}\mathbf{d}_W\right)_{xy} \quad \text{and} \quad \mathbf{d}_W = \frac{h_W}{-\hat{\mathbf{n}}_W \cdot \hat{\mathbf{r}}_W} \tag{2}$$

$\mathbf{uv}$ is the original texture coordinate, $\mathbf{M}$ is the eyeball model transform, $\mathbf{d}_W$ is the world-space offset, h_W is the virtual height between cornea and iris, $\hat{\mathbf{n}}_W$ is the eyeball's gaze direction, and $\hat{\mathbf{r}}_W$ is the refracted view direction through the iris (Figure 3). This allowed us to efficiently simulate the complex multi-layered structure of the eyeball using a single 3D mesh.

Varying eyeball shape and texture

Eyeballs vary both in shape (iris width, pupillary contraction and dilation) and texture (iris color). We varied iris width by scaling iris boundary vertices about their 3D center. Pupil size was varied using the fragment shader, by scaling the texture-space offset $\mathbf{d}_L$ to simulate radial expansion or contraction of the iris. We extracted a collection of iris textures from photographs, and randomly choose between them at run-time. These textures were used with the renderer's built-in *physically based shader*[2] that models how light behaves in reality to achieve consistent effects under various lighting conditions. The eyeball is wet so reflects the environment and surrounding eye area. These reflections can affect gaze estimation algorithms, so training data should include such examples.

4 Generative eye region model

Eye images also include nearby parts of the face, so we built a generative model for synthesizing realistic eye regions. This is important for a resulting gaze estimation system to work across users of different gender, age, and ethnicity. Our aim was to produce a model that can represent both 1) the large variety in facial shape and appearance, and 2) eyelid motion during eyelid saccades . For the former we generated novel 3D eye regions using a *morphable shape model* [Blanz and Vetter 1999] – a paramaterized linear model of 3D shape, and for the latter we used anatomically-inspired procedural geometric methods.

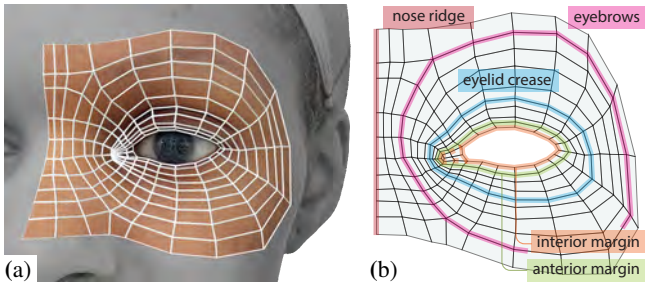

Figure 4: *(a) shows our generic eye region topology (229 vertices) over a raw scan ($\sim 5M$ vertices). (b) shows the topology in uv texture-space with important edge loops highlighted.*

Head scan registration and retopology

In order to build a generative morphable shape model, we needed 3D data of the facial eye region. We followed previous work and used high resolution 3D head scans captured by a professional photogrammetry studio[3] (10K diffuse color textures, 0.1mm resolution geometry) [Wood et al. 2015]. Scanning time is critical for capturing high-quality detailed data – while previous work took $\sim$ 1s to scan each face [Paysan et al. 2009], these models were captured in under $1/10,000$th of a second. We acquired 20 scans (5 female) covering different ages, eye shapes, bone structure, and skin tone.

Before we can build a model of 3D shape variation, the raw scan data must be brought into correspondence. To do this, we re-parameterized the original high-resolution mesh so that semantically identical points (e.g. points along the interior margin or nose ridge) shared the same space in a lower resolution domain or topology. Previous work retopologized each head-scan separately, resulting in N new topologies for N initial meshes [Wood et al. 2015]. Instead, we registered eye regions of varying shape with a single generic eye region topology – this can be seen in Figure 4. We carefully designed this generic topology so the edge loops would match the real life anatomic structure, e.g. the oculus orbicularis, whilst also faithfully capturing the original shape. These edge loops allow more realistic animation as mesh deformation matches that of real flesh and muscles [Orvalho et al. 2012]. We manually positioned the topology over the original scan in order to be as accurate as possible, and then transferred across color and displacement maps.

Morphable eye region model

Following registration, the shape of each eye region is represented by our generic eye region topology and can be expressed as a $3n$ dimensional vector:

$$\mathbf{s} = [x_1, y_1, z_1, x_2, ...y_n, z_n]^T \in \mathbb{R}^{3n} \tag{3}$$

Where x_i, y_i, z_i represent the 3D position of the ith vertex. We constructed a linear model $\mathcal{M}_s$ using *principal component analysis* (PCA) on our set of m retopologized scans. PCA uses singular value decomposition to extract an ordered set of orthogonal basis functions $\mathbf{U} \in \mathbb{R}^{3n \times m}$ that best describe the data. We then fit a Gaussian $\mathcal{N}_i(0, \sigma_i)$ to each of the m PCA basis functions using the original data resulting in a parametric shape model:

$$\mathcal{M}_s = (\boldsymbol{\mu}, \boldsymbol{\sigma}, \mathbf{U}) \tag{4}$$

[2] http://docs.unity3d.com/Manual/Shaders.html

[3] http://www.3dscanstore.com/

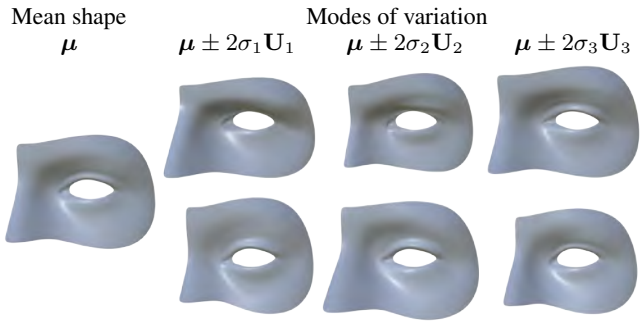

Figure 5: *The mean 3D shape along with the first three modes of variation extracted by PCA. The first mode $\mathbf{U}_1$ captures the difference between a hooded eye and a protruding eye.*

Figure 6: *Four example textures in uv space and rendered with mean shape $\boldsymbol{\mu}$, ranging from the darkest skin tone to the lightest.*

where $\boldsymbol{\mu} \in \mathbb{R}^{3n}$ is the average 3D shape and $\boldsymbol{\sigma} = [\sigma_1, \sigma_2, ...\sigma_m]$ describes the Gaussian distribution of each basis function. New eye region shapes $\mathbf{s}$ can then be generated from basis function coefficients $\boldsymbol{\alpha} \in \mathbb{R}^m$ as follows:

$$\mathbf{s}(\boldsymbol{\alpha}) = \boldsymbol{\mu} + \mathbf{U}\,\mathrm{diag}(\boldsymbol{\sigma})\;\boldsymbol{\alpha}^T \tag{5}$$

where α is described by m independent random Gausian variables with zero mean and unit variance.

Previous work on 3D morphable models also built a similar linear model for skin albedo [Blanz and Vetter 1999]. We investigated this approach, but found we did not have enough samples to derive a realistic generative texture model, especially considering the high dimensionality of the textures. Instead we randomly chose a single high-resolution texture at run-time, thus ensuring the eye region appeared realistic (Figure 6).

Procedural methods for eyelid movement

When the eyeball moves, the eyelids move with it. This is most prominent during vertical saccades which are always accompanied by lid movement. As our shape model represents neutral gaze (0° eyeball pitch and yaw), we had to deform the mesh to follow eyeball rotation. In reality, a muscle controls upper lid movement via tendons connected to the lid [Evinger et al. 1991]. Its general movement can be described as a rotation, but different parts of the lid have different rotational axes [Malbouisson et al. 2005]. We used our carefully designed topology and modelled the rotation of the jth vertex in the ith edge loop $\mathbf{v}_{ij}$ as follows:

$$\mathbf{v}'_{ij} = \mathbf{R}\left(\mathbf{a}_i, \mathbf{p}_{ij}, \theta_{ij}\right)\mathbf{v}_{ij} \tag{6}$$

Where $\mathbf{R}\left(\mathbf{a}, \mathbf{p}, \theta\right)$ describes a rotation of θ around axis $\mathbf{a}$ about pivot $\mathbf{p}$. Each edge loop's rotational axis $\mathbf{a}_i$ is defined as the offset between its eye corner vertices, and pivot $\mathbf{p}_{ij}$ interpolates between

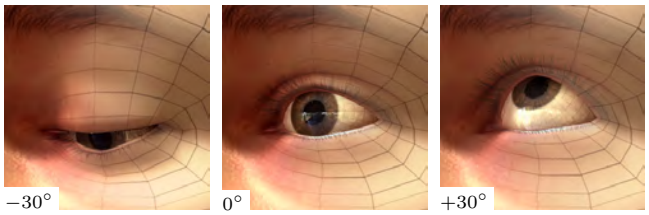

Figure 7: *We use anatomically inspired procedural geometric methods to animate eyelid, avoiding the need to manually rig the model. Shown are renderings for eyeball pitch at 0° and $\pm 30^\circ$.*

eye corners and eyeball center to ensure vertices near the eye corners are not displaced too far. Angle θ is defined separately for upper and lower lids using measurements taken from an empirical study [Malbouisson et al. 2005], and decays for the outer edge loops to simulate elastic stretching of the surrounding skin and flesh. In this way, we use the edge loops to model lid movement procedurally, avoiding manual animation [Wood et al. 2015] or more expensive mechanical-based techniques [Miller and Pinskiy 2009].

As our eye region mesh is derived from raw scans, there is no guarantee that it will touch the eyeball, and in some cases it may intersect it. Therefore we *shrinkwrapped* the inner edge loops to the eyeball surface, filling in any gap between mesh and eyeball. The innermost edge loop (the interior margin) is projected directly onto the eyeball surface, while the surrounding two loops are projected with an offset, simulating skin thickness. These collision calculations are fast as we can approximate the eyeball mesh with two sphere primitives. Finally, vertices in outer edge loops are displaced towards their inner edge loop neighbours to simulate skin elasticity and avoid discontinuous deformation.

Eye region details

We added two parts to improve realism: eye wetness and eyelashes. A thin layer of tear fluid covers the eyeball and bunches up near the eyelid. This can create strong specular highlights (see Figure 8c) around the edge of the eyelids. We procedurally deformed an eye wetness mesh to follow the interior margin (red geometry in Figure 8b), and shaded it with a transparent smooth material. In an image, eyelashes appear as dark edges around the eye and can provide a visual cue when someone is looking downwards, so we felt it was important to include this in training data. We modelled them using directed particle effects – hair particles start at the eyelid boundary and grow outwards away from the eyeball, curling up or down depending on the eyelid. We iteratively checked for collisions with the face geometry during hair growth, and redirected hair particles to avoid clipping. Computing hundreds of hair strands is an expensive operation, so we instead grew only ten guide hairs, and fitted smoothed eyelash geometry to them (Figure 8b). This geometry was then textured with a semi-transparent eyelash image.

Skin has a complex structure that appears soft as light scatters through its layers. Simple shading approaches cause skin to look hard, so we used specialized graphics techniques to improve realism. Previous work used a physically-based subsurface scattering material to simulate rays of light as they enter and exit skin [Wood et al. 2015] – this is accurate but slow. We instead used a *pre-integrated* skin shader [Penner and Borshukov 2011] – this technique pre-computes the scattering effect of light through skin for different curvatures, and efficiently applies it using a fragment shader (Figure 9). As our eye region mesh comprises 229 vertices, its edges can appear sharp, especially around the eyeball itself. Therefore, we finally smoothed the mesh with Loop subdivision

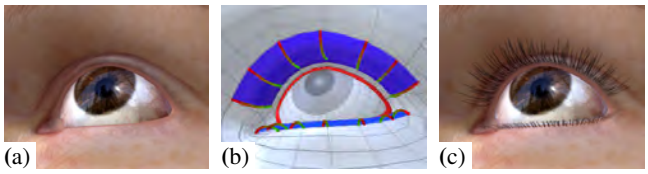

Figure 8: *We include eyelashes and eye wetness for realism. (a) shows a render without these, (b) shows eye wetness (red) and eyelash geometry (blue), (c) shows the final render.*

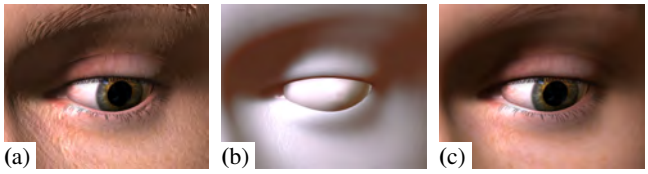

Figure 9: *We use pre-integrated skin shading for realism (c). Without it, skin appears too hard (a). (b) shows the scattered light through skin – this causes the skin to appear soft.*

[Loop 1987], using pre-computed weights for efficiency.

5 Image synthesis

Our goal was to rapidly create large, realistic, and varied datasets of eye images. Though our generative eye region captures 3D shape variation, in-the-wild images also exhibit variation depending on pose and environment. In this section we describe the software we used for synthesizing datasets much faster than previous work, as well as how we parameterized the scene to produce varied images.

Rendering our models

We used the *Unity 5.2*[4] game engine to render our eyeball and generative eye region model. Our contribution here is a massive speed-up in rendering time compared to previous work. This allows us to easily generate datasets several orders of magnitude larger than before – an important factor in successfully training large-scale learning systems [Zhang et al. 2015]. Sugano et al. wrote their own CPU-based rendering software to render static eye region geometry [2014] , and Wood et al. used a GPU-enabled path-tracing engine to render 120×80px images at 5.26s$/$image [2015]. We rendered and saved 400×300px images at 23ms$/$image using a commodity GPU (Nvidia GTX660) and SSD: a $200\times$ speedup. The bottleneck is writing image files to storage, image rendering itself takes only 3.6ms. This is because Unity's rasterizing renderer efficiently draws triangular meshes instead of simulating the behaviour of thousands of rays of light. The engine also provides physically-based shader materials and methods for efficiently manipulating the meshes and materials in real-time. As well as saving the rendered images, we output JSON-formatted metadata files. These describe the scene of each image fully, including gaze direction, eye region shape parameters, and lighting information, as well as 2D and 3D facial landmarks (e.g. eye-corners and eye-centre) for assisting with image-alignment.

Illuminating our models

A major source of error for appearance-based gaze estimation is lighting variation [Zhang et al. 2015]. The eye-region can appear very different depending on how it is illuminated – the brow or nose can produce shadows, and lights can create glints or large variations in image intensity. Unlike previous datasets collected in laboratories [Sugano et al. 2014], we wanted our synthesized images to cover a wide range of illumination conditions. We therefore rendered our scene with a directional light source and *image-based lighting*, a method where high dynamic range (HDR) panoramic images are used to provide ambient and specular light [Debevec 2002]. The directional light simulates bright light sources (e.g. the sun or a laptop screen), and is pointed in a random direction towards the eye region – this produces highlights and soft shadows. We then chose from a collection of 20 HDR panoramic photographs, and randomly varied their rotation and exposure levels – these were used for reflections and environmental ambient light (Figure 10). Our rendering engine then calculated lighting, shadows, reflections, and ambient occlusion for the eye region and eyeball in real-time.

[4]https://unity3d.com/

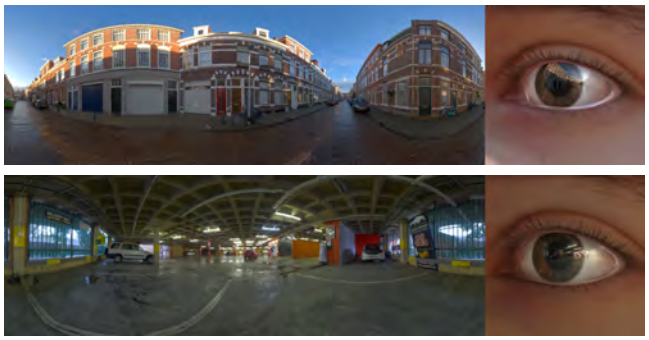

Figure 10: *We use HDR panoramic images for reflections and ambient light in the scene. Here you can see two example equirectangular panoramas, with example eye renderings.*

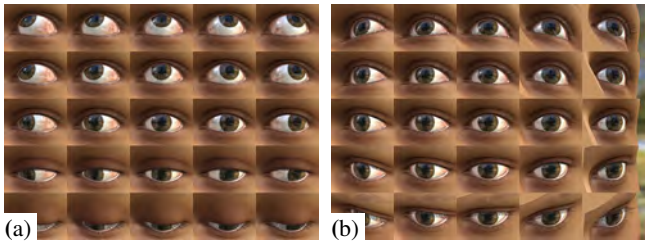

Figure 11: *We precisely control gaze direction and head pose: (a) shows renders with fixed camera position but varying gaze, (b) shows fixed gaze but varying head pose.*

Posing our models

One of the advantages of generating a dataset using computer graphics is being able to precisely position objects in the scene without the practical difficulties of real-life image capture. Including images of the eye taken under different head poses is one approach for head pose independent gaze estimation. While some previous datasets captured participants from a sparse and discrete set of different camera angles [Smith et al. 2013; Sugano et al. 2014], we instead specified the transforms of camera and eyeball continuously, allowing us to synthesize dense training data. The generative eye region was positioned at the scene origin pointing forwards, and defines neutral head pose. For each image, we randomly positioned the camera using spherical coordinates and pointed it towards the eyeball centre, simulating different head poses. We also randomly varied eyeball pitch and yaw as deviation from neutral gaze. To produce a variety of images, we modelled up to $\pm 30^\circ$ deviations in

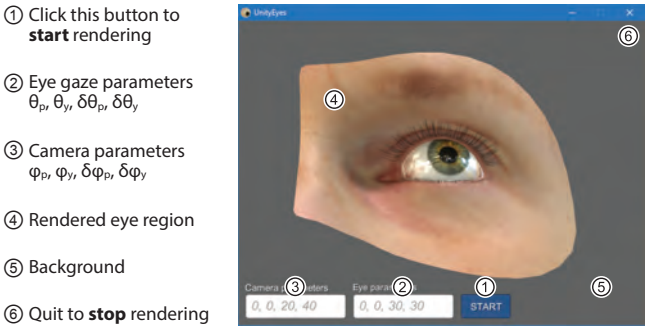

Figure 12: *UnityEyes running on* Windows 10 *at* 640×480px. *Important interface components are labelled.*

both pitch and yaw for head pose and gaze direction. We assumed an orthographic camera to simulate cropping a region of interest from a wide-angle image.

6 Using UnityEyes

We are making UnityEyes freely available for the benefit of the research community. On starting UnityEyes, users first choose a resolution for rendering images. The application then starts in *interactive mode* where users can pan around using the left mouse button, rotate the eyeball with the middle mouse button, and adjust the zoom using the scroll wheel. As can be seen in Figure 12, the 3D eye region is rendered against a 50% grey background. Once the *start* button is pressed, the application will enter *rendering mode* where it will continuously randomize the scene and save images. It will also save JSON metadata files describing the eye gaze, facial landmark, and other scene parameters.

Recent work has shown that targetting a specific use scenario in dataset synthesis can improve results [Wood et al. 2015]. We allow users to specify a gaze distribution through parameters $\{\theta_p, \theta_y, \delta\theta_p, \delta\theta_y\}$ where eyeball pitch and yaw are modelled as uniform random variables $U(\theta_p - \delta\theta_p, \theta_p + \delta\theta_p)$ and $U(\theta_y - \delta\theta_y, \theta_y + \delta\theta_y)$. Changes in head pose are simulated by rotating the camera around the eye region. Variance in the camera position is definined in a similar way using $\{\phi_p, \phi_y, \delta\phi_p, \delta\phi_y\}$.

7 Experiments

We performed a number of experiments to assess both the quality of our rendered images and their suitability for appearance based gaze estimation. We briefly outline the test datasets and the methodology used for finding best images to match the test data. Note that in all of our experiments we use a single generic rendering environment to generate training data, and we do not perform any dataset-specific targeting as was done in previous work [Wood et al. 2015].

Datasets To test the ability of UnityEyes for generating images that can be used to estimate gaze on varied and difficult eye region images, we evaluated our approach on 300-W challenge [Sagonas et al. 2013] datasets which include: **AFW** [Zhu and Ramanan 2012], **IBUG** [Sagonas et al. 2013] and **LFPW+Helen** [Belhumeur et al. 2011; Le et al. 2012], containing 135, 337, 600, and 554 images respectively. All of the datasets include uncontrolled images of faces *in-the-wild*: in indoor and outdoor environments, under varying illuminations, in presence of occlusions, under different poses, and from different quality cameras.

To quantitatively evaluate our dataset for appearance based gaze estimation we used a subset of **MPIIGaze** [Zhang et al. 2015] that had manually annotated eye corners (1500 eye images). The dataset has been collected in realistic laptop use scenarios and poses a challenging and practically relevant task for eye gaze estimation. As a comparison, we used a dataset generated by recent previous work for learning gaze estimation – **SynthesEyes**. This dataset contains 12000 synthesized images of an eye region, rendered from similar head-scan data using a highly accurate physically based renderer.

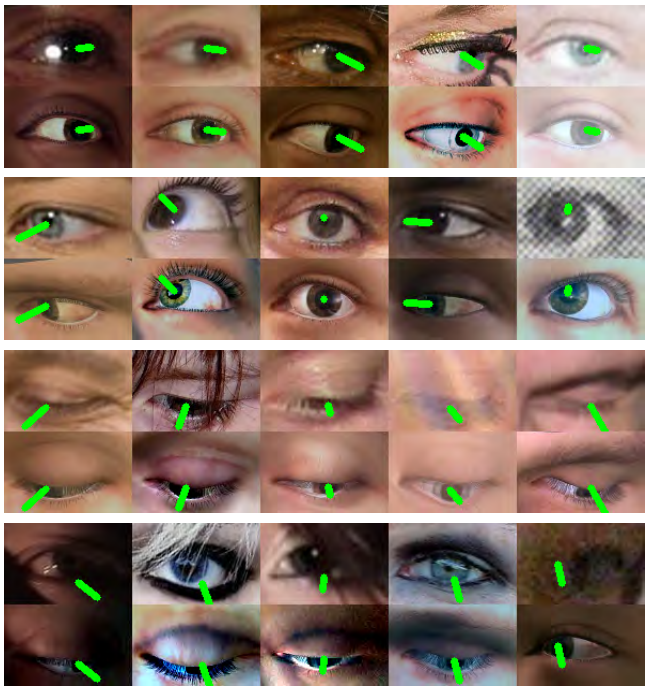

Figure 13: *Nearest-neighbour pairs showing in-the-wild images (top) and our renders (bottom) along with estimated gaze (green). The top three rows show qualitatively good gaze estimates, even under difficult lighting, low resolution, and extreme gaze angles. The bottom row shows failure cases from unmodelled variation e.g. makeup and hair.*

Methodology In all of the experiments we cropped images to 60×38px, and aligned them using a similarity transform. To align the images from 300-W and MPIIGaze datasets we used annotated eye corner locations included in the dataset. To align our UnityEyes images we used the same landmark conventions, with landmarks sampled from the 3D mesh. The estimated gaze vectors were rotated around the z-axis in-line with the similarity transform.

For matching we first converted all images to grayscale and then normalised each resulting image to have zero mean and unit variance. Finally, image matching is done using nearest-neighbour to choose the image i from the training set that closest matches the test image.

$$i = \arg\min_i \left(\text{mean}\left(||\text{train}_\text{i} - \text{test}||\right)\right) \quad (7)$$

The *pixel error* for image matching was computed using the mean absolute difference between the normalised images. The eye gaze error was computed as median angle between ground truth and estimated eye gaze vectors in degrees.

Matching eye images in-the-wild

First we demonstrate the realism of our synthesized dataset by matching in-the-wild eye images to those generated by UnityEyes.

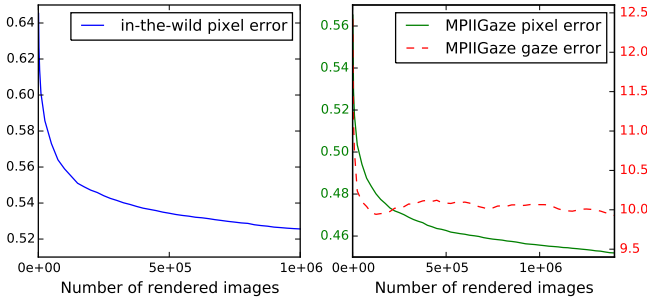

Figure 14: *Pixel and gaze errors for nearest-neighbor matching UnityEyes against in-the-wild and MPIIGaze eye images. Errors decrease as we use more rendered images.*

Table 1: *Comparison of our method to previous work for cross dataset gaze estimation on MPIIGaze. Note our improved results by using a very simple approach and our new dataset.*

MODEL	GAZE ERROR
CNN with UT [Zhang et al. 2015]	13.91°
CNN with SynthesEyes [Wood et al. 2015]	13.55°
CNN with SynthesEyes+UT [Wood et al. 2015]	11.12°
k-NN with UnityEyes **(ours)**	9.95°

We rendered over a million images and matched these to the four in-the-wild test sets. The error achieved using our approach (0.522) is comparable of that of matching from LFPW and Helen trainsets to the testest (0.511), and significantly more accurate than that of Syntheseyes (0.607). This shows UnityEyes images are closer to real-world captured images than SynthesEyes images. ANOVA with post-hoc pair-wise t-tests with Bonferroni correction revealed that all of the differences are statistically significant with $p < 0.001$.

Some example matches can be seen in Figure 13. The top rows show successful nearest-neighbor matches, allowing gaze to be estimated for unseen people in unconstrained lighting conditions. Failure cases in the bottom row include un-modelled occlusions (e.g. hair) and appearance variation (make-up). These qualitative examples show the benefit of UnityEyes over previous methods, as we estimate gaze for extreme gaze angles for which training data could not previously be collected. As nearest-neighbor image matching was carried out in a normalized grey-scale space, we color corrected (matching the mean and standard deviations of the RGB channels) the matched images to aid visual comparison.

Similarly for MPIIGaze our newly generated dataset achieves better image matching performance (0.456) than Syntheseyes (0.511) and in-the-wild (0.539) and even within dataset matching (0.511). ANOVA with post-hoc pair-wise t-tests with Bonferroni correction revealed that all of the differences are statistically significant with $p < 0.01$, except for Syntheseyes and within dataset matching.

Training data amount analysis Due to the ability of our framework to rapidly synthesize training data, we were able to generate over a million training images in less than 12 hours on commodity hardware. We demonstrate the effectiveness of this in Figure 14. We can see that both the image pixel error and gaze estimation errors decrease with an increased number of training images.

Gaze estimation

We evaluated the ability of our model to estimate eye gaze vectors by matching the rendered images to the eye images from MPIIGaze dataset (using k-Nearest-Neighbour algorithm outlined previously).

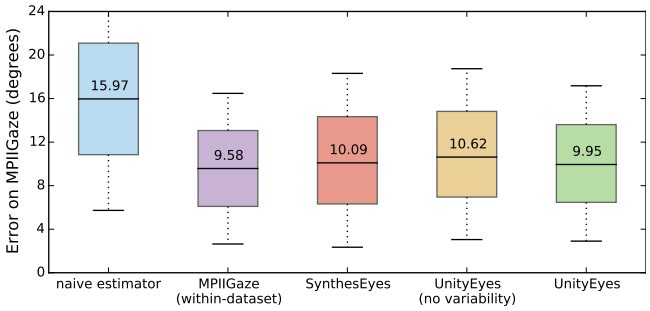

Figure 15: *Box-and-whisker plot showing gaze errors tested on MPIIGaze using a k-NN estimator. x-axis represents training set. The final two box plots show the benefits of using our generative eye region model (green) over one with no variation (yellow).*

The error rates of our model can be seen in Figure 15.

The results show that our model contains images that are closer in appearance and therefore better at predicting gaze than the SynthEseyes dataset and comparable to the predictions within the MPIIGaze dataset itself. Training with our UnityEyes dataset leads to statistically significantly results than using SynthesEyes ($p < 0.001$) when using a pair-wise t-test.

Furthermore, our simple k-Nearest-Neighbour approach achieves comparable performance to state-of-the-art deep learning based methods for cross-dataset appearance based eye gaze estimation without dataset targeting [Zhang et al. 2015; Wood et al. 2015] (see Table 1).

Shape variance

A final experiment we conducted was intended to validate the usefuleness of our morphable eye region model. We generated two sets of UnityEyes training data: a regular dataset using the full capabilities of our generative shape model, and a dataset with no shape variation, containing just the mean 3D shape $\boldsymbol{\mu}$. Both datasets contained uniform variation in eye region texture.

Our results demonstrate that using our morphable model shape variation was beneficial for both the pixel errors on the MPIIGaze dataset (0.456 vs 0.477) and the angle estimation on the same dataset (9.95 vs 10.62). Angle error differences are statistically significant ($p < 0.001$) according to a pair-wise t-test. See Figure 15 for more detailed comparisons.

Discussion

In our experiments section we demonstrate that our framework is able to generate highly complex and variable eye region images that lead to closer nearest-neighbor matches than previous work, and enables gaze estimation for difficult in-the-wild images. Furthermore, our rendered images can be used in a very simple k-Nearest-Neighbour eye gaze estimation system, achieving competitive performance compared to more complex deep learning based approaches [Zhang et al. 2015; Wood et al. 2015]. This demonstrates the utility of our framework for training appearance based gaze estimation systems.

8 Conclusion

In this paper we introduced a novel statistically-derived generative 3D model of the eye region. We also described a rendering frame-

work for rapidly synthesizing eye images 200× faster than previous work allowing us to easily generate a large number of images expressing a wide range of gaze directions. We demonstrate the utility of our eye region model and the fast generation framework by showing good image matching capabilities and by showing competitive performance for device and person independent appearance-based gaze estimation.

In future work we would like to use more varied head scans in order to better capture the shape and appearance variability. We would also like to explore the use of more advanced appearance based gaze estimation techniques using our eye region rendering framework.

Acknowledgements

This work was funded, in part, by the Cluster of Excellence on Multimodal Computing and Interaction (MMCI) at Saarland University.

References

BELHUMEUR, P. N., JACOBS, D. W., KRIEGMAN, D. J., AND KUMAR, N. 2011. Localizing parts of faces using a consensus of exemplars. In *CVPR*.

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

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

BLANZ, V., AND VETTER, T. 1999. A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 187–194.

DEBEVEC, P. 2002. Image-based lighting. *IEEE Computer Graphics and Applications 22*, 2, 26–34.

EVINGER, C., MANNING, K. A., AND SIBONY, P. A. 1991. Eyelid movements. *Invest. Ophthalmol. Vis. Sci 32*, 2.

FANELLI, G., DANTONE, M., GALL, J., FOSSATI, A., AND VAN GOOL, L. 2013. Random forests for real time 3d face analysis. *International Journal of Computer Vision*.

HUANG, Q., VEERARAGHAVAN, A., AND SABHARWAL, A. 2015. Tabletgaze: A dataset and baseline algorithms for unconstrained appearance-based gaze estimation in mobile tablets. *arXiv preprint arXiv:1508.01244*.

JIMENEZ, J., DANVOYE, E., AND VON DER PAHLEN, J. 2012. Photorealistic eyes rendering. In *SIGGRAPH Talks, Advances in Real-Time Rendering*, ACM.

LE, V., BRANDT, J., LIN, Z., BOURDEV, L., AND HUANG, T. S. 2012. Interactive facial feature localization. In *ECCV*.

LI, H., YU, J., YE, Y., AND BREGLER, C. 2013. Realtime facial animation with on-the-fly correctives. *ACM Transactions on Graphics*.

LOOP, C. 1987. Smooth subdivision surfaces based on triangles.

LU, F., SUGANO, Y., OKABE, T., AND SATO, Y. 2011. Inferring human gaze from appearance via adaptive linear regression. In *ICCV*, IEEE.

LU, F., SUGANO, Y., OKABE, T., AND SATO, Y. 2012. Head pose-free appearance-based gaze sensing via eye image synthesis. In *Pattern Recognition (ICPR)*, IEEE.

MALBOUISSON, J. M., MESSIAS, A., LEITE, L., RIOS, G., ET AL. 2005. Upper and lower eyelid saccades describe a harmonic oscillator function. *Invest. Ophthalmol. Vis. Sci 46*, 3.

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

MORA, K. A. F., AND ODOBEZ, J.-M. 2012. Gaze estimation from multimodal kinect data. In *CVPRW*, IEEE.

ORVALHO, V., BASTOS, P., PARKE, F., OLIVEIRA, B., AND ALVAREZ, X. 2012. A facial rigging survey. In *Eurographics*.

PAYSAN, P., KNOTHE, R., AMBERG, B., ROMDHANI, S., AND VETTER, T. 2009. A 3d face model for pose and illumination invariant face recognition. In *Advanced Video and Signal Based Surveillance*, IEEE.

PENNER, E., AND BORSHUKOV, G. 2011. Pre-integrated skin shading. *Gpu Pro 2*, 41–54.

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*, 69–91.

SAGONAS, C., TZIMIROPOULOS, G., ZAFEIRIOU, S., AND PANTIC, M. 2013. 300 faces in-the-wild challenge: The first facial landmark localization challenge. In *ICCV*.

SHIRLEY, P., ASHIKHMIN, M., AND MARSCHNER, S. 2009. *Fundamentals of computer graphics*. CRC Press.

SMITH, B., YIN, Q., FEINER, S., AND NAYAR, S. 2013. Gaze Locking: Passive Eye Contact Detection for HumanObject Interaction. In *UIST*, ACM.

SUGANO, Y., MATSUSHITA, Y., AND SATO, Y. 2014. Learning-by-Synthesis for Appearance-based 3D Gaze Estimation. In *Proc. CVPR*.

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

WOOD, E., BALTRUSAITIS, T., ZHANG, X., SUGANO, Y., ROBINSON, P., AND BULLING, A. 2015. Rendering of eyes for eye-shape registration and gaze estimation. In *ICCV*.

ZHANG, X., SUGANO, Y., FRITZ, M., AND BULLING, A. 2015. Appearance-Based Gaze Estimation in the Wild. In *CVPR*.

ZHU, X., AND RAMANAN, D. 2012. Face detection, pose estimation, and landmark localization in the wild. In *CVPR*.