Chapter 16
Eye Movement Synthesis

Recorded eye movements are fairly well understood from the point of view of analysis, but comparatively little has been accomplished regarding their synthesis, especially for the purpose of rendering synthetic eye motions. Most analytical approaches rely on gaze data filtering, e.g., signal smoothing, machine learning (Samadi and Cooke 2014), and/or data processing for specific event detection (Ouzts and Duchowski 2012), but they rarely, if at all, touch on signal synthesis. Signal processing approaches, however, have been used for synthesis. Yeo et al. (2012) *Eye-catch* simulation used the Kalman filter to produce gaze coupled with head motion, focusing primarily on saccades and smooth pursuits. As noted by Yeo et al., simulated gaze behavior looked qualitatively similar to captured gaze data, but comparison of synthesized trajectories showed absence of gaze jitter, a component of fixational gaze data (addressed below).

Why synthesize eye movements anyway? There are two important applications: for virtual character eye animation (Duchowski and Jörg 2016) and for testing eye movement analysis techniques (Duchowski et al. 2016). The two approaches overlap, with the latter sharing the same underlying microsaccadic jitter simulation, but amplified by simulated eye tracker noise which the former cannot use. Animation of the eye may also involve consideration of its rotation, including the modeling of Listing's and Donders' Laws within a quaternion framework (see Duchowski and Jörg 2015).

16.1 Procedural Simulation of Eye Movements

A comprehensive model of eye movement should at least require modeling of the underlying musculature along with possibly modeling of the oculomotor plant itself (e.g., see the early description of the oculomotor plant by Robinson (1968), and more recent incarnations of a Kalman-filter based model by Komogortsev and Khan (2008, 2009)).

A.T. Duchowski, *Eye Tracking Methodology*, DOI 10.1007/978-3-319-57883-5_16

For rendering and eye tracker simulation purposes, however, it is sufficient to model a dot moving within a 2D plane positioned in front of the eye. To eye tracker users, this dot is known as the calibration dot, and its movement is known as the calibration sequence. To animators, this dot can be used as a *look point* which the virtual character is meant to follow with their gaze. The 2D plane in which the dot moves is just an arbitrary plane positioned in front of the virtual character, rigged to be made stationary in relation to the character's head. Using this 2D point sequence, complex natural eye movement behaviors can also be modeled, e.g., reading. A model of reading behavior is one which directs gaze toward as yet unread or previously read words (regressions or revisits) and to lines above and below the current line of text (Campbell and Maglio 2001).

Using the concept of a look point, what is then required to model the motion of the eye is to model the motion of the point itself. Important criteria include the following:

1. a model of a saccade: how quickly does the dot jump from place to place?
2. a model of a fixation: how long does the dot rest in place, and, more importantly, how can it be made to jitter in place to model microsaccades, tremor, and/or drift?

Along saccades and fixations, smooth pursuits can be modeled (ignored here), along with random perturbations injected into various model components. Random perturbations are basically an attempt at making the movements somewhat unpredictable. Using them makes the simulation stochastic in nature. Model components are described below.

16.1.1 Modeling Saccades

A model of a symmetric acceleration function for saccades was introduced in Chap. 12. That function was then integrated twice to produce a function of position $H(t) = \frac{1}{10}t^5 - \frac{1}{4}t^4 + \frac{1}{6}t^3$, which is then be used to animate a moving dot $\mathbf{p}_t = (x_t, y_t)$ on a 2D surface. Chapter 12 also gave a `Python` code snippet for enacting the movement of the dot, or look point, between two known points in the sequence. The code that produces the movement is referred to as a simulation. When running the simulation, it is important to keep the simulation time step (h) small, e.g., $h = 0.0001$. Just before executing a saccade, set the saccade clock $t = 0$, then while $t < \Delta t$ run the following simulation steps:

1. $t = t/\Delta t$ (scale interpolant to time window)
2. $\mathbf{p}_t = \mathbf{C}_{i-1} + H(t)\mathbf{C}_i$ (advance position)
3. $t = t + h$ (advance time by the time step h)

where $\mathbf{C}_i$ denotes the i^{th} 2D look point coordinates and $\mathbf{p}_t$ is the saccade position, both in vector form, and Δt denotes the saccade duration (see below).

Given such a sequence of look points $\{\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_n\}$, several important requirements arise, namely:

1. a model of saccadic velocity (i.e., position and duration);
2. a model of the spatio-temporal fixation perturbation; and
3. control of the simulation timestep and sampling rates.

Setting time step h to an arbitrarily small value allows dissociation of the simulation clock from the sampling rate being modeled. The synthetic look point can then be sampled at arbitrary rates, 30, 60, 1000 Hz, etc., producing inter-sample periods of 33, 16, 1 etc. milliseconds, respectively.

In the simulation, Δt denotes the duration of a saccade, which is dependent on the distance between successive fixation points, known as the saccade amplitude. Saccades are ballistic and stereotyped, with the relation between amplitude (θ) and duration (Δt) expressed by the linear equation

$$\Delta t = 2.2\theta + 21 \text{ (ms)} \tag{16.1}$$

known as the *main sequence* for saccadic amplitudes up to about 20° (Bahill et al. 1975; Knox 2001).[1] The main sequence, based on empirical data, gives a model of saccade amplitude that is intuitively understood: the larger the eye rotation (θ), the more time required to rotate it.

Using the main sequence directly could lead to consistently reproducible saccadic jumps (amplitudes). This may be fine for simulating gaze and/or an eye tracker, but for animation purposes, it is often advantageous to augment the main sequence with slight random perturbations, e.g., a 10° targeting error. A small slight temporal perturbation can also be added to the predicted saccade duration, based on empirical observations. Saccade duration can thus be modeled as

$$\Delta t = 2.2\mathcal{N}(\theta, \sigma = 10^\circ) + 21 + \mathcal{N}(0, 0.01) \text{ (ms)} \tag{16.2}$$

where $\mathcal{N}(\mu, \sigma)$ denotes the normal distribution with mean μ and standard deviation σ.

16.1.2 Modeling Fixations

Fixations are never perfectly still, but instead are composed of small involuntary eye movements, namely *microsaccades*, *drift*, and *tremor* (Rolfs 2009). Eye movements in stabilized vision, i.e., during fixation, carry an image across the retinal photoreceptor mosaic (Pritchard 1961). Slow drifts curve away from the center of vision, high-frequency (150 Hz) tremor is superimposed on the drift, and microsaccades,

[1] In their development of *Eyes Alive*, Lee et al. (2002) (see also Gu et al. 2008) expressed the main sequence as $\Delta t = d\theta + D_0$ (ms) with $d \in [2, 2.7]$ ms/deg and $D_0 \in [20, 30]$ ms.

are fast *flick* movements (straight lines) back toward the center. If microsaccadic eye movements were perfectly counteracted, visual perception would rapidly fade due to adaptation (Hubel 1988; Grzywacz and Norcia 1995). Microsaccades contribute to maintaining visibility during fixation by shifting the retinal image to overcome adaptation, generating neural responses to stationary stimuli in visual neurons. For deeper reviews of microsaccades, including their role in visual perception, see Martinez-Conde et al. (2004) and Kowler (2011).

Instead of modeling drift and tremor (as Engbert (2012) does using a random-walk model), spatio-temporal perturbation of the gaze point at a fixation can be effected through stochastic simulation of microsaccadic jitter (Duchowski et al. 2015). Simply put, this jitter is a small, random offset to the fixation coordinates. Although not particularly realistic, it appears that even this type of slight jitter of the eye is consistently better rated by viewers as more realistic than an eye lacking any type of jitter whatsoever.

Duchowski et al. (2015) model fixational jitter as $1/f^{\alpha}$ noise since it has been found that recorded neural spikes are superimposed with noise that exhibits non-Gaussian characteristics that can be approximated as $1/f^{\alpha}$, or pink, noise (Yang et al. 2009). Pulse trains of nerve cells belonging to various brain structures have also been observed and characterized as $1/f$ noise (Usher et al. 1995).

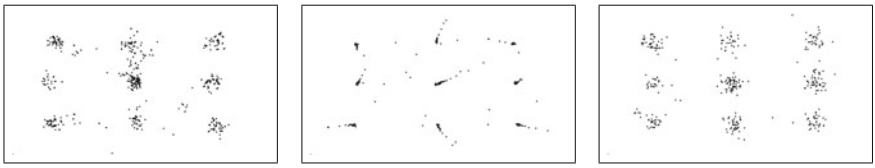
The $1/f$ regime accomplishes a tradeoff: the perceptual system amplifies and is sensitive to small fluctuations. Simultaneously, the system preserves a memory of past stimuli in the long time correlation tails. The memory inherent in the $1/f$ system possibly achieves a priming effect: successive stimuli separated by 50–100 ms at the same location elicit a stronger response to more recent stimulus. Such discrete temporal sampling may be optimal across a number of sensory systems, e.g., sniffs in rodent olfaction discretely sample sensory information every 200–300 ms and are similar in their temporal dynamics to primate saccades and microsaccades.

Duchowski et al. (2015) use a a 4th order filter for reshaping microsaccadic jitter modeled by Gaussian noise, $\mathcal{N}(0, \sigma = 12/60)$ arcmin visual angle. More formally, they define the pink noise filter as a function of two parameters, $\mathcal{P}(\alpha, f_0)$, where $1/f^{\alpha}$ describes the pink noise power spectral distribution and f_0 the filter's unity gain frequency (or more simply its gain). Setting $\alpha = 0$ produces white, uncorrelated noise, with a flat power spectral distribution, likely a poor choice for modeling biological motion such as microsaccades. Choices of $\alpha = 0.6$ and $f_0 = 0.85$ gave fairly natural microsaccadic jitter.

The pink noise model of microsaccadic jitter does not care *where* fixations are made, i.e., the model just offsets the current fixation coordinates, i.e., the current look point. Stated mathematically (Duchowski et al. 2016),

$$\mathbf{p}_{t+h} = \mathbf{p}_t + \mathcal{P}(\alpha, f_0) \tag{16.3}$$

where $\mathbf{p}_t$ is the look point at simulation time t and h is the simulation time step.



(a) recorded gaze data (b) synthetic pink noise (c) pink and white noise

Fig. 16.1 Gaze point (look point) data and simulation. These are composite renderings showing the whole calibration sequence on one frame. Actual data (**a**) as captured with an eye tracker was analyzed to find fixations. Detected fixations are then used as look points to drive synthetic simulation with pink noise jitter at the points of fixation (**b**). Synthetic data with pink noise is suitable for rendering eye motion of synthetic avatars but lacks simulation of noise that an eye tracker might produce. Adding such noise to synthetic data produces the simulation (**c**) which is fairly similar in appearance to captured data

16.2 Adding Synthetic Eye Tracking Noise

Note that eye trackers' sampling rates are not precise, or rather, eye trackers' sampling periods are generally non-uniform, most likely due to competing processes on the computer used to run the eye tracking software and/or due to network latencies. The simulation sampling period can be modeled by adding in a slight random temporal perturbation, e.g., $\mathcal{N}(0, \sigma = 0.5)$ ms.

Figure 16.1 shows the results of simulated gaze data in comparison to real data. Recorded gaze is noisy about each point of fixation. Using this data directly to render virtual characters' eye motion would result in very jerky and unnatural-looking eye movements. Smoothing of this data, e.g., with something like a Butterworth filter, produces gaze data that also lacks credibility. The procedural eye movement simulations models microsaccadic jitter and eye tracker noise separately. By withholding eye tracker noise, the simulation produces just enough jitter to make synthetic eye movements look fairly natural (see Fig. 16.1b). Adding white noise to this data produces a simulation that is now again too noisy for virtual characters but is adequate as a simulation of eye-tracked data (see Fig. 16.1c).

16.3 Summary and Further Reading

To recount, the stochastic model of eye movements is based on injection of noise at various points in the simulation:

- fixation durations, modeled by $\mathcal{N}(1.081, \sigma = 2.9016)$ (seconds), the average and standard deviation of the fixation duration Duchowski et al. (2016),
- fixation jitter, modeled by microsaccadic pink noise $\mathcal{P}(\alpha = 0.6, f_0 = 0.85)$ (degrees visual angle),
- saccade durations, modeled by (16.2), and

- sampling period $\mathcal{N}(1,000/\mathcal{F}, \sigma = 0.5)$ (milliseconds), with $\mathcal{F}$ the sampling frequency (Hz).

For rendering purposes, Duchowski et al. (2015) also add to the eye movement data stream:

- blink duration, modeled as $\mathcal{N}(120, \sigma = 70)$ (ms), and
- pupil unrest, modeled by pink noise $\mathcal{P}(\alpha = 1.6, f_0 = 0.35)$ (relative diameter).

The above sources of perturbation of the gaze point at its current location can be collectively described as

$$\mathbf{p}_{t+h} = \mathbf{p}_t + \mathcal{P}(\alpha, f_0) + \eta \tag{16.4}$$

where the primary source of microsaccadic jitter is represented by pink noise $\mathcal{P}(\alpha, f_0)$ and η represents other sources of variation, above.

For further details, see Duchowski et al. (2016) as well as Duchowski and Jörg (2016).