Introduction to Wavelets

Lecture #1: Tuesday, 30 September 1997
Lecturer: Denis Zorin
Scribe: John Owens

1 Why are wavelets useful?

Wavelets have a few interesting applications, some of which are mentioned below. However, the applications of wavelets by themselves are limited. The ideas behind wavelets, which we will be covering in this lecture and future lectures, are more important.

The most common use of wavelets is in signal processing applications. For example:

- Compression applications. If we can create a suitable representation of a signal, we can discard the "least significant" pieces of that representation and thus keep the original signal largely intact. This requires a transformation which separates the "important" parts of the signal from less important parts.

  In the simplest case, we can decompose a signal into two parts: a low frequency part, which is some sort of average of the original signal, and a high frequency part, which is what remains after the low frequency part is subtracted from the original signal. If we are interested in the low frequency part and hence discard the high frequency part, what remains is a smoother representation of the original signal with its low frequency components intact. Alternatively, if we are most interested in the high frequency part, we may be able to discard the low frequency part instead.

  This approach, that of decomposing a signal into two parts, is common for all wavelets. Also fundamental to wavelet analysis is a heirarchical decomposition, in which we may apply further transforms to an already decomposed signal.

- Edge detection. With this application it is most important to identify the areas in which the input image changes quickly. We can discard the smooth (low frequency) parts. The simplest wavelet basis, the Haar basis (to be discussed later) is suitable for this application.

  Along this vein, the book by Strang and Nguyen describes a widely used application of wavelets, fingerprint compression, in which edge detection figures prominently.

- Graphics. Two prominent uses of wavelets in graphics include

1. Curve and surface representations; and
2. Wavelet radiosity.

These two reflect two quite different uses of wavelets.

- Numerical analysis. Wavelets are used in the solution of partial differential equations and integral equations.

2 History of Wavelets

The first use of wavelets was by Haar in 1909. He was interested in finding a basis on a functional space similar to Fourier's basis in frequency space. In physics, wavelets were used in the characterization of Brownian motion. This work led to some of the ideas used to construct wavelet bases. Wavelets were also used for analysis of coherent states of a particular quantum system. Finally, in the signal processing field, S. Mallat discovered that filter banks have important connections with wavelet basis functions.

3 Filters and Filter Banks

3.1 Linearity and Time Invariance

Consider a discrete input signal $x(n)$, a filter H, and an output $y(n)$. We express the operation of H on the input signal x as $y = Hx$.

We call the filter H "linear" if scaling the input scales the output, and we call H "time invariant" if shifting the input (in time) correspondingly shifts the output. In these notes all filters will be assumed to be linear and time invariant.

3.2 Filter operation

If H is linear and time invariant, the we can express its operation as follows:

$$y(n) = \sum_k h(k)x(n-k).$$

This operation is called a convolution. The individual coefficients $h(i)$ are the "impulse responses" of the system.

The equation $y = Hx$ can also be written as an infinite matrix, with y and x column vectors and H in the following form:

$$\begin{bmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & h(1) & h(0) & h(-1) & \cdots & \cdots \\ \cdots & \cdots & h(1) & h(0) & h(-1) & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix}$$

Note that the entries of the filter are in reverse indexed order. If the input stream is finite with n elements, H is a finite $n \times n$ matrix.

3.3 What is a basis?

For a given space of functions, a frame is a collection of functions such that any function in the space is a weighted sum of the functions in the frame. In other words, if the functions in the frame are $f_0 \ldots f_n \ldots$, then any function g can be written as $g = \sum_i a_i f_i$.

A basis is also a collection of functions. Any basis is a frame, but a basis also has the property of linear independence. With a basis, the coefficients a_i of the expansion of the function g (written as a weighted sum of the basis functions $g = \sum_n a_i f_i$) are uniquely determined. Another way to state this property is that no function f_i in the basis can be written as a weighted sum of the other functions of the basis.

For a function space we can define an inner product. One example of an inner product is a dot product, used in a vector space. In a function space we define the inner product $< f, g >$ of two functions f and g as $\int f(t)g(t)dt$.

One desirable property of a basis is orthogonality. With an orthogonal basis, the inner product of two basis functions f_i and f_j is equal to zero if $i \neq j$.

A second desirable property is orthonormality, which implies that taking the inner product of a basis function f_i with itself equals 1. For our functional basis we see that $\int f^2(t)dt = 1$.

In summary, an orthogonal, orthonormal basis implies that

$$< f_i, f_j >= \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

So why is orthonormality a desired property? Let us define a function x as the weighted sum of the basis functions, $x = \sum a_i f_i$. We would like to find the coefficients a_i, and orthonormality makes this simple. We only need to take the inner product of the function x with a_i's associated basis function to find the associated coefficient a_i.

$$\begin{aligned} < x, f_j > &= \sum_i a_i < f_i, f_j > \\ &= a_j < f_j, f_j > \\ &= a_j \end{aligned}$$

3.4 Filters and wavelets, Haar filters

Each wavelet basis has two filters associated with it. In general those filters are expressed in the form $y(n) = \sum h(n)x(n)$.

The Haar basis is particularly simple. The two filters are H_0 and H_1 and are defined as follows:

$$H_0 : y(n) = \frac{1}{2}x(n) + \frac{1}{2}x(n-1)$$

$$H_1 : y(n) = \frac{1}{2}x(n) - \frac{1}{2}x(n-1)$$

For H_0, all h coefficients are zero except for $h_0(0) = 1/2$ and $h_0(1) = 1/2$. Similarly, in H_1, $h_1(0) = 1/2$ and $h_1(1) = -1/2$.

H_0 computes a moving average of its input, resulting in a sequence which is smoother than the initial sequence. Hence it is a low pass filter. H_1 computes a moving difference and serves as a high pass filter.

3.5 Scaling functions and the dilation equation

We define a function ϕ as a "scaling function". Scaling functions obey the dilation equation,

$$\phi(t) = 2\sum h_0(k)\phi(2t-k). \tag{1}$$

Note that the filter coefficients used (the h_0's) are from the first of the pair of filters. We will use the second filter later.

If this equation has an appropriate solution $\phi(t)$, then we can construct a *frame* using any integral values of j or k in the following generating relation:

$$\phi(2^j t - k). \tag{2}$$

Next we introduce the wavelet equation (3), which uses the coefficients from the second pair of filters:

$$w(t) = 2\sum h_1(k)\phi(2t-k). \tag{3}$$

We generate a wavelet *basis* with a similar generating relation

$$w(2^j t - k).$$

Note that scaling functions don't form a basis; wavelets do. But wavelets are constructed from scaling functions, so we have to find a scaling function first.

As can be seen from the wavelet equation, wavelets are defined as linear combinations between scaling functions one level below, where a "level" is defined as the set of all scaling functions generated with a given j.

For instance, the Haar scaling function satisfies the dilation equation with the solution

$$\phi(t) = \phi(2t) + \phi(2t-1).$$

And the Haar wavelet satisfies the wavelet equation with the solution

$$w(t) = \phi(2t) - \phi(2t-1).$$

The Haar basis spans the space of all functions with integrable square.

Haar wavelets are orthogonal.

4 Filters and Filter Banks

4.1 Definition of several operators

Here we will define four operators and describe their operation (Figure 1).

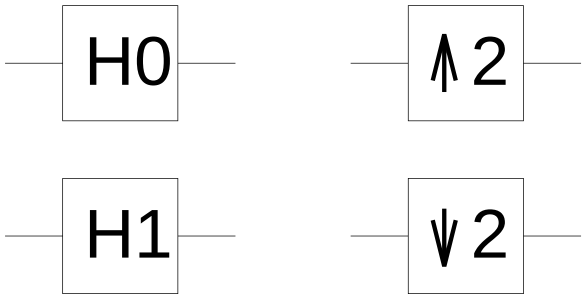


Figure 1: Our 4 operators

The first operator, $H = H_0$, is the Haar moving average filter. It outputs the average of its current input and its previous input.

The second operator, H_1, is the Haar moving difference filter. It outputs half the difference between its current input and its previous input.

The third operator, $\uparrow 2$, is an upsampling operator. It outputs each input twice, outputting at double the rate of its input.

The fourth operator, $\downarrow 2$, is a downsampling operator. It outputs every other input it receives, outputting at half the rate of the input.

4.2 Overview

We will look at two filter banks, an analysis bank and a synthesis bank. The analysis bank will use the Haar filter pair, consisting of a low pass and a high pass filter. We would like to pick a synthesis filter bank such that the output of the analysis and synthesis filter connected in series is the same as the input into the analysis filter, with perhaps a time delay.

First we will look at the analysis bank, then the synthesis bank, then we will put them together and show that the aforementioned property holds.

4.3 Haar analysis bank

Figure 2 shows the analysis bank.

The input comes in on the left side, and the output of the filter bank leaves through the two paths on the right side. It is desirable (but not necessary) that the bandwidth and storage requirements in the system be equal to those of the input stream. Hence, the downsampling operators cut the bandwidths in half (compared to that of the input

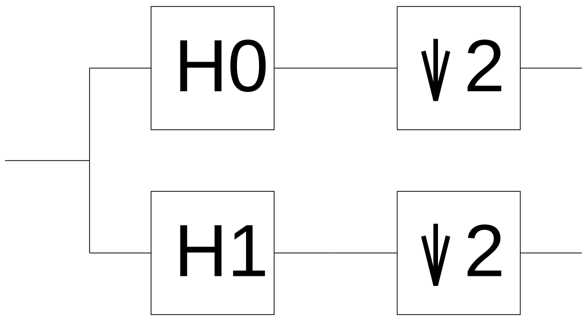


Figure 2: The Haar analysis bank

stream) on each of the two output streams, and the overall bandwidth requirement entering the analysis bank is equal to that leaving the bank. In this case the filter bank can be viewed as a linear transform: if the input signal had finite length n, the output will also have length n, consisting of two halves, each having length $n/2$.

4.4 Synthesis bank

Given the above analysis bank, what synthesis bank should we use to reproduce the input on the output? The basic analysis filter structure is shown in Figure 3. The output of each of the upsampling filters has the same bandwidth as the initial input stream, but note the outputs of filters F and G are summed to create an output stream with the same bandwidth as the original input stream.

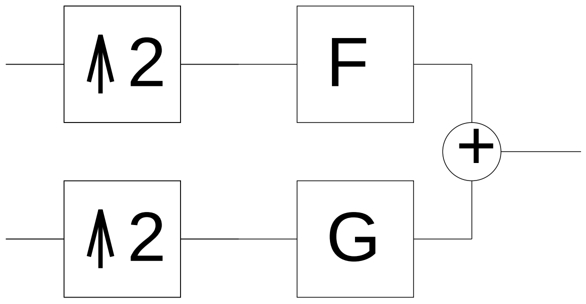


Figure 3: The basic synthesis bank

Conveniently enough, it turns out that we can use the Haar moving average filter H_0 as the filter F in Figure 3 and similarly, replace G with $-H_1$. This structure is shown in Figure 4.

The complete filtering system is shown in Figure 5.

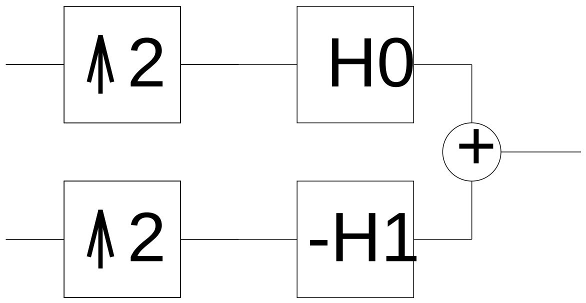


Figure 4: The Haar synthesis bank

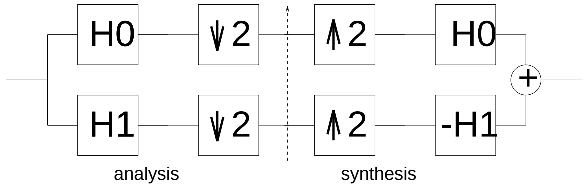


Figure 5: The complete system

5 Transforms used for analysis and construction of filters

Two related transforms are the Fourier transform and the z transform. To see more background on Fourier transforms, please consult the handout on "Antialiased Shifting and Resizing" distributed in class. Both transforms are described in detail in most signal processing books.

5.1 The Fourier Transform

The Fourier transform is defined on both a discrete signal $x(k)$ as

$$X(\omega) = \sum_k x(k)e^{-jk\omega} \tag{4}$$

and a continuous signal $x(t)$ as

$$X(\omega) = \int x(t)e^{-jk\omega}dt.$$

Following the signal processing tradition, we use j to denote $\sqrt{-1}$.

5.2 The z Transform

The z transform is similar but does not use powers of $e^{-j\omega}$. instead choosing powers of z:

$$X(z) = \sum_k x(k)z^{-k}.$$

5.3 Frequency response

The frequency response of a filter H is written as $H(\omega)$ (or $H(e^{j\omega})$). In Fourier space, the frequency response is useful for analyzing linear time invariant filters. Why? Convolution (i.e. the operation we use to describe filter operation) is simply multiplication. In other words, the operation

$$y(n) = \sum_k h(k)x(n-k)$$

in Fourier space is equivalent to

$$Y(\omega) = H(\omega)X(\omega).$$

We can see, then, that $H(\omega)$, used as a filter, serves to emphasize or deemphasize frequencies in the input signal. Each frequency component of the input $X(\omega)$ corresponding to a frequency ω is scaled by $H(\omega)$. For example, a low-pass filter passes through components with small ω ($H(\omega)$ is close to 1 when ω is small) and suppresses high-frequency components ($H(\omega)$ is close to 0 if ω is close to π).

5.4 Frequency response of the Haar basis

First we will look at the frequency response of H_0, the moving average function. Recall that the frequency response of an input stream is defined in (4). In H_0 the only coefficients that are non-zero are $h(0) = h(1) = 1/2$.

Thus we can calculate the input response using (4):

$$\begin{aligned} H_0(\omega) &= (1 + e^{-jk\omega})/2 \\ &= e^{-jk\omega/2}(e^{jk\omega/2} + e^{-jk\omega/2})/2 \\ &= e^{-jk\omega/2}\cos(\omega/2) \end{aligned}$$

and $|H_0(\omega)| = \cos(\omega/2)$.

With a similar derivation we can show that $|H_1(\omega)| = |\sin(\omega/2)|$.

Introduction to Wavelets II

Lecture #2: Thursday, 2 October 1997
Lecturer: Denis Zorin
Scribe: Scott Cohen
Reviewer: John Owens

1 Filter Banks as Transformations

A filter bank H transforms an input x into an output $y = H(x)$. Figure 1 shows the familiar example of a analysis filter bank that separates the low and high frequencies of a discrete input signal. If the filter bank is linear, then the corresponding transformation

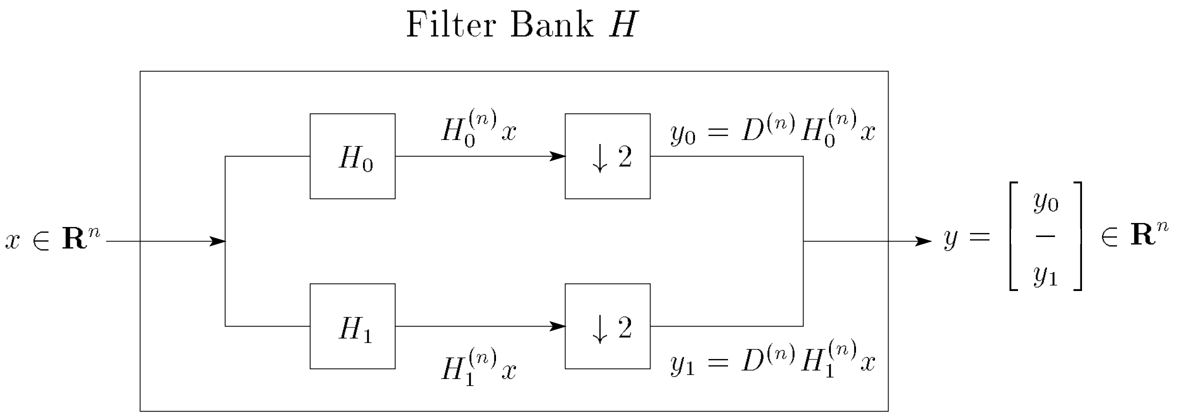

Filter Bank H
H_0
$H_0^{(n)}x$
$\downarrow 2$
$y_0 = D^{(n)}H_0^{(n)}x$
$x \in \mathbf{R}^n$
$y = \begin{bmatrix} y_0 \\ - \\ y_1 \end{bmatrix} \in \mathbf{R}^n$
H_1
$\downarrow 2$
$H_1^{(n)}x$
$y_1 = D^{(n)}H_1^{(n)}x$

Figure 1: A analysis filter bank.

can be represented as a matrix H, and applying H to x is achieved by computing the matrix product $y = Hx$.

As an example, we shall now compute the matrix H for the Haar analysis filter bank, assuming an input x consisting of n samples ($x \in \mathbf{R}^n$). Recall that the Haar low pass filter H_0 simply averages adjacent entries of its input. This operation can be represented by the matrix[1]

$$H_0^{(n)} = \frac{1}{2}\begin{bmatrix} 1 & 1 & & & & & \\ & 1 & 1 & & & & \\ & & 1 & 1 & & & \\ & & & & \ddots & & \\ & & & & & 1 & 1 \\ & & & & & & 1 \end{bmatrix} \in \mathbf{R}^{n\times n}.$$

The Haar high pass filter H_1 computes half the difference between successive input samples, and can be represented[2] as

$$H_1^{(n)} = \frac{1}{2}\begin{bmatrix} 1 & -1 & & & & & \\ & 1 & -1 & & & & \\ & & 1 & -1 & & & \\ & & & & \ddots & & \\ & & & & & 1 & -1 \\ & & & & & & 1 \end{bmatrix} \in \mathbf{R}^{n\times n}.$$

The downsampling operation is represented as the matrix[3]

$$D^{(n)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \in \mathbf{R}^{\frac{n}{2}\times n}$$

which picks out the first, third, etc. entries of x. The combined action of low pass filtering and then downsampling is represented by the matrix

$$L^{(n)} = D^{(n)}H_0^{(n)} = \frac{1}{2}\begin{bmatrix} 1 & 1 & & & & & & & \\ & & 1 & 1 & & & & & \\ & & & & \cdots & & & & \\ & & & & & 1 & 1 & & \\ & & & & & & & 1 & 1 \end{bmatrix} \in \mathbf{R}^{\frac{n}{2}\times n}.$$

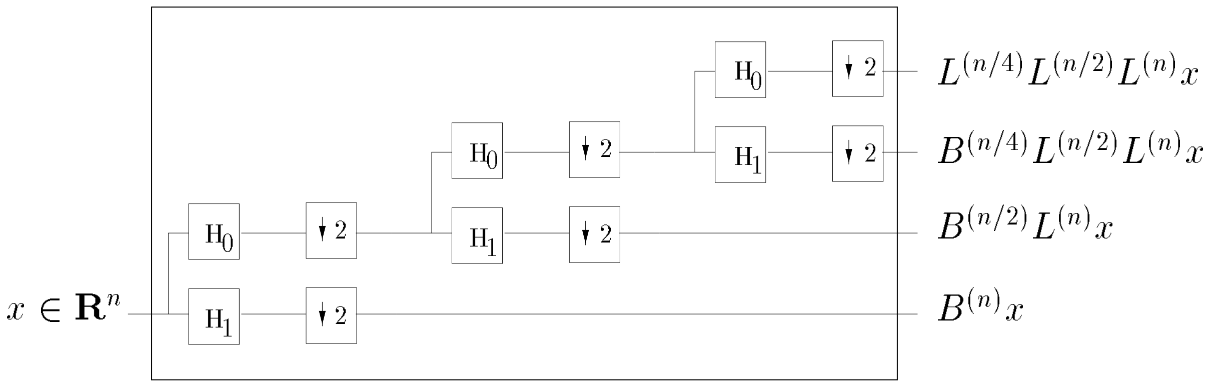
Similarly, the combined action of high pass filtering and then downsampling is represented

[1]Note that $H_0^{(n)}$ averages the final entry of x with zero since there is no next element. This is a moot point since the downsampling step that follows throws away the last entry of $H_0^{(n)}x$.

[2]The filter matrix $H_1^{(n)}$ differences the final entry of x with zero since there is no next element. This boundary anomaly is, once again, a moot point since the downsampling step that follows throws away the last entry of $H_1^{(n)}x$.

[3]Here we have tacitly assumed that n is even. Since the downsampling operation picks out the first, third, etc. components of its input, the last component will be discarded.

Filter Bank H_{sys}



$$\text{output } y = \begin{bmatrix} L^{(n/4)}L^{(n/2)}L^{(n)} \\ B^{(n/4)}L^{(n/2)}L^{(n)} \\ B^{(n/2)}L^{(n)} \\ B^{(n)} \end{bmatrix} x = H_{\text{sys}}x \in \mathbf{R}^n$$

Figure 2: Hierarchical decomposition of a signal.

as

$$B^{(n)} = D^{(n)}H_1^{(n)} = \frac{1}{2}\begin{bmatrix} 1 & -1 & & & & & & & \\ & & 1 & -1 & & & & & \\ & & & & \cdots & & & & \\ & & & & & 1 & -1 & & \\ & & & & & & & 1 & -1 \end{bmatrix} \in \mathbf{R}^{\frac{n}{2}\times n}.$$

The top and bottom branches of the filter bank produce (see Figure 1)

$$y_0 = L^{(n)}x \in \mathbf{R}^{\frac{n}{2}} \qquad \text{and} \qquad y_1 = B^{(n)}x \in \mathbf{R}^{\frac{n}{2}},$$

respectively. The output of the filter bank is

$$y = \begin{bmatrix} y_0 \\ - \\ y_1 \end{bmatrix} = \begin{bmatrix} L^{(n)}x \\ - \\ B^{(n)}x \end{bmatrix} = H^{(n)}x \in \mathbf{R}^n, \qquad \text{where } H^{(n)} = \begin{bmatrix} L^{(n)} \\ - \\ B^{(n)} \end{bmatrix} \in \mathbf{R}^{n\times n}.$$

Thus, we have expressed the operation of the Haar filter bank on an input of length n as the matrix $H^{(n)}$ shown above.

The output of the top branch of the filter bank is a coarse version of the input signal. We can build a hierarchical representation of a signal by recursively filtering the low pass output of the filter bank. This process is illustrated in Figure 2. In each step of the recursion, the signal rate decreases by a factor of two. If the signal is discrete and finite (and a power of two in length), then we eventually reach a signal with one sample. In the case of the Haar analysis filter bank, this sample will be the average of the elements of the original signal. The sum of the sizes of all the outputs from the high pass steps and the output of the final low pass step is equal to the size of the original input. The final, hierarchical representation of an input signal is a collection of signal details at various resolution levels (scales) and a coarse version of the original signal (which is the output of the final low pass filter). The entire process can be represented as a single transformation matrix H_{sys} which one can think of as rewriting the input x in terms of another basis to produce the output $y = H_{\text{sys}}x$. Figure 2 shows the H_{sys} matrix that results when the Haar filter bank is applied three times.

When the input to the system is an image (a 2D signal), the first filter bank application produces a blurred version of the image (the low pass output) and the details of the original image (sharp edges) which are not contained in the blurred version (the output of the high pass filter). The second low pass filter application takes the blurred version from step one and blurs it even further. The differences between the second blurred image and the first blurred image are captured in the output of the second high pass filter. This recursive process transforms an image into a collection of images that capture image details at different scales and one final coarse image.

2 The Haar Wavelet Basis

If a hierarchical decomposition via filter banks writes a discrete signal in terms of new basis. Can we find a similar decomposition for continuous signals? Answering this question is where the dilation equation

$$\phi(t) = 2\sum_k h_0(k)\phi(2t-k) \tag{1}$$

and the wavelet equation

$$w(t) = 2\sum_k h_1(k)\phi(2t-k) \tag{2}$$

come into the picture. The function $\phi(t)$ is called the *scaling function*, and the function $w(t)$ is called the *wavelet function*. The dilation equation and wavelet equation must hold for all t. Replacing t by $2^{j-1}t$ gives

$$\begin{aligned} \phi(2^{j-1}t) &= 2\sum_k h_0(k)\phi(2^j t-k) && (3) \\ w(2^{j-1}t) &= 2\sum_k h_1(k)\phi(2^j t-k) && (4) \end{aligned}$$

This last set of equations is more convenient for describing the construction of the wavelet basis corresponding to a filter bank.

We now illustrate the wavelet basis construction from the dilation equations using the Haar filter bank. Recall that the low pass Haar filter H_0 is defined by $h_0(0) = h_0(1) = 1/2$

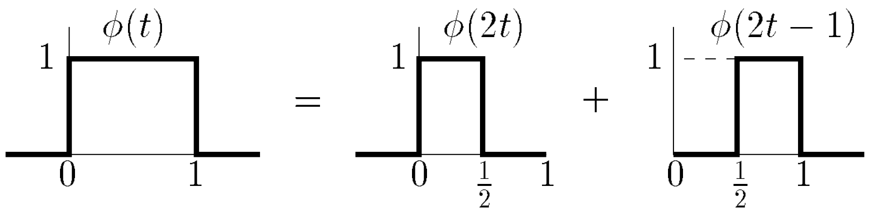


Figure 3: The Haar scaling functions $\phi(t)$, $\phi(2t)$, and $\phi(2t-1)$.

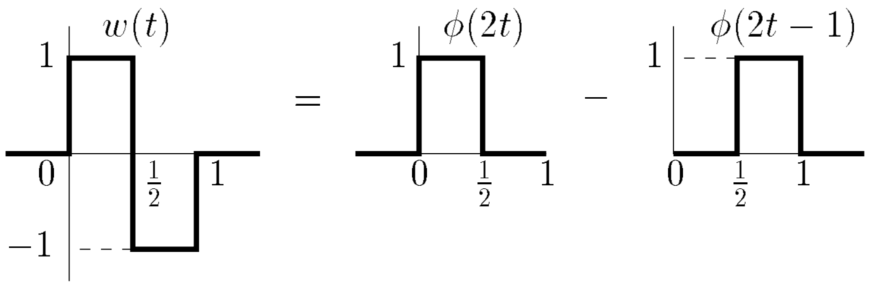


Figure 4: The Haar wavelet function $w(t)$.

(all other coefficients are zero). Substituting the Haar low pass filter into equation (1), we get

$$\phi(t) = \phi(2t) + \phi(2t-1).$$

The solution to this recurrence is the Haar scaling function

$$\phi(t) = \begin{cases} 1 & \text{if } t \in [0,1) \\ 0 & \text{otherwise} \end{cases} . \tag{5}$$

The functions $\phi(t)$, $\phi(2t)$, and $\phi(2t-1)$ are shown in Figure 3. The high pass Haar filter H_1 is defined by $h_1(0) = 1/2$ and $h_1(1) = -1/2$. Substituting into (2) yields

$$w(t) = \phi(2t) - \phi(2t-1).$$

It follows easily from (5) that the Haar wavelet function is

$$w(t) = \begin{cases} 1 & \text{if } t \in [0,1/2) \\ -1 & \text{if } t \in [1/2,1) \\ 0 & \text{otherwise} \end{cases} .$$

The wavelet function $w(t)$ is shown in Figure 4. The scaling function $\phi(t)$ is the continuous analog of the discrete low pass filter H_0. Applying $\phi(t)$ to $f(t)$ yields

$$<\phi, f> = \int_{-\infty}^{\infty} \phi(t) f(t)\, dt = \int_0^1 f(t)\, dt,$$

the average value of f over the interval $[0,1)$. The wavelet function $w(t)$ is the continuous analog of the discrete high pass filter H_1. Applying $w(t)$ to $f(t)$ yields

$$< w, f > = \int_{-\infty}^{\infty} w(t)f(t)\,dt = \int_0^{\frac{1}{2}} f(t)\,dt - \int_{\frac{1}{2}}^{1} f(t)\,dt.$$

The filter ϕ is an averaging operator, and the filter w is a differencing operator.

Now consider functions defined on the interval $[0,1)$. Let V^j denote the set of functions that are constant on the 2^j subintervals $[l/2^j, (l+1)/2^j)$, $l = 0, 1, \ldots, 2^j - 1$. Any function in V^j can be represented exactly by a linear combination of the 2^j functions

$$\phi_{jk}(t) = \phi(2^j t - k), \quad k = 0, \ldots, 2^j - 1.$$

This should be clear (at least for the case $j = 3$) from the upper left hand corner in Figure 5 which shows the $2^j = 8$ functions $\phi_{3k} = \phi(2^3 t - k)$, $k = 0, \ldots, 7$. Similarly, the wavelet functions are denoted by

$$w_{jk}(t) = w(2^j t - k), \quad k = 0, \ldots, 2^j - 1.$$

We collect the scaling and wavelet functions at fixed resolution level j in the sets

$$\Phi_j = \{\ \phi_{jk}(t)\ :\ k = 0, \ldots, 2^j - 1\ \} \quad \text{and} \quad \Omega_j = \{\ w_{jk}(t)\ :\ k = 0, \ldots, 2^j - 1\ \},$$

respectively.

As previously mentioned, Φ_j is a basis for V^j. Applying equations (3) and (4) with $j = 3$ to the functions $\phi_{3k}(t) = \phi(2^3 t - k)$, $k = 0, \ldots, 7$ yields the functions $\phi_{2k}(t) = \phi(2^2 t - k)$, $k = 0, \ldots, 3$ and $w_{2k}(t) = w(2^2 t - k)$, $k = 0, \ldots, 3$. In detail,

$$\begin{bmatrix} \phi_{20} \\ \phi_{21} \\ \phi_{22} \\ \phi_{23} \\ w_{20} \\ w_{21} \\ w_{22} \\ w_{23} \end{bmatrix} = 2H^{(8)} \begin{bmatrix} \phi_{30} \\ \phi_{31} \\ \phi_{32} \\ \phi_{33} \\ \phi_{34} \\ \phi_{35} \\ \phi_{36} \\ \phi_{37} \end{bmatrix}.$$

The results of this step are shown in the middle and last columns of the first row. The final column in the first row shows the separation of the low pass output functions Φ_2 from the high pass output functions Ω_2. Note that $\Phi_2 \cup \Omega_2$ is also a basis for V^3. The low pass output functions are further refined by once again applying equations (3) and (4), this time with $j = 2$. In matrix notation,

$$\begin{bmatrix} \phi_{10} \\ \phi_{11} \\ w_{10} \\ w_{11} \end{bmatrix} = 2H^{(4)} \begin{bmatrix} \phi_{20} \\ \phi_{21} \\ \phi_{22} \\ \phi_{23} \end{bmatrix}.$$

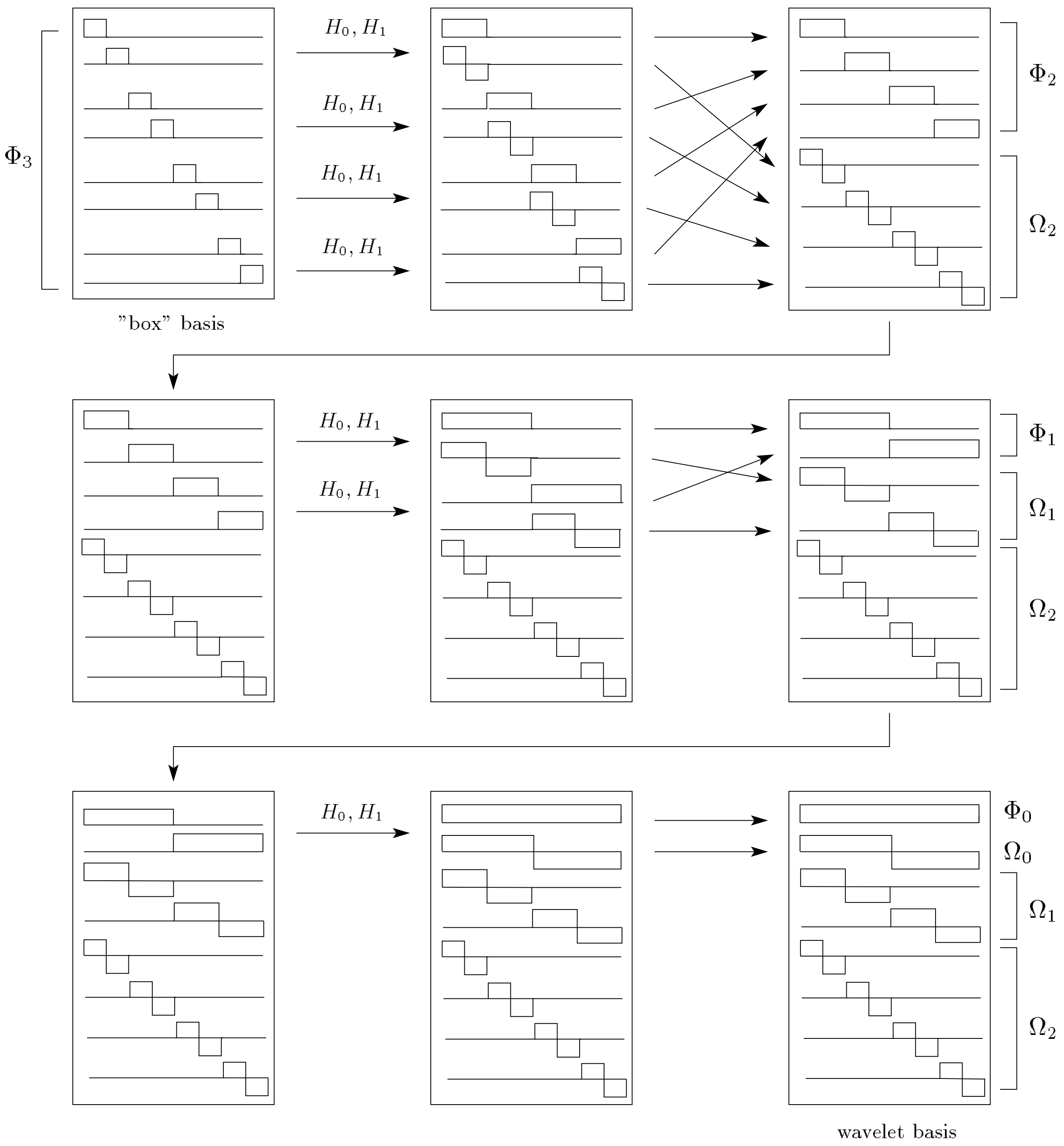


Figure 5: The Haar wavelet basis.

This defines the four new functions $\phi_{1k}(t) = \phi(2t - k)$, $k = 0, 1$, and $w_{1k}(t) = w(2t - k)$, $k = 0, 1$ shown in the second row of Figure 5. The last column of this row shows the separation of the low pass outputs Φ_1 and high pass outputs Ω_1. The set $\Phi_1 \cup \Omega_1 \cup \Omega_2$ is, again, a basis for V^3. Finally, the third row shows the result of applying the dilation and wavelet equations with $j = 1$ to $\phi_{1k}(t) = \phi(2t - k)$, $k = 0, 1$ to produce the scaling function $\phi_{0k}(t) = \phi(t - k)$, $k = 0$ and the wavelet function $w_{0k}(t) = w(t - k)$, $k = 0$. More precisely,

$$\begin{bmatrix} \phi_{00} \\ w_{00} \end{bmatrix} = 2H^{(2)} \begin{bmatrix} \phi_{10} \\ \phi_{11} \end{bmatrix}.$$

The eight functions ϕ_{00}, w_{00}, w_{10}, w_{11}, w_{20}, w_{21}, w_{22}, w_{23} in $\Phi_0 \cup \Omega_0 \cup \Omega_1 \cup \Omega_2$ form the the Haar wavelet basis for V^3. In general, the Haar wavelet basis for V^j contains the 2^j functions in $\Phi_0 \cup \Omega_0 \cup \Omega_1 \cup \cdots \cup \Omega_{j-1}$.

3 Some Examples

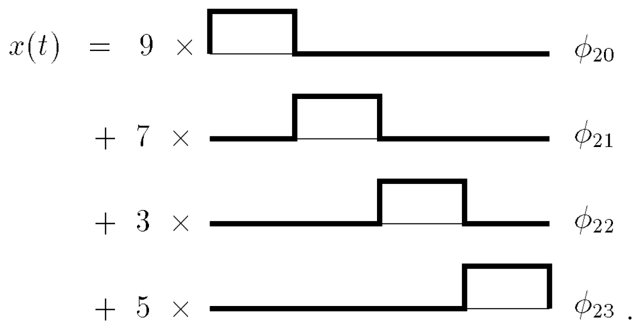
Suppose we have a function[4] $x(t)$ defined on $[0, 1)$ by

$$x(t) = \begin{cases} 9 & \text{if } t \in [0, 1/4) \\ 7 & \text{if } t \in [1/4, 1/2) \\ 3 & \text{if } t \in [1/2, 3/4) \\ 5 & \text{if } t \in [3/4, 1) \end{cases}.$$

The function $x(t)$ is in the space V^2. In terms of the basis Φ_2, $x(t)$ has representation

$$x = \begin{bmatrix} 9 \\ 7 \\ 3 \\ 5 \end{bmatrix}.$$

A graphical representation of $x(t)$ in terms of the basis Φ_2 is

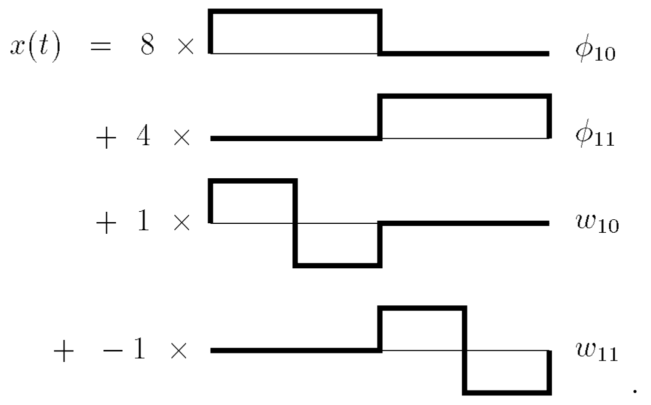


Applying the Haar sythesis filter bank $H^{(4)}$ to x gives

$$z = H^{(4)}x = \begin{bmatrix} 8 \\ 4 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} (9+7)/2 \\ (3+5)/2 \\ (9-7)/2 \\ (3-5)/2 \end{bmatrix}.$$

[4]This example is taken directly from the paper "Wavelets for Computer Graphics: A Primer (Part 1)" by Eric J. Stollnitz, Tony D. DeRose, and David Salesin in *IEEE Computer Graphics and Applications*, 15(3):76-84, May 1995. It is also available online at `http://www.cs.washington.edu/research/graphics/projects/wavelets/article/`.

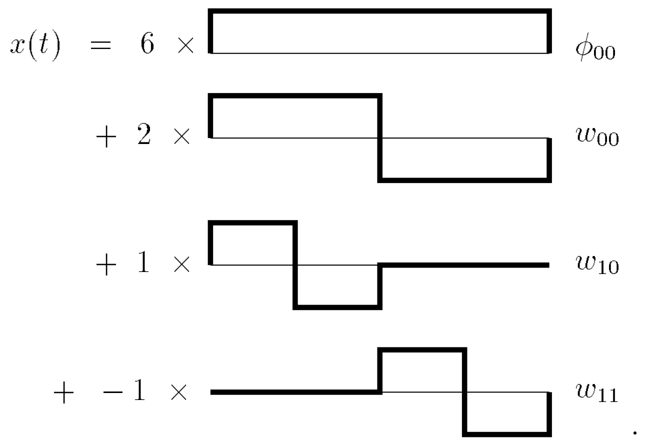
The elements of z are the coefficients of the representation of $x(t)$ in terms of the basis $\Phi_1 \cup \Omega_1$.



Applying the Haar analysis filter bank $H^{(2)}$ to the first two elements in z and leaving the high pass outputs in place gives

$$y = \begin{bmatrix} H^{(2)} \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \\ z_3 \\ z_4 \end{bmatrix} = \begin{bmatrix} 6 \\ 2 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} (8+4)/2 \\ (8-4)/2 \\ 1 \\ -1 \end{bmatrix}.$$

The elements in y are the coefficients of the representation of $x(t)$ in terms of the Haar wavelet basis $\Phi_0 \cup \Omega_0 \cup \Omega_1$.



The Haar wavelet transform of the signal $x = [\ 9\ \ 7\ \ 2\ \ 5\ ]$ is $y = [\ 6\ \ 2\ \ 1\ \ -1\ ]$.

We can also compute the wavelet transform of x by multiplying the system filter bank matrix H_{sys} by x. Figure 2 shows H_{sys} when the Haar analysis filter bank is applied three times. In the example in this section, we only need two applications and the input vector has length $n = 4$. This results in the matrix

$$H_{\text{sys}} = \begin{bmatrix} L^{(2)}L^{(4)} \\ B^{(2)}L^{(4)} \\ B^{(4)} \end{bmatrix} = \begin{bmatrix} \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} \\ \frac{1}{2} & -\frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & -\frac{1}{2} \end{bmatrix}.$$

The system output is

$$y = H_{\text{sys}}x = \begin{bmatrix} 6 \\ 2 \\ 1 \\ -1 \end{bmatrix},$$

just as we derived previously using the hierarchical construction.

In general, computing the representation of an n-dimensional vector in a different basis via an $n \times n$ matrix–vector multiplication requires $O(n^2)$ arithmetic operations. For computing the Haar wavelet transform, however, we can do better. The hierarchical construction given at the beginning of this section requires only $O(n)$ arithmetic operations. In fact, if applying the analysis filter bank to an n-sample signal requires at most kn operations for some constant k (as it does for the Haar filter bank), then the total number of operations to compute the wavelet transform is at most

$$kn + k\frac{n}{2} + k\frac{n}{4} + \cdots + k\frac{n}{2^{\log_2 n}} \leq kn(1 + \frac{1}{2} + \frac{1}{4} + \cdots) \leq 2kn = O(n).$$

Essentially, the hierarchical construction implements a fast matrix-vector multiply by taking advantage of the structure of H_{sys}.

Another informative example to consider is the Haar transform of a constant signal, say $x = [\ c\ c\ c\ c\ c\ c\ c\ c\ ]$. Computing the Haar transform of x is equivalent to decomposing the function $x(t) \equiv c$, $t \in [0, 1)$ in terms of the basis $\Phi_0 \cup \Omega_0 \cup \Omega_1 \cup \Omega_2$ for V^3. Applying the Haar analysis filter bank three times gives

$$x = \begin{bmatrix} c \\ c \\ c \\ c \\ c \\ c \\ c \\ c \end{bmatrix} \to H^{(8)}x = \begin{bmatrix} c \\ c \\ c \\ c \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \to \begin{bmatrix} H^{(4)} \begin{bmatrix} c \\ c \\ c \\ c \end{bmatrix} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} c \\ c \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \to \begin{bmatrix} H^{(2)} \begin{bmatrix} c \\ c \end{bmatrix} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} c \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = y.$$

The output y contains only one nonzero coefficient, namely the coefficient for $\phi_{00}(t) = \phi(t)$. This transform result is obvious from the functional point of view since $x(t) = c\phi(t)$. Discarding the zero coefficients in the output leaves a very compact representation of the signal x (one which is eight times smaller than the original representation). In general, the Haar basis provides a compact representation for parts of a signal with little variation.

For the final example of this section, consider the Haar transform of the delta signal

$$x = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \longrightarrow H_{\mathrm{sys}} x = \begin{bmatrix} \frac{1}{16} \\ -\frac{1}{16} \\ -\frac{1}{8} \\ 0 \\ 0 \\ -\frac{1}{4} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -\frac{1}{2} \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = y.$$

Note that 11 of the 16 output coefficients are zero. This is not the case for the Fourier transform of x which contains energy at all frequencies. The coefficients of the Haar basis functions whose support interval (i.e. the interval over which the basis function is nonzero) does not overlap with the support of the represented function are all zero. The Haar basis allows one to make spacially localized (at some scale) changes to a function by simply adjusting the coefficient(s) for the basis function(s) of the appropriate scale (j) and location (k). This is very different from the Fourier representation in which a change to a single basis function coefficient causes spacially global changes to the represented function.

4 Orthogonal Filter Banks and Wavelet Bases

In Section 1, we showed how to view a filter bank as a transformation represented by a square matrix H (assuming the input and output are the same size). We say that a filter bank is orthogonal if its corresponding matrix is orthogonal. A matrix H is orthogonal if its inverse is equal to its transpose: $H^T H = H H^T = I$. The set of column vectors and the set of row vectors of an orthogonal matrix are both orthonormal sets of vectors (it would make more sense to call such matrices orthonormal, but the historical term for such matrices is *orthogonal*). With a suitable scaling of the input before low pass and high pass filtering, the Haar filter bank is indeed orthogonal. For example, for $n = 4$ we have

$$H^{(n)} = H^{(4)} = \begin{bmatrix} L^{(4)} \\ - \\ B^{(4)} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 1 & & \\ & & 1 & 1 \\ 1 & -1 & & \\ & & 1 & -1 \end{bmatrix},$$

and

$$(H^{(n)})^T H^{(n)} = H^{(n)} (H^{(n)})^T = \frac{1}{2} I.$$

Therefore, the matrix $\sqrt{2}H^{(n)}$ is orthogonal. A modified Haar filter bank which includes a scaling by $\sqrt{2}$ before the low pass and high pass operations is orthogonal.

A basis of functions $f_0, f_1, \ldots$ (for some class of functions on the real line) is an orthonormal basis iff

$$< f_i, f_j > = \int_{t=-\infty}^{\infty} f_i(t) f_j(t)\, dt = \left\{ \begin{array}{ll} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{array} \right. .$$

It is very easy to write a function f in terms of an orthonormal basis. Taking the inner product with f_j with both sides of

$$f = \sum a_i f_i$$

gives

$$\begin{aligned} < f, f_j > &= < \sum_i a_i f_i, f_j > \\ &= \sum_i a_i < f_i, f_j > \\ < f, f_j > &= a_j. \end{aligned}$$

Therefore, the coefficient a_j in the decomposition of f is simply the inner product $< f, f_j >$ of f with the jth basis function f_j. In Section 2 we saw how a filter bank gives rise to a wavelet basis via the dilation and wavelet equations (at least in the case of the Haar filter bank). What conditions on the filters will guarantee an orthonormal wavelet basis?

It turns out that orthogonality of a filter bank implies orthonormality of the basis generated by the filter bank though the dilation and wavelet equations. The proof of this fact will be given in the next lecture. Since the normalized Haar filter bank is orthogonal, this fact implies that the corresponding normalized Haar wavelet basis is orthonormal. The wavelet basis given in Section 2 is not normalized, but its orthogonality can be verified directly from the equations for its scaling function $\phi(t)$ and wavelet function $w(t)$.

Construction of Orthogonal Wavelets

Lecture #3: Tuesday, 7 October 1997
Lecturer: Denis Zorin
Scribe: Liwei He
Reviewer: Scott Cohen

In this lecture we will discuss the the criteria used in the design of wavelet filter banks. We will show the design process using the the Daubechies wavelet as an example (Figure 1).

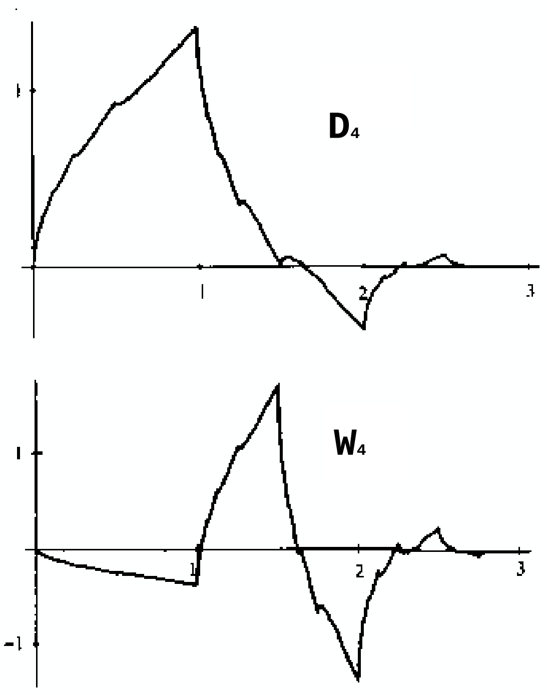


Figure 1: The scaling function of the Daubechies wavelet (for a filter of length 4) is on the top and its wavelet function is at the bottom. Note that the shape of this wavelet and the Haar wavelet resembles waves, hence the name.

1 Properties of Wavelet Bases and Filter Banks

The first step in the design process is to formulate a set of requirements for the filter bank and the associated basis. The following five properties are particularly important:

1. Finite filters. Also known as filters with Finite Impulse Response (FIR). The basis functions derived from such filters have compact support. For efficiency, we would like to make our filters as short as possible.

2. Orthogonality. For orthogonal filter banks one filter essentially defines the whole bank; choosing a synthesis filter with good numerical properties (for example, stability) guarantees that the corresponding analysis filter also has the same properties. For orthonormal bases in functional spaces it is easy to compute the coefficients: we just need to take the dot product of the input function with each basis function. Orthogonal filter banks allow implementations with minimal quantization error.

3. Good approximation. We want to have a set of basis functions that yields a good approximation to functions with as few coefficients as possible. Clearly, this cannot be achieved for arbitrary function; however, we may require that all sufficiently smooth functions are approximated with few coefficients.

4. Symmetry. Nonsymmetric low-pass filters are particularly undesirable for image-processing applications: such filters lead to "smeared" images after low-pass filtering.

5. Regularity. Sometimes it is desirable to have a smooth approximation even after many terms in the approximation are truncated. Suppose we express f as a linear combination of a wavelet basis functions f_i:

$$f = \sum_{i=0}^{\infty} a_i f_i \quad \text{where} \quad a_i \to 0 \quad \text{as} \quad i \to \infty.$$

   The function $g = \sum_{i=0}^{k} a_i f_i$ is an approximation of the function f after we truncate the coefficients after some number k. If the basis is not regular, it is likely that g will not be smooth because the only way to build a smooth function from non-smooth basis functions is to have non-smooth features of basis functions cancel each other.

Unfortunately, the constraints imposed by the requirements of orthogonality and symmetry are too restrictive: the only filter bank that satisfies both requirements is the Haar filter bank. Further, good approximation and high regularity can be achieved only at the expense of increasing the length of the filter. A trade-off has to be made depending on a particular application. For example, the Daubechies wavelet basis has compact support, is orthogonal, and has maximal approximation order among all bases generated by filters of given size.

A set of coefficients $h(k)$ can be computed according to the requirements that we set. We will show how the coefficients of the Daubechies wavelet are constructed in Section 5.

With these coefficients, we can solve for the scaling function ϕ of the wavelet basis using the dilation equation.

2 The Cascade Algorithm

For the Haar wavelet we could guess the solution of the dilation equation. In general, solutions of dilations equations cannot be expressed using elementary functions. Rather then providing formulas for solutions, we describe an algorithm that for a large class of dilation equations yields a solution when one exists. This algorithm is called the *cascade algorithm.*

Recall that the dilation equation is

$$\phi(t) = \sqrt{2}\sum_k c(k)\phi(2t-k). \tag{1}$$

Note that we use normalized filter coefficients, so the coefficient in front of the sum in the right-hand side is $\sqrt{2}$ not 2. Given a set of coefficients $c(k)$, the cascade algorithm solves the dilation equation iteratively. The iteration begins, for example, with the Haar scaling function,

$$\phi^0(t) = \begin{cases} 1 & \text{if } t \in [0,1) \\ 0 & \text{otherwise} \end{cases}.$$

In the ith step, we plug ϕ^{i-1} into the right hand side of the dilation equation to obtain ϕ^i:

$$\phi^i(t) = \sqrt{2}\sum_k c(k)\phi^{i-1}(2t-k). \tag{2}$$

Each iteration hopefully takes us closer to the scaling function we are looking for. The cascade algorithm converges when $\phi^i = \phi^{i-1}$. In the case of the Haar filter bank ($c(0) = \frac{\sqrt{2}}{2}$, $c(1) = \frac{\sqrt{2}}{2}$), convergence is achieved in the first iteration.

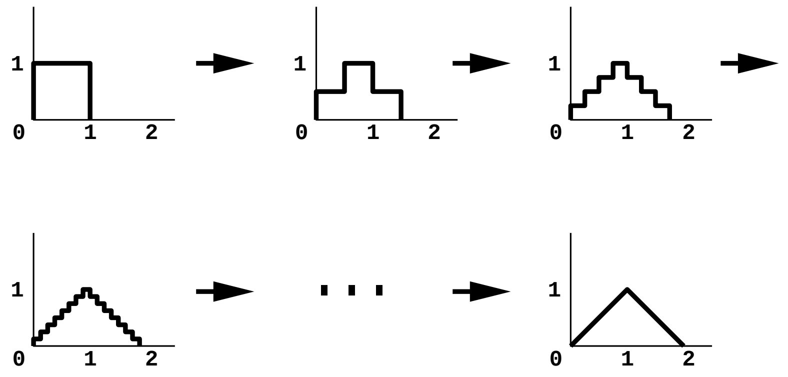


Figure 2: Cascade algorithm uses Haar scaling function as starting point and solves the scaling function iteratively. Shown here are ϕ^0, ϕ^1, ϕ^2, ϕ^3, and ϕ^∞.

A more interesting example is to use a set of coefficients where $c(0) = 1/4$, $c(1) = 1/2$, $c(2) = 1/4$.

$$\phi^1(t) = \sqrt{2}(\frac{1}{4}\phi^0(2t) + \frac{1}{2}\phi^0(2t-1) + \frac{1}{4}\phi^0(2t-2)) \tag{3}$$

In the limit, the scaling function we get, with this particular set of coefficients, is a "hat" function (see Figure 2). Remarkably, this is a linear spline basis function. All other spline basis functions can be found as solutions of dilation equations for a particular choice of coefficients.

Using the cascade algorithm, we can derive some properties of the basis functions generated by a filter bank directly from the properties of the filters, without having explicit expressions for the basis functions themselves.

3 Orthogonality of Wavelet Bases

To illustrate the idea of deriving the properties of the wavelet basis from the properties of a filter, let us prove that if a wavelet filter bank matrix is orthogonal, then the set of wavelet basis functions is orthonormal.

The coefficients to the dilation equation are found in the filter bank matrix F:

$$F = \begin{bmatrix} L \\ - \\ B \end{bmatrix} = \left[\begin{array}{cccc} c(0) & c(-1) & c(-2) & \cdots \\ c(2) & c(1) & c(0) & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \hline d(0) & d(-1) & d(-2) & \cdots \\ d(2) & d(1) & d(0) & \cdots \\ \cdots & \cdots & \cdots & \cdots \end{array}\right]$$

If F is orthogonal, the following relations among the coefficients must hold:

$$\begin{aligned} \sum_n c(n)c(n-2k) &= \delta(k) \qquad (4) \\ \sum_n d(n)c(n-2k) &= 0 \\ \sum_n d(n)d(n-2k) &= \delta(k) \end{aligned}$$

Note that $\delta(k)$ is 1 when $k = 0$ and is 0 elsewhere.

A wavelet basis is composed of the top level scaling functions $\phi(t-k)$ and the wavelet functions $w(2^i t - k)$ at all scales. In order to show that a wavelet basis orthonormal, we must prove that the inner products of between different function are 0, while the inner product of a basis function with itself is 1.

$$\begin{aligned} <\phi(t-k_1), \phi(t-k_2)> &= \delta(k_1 - k_2) \qquad (5) \\ <\phi(t-k_1), w(2^i t - k_2)> &= 0 \qquad (6) \\ <w(2^i t - k_1), w(2^j t - k_2)> &= \delta(i-j)\delta(k_1 - k_2) \qquad (7) \end{aligned}$$

We use proof by induction in the context of the cascade algorithm to show Equation (5) first. Note that it is sufficient to prove Equation (5) for $k_1 = 0$, because we can reduce the general case to the the case $t_1 = 0$ by replacing t with $t + k_1$.

We start from the base case $\phi^0(t-k)$, which are the Haar scaling functions. They are known to be orthonormal:

$$<\phi^0(t), \phi^0(t-k)> = \delta(k)$$

then we will prove that

$$<\phi^i(t), \phi^i(t-k)> = \delta(k) \quad \Longrightarrow \quad <\phi^{i+1}(t), \phi^{i+1}(t-k)> = \delta(k)$$

From the dilation equation,

$$\begin{aligned}<\phi^{i+1}(t), \phi^{i+1}(t-k)> &= \int [\sqrt{2}\sum_{k_1} c(k_1)\phi^i(2t-k_1)][\sqrt{2}\sum_{k_2} c(k_2)\phi^i(2t-k_2)] \\ &= 2\sum_{k_1 k_2} c(k_1)c(k_2)\int \phi^i(2t-k_1)\phi^i(2(t-k)-k_2)dt\end{aligned}$$

Let $T = 2t - k_1$, then $t = (T+k_1)/2$, $dt = dT/2$, and

$$<\phi^{i+1}(t), \phi^{i+1}(t-k)> = \sum_{k_1 k_2} c(k_1)c(k_2)\int \phi^i(T)\phi^i(T+k_1-k_2-2k)dT$$

By the induction hypothesis $<\phi_i(t), \phi_i(t-k)> = \delta(k)$,

$$<\phi^{i+1}(t), \phi^{i+1}(t-k)> = \sum_{k_1 k_2} c(k_1)c(k_2)\delta(k_1-k_2-2k)$$

By the definition of δ, the only non-zero terms in the summation are those when $k_2 = k_1 - 2k$,

$$<\phi^{i+1}(t), \phi^{i+1}(t-k)> = \sum_{k_1} c(k_1)c(k_1-2k)$$

Now let $n = k_1$,

$$<\phi^{i+1}(t), \phi^{i+1}(t-k)> = \sum_{n} c(n)c(n-2k)$$

This is $\delta(k)$ given by Equation (5). Thus Equation (5) is true. Equation (6) and Equation (7) can be proven by similar arguments.

4 Approximation

A good approximating basis will have only a few large coefficients for a smooth function and leave the rest relatively small. If we truncate the summation, the reconstructed function is still very close to the original f.

The quality of approximation for a basis is related to the number of the *vanishing moments* of the basis functions. We say that a function has P vanishing moments if

$$\int f_i t^k dt = 0 \quad \text{where} \quad k \in [0..P-1]$$

If $f(t)$ is smooth around some point t_0, it can be represented by a Taylor's series expansion with remainder $R(t)$,

$$f(t) = \sum_{j=0}^{P-1} \frac{f^{(j)}(t_0)(t-t_0)^j}{j!} + (t-t_0)^P R(t)$$

and the coefficient a_i for the basis function f_i can be computed by taking the dot product of $f(t)$ and each component function of the basis, since the basis is orthonormal[1]

$$\begin{aligned} a_i &= < f, f_i > \\ &= \int f(t) f_i(t) dt \\ &= \int [\sum_{j=0}^{P-1} \frac{f^{(j)}(t-t_0)^j}{j!} + (t-t_0)^P R(t)] f_i(t) dt \\ &= \sum_{j=0}^{P-1} \int \frac{f^{(j)}(t-t_0)^j}{j!} f_i(t) dt + \int (t-t_0)^P R(t) f_i(t) dt \end{aligned}$$

If the basis function f_i has P vanishing moments, all the terms in the summation will vanish since they are all linear combinations of monomials t^k where $k < P$.

$$a_i = \int (t-t_0)^P R(t) f_i(t) dt$$

Suppose the original wavelet function $w(t)$ has support I_0 and $f_i(t) = w(2^j t - k)$, with $\frac{k}{2^j} \approx t_0$. Then $f_i(t)$ has support I with $|I| = \frac{|I_0|}{2^j}$, and

$$a_i \leq C \int_I |t-t_0|^P dt \leq D 2^{-jP} \quad C, D \text{ are some constants not depending on } P$$

This formula indicates that the magnitude of coefficients rapidly decreases as P grows.

The following condition on filters ensures that the wavelets have P vanishing moments:

$$\sum_n (-1)^n n^k h_0(k) = 0 \quad \text{where} \quad k \in [0..P-1] \tag{8}$$

We state this condition without a proof.

5 Computing the Daubechies Filter Bank Coefficients

Now we are ready to compute the coefficients of the Daubechies filter bank. Using the cascade algorithm we can find the scaling function and the wavelet with arbitrary precision.

[1]All integrals without specified range are taken over $(-\infty \ldots + \infty)$.

The Daubechies filter bank is orthogonal and FIR, and has the best approximation for a given filter length. We compute the coefficients for the case when the filter length is assumed to be 4. Our filter bank matrix will look something like this,

$$\begin{bmatrix} c_0 & c_1 & c_2 & c_3 & 0 & 0 & 0 & \cdots \\ 0 & 0 & c_0 & c_1 & c_2 & c_3 & 0 & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix}$$

If the coefficients satisfy

$$\begin{aligned} c_0^2 + c_1^2 + c_2^2 + c_3^2 &= 1 && (9) \\ c_0 c_2 + c_1 c_3 &= 0, && (10) \end{aligned}$$

then the filter bank matrix will be orthogonal. As we have proved in Section 3, this guarantees that the wavelet basis is orthonormal. Since we have four coefficients, we still have two degrees of freedom left to maximize the approximation order. Using Equation (8), we get the two remaining constraints, that ensure that the wavelet has two vanishing moments:

$$\begin{aligned} c_0 - c_1 + c_2 - c_3 &= 0 \quad \text{when} \quad k = 0 && (11) \\ -c_1 + 2c_2 - 3c_3 &= 0 \quad \text{when} \quad k = 1 && (12) \end{aligned}$$

By solving the system of equations together with Equation (9) and (10), we get

$$c_0 = \frac{1+\sqrt{3}}{4\sqrt{2}}, \quad c_1 = \frac{3+\sqrt{3}}{4\sqrt{2}}, \quad c_2 = \frac{3-\sqrt{3}}{4\sqrt{2}}, \quad c_3 = \frac{1-\sqrt{3}}{4\sqrt{2}}$$

For longer filters, we can obtain bases with more vanishing moments. The equations for coefficients can be solved explicitly only for small filter lengths; for longer filters the values for coefficients can be computed numerically.

Biorthogonal Wavelets

Lecture #4: Thursday, 9 October 1997
Lecturer: Denis Zorin
Scribe: Pierre Louveaux

In the previous lecture we dropped symmetry in favor of orthogonality, and thus arrived at the Daubechies wavelets. This time around we relinquish orthogonality and settle for the next best thing, *biorthogonality.* Without orthogonality, the construction of our filters becomes subject to far fewer constraints; as a result, the process of meeting the other desired criteria is more involved.

1 Giving Up on Orthogonality

Recall the set of properties we identified as desirable for a wavelet filter:

1. Finite Impulse Response.
2. Orthogonality.
3. Good approximation.
4. Symmetry.
5. Regularity (smoothness).

Unfortunately, enforcing FIR, orthogonality, and symmetry together yields nothing better than the Haar case. To see this, consider a symmetric signal of length, say, 6:

$$\begin{array}{rcccccc} & c(0) & c(1) & c(2) & c(3) & c(4) & c(5) \\ = & c(0) & c(1) & c(2) & c(2) & c(1) & c(0) \end{array}$$

Now consider various even shifts of this signal:

$$\begin{array}{cccccccccc} c(0) & c(1) & c(2) & c(2) & c(1) & c(0) & & & & \\ 0 & 0 & c(0) & c(1) & c(2) & c(2) & c(1) & c(0) & & \\ 0 & 0 & 0 & 0 & c(0) & c(1) & c(2) & c(2) & c(1) & \cdots \end{array}$$

By hypothesis, $c(0) \neq 0$ (since the signal has length 6). Because of the orthogonality condition

$$\sum_n c(n)c(n-2k) = \delta(k)$$

we must have $c(2)c(0) = 0$ hence $c(2) = 0$, and $c(1)c(0) = 0$ hence $c(1) = 0$. Every coefficient except $c(0)$ must be 0, so all finite, symmetric, orthogonal filters have the form

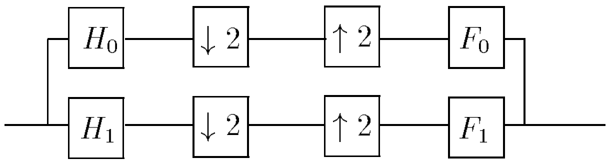
$$c(0) \quad 0 \quad 0 \quad \cdots \quad 0 \quad 0 \quad c(0) \ .$$

Of those, the only reasonable low-pass filter is the one with length 2, i.e., the Haar filter.

2 Perfect Reconstruction from General Filter Banks

In Lecture #1 we saw that if we use the Haar filter pair in our analysis bank, all we need for perfect reconstruction of the input signal is another copy of the Haar filter pair, with the sign of H_1 flipped, in our synthesis bank. But what are the conditions for perfect reconstruction in the general case?

Here is the general structure we will consider:



analysis bank synthesis bank

The analysis bank can be represented as a single matrix $\left[\frac{L}{B}\right]$, where L corresponds to H_0 followed by downsampling and B corresponds to H_1 followed by downsampling. In the case of an orthogonal filter bank, $\left[\frac{L}{B}\right]$ is orthogonal. It is also possible to use a non-orthogonal $\left[\frac{L}{B}\right]$, but for perfect signal reconstruction the synthesis bank must be of the form $\left[\ \tilde{L}\ \middle|\ \tilde{B}\ \right]$ such that

$$\left[\ \tilde{L}\ \middle|\ \tilde{B}\ \right] \cdot \left[\frac{L}{B}\right] = \tilde{L} \cdot L + \tilde{B} \cdot B = I$$

as well as

$$\left[\frac{L}{B}\right] \cdot \left[\ \tilde{L}\ \middle|\ \tilde{B}\ \right] = \left[\begin{array}{c|c} L \cdot \tilde{L} & L \cdot \tilde{B} \\ \hline B \cdot \tilde{L} & B \cdot \tilde{B} \end{array}\right] = \left[\begin{array}{c|c} I & 0 \\ \hline 0 & I \end{array}\right] \tag{1}$$

(where each I represents an identity matrix and each 0 a zero matrix). Equation (1) gives us constraints in terms of coefficients:

$$L \cdot \tilde{L} = I \iff \sum_n h_0(k) f_0(k-2n) = \delta(n)$$

$$L \cdot \tilde{B} = 0 \iff \sum_n h_0(k) f_1(k-2n) = 0 \tag{2}$$

$$B \cdot \tilde{L} = 0 \iff \sum_n h_1(k) f_0(k-2n) = 0 \tag{3}$$

$$B \cdot \tilde{B} = I \iff \sum_n h_1(k) f_1(k-2n) = \delta(n)$$

3 Alternating Flip

Conditions (2) and (3) above denote the biorthogonal nature of the system: L is orthogonal to $\tilde{B}$ and B is orthogonal to $\tilde{L}$. We can ensure that some of the biorthogonality conditions are satisfied requiring that the coefficients of the mutually orthogonal filters be the same but in reverse order and with every other sign flipped:

$$f_0(k) = (-1)^k h_1(N-k)$$

$$f_1(k) = -(-1)^k h_0(N-k)$$

(N is the order of the filters.)

It also turns out that, if we use alternating flip, $L \cdot \tilde{L} = I$ is equivalent to $B \cdot \tilde{B} = I$; so we are now left with only one condition to worry about: how to choose f_0 and h_0.

We no longer have a single dilation equation as in the orthogonal case, but rather a pair of dilation equations:

$$\phi(t) = 2\sum_k h_0(k)\phi(2t-k)$$

$$\tilde{\phi}(t) = 2\sum_k f_0(k)\tilde{\phi}(2t-k)$$

What does this mean in the continuous domain? There is a nice interpretation in terms of bases; but first we should introduce the concept of *multiresolution analysis.*

4 Multiresolution Analysis

All wavelets exhibit a multiresolution structure in the following sense: they are made up of nested sets of increasingly refined and powerful bases. Given a scaling function ϕ and its associated wavelet function w, we can form the space

$$V_0 = \text{span}\langle\phi(t-k)\rangle = \text{span}\langle\phi(2^0 t-k)\rangle$$

spanned by the translates of ϕ, and the space

$$W_0 = \text{span}\langle w(t-k)\rangle = \text{span}\langle w(2^0 t-k)\rangle$$

spanned by the translates of w. By scaling ϕ we can also build

$$V_1 = \text{span}\langle w(2t-k)\rangle = \text{span}\langle w(2^1 t-k)\rangle,$$

which (provided that w is the wavelet derived from ϕ) will be exactly the sum of V_0 and W_0:

$$V_1 = V_0 + W_0$$

What this means is that any function in V_1 can be represented as the sum of a function in V_0 and a function in W_0. Further scaling of ϕ and w by powers of 2 allows us to define an infinite sequence of spaces V_j and W_j:

$$V_j = \text{span}\langle \phi(2^j t - k)\rangle$$

$$W_j = \text{span}\langle w(2^j t - k)\rangle$$

At every level we have $V_j = V_{j-1} + W_{j-1}$, which implies the infinite nesting of the bases:

$$V_0 \subset V_1 \subset V_2 \subset \ldots$$

Note: All of the above applies in both orthogonal and biorthogonal cases; but if the V_j's and W_j's are mutually orthogonal, i.e.,

$$V_j \perp W_j \text{ for all } j$$

or (by definition)

$$\langle f_{V_j} \cdot f_{W_j}\rangle = 0 \text{ for all functions } f_{V_j} \in V_j \text{ and } f_{W_j} \in W_j,$$

then we have *orthogonal* multiresolution and we write $V_j = V_{j-1} \oplus W_{j-1}$ instead of $V_j = V_{j-1} + W_{j-1}$.

In summary, here are the defining properties of a multiresolution set $\{V_j\}$ of functional spaces:

1. $V_j \subset V_{j+1}$ and $\bigcup_j V_j = L^2$
2. $f(t) \in V_j \Longleftrightarrow f(2t) \in V_{j+1}$
3. $f(t) \in V_j \Longleftrightarrow f(t-k) \in V_j$

5 Dual Bases

We start with finite-dimensional spaces. Let y be a vector such that $y = \sum a_i y_i$ for some set of coefficients a_i. If the y_i's form an orthonormal basis, finding the coefficients a_i is particularly straightforward: each a_i is equal to the inner product $\langle y_i, y\rangle$.

If the $\{y_i\}$ basis is not orthogonal, however, we lose the luxury of such a simple computation. But it turns out that it is possbile to find a set of vectors $\{\tilde{y}_i\}$ such that

$$a_i = \langle \tilde{y}_i, y\rangle. \tag{4}$$

Since $y = \sum a_i y_i$, we have

$$\langle \tilde{y}_i, y\rangle = \langle y, \tilde{y}_i\rangle = \sum_j a_i \langle y_i, \tilde{y}_j\rangle$$

and the $\tilde{y}_i$ vectors are therefore determined by the set of conditions

$$\langle y_i, \tilde{y}_j \rangle = \delta(i-j) \tag{5}$$

or, in matrix notation,

$$Y \cdot \tilde{Y} = I.$$

This property makes $\{y_i\}$ and $\{\tilde{y}_i\}$ *dual bases*. Two bases are said to be duals of each other if there exists a one-to-one correspondence between them such that the inner product of corresponding basis elements is 1 while the inner product of non-corresponding elements is 0.

The same concept applies to functional spaces. If a filter bank provides perfect reconstruction of the input signal, the bases generated by the synthesis functions $\tilde{\phi}$ and $\tilde{w}$ are duals of the bases generated by the analysis functions ϕ and w. Given $\{V_i\}$ and $\{W_i\}$ as defined in Section 4, this duality can be seen in the relations

$$V_i \perp \tilde{W}_i$$

and

$$\tilde{V}_i \perp W_i.$$

Since the dual wavelets at level i are orthogonal to the primal scaling functions at the same level, i.e.,

$$\langle \tilde{w}(2^i t - k), \phi(2^i t - k) \rangle = 0,^1$$

$\tilde{\phi}$ and $\tilde{w}$ give us a way to compute the coefficients of ϕ and w much in the same way that property (5) allows us to compute the a_i's in (4). To see this, consider the following analysis of an input signal f:

$$f = \sum_k a_k \phi(t-k) + \sum_{i,k} b_{ik} w(2^i t - k)$$

We can obtain the a_k's and b_{ik}'s as follows:

$$\begin{aligned} a_k &= \langle f, \tilde{\phi}(t-k) \rangle \\ b_{ik} &= \langle f, \tilde{w}(2^i t - k) \rangle \end{aligned}$$

6 Where to Enforce Which Conditions

The desirable conditions we started out with were formulated for single wavelet bases, whereas we are now dealing with two wavelet bases — a primary and a dual. How does that change the way we enforce the conditions?

Symmetry is not a problem: an immediate consequence of the alternating flip is that each filter in the primary bank is symmetric with respect to the corresponding filter in the dual bank.

[1] In fact $\langle \tilde{w}(2^i t - k), \phi(2^j t - k) \rangle = 0$ for any $j \leq i$.

Good approximation, in the case of a single wavelet, required a large number of vanishing moments. Now that we have two distinct wavelets w and $\tilde{w}$, which one should have many vanishing moments? The same argument applies as in the single-basis case of Lecture #3; there it was shown that a coefficient a_i would be small if the basis function f_i had many vanishing moments. The derivation hinged on the condition

$$a_i = \langle f, f_i \rangle$$

in order to arrive at the expression

$$a_i = \int (t - t_0)^P R(t) f_i(t) dt$$

and, ultimately, at a bound on the magnitude of a_i. With our dual wavelets, a similar derivation would start out with

$$b_{ik} = \langle f, \tilde{w}(2^i t - k) \rangle.$$

Immediately we see that the dual wavelet $\tilde{w}$ (as opposed to the primal w) is the one that should have many vanishing moments.

As for smoothness, recall that when a function gets projected onto a basis and some of the projected elements are discarded, the truncated projection takes on the local properties of the basis — in particular, smoothness or jaggedness. For instance, if a function

$$f = \sum_{i=0}^{\infty} a_i f_i$$

is approximated by

$$\tilde{f} = \sum_{i=0}^{J} a_i f_i,$$

$\tilde{f}$ takes on the smoothness properties of the f_i's. In compression applications, truncation is applied after the analysis stage, which means that the truncated terms come from the primary basis. Therefore we should strive for a smooth analysis filter. In other words, the smoothness condition should be enforced on the primal w.

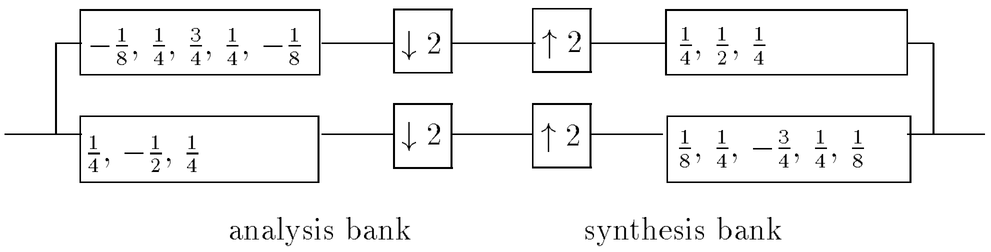
7 Spline Wavelets

We conclude with a simple example of a biorthogonal wavelet basis. As we have seen in the beginning, orthogonality prevents us from constructing finitely supported scaling functions; in particular, we cannot use spline basis functions as scaling functions. Recall our discussion of the cascade algorithm: a linear spline satisfies the scaling equation with coefficients 1/4, 1/2 and 1/4. In addition to symmetry, splines have one more very nice feature: out of all scaling functions with given support, they have maximal smoothness.

This is exactly one of our requirements for the scaling functions/wavelets of the synthesis filter bank.

In the simplest spline filter bank, F_0 is the linear spline filter 1/4, 1/2, 1/4. Recall that we can get the analysis filter H_1 by alternating flip. This leaves us one filter to choose, say, H_0. The simplest filter satisfying Equation (2) has just one coefficient $h(0) = 1$. This completely determines the filter. However, it is clear that the analysis filters are quite bad: the low-pass part of the signal is just subsampled version of the filter. Note that the high-pass filter of the synthesis filter has one vanishing moment. By increasing the support of the analysis filters, we can improve them, ensuring existence of corresponding wavelet basis and greater number of vanishing moments. For example, if the support size is increased to 4, we obtain coefficients $-1/8$, $1/4$, $3/4$, $1/4$, $-1/8$ for the analysis filter H_0. This is so-called 5/3 spline filter bank (the numbers refer to the number coefficients in the analysis and synthesis low-pass filters.)

This filter bank is shown below.


$-\frac{1}{8}, \frac{1}{4}, \frac{3}{4}, \frac{1}{4}, -\frac{1}{8}$
$\frac{1}{4}, -\frac{1}{2}, \frac{1}{4}$
$\downarrow 2$
$\downarrow 2$
$\uparrow 2$
$\uparrow 2$
$\frac{1}{4}, \frac{1}{2}, \frac{1}{4}$
$\frac{1}{8}, \frac{1}{4}, -\frac{3}{4}, \frac{1}{4}, \frac{1}{8}$
analysis bank
synthesis bank

Application of Wavelets in Graphics

Lecture #5: Tuesday, Oct 14 1997
Lecturer: Eric Veach
Scribe: Li-Yi Wei
Reviewer: Li-Wei He

In this lecture we will talk about various applications of wavelets in graphics, with concentration on wavelet radiosity. Typical applications of wavelets in graphics and image processing include:

1. Image editing, compression, and retrieval [3, 4].
2. Curve and surface editing, compression, and variational modeling [2, 6].
3. Global illumination problems (e.g. radiosity) [7, 8, 9].

The book *Wavelets for Computer Graphics: Theory and Applications* by Stollnitz, DeRose, and Salesin [1] gives an overview of these applications.

1 Image Compression using Wavelets

One successful application of wavelets in image compression is finger print file compression [5]. The FBI has about 30 million finger print images, each with size about 10M bytes. Wavelet compression outperforms JPEG in compressing those finger print files. The main problem with JPEG is that it introduces blocking artifacts at high compression ratios, which can obscure the fingerprint "minutiae" (e.g. ridge endings).

Generally, image compression consists of the following steps:

1. Choose a basis. First we may want to divide the image into fixed sized blocks (e.g. as in JPEG). Then we choose a set of basis functions that has the desired properties, such as smoothness, vanishing moments, symmetry.
2. Transformation of the image, by projecting it into the chosen basis.
3. Quantization of the coefficients.
4. Coding. After quantization we could further compress the coefficients using lossy or lossless compression techniques.

The main reason for dividing the image into fixed sized blocks is to make the basis functions local. With wavelets this is not necessary since the basis functions already have local support.

Given a set of 1D scaling functions $\{\phi_{i,j}(x)\}$ and wavelets $\{\psi_{i,j}(x)\}$, there are two ways to construct a 2D basis: standard basis and non-standard basis.

1.1 Standard Basis

In this case, we first apply a wavelet transform to all the rows of the original image, then we apply a wavelet transform to all the columns of the resulting coefficients. Let

$$\begin{aligned} V_i &= \text{Span}\{\phi_{i,j}(x)\}, & W_i &= \text{Span}\{\psi_{i,j}(x)\}, \\ V_i' &= \text{Span}\{\phi_{i,j}(y)\}, & W_i' &= \text{Span}\{\psi_{i,j}(y)\} \end{aligned}$$

from previous lectures we know

$$\begin{aligned} V_n &= V_0 \oplus W_0 \oplus W_1 \cdots \oplus W_{n-1} \\ V_n' &= V_0' \oplus W_0' \oplus W_1' \cdots \oplus W_{n-1}' \end{aligned}$$

Assume that the initial image $I \in V_n \times V_n'$, we can expand $V_n \times V_n'$ to be

$$V_n \times V_n' = V_0 \times V_0' \quad \oplus \quad \sum_{j=0}^{n-1} V_0 \times W_j' \quad \oplus \quad \sum_{j=0}^{n-1} W_j \times V_0' \quad \oplus \quad \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} W_i \times W_j'$$

In other words, we project the original image $I \in V_n \times V_n'$ into the following set of basis functions:

$$\begin{array}{ll} \phi_{00}(x)\phi_{00}(y) & \phi_{00}(x)\psi_{j'k'}(y) \\ \phi_{jk}(x)\psi_{00}(y) & \psi_{jk}(x)\psi_{j'k'}(y) \end{array}$$

where $j = 0 \dots n-1$, $k = 0 \dots 2^j - 1$, and similarly for j', k'.

The disadvantage of the above set of basis functions is that they have bad support shape. For example, for those $\phi_{00}(x)\psi_{j'k'}(y)$, they could be wide in the x dimension and narrow in the y dimension(they look like horizontal strips). Those basis functions won't represent the original image locally, so they are a bad set of basis functions.

1.2 Non-standard Basis

In contrast with standard basis function, non-standard basis transforms the original image $I_n \in V_n \times V_n'$ in an interleaved fashion. First we use one pass of wavelet transform to all the rows, then we do the same thing for all the columns. After one pass the image will look like this:

$$\begin{pmatrix} \begin{pmatrix} V_{n-1} \times V_{n-1}' \end{pmatrix} & \begin{pmatrix} V_{n-1} \times W_{n-1}' \end{pmatrix} \\ \begin{pmatrix} W_{n-1} \times V_{n-1}' \end{pmatrix} & \begin{pmatrix} W_{n-1} \times W_{n-1}' \end{pmatrix} \end{pmatrix}$$

We can repeat the above process to the upper-left subimage $I_{n-1} \in V_{n-1} \times V_{n-1}'$ and conitue in similar fashion until we have $I_0 \in V_0 \times V_0'$. The above operations could be written algebraically as:

$$\begin{aligned} V_n \times V_n' &= V_0 \times V_0' \quad \oplus \quad \sum_{i=1}^{n}(V_i \times V_i' - V_{i-1} \times V_{i-1}') \\ &= V_0 \times V_0' \quad \oplus \quad \sum_{i=1}^{n}(V_{i-1} \times W_{i-1}' \quad \oplus \quad V_{i-1}' \times W_{i-1} \quad \oplus \quad W_{i-1} \times W_{i-1}') \end{aligned}$$

In other words, we project the original image $I \in V_n \times V_n'$ into the following set of basis functions:

$$\begin{array}{ll} \phi_{00}(x)\phi_{00}(y) & \phi_{jk}(x)\psi_{jk'}(y) \\ \phi_{jk}(x)\psi_{jk'}(y) & \psi_{jk}(x)\psi_{jk'}(y) \end{array}$$

where $j = 0 \ldots n-1$ and $k, k' = 0 \ldots 2^j - 1$.

For non-standard basis, the filter supports are more localized than standard basis since all the products only involve scaling functions and wavelet functions of the same level. As you might expect the non-standard basis usually gives more compression ratio than the standard basis.

2 Wavelet Radiosity

The final section will talk about wavelet radiosity. For more information, please consult [8], [9] and [10].

The radiosity equation could be written as:

$$B(x) = E(x) + \int K(x,y)B(y)\,dy \tag{1}$$

where x and y are points on 2D scene surfaces.

Since the above equation is an integral equation, we have to project it into a finite basis to solve it. In traditional radiosity method, we first divide the environment into N surface patches, associate N basis functions with those patches, project the above equation into a matrix form, and use numerical algorithms like Jacobian iteration to solve the linear system. The problem is that it's very expensive to project the kernel $K(x,y)$ into a $N \times N$ matrix. In [7] they use hierarchical algorithm to reduce the computation time from $O(N^2)$ to $O(k^2 + N)$ using box basis functions, where k is the number of initial patches and N is the number of subdivided elements. [8] and [9] further extends the ideas by using wavelet basis functions. The merit is that by doing so we can get a better discretization of the kernel $K(x,y)$, thus throwing away those smaller coefficients does not hurt too much.

Let $V = \text{Span}\{B_i(x), i = 1 \ldots N\}$ to be the function space spanned by basis $\{B_i(x)\}$. The approximate solution $\hat{B}(x)$ is represented as a linear combination of these basis functions:

$$\hat{B}(x) = \sum_{i=1}^{N} b_i B_i(x) \;\; .$$

Assume that $\{B_i(x)\}$ is orthonormal, and define operator P_v to be the projection of a function f to V, i.e.

$$P_v(f) = \sum_i \langle f, B_i \rangle B_i$$

Another way to think about this is that P_v transforms a function f into a column vector V_f which is of dimension N and contains the coefficients of the projection.

We now solve the approximate equation

$$\hat{B}(x) = (P_v E)(x) + (P_v K P_v \hat{B})(x)$$

where $\hat{K} = P_v K P_v$ is the approximate transport operator. Because of the leftmost P_v, $\hat{B}(x)$ lies in the subspace V and can be represented by its coefficients b_i.

Mathematically, let

$$\begin{aligned}
\hat{E}(x) &= \sum_{i=1}^{N} \langle E(x), B_i(x)\rangle B_i(x) \;=\; \sum_{i=1}^{N} e_i B_i(x) \\
\hat{K}(x,y) &= \sum_{i=1}^{N}\sum_{j=1}^{N} \langle\langle K(x,y), B_i(x)\rangle, B_j(y)\rangle B_i(x) B_j(y) \\
&= \sum_{i=1}^{N}\sum_{j=1}^{N} k_{ij} B_i(x) B_j(y)
\end{aligned}$$

Then

$$\begin{aligned}
\int \hat{K}(x,y)\hat{B}(y)\,dy &= \int (\sum_{i=1}^{N}\sum_{j=1}^{N} k_{ij} B_i(x) B_j(y))(\sum_{k=1}^{N} b_k B_k(y))\,dy \\
&= \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N} k_{ij} b_k B_i(x) \int B_j(y) B_k(y)\,dy \\
&= \sum_{i=1}^{N}\sum_{j=1}^{N} k_{ij} b_j B_i(x)
\end{aligned}$$

since $\{B_i\}$ is orthonormal.

If we put the above result into equation (1) we have

$$b_i = e_i + \sum_{j=1}^{N} k_{ij} b_j$$

or in matrix form

$$\hat{B} = \hat{E} + \hat{K}\hat{B}, \quad \text{where}$$

$$\hat{B} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{pmatrix} \quad \hat{E} = \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_N \end{pmatrix} \quad \hat{K} = \begin{pmatrix} k_{11} & k_{12} & \cdots & k_{1N} \\ k_{21} & k_{22} & \cdots & k_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ k_{N1} & k_{N2} & \cdots & k_{NN} \end{pmatrix}$$

The above linear system is the discretized version of equation (1). We could solve it using other numerical algorithms and get an approximate solution for the projected radiosity function $\hat{B}$.

For wavelet radiosity, wavelets are used for the basis functions $B_i(x)$. Note that for 3D scene, the variables x and y are actually 2D surface points, so the kernel $K(x, y)$ is a 4D function. Thus we need to project the kernel onto basis functions that are a product of four 1D basis functions. This can be done in either the standard or non-standard ways as described above.

If we compare image compression using wavelets and wavelet radiosity we can find many similarities between them. The digital image is a discretized version of some continous image. In otherwords, we can think we project the original continous image into a set of basis functions and use the coefficients to represent the original image. In both image compression and wavelet radiosity, we try to find a good basis so that we can throw away many coefficients with sacrificing the quality.

The following compares the results of wavelet radiosity in flatland using different bases:

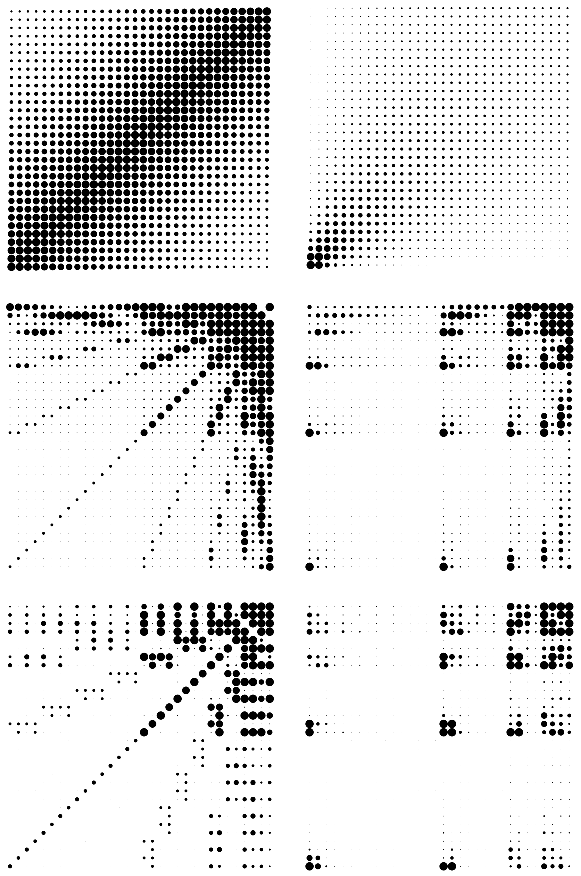


Figure 1: The left column shows the matrix of the flatland kernel for two parallel line segments in the finest level basis, the standard Haar basis and the standard F_2 basis(top to bottom). The right column shows the matrix of the flatland kernel for two line segments meeting at right angles in the same three bases.

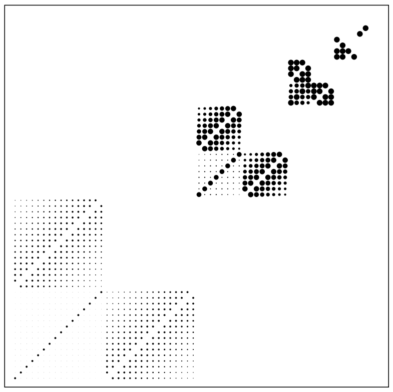


Figure 2: The kernel for two parallel lines realized in the non-standard Haar basis.

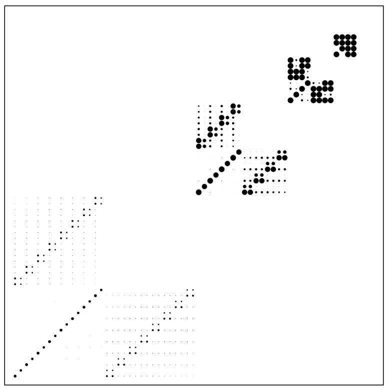


Figure 3: The kernel for two parallel lines realized in the non-standard F_2 basis.

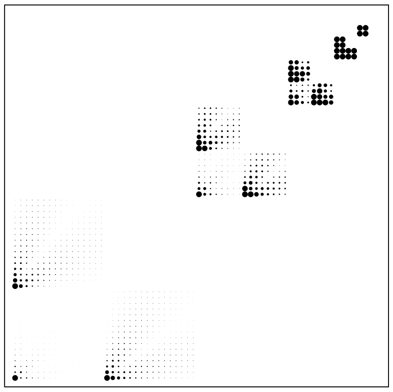


Figure 4: The kernel for two perpendicular lines realized in the non-standard Haar basis.

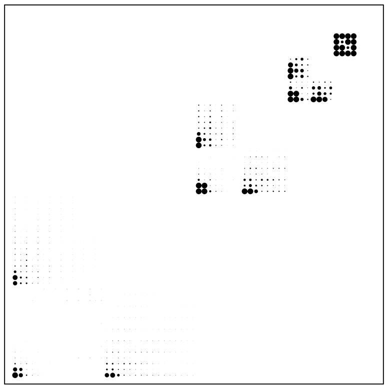


Figure 5: The kernel for two perpendicular lines realized in the non-standard F_2 basis.

Figure 1 compares Haar basis with F_2 basis for standard case. We can see F_2 gives better result since the resulting matrix contains more small elements. Figure 2, 3, 4, 5 show that for non-standard case, F_2 basis gives better result than Haar basis. Figure 6, 7 compares more different bases in wavelet radiosity.

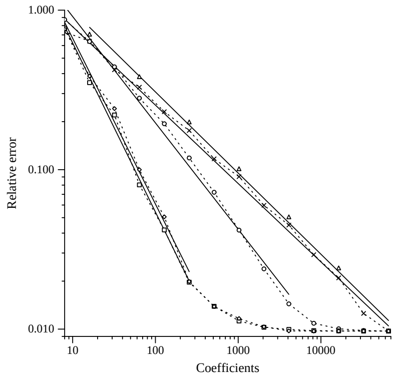


Figure 6: Parallel segments: Relative L^1 error plotted against number of coefficients used with full matrices, non-standard haar, standard haar, non-standard F_2, and standard F_2 bases (top to bottom).

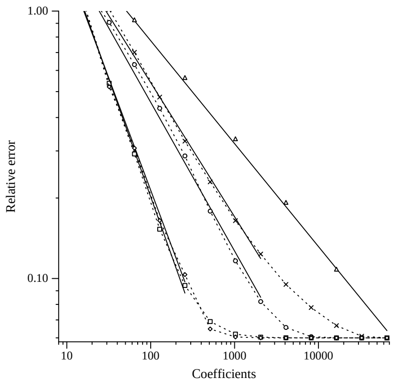


Figure 7: Perpendicular segments: Relative L^1 error plotted against number of coefficients used with full matrices, non-standard haar, standard haar, non-standard F_2, and standard F_2 bases (top to bottom).

References

[1] Eric J. Stollnitz, Tony D. DeRose, and David H. Salesin *Wavelets for Computer Graphics: Theory and Applications.* Morgan Kauffmann, San Francisco, 1996.

[2] Adam Finkelstein, David H. Salesin. *Multiresolution Curves.* In Proceedings of SIGGRAPH '94, pages 261-268. ACM, New York, 1994.

[3] Deborah F. Berman, Jason T. Bartell, David H. Salesin. *Multiresolution Painting and Compositing.* Proceedings of SIGGRAPH 94, in Computer Graphics Proceedings, Annual Conference Series, 85-90, July 1994.

[4] Charles E. Jacobs, Adam Finkelstein, David H. Salesin. *Fast Multiresolution Image Querying.* Proceedings of SIGGAPH 95, in Computer Graphics Proceedings, Annual Conference Series, pages 277-286, August 1995.

[5] *Fingerprints go digital.* Survey article directed towards mathematicians. In: Notices Amer. Math. Soc., v. 42 no. 11, pp. 1278-1283, Nov. 1995. (formerly AAAS95) Author: C. Brislawn.

[6] Gortler, S.J. et al. *Hierarchical and variational geometric modeling with wavelets* Proceedings 1995 Symposium on Interactive 3D Graphics. Held: Monterey, CA, USA 9-12 April 1995. Proceedings 1995 Symposium on Interactive 3D Graphics (1995) p. 35-42, 205 (Conference Paper)

[7] Hanrahan, P., Salaman, D., and Aupperle, L. *A Rapid Hierarchical Radiosity Algorithm. Computer Graphics 25, 4* (July 1991), 197-206

[8] Steven J. Gortler, Peter Schroder, Michael F. Cohen and Pat Hanrahan. *Wavelet Radiosity.* In *Computer Graphics, Annual Conference Series, 1003* (August 1993), Siggraph.

[9] Peter Schroder, Steven J. Gortler, Michael F. Cohen and Pat Hanrahan. *Wavelet Projections for Radiosity.*

[10] Michael F. Cohen and John R. Wallace. *Radiosity and Realistic Image Synthesis.* Academic Press Professional, 1993

Introduction to Monte Carlo Integration

Lecture #6:	Thursday, 16 October 1997
Lecturer:	Eric Veach
Scribe:	Lucas Pereira
Reviewer:	Li-Yi Wei

1 The Development of Monte Carlo Methods

Monte Carlo integration methods were first used on a computer, the ENIAC, just after World War II. The ENIAC, completed in 1946, was the first electronic (as opposed to electromechanical) computer in the United States, following the Colossus completed in England in 1943. It was huge (24 meters long, 18,000 vacuum tubes), and could run at a (then) phenomenal rate of 5000 operations per second.

At that time, scientists at the Los Alamos National Laboratory were working on building an H-bomb, and were considering how the ENIAC could help them. Stanislaw Ulam suggested that random sampling could be used to simulate the flight paths of neutrons. John Von Neumann expanded on the idea, and came up with a detailed proposal in 1947. He estimated that, following 100 neutrons, they could compute one round of collisions in 3 minutes. This was fast enough so that a simulation of 100 collisions could be completed in under 5 hours. Nick Metropolis named the new method "Monte Carlo", after the city in Monaco famous for its casinos.

In 1949, Metropolis and Ulam published a paper on Monte Carlo methods, which sparked a lot of work on the methods in the 50's. Unfortunately, the computers at that time were only capable of handling trivial examples of many applications. Also, some researchers tried to apply Monte Carlo methods to every problem in sight, even those that were already well solved using previous techniques. These mis-applications gave Monte Carlo methods a bad reputation for a while. However, things gradually improved over the 60's as people discovered more efficient Monte Carlo techniques, and they learned which problems were and were not appropriate for the methods. Monte Carlo methods are now an important tool for solving many numerical problems.

2 A Brief History of Random Sampling

In isolated instances, random sampling had been used much earlier to solve numerical problems. For example, in 1777 the Comte de Buffon computed that, given parallel

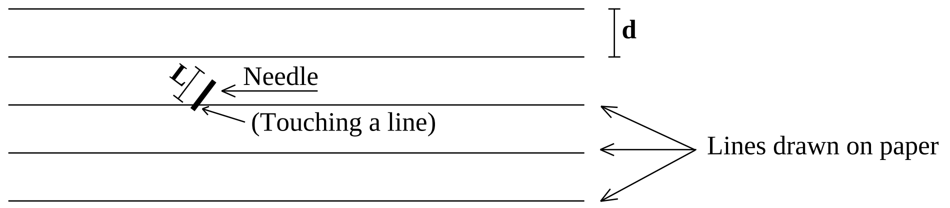


Figure 1: Using a needle to estimate π.

ruled lines separated by a distance d, and a needle of length L (where $L < d$) dropped randomly, the probability of the needle touching a line would be:

$$P_{touch} = \frac{2L}{\pi d} \ .$$

He verified this experimentally. Laplace, 100 years later, noticed that you could actually use this method to estimate π. In the latter part of the 19th century, this became quite a popular experiment (or party game).

Similarly, in 1900 Lord Kelvin was researching the kinetic theory of gasses, and used random sampling to help with his calculations. His random number generator consisted of writing numbers on slips of paper, putting them in a jar, and pulling them out one at a time. He was quite concerned that they weren't getting mixed well enough, due to such effects as static electricity.

As a final example, Student (an alias for W.S. Gosset) used random sampling in 1908 as an aid to deriving his famous t-distribution. He did not know the exact distribution, so he did some experiments to help him guess the analytic form.

3 Numerical Quadrature

Suppose we want to calculate some integral,

$$I = \int_a^b f(x)\, dx \ .$$

(We will do this example in 1-D, to keep things simple.)

The best way to calculate this integral would be to solve it analytically, and get:

$$I = \int_a^b f(x)\, dx = F(b) - F(a) \ .$$

However, there are many functions we would like to integrate that we cannot do analytically. So people have developed methods for approximating the integral through *quadrature rules* of the form:

$$\hat{I} = \sum_{i=1}^{n} w_i f(x_i) \ ,$$

which is essentially a weighted sum of samples of the function at various points. There are many different quadrature rules, each with their own sampling patterns and weights. A few common ones include:

3.1 Midpoint Rule:

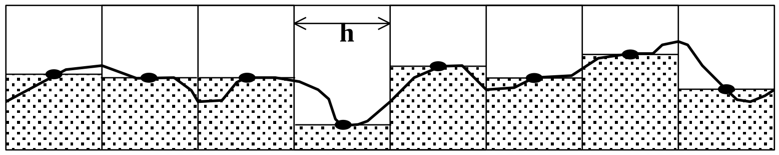


Figure 2: The midpoint rule.

We divide up the interval into some fixed number n of intervals, each of size $h = (b-a)/n$. We then choose one sample point at the midpoint of each interval:

$$\begin{aligned}\hat{I} &= h\sum_{i=1}^{n} f(a+(i-\frac{1}{2})h) \\ &= h\left[f(a+\frac{h}{2}) + f(a+\frac{3h}{2}) + \cdots + f(b-\frac{h}{2})\right].\end{aligned}$$

3.2 Trapezoid Rule:

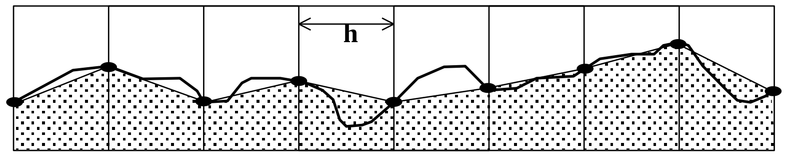


Figure 3: The trapezoid rule.

This is similar to the midpoint rule, except we sample the function at the ends of each interval, and compute the area of a trapezoid for each interval.

$$\begin{aligned}\hat{I} &= \sum_{i=1}^{n} \frac{h}{2}\left[f(a+(i-1)h) + f(a+ih)\right] \\ &= h\left[\frac{1}{2}f(a) + f(a+h) + f(a+2h) + \cdots + f(b-h) + \frac{1}{2}f(b)\right].\end{aligned}$$

3.3 Simpson's Rule:

This is similar to the trapezoid rule, except we compute the area under a quadratic polynomial approximation (instead of a linear approximation for the trapezoid). The equation is:

$$\begin{aligned}\hat{I} &= h\left[\frac{1}{3}f(a) + \frac{4}{3}f(a+h) + \frac{2}{3}f(a+2h) + \frac{4}{3}f(a+3h) + \frac{2}{3}f(a+4h) + \right. \\ &\qquad \left. \cdots + \frac{4}{3}f(b-h) + \frac{1}{3}f(b)\right] .\end{aligned}$$

3.4 Convergence:

The Midpoint Rule is exact for constant or linear functions ($f \sim 1, x$). Its error is given by

$$\hat{I} - I = -\frac{(b-a)^3}{24n^2} f''(\xi) = O(n^{-2})$$

where $a \le \xi \le b$, provided that f has at least two continuous derivatives on $[a,b]$. For the trapezoid rule, the error is

$$\hat{I} - I = \frac{(b-a)^3}{12n^2} f''(\xi^*) = O(n^{-2}) .$$

So, the midpoint and trapezoid rules have the same convergence rate, and their errors have opposite signs if $f'' > 0$ (i.e. the true answer is bounded between them).

Simpson's Rule is exact for polynomial functions up to cubics ($f \sim 1, x, x^2, x^3$). The error can be bounded by the fourth derivative:

$$|\hat{I} - I| = \frac{(b-a)^5}{180(2n)^4} f^{(4)}(\xi) = O(n^{-4}) .$$

This converges very quickly, assuming that f has a continuous fourth derivative. There are higher-order rules called *Newton-Cotes* rules that can achieve even faster convergence, but require the function to be even smoother. Another popular family of integration rules are the *Gauss-Legendre rules*, which optimize the sample locations as well as the weights to get better convergence.

3.5 Multi-Dimensional Integration:

Multi-dimensional integration is more complex. A common way to extend a one-dimensional quadrature rule is to use a *tensor product rule.* These have the form

$$\hat{I} = \sum_{i_1=1}^{n} \sum_{i_2=1}^{n} \cdots \sum_{i_s=1}^{n} w_{i_1} w_{i_2} \cdots w_{i_s} f(x_{i_1}, x_{i_2}, \ldots, x_{i_s}) ,$$

where s is the dimension, and the w_i and x_i are the weights and sample locations for a given one-dimensional quadrature rule. For example, if we needed to evaluate the 5-D integral

$$I = \int_0^1 \int_0^1 \int_0^1 \int_0^1 \int_0^1 f(x_1, x_2, x_3, x_4, x_5)\, dx_1\, dx_2\, dx_3\, dx_4\, dx_5 ,$$

we would compute a sum of the form

$$I = \sum_{i_1=1}^{n} \sum_{i_2=1}^{n} \sum_{i_3=1}^{n} \sum_{i_4=1}^{n} \sum_{i_5=1}^{n} w_{i_1} w_{i_2} w_{i_3} w_{i_4} w_{i_5} f(x_{i_1}, x_{i_2}, x_{i_3}, x_{i_4}, x_{i_5}) \; .$$

The 2-D version of Simpson's rule looks like this:

$$h^2 \cdot \frac{1}{9} \cdot \begin{bmatrix} 1 & 4 & 2 & 4 & 2 & \cdots & 4 & 1 \\ 4 & 16 & 8 & 16 & 8 & \cdots & 16 & 4 \\ 2 & 8 & 4 & 8 & 4 & \cdots & 8 & 2 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 4 & 16 & 8 & 16 & 8 & \cdots & 16 & 4 \\ 1 & 4 & 2 & 4 & 2 & \cdots & 4 & 1 \end{bmatrix}$$

However, this does not work very well in high dimensions. If we start with an n-point quadrature rule in 1-D, we need $N = n^s$ sample points for an s-dimensional integral. Thus we can see that when the dimensionality is large (say $s = 20$), the number of required samples grows astronomically:

$$\begin{aligned} 1^{20} &= 1 \\ 2^{20} &= 1048576 \\ 3^{20} &= 3486784401 \\ 4^{20} &= 1099511627776 \\ \ldots & \end{aligned}$$

Furthermore, suppose that the 1-D rule has a convergence rate of $O(n^{-r})$. The s-dimensional rule doesn't work any better than the one-dimensional rule along each axis, so it converges at $O(n^{-r})$ as well. However, the total number of sample points used is much larger ($N = n^s$), so that in terms of the total number of samples the convergence is only $O(N^{-r/s})$. For example, the rate of convergence for Simpson's method is:

$$\begin{aligned} s = 1: &\quad O(n^4) \\ s = 2: &\quad O(n^2) \\ s = 4: &\quad O(n^1) \\ s = 8: &\quad O(n^{1/2}) \\ \ldots & \end{aligned}$$

So even with Simpson's assumption of 4 continuous derivatives, the convergence is already looking bad for high dimensions.

If we throw in a discontinuity in f, things get even worse. Consider the function

$$f(x) = \begin{cases} 1 & \text{if } x < X^* \\ 0 & \text{if } x > X^* \end{cases} \; .$$

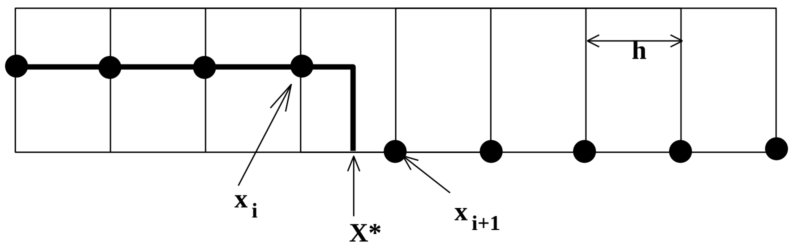


Figure 4: A discontinuous f.

and suppose that the quadrature points x_i are evenly spaced.

We see that the value computed the quadrature rule does not change when X^* is anywhere between x_i and x_{i+1}. Thus the error of *any* fixed quadrature rule is directly proportional to $h = 1/n$. So if f has a discontinuity, the convergence rate is $O(n^{-1})$ even in one dimension. Larger dimensions only compound this problem, leading to a convergence rate of $O(N^{-1/s})$.

3.6 Bakhvalov's Theorem:

There is an important result which limits the convergence rate for any deterministic quadrature rule, called *Bakhvalov's theorem.* Essentially, it says that given any s-dimensional quadrature rule, there is function f with r continuous, bounded derivatives, for which the convergence rate of the chosen quadrature rule is only $O(N^{-r/s})$.

Specifically, let C_M^r denote the set of function on $[0,1]^s$ with r continuous, bounded derivatives. That is, we require

$$\left| \frac{\partial^r}{\partial x_1^{a_1} \cdots \partial x_s^{a_s}} f \right| \le M$$

for all $a_1, \ldots, a_s$ such that $\sum a_i = r$.

Now consider any N-point quadrature rule, and let

$$\hat{I}(f) = \sum_{i=1}^{N} w_i f(x_i)$$

be the approximation to the true integral

$$I(f) = \int_{[0,1]^s} f(x_1, \ldots, x_s)\, dx_1 \cdots dx_s \,.$$

Then there exists a function $f \in C_M^r$ such that the error is

$$\left| \hat{I}(f) - I(f) \right| > k \cdot N^{-r/s} \,,$$

where $k > 0$ depends only on M and r.

4 Monte Carlo Integration

The basic Monte Carlo method (in 1-dimension, for simplicity) is:

$$\int_a^b f(x)\,dx \approx \frac{b-a}{N}\sum_{i=1}^{N} f(X_i) \tag{1}$$

where the points X_i are chosen independently and uniformly at random on the interval $[a, b]$. As we will see, this method has a convergence rate of $O(N^{-1/2})$ in any dimension, *regardless* of the smoothness of the function f. This is particularly useful in graphics, where we often need to calculate multi-dimensional integrals of discontinuous functions, where Simpson's rule and Gaussian quadrature don't do well.

4.1 A Bit of Probability Theory

First we review some terms from that probability course you took a long time ago.

A *random variable* X is simply a quantity that is chosen by some random process. (This is good enough for our purposes; to define it more precisely involves a lot of math.)

Given a random variable X, its **cumulative distribution function** $P(x)$ is defined by

$$P(x) = Pr\{X_i \le x\}$$

i.e. $P(x)$ is the probability that the random variable is not greater than a given value x.

So, for example, the cumulative distribution function $U[0, 1]$ is:

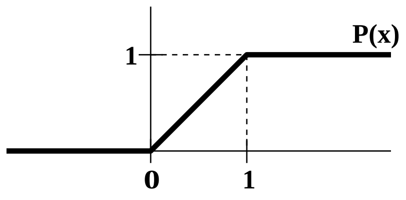


Figure 5: The cumulative distribution of $U[0, 1]$.

Next, we define the *probability density function* $p(x)$:

$$p(x) = \frac{dP(x)}{dx}\ ,$$

which is often just called a *density function* or *pdf.* From these definitions, we get the important relationship

$$Pr\{\alpha \le X \le \beta\} = \int_\alpha^\beta p(x)\,dx = P(\beta) - P(\alpha)\ .$$

Let $Y = f(X)$ (note that Y is also a random variable). The **expectation** of Y is defined as

$$E[Y] = \int f(x)p(x)\,dx\ ,$$

where $p(x)$ is the density function for X. The **variance** of Y is defined as

$$V[Y] = E[(Y - E[Y])^2] ,$$

which is to say, the variance of Y is the expected squared value of the difference between Y and its mean.

From these definitions, it is easy to see that

$$E[aY] = a\,E[Y] \qquad \text{and} \qquad V[aY] = a^2\,V[Y]$$

for any constant a. Another important rule that always holds is

$$E\left[\sum_i Y_i\right] = \sum_i E[Y_i] .$$

For example, we can use these rules to derive a simpler expression for the variance:

$$\begin{aligned} V[Y] &= E[(Y - E[Y])^2] \\ &= E[Y^2 - 2Y\,E[Y] + E[Y]^2] \\ &= E[Y^2] - E[Y]^2 \end{aligned}$$

noting that the inner $E[Y]$'s are just constants.

The following rule, however, holds only when the Y_i are independent:

$$V\left[\sum_i Y_i\right] = \sum_i V[Y_i]$$

Two variables Y_1 and Y_2 are defined to be **independent** if:

$$Pr\{Y_1 \le x_1 \text{ and } Y_2 \le x_2\} = Pr\{Y_1 \le x_1\} \cdot Pr\{Y_2 \le x_2\}$$

for all values of x_1 and x_2.

4.1.1 Basic Monte Carlo

Let's return now to the basic Monte Carlo estimate (1), namely

$$F_N = \frac{b-a}{N} \sum_{i=1}^{N} f(X_i) . \tag{2}$$

Our first task is to show that this gives the correct answer on average, i.e.

$$E[F_N] = \int_a^b f(x)\,dx = I .$$

We have used the notation F_N (rather than $\hat{I}$) to emphasize that the result is a random variable, and that its properties depend on how many sample points are chosen.

Recall that th X_i are independent samples from $U[a,b]$ (the uniform distribution on $[a,b]$). The density function for each X_i is

$$p(x) = \begin{cases} 1/(b-a) & \text{when } a \le x \le b \ , \\ 0 & \text{otherwise .} \end{cases}$$

We can now compute the expected value of F_N:

$$\begin{aligned} E[F_N] &= E\left[\frac{b-a}{N}\sum_{i=1}^{N} f(X_i)\right] \\ &= \frac{b-a}{N}\sum_{i=1}^{N} E[f(X_i)] \\ &= \frac{b-a}{N}\sum_{i=1}^{N}\int_{-\infty}^{\infty} f(x)p(x)\,dx \\ &= \frac{1}{N}\sum_{i=1}^{N}\int_{a}^{b} f(x)\,dx \\ &= I \ , \end{aligned}$$

which is what we wanted.

It is actually not necessary to choose the samples uniformly to make this work. Suppose that rather than choosing the X_i uniformly, we choose them according to some arbitrary density function $p(x)$ on the interval $[a,b]$. (We will discuss how to do this in the next lecture.) Now consider the estimate

$$F'_N = \frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p(X_i)} \ , \tag{3}$$

i.e. we compute f/p at each sample point and average all of these values together. The expected value of F'_N is

$$\begin{aligned} E[F'_N] &= \frac{1}{N}\sum_{i=1}^{N}\int_{-\infty}^{\infty}\frac{f(x)}{p(x)}\,p(x)\,dx \\ &= \frac{1}{N}\sum_{i=1}^{N}\int_{a}^{b} f(x)\,dx \\ &= I \ . \end{aligned}$$

This is an important generalization that is often used in practice. The only condition we require on the density $p(x)$ is that $p(x) > 0$ whenever $f(x) \neq 0$. The easiest way to achieve this is to have $p(x) > 0$ on the whole interval $[a,b]$. (If $p(x) = 0$ for some interval where $f(x) \neq 0$, then we will miss sampling some of the integral.)

4.2 Convergence of Monte Carlo Methods:

From the previous discussion, we know that F_N gives the right answer on average. However, we would also like to know how large an error we can expect, and what the rate of convergence is.

To simplify the notation, let $Y_i = f(X_i)/p(X_i)$, so that the estimate F_N becomes

$$F_N = \frac{1}{N}\sum_{i=1}^{N} Y_i \; .$$

We also let Y be a synonym for Y_1. The value that we are trying to approximate is $E[Y] = I$.

First, we have the **strong law of large numbers**, which says that given enough samples, the mean will converge to the expected value with probability 1:

$$Pr\left\{ \lim_{N\to\infty} \frac{1}{N}\sum_{i=1}^{N} Y_i = E[Y] \right\} = 1$$

assuming that the Y_i are i.i.d. (independent and identically distributed). This says that if we take enough samples, F_N is guaranteed to converge to the correct answer.

To determine the rate of convergence, we need **Chebychev's inequality**:

$$Pr\left\{ |F - E[F]| \geq \left(\frac{V[F]}{\delta}\right)^{1/2} \right\} \leq \delta \; ,$$

where F is any random variable such that $V[F] < \infty$. We will apply this inequality to F_N to get a bound on the error.

First, we compute the variance of F_N is terms of the variance of Y (a single sample):

$$\begin{aligned} V[F_N] &= V\left[\frac{1}{N}\sum_{i=1}^{N} Y_i\right] \\ &= \frac{1}{N^2} V\left[\sum_{i=1}^{N} Y_i\right] \\ &= \frac{1}{N^2}\sum_{i=1}^{N} V[Y_i] \\ &= \frac{1}{N} V[Y] \end{aligned}$$

where we have used $V[aY] = a^2 V[Y]$ and the fact that the Y_i are independent samples. Thus, the variance decreases linearly with N.

Plugging this into Chebychev's inequality, we get

$$Pr\left\{ |F_N - I| \geq N^{-1/2} \left(\frac{V[Y]}{\delta}\right)^{1/2} \right\} \leq \delta \; .$$

Thus for any fixed threshold δ, we see that the error decreases at the rate $N^{-1/2}$.

It is possible to get much tighter bounds on the error using the **Central Limit Theorem**, which states that F_N converges to a normal distribution as $N \to \infty$. It is most conveniently stated in terms of the *standard deviation* σ_{F_N} of F_N, which is simply the square root of the variance:

$$\sigma_{F_N} = (V[F_N])^{1/2} = N^{-1/2}\sigma_Y \ .$$

The standard deviation itself is a common measurement of error (it is sometimes called RMS error), and notice that it also converges at the $O(N^{-1/2})$ rate.

The central limit theorem then states that

$$\lim_{N\to\infty} Pr\left\{ \frac{1}{N}\sum_{i=1}^{N} Y_i - E[Y] \le t\,\frac{\sigma_Y}{\sqrt{N}} \right\} = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{t} e^{-x^2/2}\,dx \ ,$$

where the expression on the right is the *normal distribution* (the integral of the familiar "bell curve" or Gaussian).

This equation can be rearranged to give

$$Pr\left\{ |F_N - I| \ge t\sigma_{F_N} \right\} = \sqrt{\frac{2}{\pi}} \int_{t}^{\infty} e^{-x^2/2}\,dx \ .$$

The integral on the right decreases very quickly with t; for example when $t = 3$ the right-hand side is approximately 0.003. Thus, there is only about a 0.3% chance that F_N will differ from its mean by more than three standard deviations, provided that N is large enough for the central limit theorem to hold. Recall that the standard deviation is $\sigma_{F_N} = O(N^{-1/2})$, so this also verifies the $O(N^{-1/2})$ convergence.

4.3 Properties of Estimators:

The purpose of a Monte Carlo estimator is to approximate the value of some *quantity of interest* Q. Normally Q will be the value of an integral, although more general situations are possible (e.g. Q could be the ratio of two integrals).

An *estimator* is now a function of the form

$$F_N \ = \ F_N(X_1, \ldots, X_N) \ , \tag{4}$$

where the X_i are random variables. A particular numerical value of F_N is called an *estimate*. So far, we have only considered estimators where the X_i are independent and identically distributed (i.i.d.). However, in general the X_i can depend on each other, and they can have different distributions.

An estimator F_N is called *unbiased* if

$$E[F_N] = Q \qquad \text{for all values of } N \ .$$

The estimators (2) and (3) we have already discussed are unbiased.

Otherwise, the *bias* of the estimator is defined as

$$\beta[F_N] = E[F_N] - Q \ .$$

If the bias goes to zero as N increases, then the estimator is called *consistent*:

$$\lim_{N\to\infty} \beta[F_N] = 0 \ ,$$

or equivalently

$$\lim_{N\to\infty} E[F_N] = Q \ .$$

Such an estimator will ultimately converge to the right answer, but it may take many samples.

4.3.1 Example: a biased, consistent estimator

Suppose we are attempting to do antialiased sampling on a (1-D) pixel. To determine the pixel value, we need to evaluate an integral

$$I = \int_0^1 w(x) f(x)\, dx \tag{5}$$

where f is the image function, and w is the filter function, which is assumed to integrate to one:

$$\int_0^1 w(x)\, dx = 1 \ .$$

For example, w could be a tent filter:

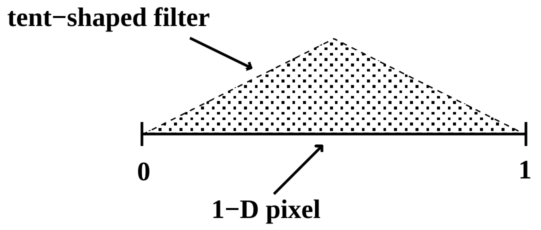


Figure 6: Antialiasing samples in a 1-D pixel with a tent filter.

In graphics, a common way to evaluate this integral is

$$F_N = \frac{\sum_{i=1}^N w(X_i) f(X_i)}{\sum_{i=1}^N w(X_i)}$$

where the X_i are uniformly distributed on $[0, 1]$. This is an example of a biased estimator, since when $N = 1$ we get

$$E[F_1] = E\left[\frac{w(X_1) f(X_1)}{w(X_1)}\right] = E[f(X_1)] = \int_0^1 f(x)\, dx \ .$$

This is not the same as the desired answer I given by (5), so the estimator is biased. Similarly, when $N = 2$ we get

$$E[F_2] = \int_0^1 \int_0^1 \frac{w(x_1)f(x_1) + w(x_2)f(x_2)}{w(x_1) + w(x_2)} \, dx_1 \, dx_2 \; ,$$

which is difficult to evaluate but is certainly not equal to I. In general, this estimator is biased for every value of N.

However, as $N \to \infty$ the bias goes to zero. To see this, we rewrite F_N as

$$F_N = \frac{(1/N) \sum_{i=1}^N w(X_i) f(X_i)}{(1/N) \sum_{i=1}^N w(X_i)}$$

and then evaluate

$$\begin{aligned}
\lim_{N\to\infty} E[F_N] &= \frac{\lim_{N\to\infty} (1/N) \sum_{i=1}^N w(X_i) f(X_i)}{\lim_{N\to\infty} (1/N) \sum_{i=1}^N w(X_i)} \\
&= \frac{\int_0^1 w(x) f(x) \, dx}{\int_0^1 w(x) \, dx} \\
&= \int_0^1 w(x) f(x) \, dx \\
&= I \; .
\end{aligned}$$

Next time, we will look at some other important properties of estimators, including the *mean squared error* and *efficiency* of an estimator.

Sampling Random Variables

Lecture #7: Tuesday, 21 October 1997
Lecturer: Eric Veach
Scribe: Menelaos Karavelas
Reviewer: Lucas Pereira

1 Properties of Monte Carlo estimators

Recall from the previous lecture that the purpose of a Monte Carlo estimator is to approximate the value of some quantity of interest Q. Most commonly Q will be an integral of the form

$$Q = \int_\Omega f(x)\,dx \tag{1}$$

where Ω is some arbitrary domain. An *estimator* is then defined to be a function of the form

$$F_N = F_N(X_1, \ldots, X_N)$$

(where the X_i are random variables), such that the mean of F_N is a usable approximation of the quantity we want to estimate.

The estimator F_N is said to be *unbiased* if

$$E[F_N] = Q \quad \text{for all } N.$$

For example, suppose that Q is given by (1), and let X be a random variable on Ω with density $p(x)$. Furthermore, suppose that $p(x) > 0$ whenever $f(x) \neq 0$. Then

$$F = \frac{f(X)}{p(X)}$$

is an unbiased estimator of Q, since

$$E[F] = \int_\Omega \frac{f(x)}{p(x)}\,p(x)\,dx = I.$$

Otherwise, the quantity

$$\beta[F_N] = E[F_N] - Q$$

is called the *bias*. If the bias goes to zero as the number of samples N is increased, then the estimator is called *consistent*:

$$Pr\left\{\lim_{N\to\infty} F_N = Q\right\} = 1,$$

i.e., F_N converges in probability to Q.

1.1 Example: a biased, consistent estimator

Recall the example of antialiased sampling of a 1D pixel from the previous lecture. The quantity to be estimated is the pixel value I, which is defined by an integral

$$I = \int_0^1 w(x) f(x)\, dx. \tag{2}$$

Here f is the image function on the domain $[0,1]$, and w is the filter function, which satisfies

$$\int_0^1 w(x)\, dx = 1.$$

A common way to evaluate the integral (2) is to use the estimator:

$$F_N = \frac{\sum_{i=1}^N w(X_i) f(X_i)}{\sum_{i=1}^N w(X_i)}, \tag{3}$$

where the X_i are independent uniformly distributed samples on $[0,1]$.

We first show that this estimator is in general biased. Suppose that just one sample is taken ($N = 1$). Then

$$F_1 = \frac{w(X_1) f(X_1)}{w(X_1)} = f(X_1),$$

and

$$E[F_1] = \int_0^1 f(x) p(x)\, dx = \int_0^1 f(x)\, dx,$$

noting that $p(x) = 1$ is the density function for X_1 on the domain $[0,1]$. Since this expected value is in general different from I (see (2)), the estimator F_1 is biased.

However, the estimator F_N is consistent since by the law of large numbers

$$\begin{aligned}
\lim_{N\to\infty} \frac{1}{N} \sum_i w(X_i) f(X_i) &= \int_0^1 w(x) f(x)\, dx, \\
\lim_{N\to\infty} \frac{1}{N} \sum_i w(X_i) &= \int_\Omega w(x)\, dx \;=\; 1.
\end{aligned}$$

Thus we have

$$\lim_{N\to\infty} E[F_N] = \int_0^1 w(x) f(x)\, dx,$$

i.e. we obtain the right answer as N becomes large.

A numerical example. To further understand how this estimator behaves, suppose that we have an image composed of identical pixels, where the left half of each pixel is white and the right half of is black. This corresponds to the image function

$$f(x) = \begin{cases} 1, & x \in [0, \frac{1}{2}) \\ 0, & x \in [\frac{1}{2}, 1] \end{cases} ,$$

where 1 represents white and 0 represents black. Suppose now that we use a filter $w(x)$ that puts a lot of weight on the right half of the each pixel, specifically:

$$w(x) = \begin{cases} 2 - 2\epsilon, & x \in [0, \frac{1}{2}) \\ 2\epsilon, & x \in [\frac{1}{2}, 1] \end{cases} , \tag{4}$$

where ϵ is a small number in the range $0 < \epsilon < 1$. Notice that this weighting function integrates to one, as required.

Then it is easy to see that the correct value for the filtered pixel is:

$$I = \int_0^1 f(x)w(x)\,dx = \frac{1}{2}(1)(2\epsilon) + \frac{1}{2}(0)(2 - 2\epsilon) = \epsilon,$$

and this is exactly the value that we want our Monte Carlo method to estimate (i.e. $Q = I$). However, if we use the biased estimator described above to evaluate the integral I at each pixel, then using just one sample X_1 per pixel leads to

$$F_1 = \begin{cases} 1, & X_1 \in [0, \frac{1}{2}) \\ 0, & X_1 \in [\frac{1}{2}, 1] \end{cases} .$$

This produces an image with a checkerboard appearance, where approximately half the pixels are black and half are white (see Figure 1). The resulting image has a mean value of 1/2, as opposed to the correct filtered image which has a mean value of ϵ (a uniform image of low gray-level).

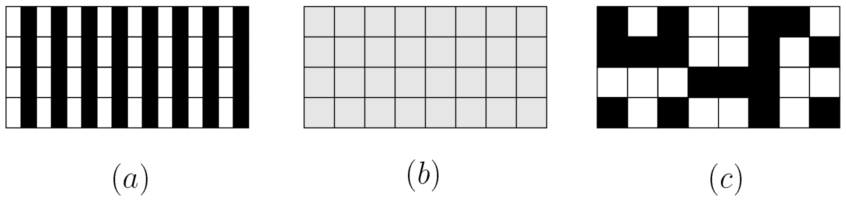


Figure 1: (a) A 4×8-pixel image, each pixel of which is half white and half black. (b) The filtered image using the filter w in (4), which has a mean value of ϵ. (c) The filtered image using the Monte Carlo estimator (3) with $N = 1$, which has a mean value of 1/2.

1.2 Mean squared error

The main reason for preferring unbiased estimators is that it is easier to estimate the error. Typically our goal is to minimize the *mean squared error* (MSE), defined by

$$MSE[F] \;=\; E[(F - Q)^2] \tag{5}$$

(where we have dropped the subscript N). We would like to find an estimator such that the mean squared error is small, relative to Q^2:

$$MSE[F_N] \ll Q^2.$$

In general, the mean squared error can be rewritten as

$$\begin{aligned} MSE[F] &= E[(F-Q)^2] \\ &= E[(F-E[F]+E[F]-Q)^2] \\ &= E[(F-E[F])^2] + 2E[F-E[F]](E[F]-Q) + (E[F]-Q)^2 \\ &= V[F] + \beta[F]^2 \end{aligned}$$

(since $E[F-E[F]] = 0$). Thus, to estimate the error we must have an upper bound on the possible bias. In general, this requires additional knowledge about the estimand Q, and it is often difficult to find a suitable bound.

On the other hand, for unbiased estimators we have $E[F] = Q$, so that the mean squared error is identical to the variance:

$$MSE[F] \;=\; V[F] \;=\; E[(F-E[F])^2] \;.$$

This makes it far easier to obtain error estimates, by simply taking several independent samples. Letting $Y_1, \ldots, Y_N$ be independent samples of an unbiased estimator Y, and letting

$$F_N \;=\; \frac{1}{N}\sum_{i=1}^{N} Y_i$$

as before (which is also an unbiased estimator), then the quantity

$$\hat{V}[F_N] \;=\; \frac{1}{N-1}\left\{\left(\frac{1}{N}\sum_{i=1}^{N} Y_i^2\right) - \left(\frac{1}{N}\sum_{i=1}^{N} Y_i\right)^2\right\}$$

is an unbiased estimator of the variance $V[F_N]$ (see Kalos and Whitlock). Thus, error estimates are easy to obtain for unbiased estimators.

2 Sampling random variables

There are a variety of techniques for sampling random variables, including:

- the method of *transformation of variables* (also known as the *inversion method*),
- the *rejection method*, and
- the *Metropolis method* (which will be discussed in another lecture).

There are also a variety of tricks that are useful for sampling particular distributions, such as the normal distribution. We give some simple examples of such tricks below.

With all of these methods, it is assumed that a supply of random variables $U_1, U_2, \ldots$ is available, where U_i is independent and uniformly distributed on $[0,1]$ ($U_i \sim U[0,1]$).

2.1 Transformation of variables

Our goal is to generate a random variable Y that is distributed according to a given density function $p(y)$. Let $P(y)$ be the cumulative distribution of Y, that is,

$$P(y) = Pr\{Y \leq y\}.$$

The *inversion method* consists of first choosing a random variable U that is uniformly distributed on $[0, 1]$, and then finding a value of Y such that $P(Y) = U$. Equivalently, we can write

$$Y = P^{-1}(U),$$

where P^{-1} denotes the inverse function of P. This procedure is shown graphically in Figure 2.

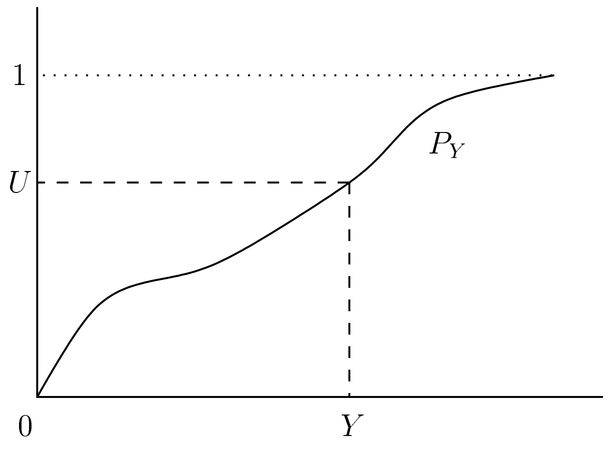


Figure 2: The inversion method.

It is easy to check that Y has the desired distribution, since

$$\begin{aligned} Pr\{Y \leq y\} &= Pr\{P(Y) \leq P(y)\} \\ &= Pr\{U \leq P(y)\} \\ &= P(y), \end{aligned}$$

where the first equality holds because P is a non-decreasing function, and the last equality holds because U is uniform on $[0, 1]$.

An example. To see how the method works, suppose we want to evaluate the integral

$$I = \int_0^\infty e^{-cx} g(x)\, dx.$$

Integrals of this type arise in volume rendering: x represents the distance along a ray, $g(x)$ represents the amount of light sent toward the viewer at the distance x, and e^{-cx} represents the attentuation of this light as it travels through a medium of constant opacity.

To evaluate this integral, we would like to sample a random distance X according to the density

$$p(x) = ce^{-cx}$$

(the factor of c is required so that p integrates to one). The cumulative distribution of X is

$$P(x_0) = \int_0^{x_0} p(x)\,dx = \int_0^{x_0} ce^{-cx}\,dx = -\frac{c}{c}\,e^{-cx}\Big|_0^{x_0} = 1 - e^{-cx_0}.$$

Thus using the inversion method, we have

$$\begin{aligned} 1 - e^{-cX} = U \quad &\Rightarrow \quad -cX = \ln(1-U) \\ &\Rightarrow \quad X = -(1/c)\,\ln(1-U). \end{aligned}$$

Equivalently, we could have started by setting $P(X) = 1 - U$, since if U is uniformly distributed on $[0,1]$ then so is $1-U$. This yields the slightly simpler result

$$X = -(1/c)\,\ln U.$$

Note that using this strategy, there is a high probability of sampling small values of X (due to the logarithm calculation). This corresponds to the fact that attenuation factor e^{-cx} goes to zero rapidly with increasing values of x.

If we sample N different distances X_i in this way, i.e.

$$X_i = -(1/c)\,\ln U_i$$

for independent values of $U_i \sim U[0,1]$, then corresponding Monte Carlo estimator for the integral I is

$$\begin{aligned} F_N &= \frac{1}{N}\sum_{i=1}^{N} \frac{e^{-cX_i} g(X_i)}{ce^{-cX_i}} \\ &= \frac{1}{cN}\sum_{i=1}^{N} g(X_i). \end{aligned}$$

So, the whole technique is very easy to apply.

Note that the inversion technique can easily be extended to several dimensions, by computing marginal and conditional distributions and inverting each dimension separately. (Examples will be given in the next lecture.)

Transformation of variables. There is a generalization of the inversion method called *transformation of variables*. Suppose that we start with a random variable X, with a known density function $p_X(x)$. (The inversion method was a special case of this with $X = U$.) Now let $Y = y(X)$, where y is some known function. Then it is natural to ask, what is the density function of Y, in terms of the density function of X? (Note that Y is also a random variable.)

Let P_Y be the cumulative distribution of Y, that is,

$$P_Y(y) = Pr\{Y \leq y\}.$$

We will assume that $y(x)$ is a continuous, non-decreasing function of x (which implies that $y'(x) \geq 0$). Then clearly we have

$$Pr\{Y \leq y(x)\} = Pr\{X \leq x\},$$

which implies that

$$P_Y(y) = P_X(x)$$

where y = y(x). Differentiating with respect to x, we obtain the following expression that relates the density function of X with that of Y:

$$p_Y(y)\,(dy/dx) = p_X(x)$$

which implies that

$$p_Y(y) = |dy/dx|^{-1}\ p_X(x), \tag{6}$$

where the absolute value signs allow for possibility that y is non-increasing rather than non-decreasing. This relationship is often written as

$$|p_Y(y)\,dy| = |p_X(x)\,dx|,$$

where as always, x and y are implicitly related by $y = y(x)$.

To illustrate how this identity is used, suppose that X is distributed according to the density $p_X(x) = 2x$ over the domain $[0, 1]$, and let $Y = \sin(X)$. What is the density function for Y? We have

$$\begin{aligned} p_Y(y) &= |d\sin(x)/dx|^{-1} p_X(x) \\ &= |\cos(x)|^{-1} 2x \\ &= 2\arcsin(y)\Big/\cos(\arcsin(y)) \\ &= 2\arcsin(y)\Big/(1-y^2)^{1/2}, \end{aligned}$$

where we have used the facts that (a) $\cos(x) > 0$ on $[0, 1]$, (b) $y = \sin(x)$ implies $x = \arcsin(y)$, and (c) $\cos(x) = (1 - \sin^2(x))^{1/2}$ on $[0, 1]$.

Deriving a transformation. Equation (6) suggests that given a random variable X, we can sample from an arbitrary density p_Y by finding an appropriate transformation $Y = y(X)$. In one dimension, there is a straightforward procedure for deriving such a transformation: we simply find a value of Y such that

$$P_Y(Y) = P_X(X),$$

or in other words we let y be the function

$$y(x) = P_Y^{-1}(P_X(x)).$$

(This assumes that we can find analytic expressions for the cumulative distributions P_X and P_Y.) The reason this works is that if X is any random variable, then the random variable $Z = P_X(X)$ has the uniform distribution:

$$Pr\{Z \le x\} = Pr\{P_X(X) \le x\} = Pr\{X \le P_X^{-1}(x)\} = P_X(P_X^{-1}(x)) = x,$$

for any $x \in [0,1]$. Thus, setting $P_Y(Y) = P_X(X)$ is equivalent to setting $P_Y(Y) = U$.

In one dimension, the main benefit of the transformation method over the simpler inversion method is that we can start with non-uniform samples. So for example, if we already know to generate a sample X with a normal distribution, then we can apply transformations to obtain various other distributions p_Y. In higher dimensions, the transformation method can be used to map random points from one domain to another (e.g. from a disc to a hemisphere). In this case, the factor $|dy/dx|$ in equation (6) becomes the *Jacobian determinant*, as we will see in the next lecture.

The main advantage of the inversion/transformation technique compared to the methods below is that it allows samples to be stratified easily (see the next lecture). This is done by stratifying the canonical parameter space $[0,1]^s$, and then mapping these samples into the given domain Ω with the desired density p_Y. Another advantage is that the technique has a fixed cost per sample, which can easily be estimated. The main disadvantage is that the density $p(x)$ must be integrated analytically, which is not always possible. It is also preferable for the cumulative distribution to have an analytic inverse, since numerical inversion is typically slow.

2.2 The rejection method

The idea behind the rejection method is to propose a trial value for the random variable that we want to sample. The value is then subjected to a test, and it may be either accepted or rejected. If rejection takes place, another trial value is chosen and tested, and this procedure goes on until an acceptable value is chosen.

Specifically, let $p(x)$ be the desired density function. Suppose that we can find another density $q(x)$ that we know how to sample, and such that

$$p(x) \le Mq(x)$$

for some constant M. Then the rejection method consists of the following algorithm:

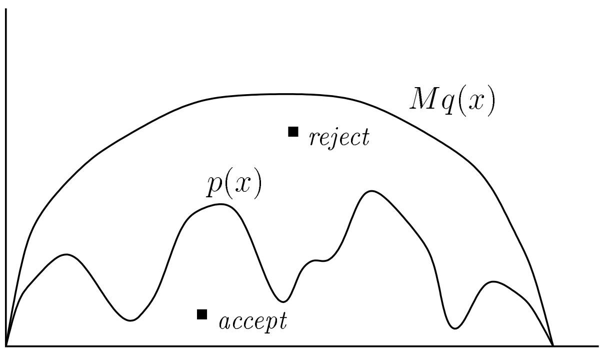


Figure 3: The rejection method.

```
Algorithm Reject;
    for $i = 1$ to $\infty$ do begin
        draw $X_i \sim q$;
        draw $U_i \sim U[0,1]$;
        if $U_i \leq \frac{p(X_i)}{M\,q(X_i)}$ then return $X_i$;
    end;
end.
```

This is illustrated graphically in Figure 3. A pair of random variables (X_i, U_i) is sampled, and if the point $(X_i, U_i \cdot Mq(X_i))$ lies under the curve $p(x)$, then it is accepted (by returning X_i). Otherwise, we choose a new pair (X_i, U_i) and try again.

Here is why the method works: on each iteration, the density of generating a sample at X_i is $q(X_i)$. This sample is returned if

$$U_i \leq \frac{p(X_i)}{Mq(X_i)},$$

which happens with probability $p(X_i)/(Mq(X_i))$. So, the probability density of returning a sample at x on a particular iteration is

$$q(x)\,\frac{p(x)}{Mq(x)} = \frac{p(x)}{M},$$

which integrates to $1/M$. Thus, when a sample is returned (with probability $1/M$), then X_i is distributed according to $p(x)$.

On the other hand, there is a $1 - 1/M$ probability that the sample will be rejected. Thus the average number of samples required until acceptance is the sum of a geometric series,

$$\sum_{i=0}^{\infty} \left(1 - \frac{1}{M}\right)^i = M,$$

where the term $(1-1/M)^i$ is the probability that at least $i+1$ samples will be required.

The main advantage of rejection sampling is that it can be used with any density function, even those that cannot be integrated analytically. However, if it is applied naively then it can be very inefficient, since we may iterate a lot of times until we find an acceptable value. Clearly, the efficiency depends on the size of the constant M, i.e. how tightly $Mq(x)$ bounds $p(x)$. The goal is to find a density $q(x)$ that we know how to sample from, and that is also a reasonably good match for $p(x)$. Another disadvantage of the rejection method is that it is difficult to apply with stratification.

An important special case. Let $p(x)$ be defined on the domain $[0,1]$, and suppose that $p(x) \le M$ for all x. In this case we can let $q(x) = 1$ (that is, X_i is drawn from the uniform distribution $U[0,1]$). The acceptance probability for X_i then becomes

$$\frac{p(X_i)}{Mq(X_i)} = \frac{p(X_i)}{M}.$$

Graphically, the acceptance probability is simply the ratio of the heights of the curves $p(x)$ and $Mq(x) = M$ at the given point X_i (see Figure 4). It is clear that the accepted samples have a density proportional to $p(x)$.

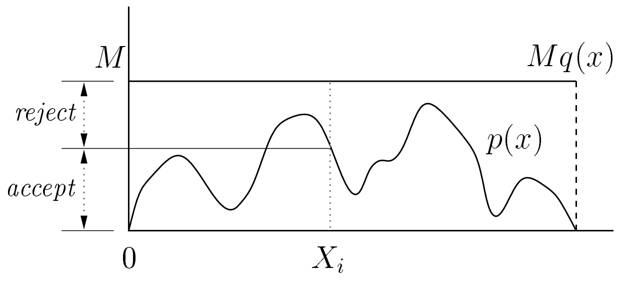


Figure 4: Each sample is accepted with probability $p(X_i)/M$, and rejected otherwise.

A geometric example. The rejection method can also be applied in geometric settings that do not correspond exactly to the general framework above. For example, suppose that we want to sample uniformly from a unit disc centered at the origin (see Figure 5).

Letting U_1 and U_2 be independent samples from $U[0,1]$, the rejection method consists of the following:

```
Algorithm Reject-Circle;
    do forever
        $X = 2U_1 - 1$;
        $Y = 2U_2 - 1$;
        if $X^2 + Y^2 \le 1$ then  return $(X, Y)$;
    end;
end.
```

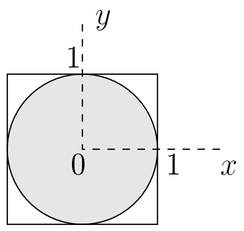


Figure 5: Sampling from a unit disc.

This can be considered a method for sampling the two-dimensional density function

$$p(x, y) = \begin{cases} 1/\pi & \text{if } x^2 + y^2 \leq 1, \\ 0 & \text{otherwise,} \end{cases}$$

by drawing samples from the uniform density $q(x)$ on $[-1, 1] \times [-1, 1]$. We do not need an extra random variable U_3 to decide whether to accept the samples, since the acceptance probability is always 0 or 1.

2.3 Tricks

There are a variety of tricks for sampling from specific distributions, usually by mapping several uniform random variables into a single output variable Y.

For example, let $U_1, \ldots, U_k$ be independent samples from $U[0, 1]$, and let

$$Y = \max\{U_1, U_2, \ldots, U_k\}.$$

Then the cumulative distribution of Y is:

$$P_Y(x) = Pr\{Y \leq x\} = \prod_{i=1}^{k} Pr\{U_i \leq x\} = x^k,$$

and thus the density function of Y is

$$p_Y(x) = P'_Y(x) = kx^{k-1}.$$

Therefore if we want to sample from a density proportional to x^k, we can do this by choosing $k+1$ random variables uniformly distributed on $[0, 1]$ and taking their maximum.

As another example, consider the random variable $Y = U_1 + U_2$. We can find the cumulative distribution of Y graphically, by thinking of U_1 and U_2 as the x and y coordinates of a unit square (see Figure 6). Specifically, we have

$$P_Y(z) = Pr\{Y \leq z\} = Pr\{U_1 + U_2 \leq z\},$$

which is simply the area below the line $U_1 + U_2 = z$. Using this fact we can easily derive the density function of Y, which is

$$p(x) = \begin{cases} x, & x \in [0, 1) \\ 2 - x, & x \in [1, 2] \end{cases} .$$

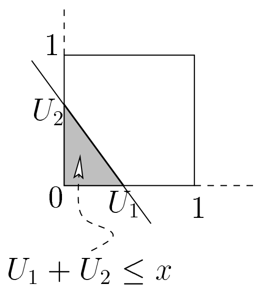


Figure 6: The density of a sum of two uniform random variables.

3 Sampling uniformly over a hemisphere

In graphics we often want to sample functions $f(\omega)$ which depend on a direction ω. We will represent ω as a unit vector (x, y, z), so that the set of all possible directions is the unit sphere

$$\mathcal{S}^2 = \{(x, y, z) \mid x^2 + y^2 + z^2 = 1\}.$$

Normally we are only interested in directions on one side of a given surface, which form a unit hemisphere centered around the surface normal $\mathbf{N}$. Such a hemisphere will be denoted Ω.

As a simple example, suppose that we want to sample a direction ω uniformly with respect to solid angle. Recall that the solid angle of a set of directions is simply the area of the corresponding set of points on the unit sphere. For example, the solid angle of a hemisphere is 2π.

For convenience, suppose that the hemisphere Ω is centered around the normal $\mathbf{N} = (0, 0, 1)$. A direction $\omega = (x, y, z)$ can then be parameterized by its spherical coordinates (θ, ϕ), where

$$\begin{aligned} x &= \sin\theta\cos\phi. \\ y &= \sin\theta\sin\phi, \\ z &= \cos\theta. \end{aligned}$$

With these coordinates, an element of area (solid angle) on the hemisphere is

$$d\omega = \sin\theta\, d\theta\, d\phi.$$

A *cap* of the hemisphere is defined as the set of directions where $0 \leq \theta \leq \theta_0$, for some given value of θ_0. The area $A(\theta_0)$ of this cap is:

$$\begin{aligned} A(\theta_0) &= \int_0^{2\pi} \int_0^{\theta_0} \sin(\theta)\, d\theta\, d\phi \\ &= \left(\int_0^{2\pi} d\phi\right) \left(\int_0^{\theta_0} \sin(\theta)\, d\theta\right) \\ &= 2\pi\,(-\cos\theta)\Big|_0^{\theta_0} = 2\pi(1 - \cos\theta_0). \end{aligned}$$

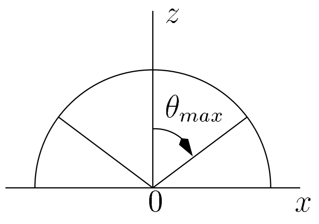


Figure 7: Sampling a cap of the hemisphere.

Recall that our goal is to sample a direction ω uniformly with respect to solid angle (i.e. area). Thus, we want the probability of sampling an angle θ to obey

$$Pr\{\theta \leq \theta_0\} \propto A(\theta_0).$$

Thus we can obtain the desired cumulative distribution for θ by normalizing $A(\theta)$:

$$P(\theta) = \frac{A(\theta)}{A(\pi/2)} = \frac{2\pi(1-\cos\theta)}{2\pi} = 1 - z,$$

recalling that $z = \cos(\theta)$. This result implies that area on the hemisphere is uniformly distributed with respect to z, i.e. if we look at any two horizontal slices with vertical thickness dz, then both slices have the same surface area.

Using the inversion method, we can now sample uniformly on the hemisphere by setting $P(\theta) = U_1$, or equivalently $z = U_1$ (since $1 - U_1$ is also uniform). If we then choose ϕ uniformly, we get

$$\begin{aligned} Z &= U_1 \\ X &= R\cos(2\pi U_2), \\ Y &= R\sin(2\pi U_2) \end{aligned}$$

where U_1 and U_2 are uniform, and $R = \sqrt{1 - U_1^2}$. This will generate a uniform distribution of points on the hemisphere with respect to solid angle.

We can also use this technique to sample uniformly on a cap of the hemisphere (i.e. the set of directions lying with some infinite cone). Let θ_{max} be the cap angle (see Figure 7), so that $\theta \in [0, \theta_{max}]$ and $\phi \in [0, 2\pi]$. To sample this cap uniformly, we only need to change the way that z is evaluated:

$$\begin{aligned} Z &= 1 - U_1(1 - \cos\theta_{max}) \\ X &= R\cos(2\pi U_2), \\ Y &= R\sin(2\pi U_2) \end{aligned}$$

where as before $R = \sqrt{1 - U_1^2}$. This technique can also be used to generate a random point on the entire sphere, by letting $\theta_{max} = \pi$.

Variance Reduction I

Lecture #8: Thursday, 23 October 1997
Lecturer: Eric Veach
Scribe: James Davis

1 Transformation of variables in several dimensions

Last lecture we talked about methods to sample random variables according to a given density function. One such method is transformation of variables. We discussed this method in a single dimension previously. Now let us generalize to the multidimensional case.

First, we examine how density functions are affected by transformations. Suppose we have a random variable $\mathbf{X} = (X_1, \ldots, X_n)$ on some domain $\Omega_{\mathbf{X}} \subset \mathbb{R}^n$, and let $p_{\mathbf{X}}(\mathbf{x})$ be the density function of $\mathbf{X}$ (where $\mathbf{x} = (x_1, \ldots, x_n)$ denotes an n-dimensional point). Now let $\mathbf{Y} = T(\mathbf{X})$ be another random variable, where T is a bijective, differentiable function from $\Omega_{\mathbf{X}}$ to some other domain $\Omega_{\mathbf{Y}}$. Then we can ask, what is the density function of $\mathbf{Y}$?

Similar to the single variable case, we know that

$$Pr\{\mathbf{Y} \in T(D)\} = Pr\{\mathbf{X} \in D\}$$

(since T is a bijection). We can write this in terms of the density function $p_{\mathbf{X}}$ and $p_{\mathbf{Y}}$ as

$$\int_{T(D)} p_{\mathbf{Y}}(\mathbf{y})\, d\mathbf{y} = \int_D p_{\mathbf{X}}(\mathbf{x})\, d\mathbf{x}\ . \tag{1}$$

Now we apply the change of variables $\mathbf{y} = T(\mathbf{x})$ to the left hand side, which yields

$$\int_D p_{\mathbf{Y}}(T(\mathbf{x}))\, |J_T(\mathbf{x})|\, d\mathbf{x} = \int_D p_{\mathbf{X}}(\mathbf{x})\, d\mathbf{x}\ , \tag{2}$$

where $J_T(\mathbf{x})$ is the Jacobian of the transformation $\mathbf{y} = T(\mathbf{x})$,

$$J_T = \begin{bmatrix} \partial y_1/\partial x_1 & \cdots & \partial y_1/\partial x_n \\ \vdots & \ddots & \vdots \\ \partial y_n/\partial x_1 & \cdots & \partial y_n/\partial x_n \end{bmatrix}$$

and $|\cdot|$ denotes the determinant of a matrix.

Finally, since equation (2) holds for any region D, the quantities being integrated must be the same:

$$p_{\mathbf{Y}}(T(\mathbf{x})) \cdot |J_T(\mathbf{x})| = p_{\mathbf{X}}(\mathbf{x}) \ .$$

This tells us how the density functions of $\mathbf{X}$ and $\mathbf{Y}$ are related. We can rewrite this as

$$p_{\mathbf{Y}}(\mathbf{y}) = \frac{1}{|J_T(\mathbf{x})|} \, p_{\mathbf{X}}(\mathbf{x}) \tag{3}$$

where $\mathbf{x} = T^{-1}(\mathbf{y})$. This identity is useful for figuring out what happens to densities when points are mapped from one coordinate representation to another.

1.1 Example: polar coordinates

Often it is convenient to choose points on the plane using polar coordinates (for example, when the density function we want to sample from is radially symmetric). Polar coordinates are defined by the mapping

$$\begin{aligned} (x, y) &= T(r, \theta) \\ \text{where} \qquad x &= r\cos\theta \\ y &= r\sin\theta \ . \end{aligned}$$

So, suppose that we choose points according to some density $p(r, \theta)$. What is the corresponding density $p(x, y)$?

The Jacobian of the transformation $(x, y) = (r\cos\theta, r\sin\theta)$ is

$$J_T(r, \theta) = \begin{bmatrix} \partial x/\partial r & \partial x/\partial \theta \\ \partial y/\partial r & \partial y/\partial \theta \end{bmatrix} = \begin{bmatrix} \cos\theta & -r\sin\theta \\ \sin\theta & r\cos\theta \end{bmatrix}$$

and the corresponding determinant is

$$|J_T(r, \theta)| = r\cos^2\theta + r\sin^2\theta = r \ .$$

So according to (3), the densities are related by

$$p(x, y) = \frac{1}{r}\, p(r, \theta) \ .$$

Most often we apply this the other way around: we are given a density function $p(x, y)$ that is measured with respect to surface area ($dA = dx\, dy$), and we want to convert it into polar coordinates to make the sampling easier. In this case we get

$$p(r, \theta) = r\, p(x, y) \ , \tag{4}$$

in other words we need to add a factor of r to the given density $p(x, y)$. A special case of this is sampling uniformly with respect to area: to do this in polar coordinates, we sample according to a density that is proportional to r.

An easy way to handle this kind of manipulation is with differentials. We start with the fundamental identity

$$p_{\mathbf{Y}}(\mathbf{y})\,d\mathbf{y} = p_{\mathbf{X}}(\mathbf{x})\,d\mathbf{x}$$

(which corresponds to equation (1)), and write the relationship between the differential volumes $d\mathbf{y}$ and $d\mathbf{x}$ as

$$d\mathbf{y} = |J_T(\mathbf{x})|\,d\mathbf{x}\ .$$

So for polar coordinates we have

$$p(x,y)\,dx\,dy = p(r,\theta)\,dr\,d\theta$$

and

$$dA = dx\,dy = r\,dr\,d\theta$$

(where dA is an element of area). These equations can be manipulated to give the same result (4) between the densities.

1.2 Example: spherical coordinates

Similarly, for sampling in three dimensions we often use spherical coordinates:

$$\begin{aligned} x &= r\sin\theta\cos\phi \\ y &= r\sin\theta\sin\phi \\ z &= r\cos\theta\ . \end{aligned}$$

The transformation $(x,y,z) = T(r,\theta,\phi)$ has a Jacobian determinant of

$$|J_T| = r^2\sin\theta\ ,$$

which is often written as the relationship

$$dV = dx\,dy\,dz = r^2\sin\theta\,dr\,d\theta\,d\phi$$

where dV is an element of volume.

We can also use spherical coordinates to represent points on the surface of a unit sphere, by setting $r = 1$:

$$\begin{aligned} x &= \sin\theta\cos\phi \\ y &= \sin\theta\sin\phi \\ z &= \cos\theta\ . \end{aligned}$$

The reason that this is important in graphics is that a direction ω can be represented as a unit vector (x,y,z). The solid angle of a set of directions is given by the area of the corresponding set of points on the unit sphere. In spherical coordinates, solid angle is measured by[1]

$$d\omega = \sin\theta \, d\theta \, d\phi \ .$$

So, if we are given a density function $p(\omega)$ that is measured with respect to solid angle, which simply means that

$$\int_D p(\omega) \, d\omega$$

is the probability of selecting a direction $\omega \in D$, then the corresponding density with respect to (θ, ϕ) satisfies

$$\begin{array}{rrcl} & p(\theta,\phi)\, d\theta\, d\phi & = & p(\omega)\, d\omega \\ \Rightarrow & p(\theta,\phi) & = & \sin(\theta)\, p(\omega) \ . \end{array}$$

In other words, we need to add an extra factor of $\sin\theta$ in order to sample a density $p(\omega)$ that is given with respect to solid angle.

2 Sampling by transformation in two dimensions

We now give several examples of sampling two-dimensional densities: uniform points on a hemisphere, uniform points in a disc, and cosine-weighted points on a hemisphere. The first two examples use the inversion method (recall that this is a special case of transformation, where we start with uniform random variables U_i), while the third example uses a more general transformation (it starts with uniformly distributed points in a unit disc).

2.1 Marginal and conditional densities

Suppose that we have some joint density function $p(x, y)$. To draw a sample (X, Y) from this density using the inversion method, we need to introduce the idea some new concepts. The *marginal density function* $p(x)$ is defined by

$$p(x) = \int p(x, y) \, dy \ ,$$

which is simply the density function for X alone, while the *conditional density function* $p(y \mid x)$ is defined by

$$p(y \mid x) = \frac{p(x, y)}{p(x)} \ ,$$

which gives the density function for Y once we have fixed a particular value of x.

[1]Note that this identity cannot be derived using a Jacobian determinant, since we have a mapping from $\mathbb{R}^2$ to a surface in $\mathbb{R}^3$. It can be derived from a similar formula for integration on a surface with respect to a particular parameterization.

These are the tools we need to sample from a joint probability distribution, since we can use the marginal density to isolate a single variable's distribution. Then we can determine the value of the other variable according to the conditional density. Examples will be given below.

Another useful concept which we will need later is *conditional expectation* of a random variable $F = f(X, Y)$, which is defined by

$$E_Y[F] = \int f(x, y)\, p(y \mid x)\, dy \ .$$

This is just the usual expression for the expected value, except that x is treated as a parameter (i.e. the result is a function of x).

It is easy to see that the expected value obeys

$$E[F] = E_X[E_Y[F]]$$

where $E[F]$ is the ordinary expected value

$$E[F] = \int f(x, y)\, p(x, y)\, dx\, dy \ ,$$

and E_X denotes the expected value with respect to X (using the marginal density $p(x)$).

2.2 Example: uniform sampling of a hemisphere

Recall the example from last time of sampling a hemisphere uniformly with respect to solid angle. We show how to derive this using marginal and conditional densities (which is really what we did last time, except that we didn't say so explicitly).

To sample uniformly with respect to solid angle, the density function is

$$p(\omega) = 1/(2\pi)$$

for ω in the unit hemisphere centered around around the positive z-axis. (Recall that this hemisphere has a solid angle (area) of 2π.) The corresponding joint probability density function with respect to (θ, ϕ) is

$$p(\theta, \phi) = \frac{1}{2\pi} \sin\theta \qquad 0 \le \theta \le \frac{\pi}{2}, \quad 0 \le \phi \le 2\pi \ .$$

To sample according to this density we first find the marginal density $p(\theta)$:

$$p(\theta) = \int_0^{2\pi} p(\theta, \phi)\, d\phi = \sin\theta$$

Supposing that we know how to sample a random variable θ from this density, we then need to sample ϕ from the conditional density

$$p(\phi \mid \theta) = \frac{p(\theta, \phi)}{p(\theta)} = \frac{1}{2\pi}$$

Since we now have density functions for each variable individually, we can use the single variable methods introduced last lecture to find samples. To use the *inversion method,* we first must find the cumulative distributions $P(\theta_0)$ and $P(\phi \mid \theta)$:

$$\begin{aligned} P(\theta_0) &= \int_0^{\theta_0} p(\theta)\, d\theta = \int_0^{\theta_0} \sin\theta\, d\theta = 1 - \cos\theta_0 \\ P(\phi_0 \mid \theta) &= \int_0^{\phi_0} \frac{1}{2\pi}\, d\phi = \frac{\phi_0}{2\pi} \end{aligned}$$

Now, applying inversion to θ, we set $P(\theta) = 1 - U_1$ (recalling that U_1 and $1 - U_1$ have the same distribution). This gives

$$\begin{aligned} 1 - \cos\theta &= 1 - U_1 \\ \cos\theta &= U_1 \\ \theta &= \cos^{-1} U_1 \end{aligned}$$

Now applying inversion to ϕ, we have

$$\begin{aligned} P(\phi \mid \theta) &= U_2 \\ \frac{\phi}{2\pi} &= U_2 \\ \phi &= 2\pi U_2 \end{aligned}$$

So we can find a point uniformly on a hemisphere by picking two uniform random variables $U_1$and U_2 and solving for (ϕ, θ).

2.3 Example: uniform samples in a disc

As a second example, suppose that we want to sample a point uniformly with respect to area in a unit disc. This corresponds to the joint density function

$$p(x, y) = \begin{cases} 1/\pi & \text{if } x^2 + y^2 \le 1, \\ 0 & \text{otherwise.} \end{cases}$$

which in terms of polar coordinates (r, θ) is

$$p(r, \theta) = \frac{1}{\pi} r$$

where $r \in [0, 1]$ and $\theta \in [0, 2\pi]$. In order to sample r and θ, we first find the marginal density for r:

$$p(r) = \int_0^{2\pi} \frac{1}{\pi} r\, d\theta = 2r$$

which is then integrated to find the cumulative distribution

$$P(r_0) = \int_0^{r_0} 2r\, dr = r_0^2\ .$$

Assuming now that r has already been chosen, the conditional density of θ is

$$p(\theta \mid r) = \frac{p(r,\theta)}{p(r)} = \frac{1}{2\pi}$$

and its cumulative distribution is

$$P(\theta_0 \mid r) = \int_0^{\theta_0} \frac{1}{2\pi}\, d\theta = \frac{\theta_0}{2\pi}$$

Applying the inversion method, we first find r_0 by setting $P(r_0) = U_1$:

$$\begin{aligned} r_0^2 &= U_1 \\ r_0 &= \sqrt{U_1} \end{aligned}$$

We then apply the inversion method again, this time to θ, by setting $P(\theta_0 \mid r_0) = U_2$:

$$\begin{aligned} \frac{\theta_0}{2\pi} &= U_2 \\ \theta_0 &= 2\pi U_2 \end{aligned}$$

A couple of comments are in order. Since both of these examples are trivial, we could easily have switched the order and found the marginal density of θ instead of r. In general, however, the calculations may be easier one way than the other (with respect to being able to do the integrations analytically). Also note that sometimes special tricks can be helpful for sampling from the density functions. For instance, the distribution $P(r_0) = r_0^2$ can be sampled by letting $R = max(U_3, U_4)$, as we showed in the last lecture.

2.4 Example: cosine-weighted hemisphere

We would often like to evaluate an integral such as the *reflectance equation*:

$$L_o(\omega_o) = \int f_r(\omega \to \omega_o) L_i(\omega) \cos\theta\, d\omega\ ,$$

where $L_o(\omega_o)$ is the radiance reflected toward the viewer, L_i is the incoming radiance, f_r is the bidirectional reflectance distribution function, and θ is the angle between the incident vector ω and the surface normal $\mathbf{N}$.

In the case of a diffuse surface $f_r = K_d$, so the integral reduces to

$$L_o(\omega_o) = \int K_d\, L_i(\omega)\, \cos\theta\, d\omega\ .$$

Since the function $L_i(\omega)$ is unknown, a reasonable importance sampling strategy is to choose ω according to a density proportional to $\cos\theta$:

$$p(\omega) \propto \cos\theta\ .$$

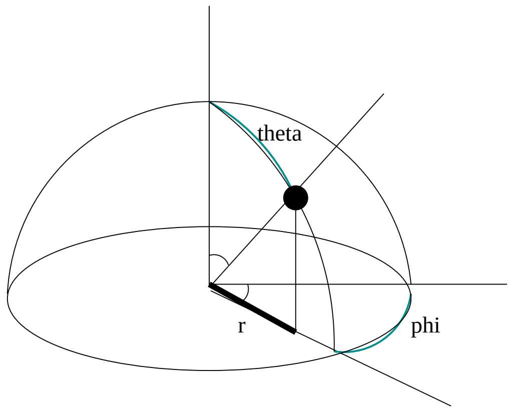
Converting this into a density with respect to (θ, ϕ) we get

$$p(\theta, \phi) \propto \cos\theta \, \sin\theta \; ,$$

where as usual $\theta \in [0, \pi/2]$ and $\phi \in [0, 2\pi]$. By integrating this density we find that the normalization constant is $1/\pi$, so that the desired joint density function is

$$p(\theta, \phi) = \frac{1}{\pi} \cos\theta \, \sin\theta \; .$$

It turns out that we can sample from this density by picking a point uniformly at random on the unit disc, and then projecting it vertically up onto the hemisphere.



Let (r, ϕ) be the polar coordinates of a point in the unit disc (note that we have switched from θ to ϕ). Recall that uniform sampling of the unit disc is represented by the joint density function

$$p(r, \phi) = \frac{1}{\pi} r$$

where $r \in [0, 1]$ and $\phi \in [0, 2\pi]$. We map this into a point ω on the hemisphere with spherical coordinates (θ, ϕ), where the vertical projection implies that r and θ are related by

$$\sin\theta = r \; .$$

(This can be verified by inspecting the diagram above, or by noting that $\sin\theta$ is the distance of ω from the z-axis.)

The Jacobian determinant of the transformation $(r, \phi) = (\sin\theta, \phi)$ is:

$$|J_T| = \begin{vmatrix} \partial r/\partial\theta & \partial r/\partial\phi \\ \partial\phi/\partial\theta & \partial\phi/\partial\phi \end{vmatrix} = \begin{vmatrix} \cos\theta & 0 \\ 0 & 1 \end{vmatrix} = \cos\theta$$

so the densities $p(r, \phi)$ and $p(\theta, \phi)$ are related by

$$p(r, \phi) = \frac{1}{|J_T|} p(\theta, \phi) \; .$$

This can be rearranged to give

$$p(\theta, \phi) \;=\; |J_T|\, p(r, \phi) \;=\; \cos\theta\, \frac{r}{\pi} \;=\; \frac{1}{\pi}\, \sin\theta\, \cos\theta\ .$$

So amazingly enough, this generates the desired distribution over the hemisphere. To summarize, we can generate cosine-weighted samples on a hemisphere by first choosing (X, Y) uniformly within the unit disc (such that $X^2 + Y^2 \le 1$), and then letting

$$Z = \sqrt{1 - (X^2 + Y^2)}\ .$$

Note that this works for any method of generating samples in the disc, e.g. rejection sampling, or Shirley's mapping from a square to the unit disc (as discussed below).

3 Variance Reduction

The goal of variance reduction is to improve the efficiency of our sampling. The *efficiency* of a Monte Carlo estimator F depends on both its variance $V[F]$, and the time $T[F]$ that is required to evaluate it:

$$\epsilon[F] = \frac{1}{V[F]\, T[F]}$$

According to this definition, F_1 is more efficient than F_2 if it yields a smaller variance in the same running time. (Similarly, an efficient estimator requires less time to achieve a given fixed variance.)

This definition is motivated by the fact that when we take N independent samples, variance is reduced by a factor of N, but computation time is increased by a factor of N. Letting

$$F_N = \frac{1}{N} \sum_{i=1}^{N} Y_i$$

where the Y_i are independent and identically distributed samples, we have

$$\begin{aligned} V[F_N] &= \frac{1}{N}\, V[F_1] \\ T[F_n] &= N\, T[F_1]\ . \end{aligned}$$

The definition of efficiency takes into account this tradeoff, by saying that F_N and F_1 are equally efficient:

$$\epsilon[F_N] = \epsilon[F_1]\ .$$

The main goal of unbiased Monte Carlo integration is to maximize efficiency: to find estimators whose variance is small and which are fast to evaluate. Methods for doing this are often simply called *variance reduction methods.*

Some of the most powerful variance reductions techniques are based on the idea of analytically integrating a function that is similar to the integrand. There are several ways to take advantage of this idea, of which the most important are *the use of expected values*, *importance sampling*, and *control variates.*

3.1 The use of expected values

Perhaps the most obvious way to reduce variance is to use analytic integration where possible. This idea is commonly referred to as *the use of expected values.* Specifically, it consists of replacing an estimator of the form

$$F \;=\; f(X,Y) \,/\, p(X,Y) \tag{5}$$

with one of the form

$$F' \;=\; f'(X) \,/\, p(X) \;\; , \tag{6}$$

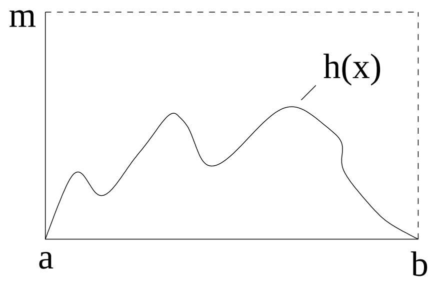
where $f'(x)$ and $p(x)$ are defined by

$$\begin{aligned} f'(x) &= \int f(x,y)\, dy \\ p(x) &= \int p(x,y)\, dy \;\; . \end{aligned}$$

Thus, to apply this technique we must be able to integrate both f and p with respect to y. We also must be able to generate samples according to the marginal density $p(x)$.

3.1.1 Example

Suppose that we want to integrate a function $h(x)$ on the interval $[a,b]$, such that $0 \le h(x) \le m$ for all $x \in [a,b]$.



Perhaps the simplest technique for doing this is the *hit-or-miss method.* The idea is to pick uniform random points inside the rectangle $[a,b] \times [0,m]$, and count how many fall under the curve $h(x)$. Formally, we can represent this strategy by the two-dimensional function

$$f(x,y) = \begin{cases} 1 & \text{if } 0 \le y \le h(x), \\ 0 & \text{otherwise,} \end{cases}$$

and letting $p(x,y)$ be the uniform density

$$p(x,y) = \frac{1}{m(b-a)}$$

on the rectangle $[a, b] \times [0, m]$. We can then estimate the integral

$$I = \int f(x, y)\, dx\, dy$$

by picking N sample points (X_i, Y_i) according to the density $p(x, y)$, and using the standard estimator

$$\begin{aligned} \hat{I} &= \frac{1}{N} \sum_{i=1}^{N} \frac{f(X_i, Y_i)}{p(X_i, Y_i)} \\ &= m(b-a) \cdot \frac{1}{N} \sum_{i=1}^{N} f(X_i, Y_i)\ , \end{aligned}$$

which is simply the area of the rectangle multiplied by the fraction of the sample points that lie under the curve $h(x)$.

Now, suppose that we apply the expected values technique by integrating analytically with respect to y. We have

$$f'(x) = \int f(x, y)\, dy = \int_0^{h(x)} 1\, dy = h(x)$$

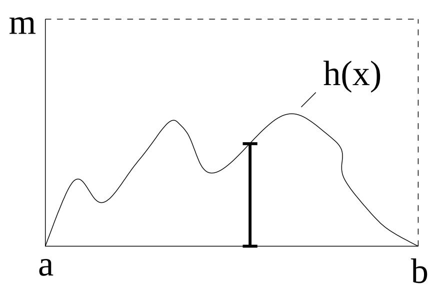
and the marginal density of X is

$$p(x) = \int_0^m p(x, y)\, dy = \frac{1}{b-a}\ .$$

This leads to the new estimator

$$\hat{I}' = \frac{1}{N} \sum_{i=1}^{N} \frac{f'(X_i)}{p(X_i)} = \frac{b-a}{N} \sum_{i=1}^{N} h(X_i)$$

which is simply the standard Monte Carlo estimator for one-dimensional integration that we have been using all along. (It is sometimes called the *sample-mean method.*)



Below, we will show that the expected values method always reduces variance. So in particular, this means that the sample-mean method always gives a lower variance than the hit-or-miss method.

3.1.2 Variance analysis

The name *expected values* comes from the fact that the estimator F' is simply the expected value of F with respect to Y:

$$\begin{aligned} E_Y[F] &= E_Y\left[\frac{f(X,Y)}{p(X,Y)}\right] \\ &= \int \frac{f(X,y)}{p(X,y)}\, p(y \mid X)\, dy \\ &= \int \frac{f(X,y)}{p(X,y)}\, \frac{p(X,y)}{p(X)}\, dy \\ &= \frac{\int f(X,y)\, dy}{p(X)} \\ &= f'(X) \,/\, p(X) \\ &= F' \,. \end{aligned}$$

To determine the variance of F' relative to F, we use the following identity (homework problem #1), where V_Y is the conditional variance with respect to Y, and E_Y is the conditional mean with respect to Y:

$$V[F] = E_X[V_Y[F]] + V_X[E_Y[F]] \,.$$

Using the fact that $F' = E_Y F$, we immediately obtain

$$\begin{aligned} V[F] - V_X[E_Y[F]] &= E_X V_Y[F] \\ V[F] - V[F'] &\geq 0 \\ V[F] &\geq V[F'] \end{aligned}$$

To get from the first to the second line, we used the fact that $E_X V_Y[F]$ is always non-negative (since the variance of a random variable is always non-negative). Furthermore $E_Y[F]$ is a function only of X, so $V[F'] = V_X[F']$.

So, we find that the variance of F' is always less than the variance of F. This suggests that we should always integrate those parts that we can, as we have done in the example above.

3.2 Importance Sampling

Suppose we want to evaluate $I = \int f(x)\, dx$. As we have seen, one method is to choose N random variables X_i uniformly over the domain, and estimate $\hat{I} = (1/N) \sum f(X_i)$. Suppose that instead, we choose the X_i according to some other density function $p(x)$, and define a new random variable

$$F = \frac{1}{N} \sum_{i=1}^{N} Y_i \qquad \text{where} \qquad Y_i = \frac{f(X_i)}{p(X_i)} \,.$$

This is unbiased, since

$$E[Y_i] = \int \frac{f(x)}{p(x)}\, p(x)\, dx = \int f(x)\, dx$$

provided that $p(x) > 0$ whenever $f(x) \neq 0$.

Now why would we want to do this? Suppose that $p(x)$ were exactly proportional to $f(x)$. Then we would have:

$$\begin{aligned} p(x) &= cf(x) \\ Y_i &= \frac{f(X_i)}{p(X_i)} = \frac{1}{c} \end{aligned}$$

so that Y_i always has the same value. This means that the variance of Y_i (and F) is zero, which is the best possible.

Unfortunately we can't do this exactly, since it turns out that in order for p to be normalized, we need

$$c = \frac{1}{\int f(x)\, dx}\ .$$

So if we could evaluate c, then we would already know the answer to our original problem, and there would be no need for sampling.

However, if we can find a density p that is approximately proportional to f (and that we can sample from), we can often reduce the variance of our estimator. This principle is called *importance sampling.* It should be noted that the argument above assumes that $f(x) \geq 0$, since $p(x)$ can't be negative. If $f(x)$ is sometimes negative, it turns out that the best thing is to make $p(x)$ proportional to $|f(x)|$, but this does not give zero variance even if we can sample this way exactly.

Importance sampling is a very useful technique in graphics. It is used to sample functions that are non-uniform, e.g. a peaked specular lobe of a BRDF, or a solid angle containing small light source. The cosine-weighted hemisphere was a simple example of this, where we sampled according to the $\cos\theta$ factor that appears in the integral for reflected radiance. Other examples are given in the homework problems.

3.3 Control Variates

Again suppose we want to evaluate $I = \int f(x)\, dx$. The idea of *control variates* is find some function $\tilde{f}$ similar to f that we can integrate analytically, and then subtract it. This is done by rewriting the integral as

$$I = \int \tilde{f}(x)\, dx + \int f(x) - \tilde{f}(x)\, dx$$

where the integral on the left can be evaluated exactly. This leads to a new estimator of the form

$$F' = \int \tilde{f}(x)\, dx + \frac{1}{N} \sum_{i=1}^{N} \frac{f(X_i) - \tilde{f}(X_i)}{p(X_i)}$$

where $p(x)$ is the density that the X_i are sampled from.

The new estimator F' will have a lower variance than the basic estimator F whenever

$$V\left[\frac{f(x_i) - \tilde{f}(X_i)}{p(X_i)}\right] \leq V\left[\frac{f(X_i)}{p(X_i)}\right] \tag{7}$$

Notice that given some analytic approximation $\tilde{f}$, we also have the option of using it for importance sampling. This is done by letting $p(x)$ be the density function

$$p(x) = \frac{\tilde{f}(x)}{\int \tilde{f}(x)\,dx}\ ,$$

assuming that we can find a way to generate samples with this density.

However, it doesn't make sense to use $\tilde{f}$ for control variates and importance sampling at the same time, since if $\tilde{f}$ is proportional to p, then the two estimators in equation (7) differ by a constant and their variance is the same. Thus if $\tilde{f}$ is already being used for importance sampling, it is not helpful to use it as a control variate as well.

So if we have found a function $\tilde{f}$ that is similar to f, which should we use: importance sampling, or control variates? The answer depends on the exact functions involved, but in general, if $f - \tilde{f}$ is a nearly constant function, then it is better to use control variates. If instead the ratio $f/\tilde{f}$ is nearly constant, then it is better to use importance sampling.

Another consideration is that with control variates, $\tilde{f}$ needs to be an approximation to the *entire integrand*. For example, if we wanted to apply control variates to the integral

$$\int_\Omega f_r(\omega \to \omega_o)\, L_i(\omega)\, \cos\theta\, d\omega\ ,$$

then we would need to approximate both the BRDF and incident light. It would not make sense to use the BRDF alone as a control variate, since it is scaled by an arbitrary factor L_i in the integral. (Subtracting the BRDF effectively replaces $L_i(\omega)$ by $L_i(\omega) - 1$, and with no knowledge of the magnitude of L_i there is no reason to think that this is an improvement.) With importance sampling, on the other hand, using the BRDF as a sampling density would be perfectly reasonable.

3.3.1 Example

In ray tracing, biased results can be avoided by choosing the recursion depth randomly. Typically this is done by defining a probability $1 - q$ that the recursion will stop at each depth, and increasing the sample weight by a factor of $1/q$ in the event that the recursion continues. (This technique is called *Russian roulette.*)

Typically, when the recursion stops we just return a value of 0. But suppose that we had some estimate L^* of the ambient light in the scene. In this case, we could apply the control variates technique, by returning the value

$$\int f_r(\omega \to \omega_o)\, L^*\, \cos\theta\, d\omega$$

instead. We compensate for this by rewriting the integral as

$$\begin{aligned} L_o(\omega_o) &= \int f_r(\omega \to \omega_o)\, L_i(\omega)\, \cos\theta\, d\omega \\ &= \int f_r(\omega \to \omega_o)\, L^*\, \cos\theta\, d\omega + \int f_r(\omega \to \omega_o)\, (L_i(\omega) - L^*)\, \cos\theta\, d\omega\ , \end{aligned}$$

i.e. if the recursion continues, we subtract L^* from the actual radiance samples $L_i(\omega)$.

One needs to be careful of course, since $L_i(\omega) - L^*$ could produce a negative value, and some systems may be unhappy with the concept of negative radiance. Another consequence is that control variates can lead to large relative errors when estimating values close to zero, for instance pixels in a dark region of the image.

4 Stratified Sampling

Stratified sampling is another way to reduce variance. We only introduce the basic idea here, and discuss its variance reduction properties in the next lecture.

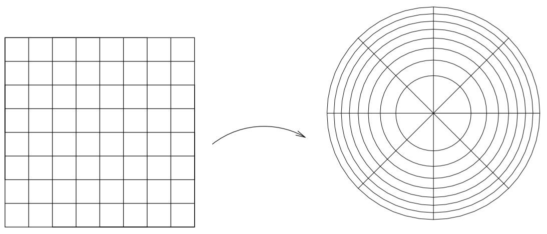
Given a domain Ω, the idea of stratified sampling is to split the domain into non-overlapping subregions $S_1, ..., S_k$ called *strata* such that

$$\bigcup_{i=1}^{n} S_i = \Omega\ .$$

We then take a fixed number of samples n_i to estimate the integral in each region, and add up the results. For example in graphics we often supersample a pixel, by breaking it into a $k \times k$ grid and choosing a random point in each of these strata. This is better than choosing k^2 points completely at random within the pixel, as we will see next time.

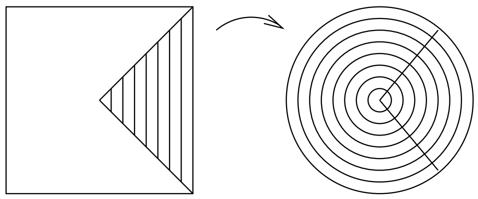
For now, let's consider how to pick a good set of strata. For instance, we found earlier that we could sample a disc uniformly with the mapping

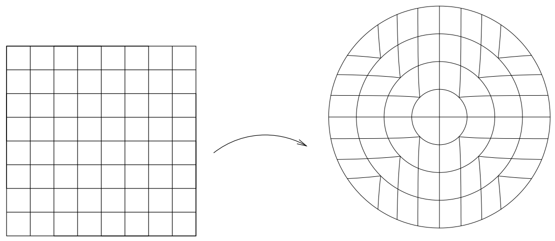
$$\begin{aligned} r &= \sqrt{U_i} \\ \theta &= 2\pi U_2 \end{aligned}$$



We can see that this results in some long thin regions. As we will show next time, it is better for the strata to have compact shapes, since this tends to result in the integrand having a more constant value over each stratum.

Peter Shirley proposed the following alternative mapping from a square to the unit disc, which results in more uniform strata. This mapping takes concentric squares into concentric circles, while preserving their relative area.





Mathematically, the mapping takes a point (u, v) in the square $[-1, 1] \times [-1, 1]$ to a point (r, θ) in the unit disc. It is defined as

$$\left.\begin{array}{rcl} r &=& u \\ \theta &=& \frac{\pi}{4}\frac{v}{u} \end{array}\right\} \quad \text{when } |u| \geq |v|,$$
$$\left.\begin{array}{rcl} r &=& v \\ \theta &=& \frac{\pi}{4}\left(2 - \frac{u}{v}\right) \end{array}\right\} \quad \text{otherwise.}$$

This certainly looks like a better stratification of a disc, but how do we know that these strata have uniform area? We start with a sample (U, V) whose density function is

$$p(u, v) = 1/4 \ ,$$

since (U, V) is chosen uniformly on $[-1, 1] \times [-1, 1]$. We then transform this to a sample (R, θ) in polar coordinates using the mapping defined above. If we consider just the case $|u| \geq |v|$, the Jacobian determinant for the transformation $(r, \theta) = T(u, v)$ is

$$|J_T| = \begin{vmatrix} \partial r/\partial u & \partial r/\partial v \\ \partial\theta/\partial u & \partial\theta/\partial v \end{vmatrix} = \begin{vmatrix} 1 & 0 \\ \text{irrelevant} & \pi/(4u) \end{vmatrix} = \frac{\pi}{4r} \ .$$

So, the density $p(r, \theta)$ in terms of $p(u, v)$ is

$$p(r, \theta) = \frac{1}{|J_T|}\, p(u, v) = \frac{4r}{\pi}\frac{1}{4} = \frac{1}{\pi}\, r \ ,$$

which is the density for sampling uniformly on a unit disc, as we saw in Section 2.3. If we take this one step further and map (r, θ) to a point (x, y), the corresponding density function is

$$p(x, y) = \begin{cases} 1/\pi & \text{if } x^2 + y^2 \leq 1, \\ 0 & \text{otherwise.} \end{cases}$$

So, Shirley's mapping sends strata of equal area in (u, v)-space into strata of equal area in (x, y)-space. This is called a *constant Jacobian mapping*, since the Jacobian determinant of the combined transformation $(x, y) = T(u, v)$ is $|J_T| = \pi/4$.

CS448: Topics in Computer Graphics Lecture #9
Mathematical Models for Computer Graphics
Stanford University Thursday, 6 November 1997

Variance Reduction II

Lecture #9:	Tuesday, 28 October 1997
Lecturer:	Eric Veach
Scribe:	David Hoffman
Reviewer:	James Davis

1 Transformation of the Unit Square into the Unit Disk

Often it is useful to be able to sample a circular domain with uniform density. For instance, a cosine-weighted sampling of the hemisphere can be obtained by first generating samples uniformly within a unit disc, and then projecting them vertically up onto the hemisphere (as we saw in the last lecture). This is useful for sampling reflections from diffuse surfaces.

To achieve uniform sampling on the unit disk, one might use a simple parameterization such as

$$r = u$$
$$\theta = 2\pi v$$

and sample (u, v) on a regular grid in the unit square. But the strata produced by this method do not have uniform (or even similar) areas, as figure 1 shows.

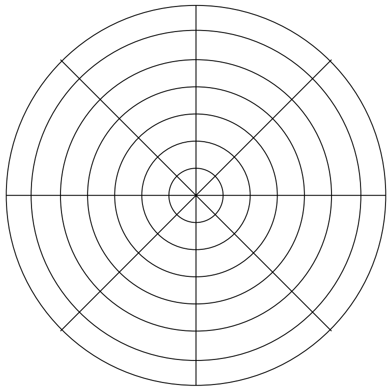


Figure 1: Image of the unit square under the mapping $(r, \theta) = (u, 2\pi v)$

One way to correct this problem is to re-parameterize as

$$r = \sqrt{u}$$
$$\theta = 2\pi v$$

which produces strata of equal area (figure 2). But these strata are less and less compact away from the center of the disk.

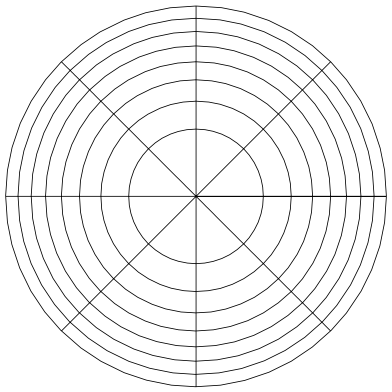


Figure 2: Image of the unit square under the mapping $(r, \theta) = (\sqrt{u}, 2\pi v)$

There is a better mapping that transforms compact areas of the square of side 2 ($[-1, 1] \times [-1, 1]$) into compact areas of the unit disk and also preserves their area (or rather, their fractional area, since the area of the unit disk is $\pi/4$ times that of the side-2 square). The triangular region below $v = u$ and above $v = 0$ maps to the region $\theta \in [0, \pi/4]$ using the map

$$r = u$$
$$\theta = \frac{\pi}{4}\frac{v}{u}$$

as shown in figure 3.

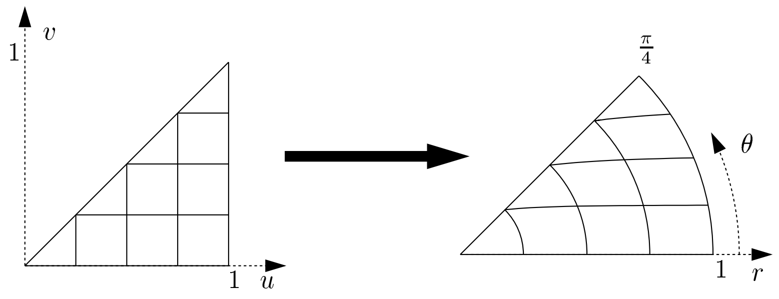


Figure 3: Transformation of part of the first quadrant by the mapping $(r, \theta) = (u, \frac{\pi}{4}\frac{v}{u})$

The mapping can then be reflected and rotated to cover the remaining 7 regions of the disk as in figure 4. The fact that this transformation preserves areas can be verified by computing its Jacobian determinant, as we did in the previous lecture.

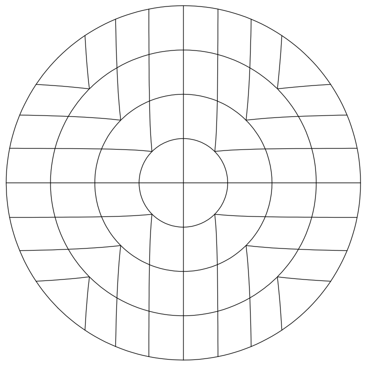


Figure 4: Completion of the $(r, \theta) = (u, \frac{\pi}{4}\frac{v}{u})$ mapping by rotation and reflection

2 Why (and when) Stratified Sampling reduces variance

Since mappings like the above are readily available for specific domains, most discussion of stratified sampling assumes that the domain is an s-dimensional unit cube. This domain is split into *strata* labeled $S_1 \ldots S_k$ which are disjoint and completely cover the domain:

$$\begin{aligned} S_i \bigcap S_j &= \emptyset \qquad \text{for all } i \neq j, \\ \bigcup_{i=1}^{k} S_i &= [0,1]^s \,. \end{aligned}$$

In general there are no other shape restrictions on the strata, but in practice they are most often rectangular regions. We denote the volume of a stratum as v_i, and clearly $\sum_i v_i = 1$. There will be a certain number of samples within each stratum, and we denote this number as n_i. Also, the samples are uniformly distributed, so

$$p_i(x) = \begin{cases} 1/v_i & \text{if } x \in S_i \\ 0 & \text{otherwise} \end{cases}$$

By taking n_i samples in stratum S_i, we get an estimated average value of

$$\hat{I}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} f(X_{i,j})$$

for the integrand in S_i, where $X_{i,j}$ is the jth sample in that stratum (which has the density function p_i defined above). The overall estimate will be the sum of the averages, each weighted by the volume of its stratum:

$$\hat{I} = \sum_{i=1}^{k} v_i \hat{I}_i$$

The actual average value of the integrand within each stratum is

$$\mu_i = E[f(X_{i,j})] = \frac{1}{v_i}\int_{S_i} f(x)dx$$

and this is the expected value of the estimator for the stratum, so $E[\hat{I}_i] = \mu_i$. The variance of the function within each stratum is

$$\sigma_i^2 = V[f(X_{i,j})] = \frac{1}{v_i}\int_{S_i} (f(x)-\mu_i)^2 dx \ ,$$

so the variance of the estimator for a stratum with n_i averaged samples is

$$V[\hat{I}_i] = \sigma_i^2/n_i \ ,$$

and the variance of the overall stratified estimator is

$$V[\hat{I}] = V\left[\sum_{i=1}^{k} v_i\hat{I}_i\right] = \sum_{i=1}^{k} V[v_i\hat{I}_i] = \sum_{i=1}^{k} v_i^2 V[\hat{I}_i] = \sum_{i=1}^{k} \frac{v_i^2\sigma_i^2}{n_i}$$

If we assume that the number of samples in each stratum is proportional to its volume, that is, $n_i = v_i N$ where N in the total number of samples, then this simplifies to

$$V[\hat{I}_s] = \frac{1}{N}\sum_{i=1}^{k} v_i\sigma_i^2 \tag{1}$$

(We have added the subscript s to indicate that this estimator is stratified.)

If the same number of samples is taken without stratification (i.e. where each sample is chosen independently and uniformly over $[0,1]^s$), what would the variance be? We have already defined the mean and variance within each stratum. The variance across the entire domain can be expressed as the mean of the individual variances plus the variance of the means, or[1]

$$V[f(X)] = \sum_{i=1}^{k} v_i\sigma_i^2 + \sum_{i=1}^{k} v_i(\mu_i - I)^2 \tag{2}$$

where X is uniformly distributed on $[0,1]^s$, and where I is the expected value of the means, which is the sought-after integral

$$I = \int_{[0,1]^s} f(x)\, dx \ .$$

[1]This formula can be proven by expanding the right-hand side directly, or observing that an unstratified sample X is equivalent to first choosing a random stratum I according to the discrete probabilities v_i, and then choosing choosing X uniformly within S_I. From this point of view, X is chosen conditionally on I, and we can obtain the given formula using the standard identity $V[F] = E_I V_{X_I} F + V_I E_{X_I} F$ where $F = f(X_I)$.

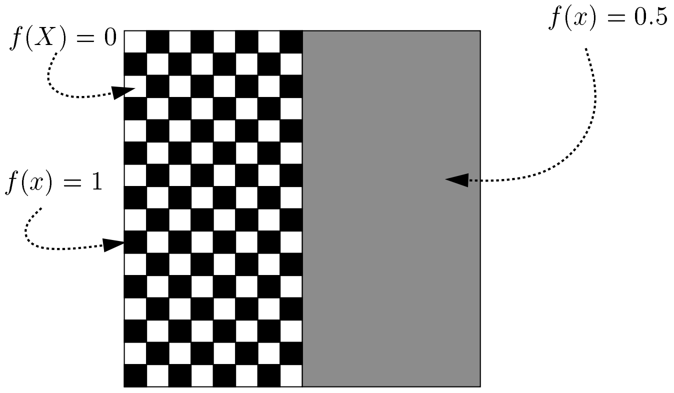


Figure 5: In both strata of this 2-dimensional function, the mean is 0.5

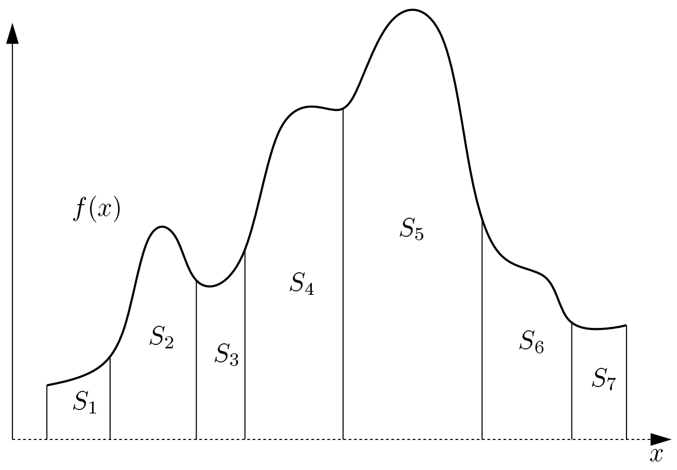


Figure 6: Well chosen strata can have widely varying means

Then the variance of the non-stratified estimator that averages N samples is

$$V[\hat{I}_{ns}] = \frac{1}{N}\left[\sum_{i=1}^{k} v_i\sigma_i^2 + \sum_{i=1}^{k} v_i(\mu_i - I)^2\right] \tag{3}$$

From equations 1 and 3 we can see that the non-stratified estimator's variance is larger than that of the stratified estimator by $\frac{1}{N}\sum_{i=1}^{k} v_i(\mu_i - I)^2$. Thus we only get a reduction in variance when the means μ_i of the strata differ from the overall mean I. For example, in figure 5, the mean in both strata is 0.5, so the variance of the mean is zero and there is no advantage in stratification (at least when the number of samples in each stratum is proportional to its volume). But in figure 6, the means vary widely between strata, so the variance reduction will be large if stratified sampling is used.

The need for widely varying means explains why compact strata are more desirable when little is known of the underlying function to be integrated. If the strata have

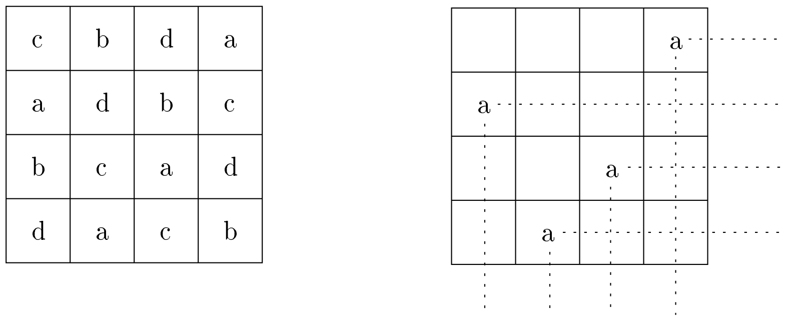


Figure 7: A single symbol from a 4×4 Latin square stratifies both the x and y coordinates

wide support in any direction, there is likely to be more variation within the strata and therefore less variation of the means between strata. By using compact strata, the variance of the means is likely to be increased, thereby increasing the variance reduction achieved by stratified sampling.

3 Latin Hypercube Sampling

In graphics, a typical application of stratified sampling is the selection of ray directions for ray tracing. For example, when there are multiple light sources or area light sources, the strata can be chosen to direct rays toward the light. Similarly, anti-aliasing can be performed by subdividing each pixel into several strata. Depth of field effects are achieved by sampling over the area of the aperture. Surface reflections are handled by sampling a hemisphere of directions according to a bidirectional reflectance distribution function, and motion blur by sampling over time. A ray tracer that includes all of these effects will be sampling in at least 9 dimensions (2 for pixels, 2 for the aperture, 2 for the reflected direction, 2 for the direction to the light source, and 1 for time). If each dimension is split into just 2 strata, there are 512 samples required per pixel! As the dimensionality of the space increases, the rate of convergence of the estimator decreases with respect to any particular dimension, just as in numerical integration. One technique to circumvent this problem of dimensionality is *Latin Hypercube Sampling.*

A "Latin square" is an $n \times n$ array of symbols – drawn from an alphabet of size n – in which each symbol appears exactly once in each row and once in each column. Figure 7 shows a 4×4 example. If we consider this a square 2-dimensional domain divided into 16 square strata, we could take a sample in each square. But we could also take samples just in those squares labeled "**a**". With only 4 samples, we would still have a sample in each row and a sample in each column.

This sampling strategy is easy to extend to s dimensions without the need to construct a full Latin hypercube. In each dimension the range $[0, 1]$ is divided into N intervals, resulting in a domain with N^s subcubes. By choosing s permutations Π_j of the integer range $\{1, \ldots, N\}$ and using the ith number in Π_j as the jth coordinate of sample X_i (where $1 \le i \le N$), we guarantee that in each dimension we have sampled all N intervals of $[0, 1]$ – in other words, the samples are stratified on each of the dimensions individually. We write the jth coordinate of the ith sample as

$$X_i^j = \frac{\Pi_j(i) - U_{ij}}{N}$$

where U_{ij} is a uniform random number in $[0, 1]$, making the probability distribution uniform in each one-dimensional interval and therefore in each s-dimensional subcube. So in the right side of figure 7, we have $\Pi_1 = \{1, 2, 3, 4\}$ and $\Pi_2 = \{3, 1, 2, 4\}$.

To analyze the convergence of this method, let us first suppose that $f(x)$ is an *additive function*, i.e. a function of the form

$$f(x) = \sum_{j=1}^{s} f_j(x^j) \tag{4}$$

where x^j denotes the jth coordinate of x. In other words, an additive function is a sum of functions f_j that each depend on only one coordinate of the vector x. To get an idea of how Latin hypercube sample converges for this type of function, let

$$\hat{I} = \frac{1}{N} \sum_{i=1}^{N} f(X_i)$$

and suppose that we fix every sample at the center of its cell by setting $U_{ij} = 1/2$. This yields

$$\hat{I} = \sum_{j=1}^{s} \frac{1}{N} \sum_{i=1}^{N} f_j\left(\frac{i - \frac{1}{2}}{N}\right) , \tag{5}$$

noting that under summation over i we can replace $\Pi_j(i)$ with i. Equation (5) is simply the sum of s midpoint integration rules, one for each f_j, which has an error of $O(N^{-2})$ provided that the f_j are smooth. It is also possible to show that if U_{ij} is a uniform random number in $[0, 1]$ (rather than being fixed at $1/2$), the error is $O(N^{-3/2})$ with high probability (for smooth functions). In either case, the convergence is better than that of standard Monte Carlo,which is $O(N^{-1/2})$.

To generalize this result, suppose that f is decomposed into a sum

$$f(x) = f_{add}(x) + f_{res}(x)$$

where $f_{add}(x)$ is an additive function, and $f_{res}(x)$ is the "residual" which is orthogonal to all additive functions, i.e.

$$\int_{[0,1]^s} f_{res}(x)\, g_{add}(x)\, dx = 0$$

for any additive function $g_{add}(x)$. This decomposition is unique, and we will give an explicit formula for $f_{add}(x)$ in terms of $f(x)$ below.

Given this decomposition, the variance of Latin hypercube sampling is

$$V[\hat{I}] = \frac{1}{N} \int f_{res}^2(x) dx + o\left(\frac{1}{N}\right)$$

for any square-integrable function $f(x)$. The second term is a function that decreases more rapidly that c/N for any constant c. Thus, Latin hypercube sampling increases the rate of convergence on the components of the function that depend on just one coordinate at a time. The closer that f is to being a purely additive function (i.e. the smaller the residual part $f_{res}(x)$ is), the better Latin hypercube sampling will do.

3.1 Analysis of variance decompositions

We can determine an explicit formula for decomposing f into its components f_{add} and f_{res}, using an *analysis of variance decomposition* (usually abbreviated *anova*). Let S be the set $\{1, \ldots, s\}$ of all coordinate indices, and consider all the possible subsets $U \subseteq S$ of these coordinates (there are 2^s possible subsets in all). In the anova framework, an arbitrary function f is expressed as a sum

$$f(x) = \sum_{U \subseteq S} f_U(x^U)$$

where each function f_U depends only on the coordinates x^U, and all the f_U are orthogonal. The function when $U = \emptyset$ does not depend on any variables, and is called the *grand mean*:

$$\mu = f_\emptyset = \int_{[0,1]^s} f(x) dx$$

The other $2^s - 1$ subsets of U are called *sources of variation*. For example, the components of f that depend on just one variable are called the *main effects* and are defined as

$$f_j(x^j) = \int (f(x) - \mu) \prod_{i \neq j} dx^i$$

Notice that all of these functions are orthogonal to the constant function $f_\emptyset = \mu$. The best additive approximation to f can then be written as

$$f_{add}(x) = \mu + \sum_{j=1}^{s} f_j(x^j) \ ,$$

where the residual $f_{res} = f - f_{add}$ is orthogonal to all additive functions.

Similarly, one can define the functions

$$f_{j,k}(x^{j,k}) = \int \left(f(x) - \mu - f_j(x^j) - f_k(x^k)\right) \prod_{i \neq j,k} dx^i$$

which represent the parts of f that depend on two particular variables together (the *two-factor interactions*). These functions are orthogonal to $f_\emptyset$ and to all the f_j.

In general, f_U is defined by

$$f_U(x^U) \;=\; \int \left(f(x) - \sum_{V \subset U} f_V(x^V) \right) dx^{S-U} \tag{6}$$

where the sum is over all proper subsets of U $(V \neq U)$. This yields a set of function f_U that are all orthogonal, i.e.

$$\int f_U(x^U)\, f_V(x^V)\, dx \;=\; 0$$

whenever $U \neq V$. This implies that

$$\int f^2(x)\, dx \;=\; \sum_{U \subset S} \int f_U^2(x^U)\, dx$$

so that the variance of f can be written as

$$\int (f(x) - \mu)^2\, dx \;=\; \sum_{|U|>0} \int f_U^2(x^U)\, dx \;\; .$$

With Latin hypercube sampling, the components of the variance where $|U| = 1$ converge at a rate faster than $1/N$, leaving only the residual component

$$\int f_{res}^2(x)\, dx \;=\; \sum_{|U|>1} \int f_U^2(x^U)\, dx \;\; .$$

4 Orthogonal Arrays

This above analysis suggests that one might try to select samples that are stratified in every *pair* of variables, in addition to every single variable. This is, in fact, possible using a version of Latin hypercube sampling based on *orthogonal arrays*.

An orthogonal array is an $N \times s$ matrix whose entries – from an alphabet of b symbols – are such that every $N \times t$ submatrix contains exactly the same number of copies of all b^t possible rows.[2] If we let λ be the number of times that each row appears in this submatrix, then clearly the total number of rows is $N = \lambda b^t$. The parameter t is called the *strength* of the orthogonal array. The following table shows an example of an orthogonal array A_{ij} whose parameters are $(N, s, b, t) = (9, 4, 3, 2)$.

[2]The submatrix is not necessarily contiguous; it can consist of any subset of the columns.

0	0	0	0
0	1	1	2
0	2	2	1
1	0	1	1
1	1	2	0
1	2	0	2
2	0	2	2
2	1	0	1
2	2	1	0

This array was constructed by the rule

$$\begin{aligned} A_{i1} &= x \\ A_{i2} &= y \\ A_{i(j+2)} &= x + jy \qquad \text{for } j = 1 \ldots b-1, \end{aligned}$$

where $i = bx+y+1$, and $0 \le x, y \le b-1$. This technique works in general for constructing an orthogonal array with $N = b^2$, $s = b+1$, $t = 2$, and b symbols, for any prime number b. There are a variety of other techniques known for constructing such arrays.

So, how do we use such an array for sampling? Suppose that we divide the domain into b^s cells, by splitting each axis into b intervals of equal size. Also, assume that the symbols in the orthogonal array are $\{0, 1, \ldots, b-1\}$. Then each $1 \times s$ row of the array can be considered an index that specifies one of the b^s possible cells. The overall strategy is to generate one sample in each of the N cells specified by the rows of the orthogonal array. Specifically, the j-th coordinate of sample X_i is given by

$$X_i^j = \frac{A_{ij} + U_{ij}}{b}$$

where U_{ij} is a uniform random number in $[0, 1]$.

Next, consider the projection of these samples onto any set of t coordinates. (There are $\binom{s}{t}$ possible projections in all.) Then from the orthogonal array property, these samples will be uniformly distributed among the b^t cells obtained by stratifying these axes into b equal-sized intervals. To see this, observe that the coordinates of the projected samples are specified by the rows of a particular $N \times t$ submatrix. That is, each $1 \times t$ row specifies one of the b^t cells in the t-dimensional projection of the domain. By the definition of orthogonal arrays, every possible $1 \times t$ row appears λ times in this submatrix, so each of the b^t cells occurs exactly λ times.

Therefore, orthogonal array sampling produces samples that are stratified with respect to *every* possible subset of t coordinates. We should note here that given an orthogonal array, one can create another with the same parameters simply by permuting the columns, or by permuting the alphabet within each column. This is significant because we don't want to always sample the same set of N cells out of the $\binom{b^s}{N}$ sets that are possible.

The variance of orthogonal array sampling can be written as

$$V[\hat{I}] = \frac{1}{N} \sum_{|U|>t} \int f_U^2(x^U)\,dx \;+\; o\left(\frac{1}{N}\right) .$$

Thus, it improves the rate of convergence with respect to all components of the integrand that depend on t coordinates or less. The case $t = 2$ is particularly interesting for graphics. For example, if we apply this technique to distribution ray tracing, it ensures that all the two dimensional projections are well stratified (e.g. over the pixel, lens aperture, light source, etc.) This also includes combinations of dimensions such as the x-coordinate of the current pixel and the x-coordinate of the lens aperture.

4.1 Orthogonal array-based Latin hypercube sampling

Notice that because we get such good stratification with respect to t-dimensional projections, we also get some stratification with respect to w-dimensional projections for $w < t$. For simplicity, assume $\lambda = 1$. Then in a w-dimensional projection, each cell receives b^{t-w} samples. For example, this says that in each one-dimensional interval of width $1/b$ along any axis j, there are exactly b^{t-1} samples. Notice that this is not as good as Latin hypercube sampling, which places one sample in each interval of width $1/b^t$.

There is a simple modification to orthogonal array sampling that yields the same one-dimensional stratification properties as Latin hypercube sampling. (The result, logically enough, is called *orthogonal array-based Latin hypercube sampling.*) Again assume that $\lambda = 1$, so that the array has b^t rows. Then observe that within each column, the values $\{0, \ldots, b-1\}$ each appear b^{t-1} times. The technique works by remapping these symbols into a single sequence $\{0, \ldots, b^t - 1\}$, by letting the b^{t-1} identical copies of each value m map to the sequence $(b^{t-1}m, \ldots, b^{t-1}(m+1)-1)$ according to some random permutation. This process is repeated for each column separately. Calling the modified array $\hat{A}$, we then change the coordinate assignment to

$$X_i^j = \frac{\hat{A}_{ij} + U_{ij}}{b^t} .$$

This will ensure that the samples are maximally stratified for each one-dimensional projection, as well as for each t-dimensional projection.

Quasi Monte Carlo

Lecture #10:	Thursday, 30 October 1997
Lecturer:	Eric Veach
Scribe:	Matt Pharr
Reviewer:	David Hoffman

1 Overview

For Monte Carlo integration to be effective, we want to use sample points that are uniformly distributed across the domain of integration. The last lecture introduced Latin hypercube sampling, which gives samples that are well-stratfied with respect to any one dimension, and orthogonal array sampling, which gives samples that are well-distributed with respect to any combination of two, three, or more dimensions.

Quasi Monte Carlo (QMC) is based on the approach of removing randomness from the generation of sampling sequences; instead, the idea is to look for fixed sequences of points that perform *better* than random sequences, at least for certain classes of functions. Techniques have been developed to generate both finite and infinite sequences of well distributed quasi-random points; in general, slightly better distributions are possible if the number of sample points needed is known in advance.

2 Discrepancy

The first question we have to answer is, given a sequence of samples, how well distributed are the samples (and thus, how effective will they be in estimating the values of the integrals we want to compute.) Discrepancy is a measure of irregularity of distribution which is used to evaluate the quality of QMC sampling sequences. The basic idea is to look at various regions of the domain and compare the volume of these regions to the number of sample points inside them. In general, one fourth of the volume should correspond to one fourth of the sample points, etc.

2.1 Definition

Assume we're evaluating a set of points defined over $[0,1]^s$. To compute the discrepancy of the points, we first pick a family of shapes $\mathcal{B}$ which are subsets of $[0,1]^s$. For example, boxes with one corner at the origin are often used. This corresponds to:

$$\mathcal{B} \;=\; \{B = [0,v_1] \times [0,v_2] \times \cdots \times [0,v_s] \mid 0 \le v_i \le 1 \text{ for all } i\}\;\;.$$

Given a sequence of sample points $P = x_1, \ldots, x_N$, the discrepancy of P with respect to $\mathcal{B}$ is

$$D_N(B, P) = \sup_{B \in \mathcal{B}} \left| \frac{\#\{x_i \in B\}}{N} - \lambda(B) \right|,$$

where $\lambda(B)$ is the volume of B. In other words, we're finding the maximum difference between the fraction of points inside one of the shapes and the volume of the shape. When the set of shapes $\mathcal{B}$ is the set of boxes with a corner at the origin (described above), this is called the *star discrepancy* $D_N^*(P)$.

Other popular sets of shapes to use to compute discrepancy include arbitrary axis aligned boxes, hyperplanes that cut the domain into two pieces, etc.

2.2 Some Examples

Consider the set of points in one dimension

$$x_i = i/N.$$

We can see that the star discrepancy of x_i is

$$D_N^*(x_1, \ldots, x_n) = \frac{1}{N}.$$

For example, take the interval $B = [0, \frac{1}{N})$. Then $\lambda(B) \approx \frac{1}{N}$, but $\#\{x_i \in B\} = 0$. This is the interval (along with the intervals $[0, \frac{2}{N})$, etc.) with the largest difference between volume and the fraction of points contained.

We can improve on the star discrepancy of this sequence by modifying it slightly:

$$x_i = \frac{i - \frac{1}{2}}{N}.$$

Then

$$D_N^*(x_i) = \frac{1}{2N}.$$

A theorem due to H. Niederreiter (one of the main pioneers of QMC) provides some bounds for the star discrepancy of a sequence of points in 1D:

$$D_N^*(x_i) = \frac{1}{2N} + \max_{1 \le i \le N} \left| x_i - \frac{2i-1}{2N} \right|.$$

And thus, the second example's sequence has the lowest possible discrepancy for a sequence in 1D. In general, it is much easier to analyze and compute bounds for the discrepancy of sequences in 1D than in higher dimensions.

Another often used type of discrepancy is the *extreme discrepancy*, which uses arbitrary axis-aligned boxes for the set of shapes. The extreme discrepancies of the 1D examples given above are both $\frac{1}{N}$.

3 Theoretical Performance of QMC

The Koksma-Hlawka theorem gives an upper bound for the error in QMC evaluation of integrals, as a product of two factors: one factor depends on the discrepancy of the point set used, while the other depends on the function being integrated. Given an integral

$$I = \int_{[0,1]^s} f(x_1, \ldots, x_s)\, dx_1 \cdots dx_s,$$

and a set of sample points $P = (p_1, \ldots, p_N)$, consider an estimate of the form

$$\hat{I} = \frac{1}{N} \sum_{i=1}^{N} f(p_i).$$

The Koksma-Hlawka theorem says that

$$|I - \hat{I}| \leq V(f)\, D_N^*(P). \tag{1}$$

Thus, the error is split into a component $V(f)$ that depends only on the function being integrated and a component $D_N^*(P)$ that depends only on the point sequence. Therefore, so long as $V(f)$ is bounded (and it isn't always bounded), the lower we can make the discrepancy of the points, the lower the maximum error will be.

In s dimensions, it is possible to get sequences such that

$$D_N^*(P) = O\left(\frac{(\log N)^{s-1}}{N}\right).$$

In particular, note that for $s = 1$,

$$D_N^*(P) = O\left(\frac{1}{N}\right).$$

As the number of dimensions increases, we can't do as well as we can in 1D, but it's nearly as good. Note that this convergence rate is much better than the $O(N^{-\frac{1}{2}})$ that standard Monte Carlo gives. However, in higher dimensions the $(\log N)^{s-1}$ factor can be quite large. For example if $s = 6$, then $(\log N)^{s-1}/N > N^{-1/2}$ for all $N < 10^15$. So, this error bound is not always very useful in practice.

$V(f)$ is called the *variation of f in the sense of Hardy and Krause.* It's easy to define in one dimension:

$$V(f) = \int_0^1 |f'(x)|\, dx,$$

as long as the derivative $f'(x)$ is continuous. Basically, it's the sum of the heights of all the monotonic segments of f (see Figure 1).

In two or more dimensions, if f is discontinuous, the variation is infinite and the bounds 1 are meaningless. In three or more dimensions, if f' is discontinuous, the variation is infinite. In general, in s dimensions, the first $s - 2$ derivatives of f must be

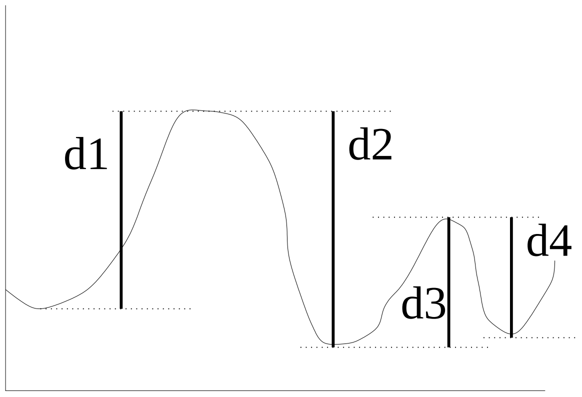


Figure 1: $V(f) \approx d1 + d2 + \ldots$

continuous for $V(f)$ to be bounded. In spite of the lack of theoretical bounds when $V(f)$ is unbounded, however, QMC can still do better than standard MC in practice.

4 Constructing Low Discrepancy Sequences

4.1 1D: Radical inverse sequences

One of the simplest low discrepancy sequences to construct is the *radical inverse sequence.* The idea is to write down an integer in a given base b, and then reflect the digits around the decimal point. When the base used is $b = 2$, this is called the *van der Corput sequence*:

1	.1	1/2
10	.01	1/4
11	.11	3/4
100	.001	3/8
101	.101	5/8
⋮	⋮	⋮

It recursively splits the intervals of the 1D line in half. The discrepancy of this sequence is

$$D_N^*(P) = O\left(\frac{\log N}{N}\right).$$

This matches the best discrepancy that has been attained for infinite sequences in s dimensions:

$$D_N^*(P) = O\left(\frac{(\log N)^s}{N}\right).$$

For finite sequences, better discrepancy is possible; the bound is

$$D_N^*(P) = O\left(\frac{(\log N)^{s-1}}{N}\right).$$

In terms of theoretical bounds on discrepancy, it is believed that these attained bounds are as good as it gets, but the best proven lower bound for sequences (in an arbitrary dimension s) is

$$D_N^*(P) = \Omega\left(\frac{(\log N)^{\frac{s}{2}}}{N}\right)$$

while for a finite point set it is

$$D_N^*(P) = \Omega\left(\frac{(\log N)^{\frac{s-1}{2}}}{N}\right).$$

Thus, most of the continuing work in constructing low-discrepancy sequences centers around reducing the constant factor k (in one dimension):

$$D_N^*(P) = O\left(\frac{\log N}{N}\right) \leq k\frac{\log N}{N}$$

for $N \geq 2$. The best k that has been attained so far is 0.224; it has been proven that $k = 0.06$ is unattainable.

4.2 Two and more dimensions

4.2.1 Halton Sequences

Halton sequences build on the radical inverse used in Van Der Corput sequences and introduce $\phi_b(i)$, the radical inverse of i in base b. To generate an n dimensional sequence, we use the radical inverse base b, with a different base for each dimension:

$$x_i = (\phi_2(i), \phi_3(i), \phi_5(i), \ldots, \phi_{p_s}(i)).$$

The first s prime numbers, $p_1, \ldots, p_s$ are used as the bases for the s dimensions. The discrepancy of these sequences is good:

$$D_N^*(x_i) = O\left(\frac{(\log N)^s}{N}\right).$$

Note that in each dimension, the Halton sequence fills in the sample points from left to right as the unit digit increases from 0 to $b-1$. Permutations of the sequence can improve their performance in practice.

4.2.2 Hammersley Points

If you know the number of samples to be taken in advance, discrepancy can be improved slightly. Hammersley point sets are defined by:

$$x_i = (\frac{i}{N}, \phi_2(i), \phi_3(i), \ldots, \phi_{s-1}(i)),$$

where N is the total number of samples to be taken. See "Ray Tracing and Irregularities of Distribution", by Don Mitchell, in the proceedings of the Third Eurographics Rendering Workshop for more on the uses of Hammersley points in graphics (more precisely, the Hammersley-Zaremba points, which use permutations as mentioned above).

4.2.3 (t, s)-sequences and (t, m, s)-nets

A family of low discrepancy sequences called (t, s)-sequences and (t, m, s)-nets has been constructed based on looking at the distributions of points with respect to b-ary boxes: these are axis aligned boxes, coincident with the lines of $(\frac{1}{b})^i$. They are defined by:

$$E = \prod_{i=1}^{s} [a_i b^{-d_i}, (a_i + 1) b^{-d_i}],$$

where $d_i \geq 0$, $0 \leq a_i < b^{d_i}$, and $\prod$ denotes a product of intervals over all dimensions.

For example, with $b = 5$ and $s = 3$, valid boxes include those of size $1 \times \frac{1}{5} \times \frac{1}{5}$, $\frac{1}{25} \times \frac{1}{125} \times 1$, etc., where the box is aligned on each axis to an integer multiple of its dimension on that axis.

(t, s)-sequences are infinite sequences with low discrepancy with respect to b-ary boxes, and (t, m, s)-nets are finite sequences with similarly good discrepancy; details of the construction of these sequences was beyond the scope of the lecture.

(t, m, s)-nets have some particularly nice properties. By definition, s is the number of dimensions we are integrating over, $0 \leq t \leq m$, and the total number of points $N = b^m$. (t, m, s)-nets are constructed so that if E is a b-ary box with volume $\lambda(E) = b^{t-m}$, then

$$\#\{x_i \in E\} = b^t.$$

The best case of this is when $t = 0$; then any b-ary box of size b^{-m} will have exactly one point in it–exactly what we'd want!

5 Summary

QMC can be a big win when doing numerical integration; keep in mind, though, that discontinuities in the integrand prevent it from being as powerful as one might expect from theoretical bounds in the presence of smooth integrands. Another complication is that classic estimates of variance can't be computed when using QMC, since computing an estimate again will always give the same result.

The Metropolis Sampling Algorithm

Lecture #11: Tuesday, 4 November 1997
Lecturer: Eric Veach
Scribe: Homan Igehy

1 Applications of Monte Carlo Methods

Monte Carlo methods were developed in order to solve difficult integration problems of high dimensionality. The techniques created came out of practical needs rather than theoretical exercises. Before going into the details of the Metropolis Sampling Algorithm, we shall first look at a few typical problems that require the use of Monte Carlo techniques. Monte Carlo is definitely not a solution in wait of a problem.

1.1 Computer Graphics

A large part of computer graphics deals with rendering and the transport of light. The light transport problem is a high dimensional problem: light leaving an emitter makes its way through several bounces on several surfaces (each with its own surface reflectance properties) before hitting the image plane. To make matters more complicated, discontinuities show up in a variety of ways. Within a BRDF, a singularity is present in the case of mirror reflections. A discontinuity in the surface normal of an object causes a discontinuous change in the amount of light reaching the object (e.g. along the edge of a cube). Even if all the BRDF's and surface normals are smooth, object boundaries form discontinuities. On the image plane, these discontinuities manifest themselves as silhouette edges. On a surface, these discontinuities form shadow boundaries. In the case of point light sources, the shadows are discontinuous in value, whereas in the case of area light sources, the shadows are discontinuous in their first or second derivatives. The presence of discontinuities in such a high dimensional problem accentuates the need for Monte Carlo techniques. Let us take a look at the dimensions involved in typical rendering methods.

1.1.1 Distribution Ray Tracing

Distribution ray tracing is similar to standard ray tracing in the sense that rays are traced from the eye towards the light source. The difference is that at each step of the light transport problem, an integral is evaluated. As illustrated in Figure 1, distribution ray tracing can be thought of as a simultaneous integration over:

Variable	Dimensions
time	1
pixel area	2
lens aperture	2
area light source	2
surface reflections	2 each

Thus, in distribution ray tracing we must evaluate a $(7 + 2r)$-dimensional integral, where r is the number of surface reflections considered.

1.1.2 Path Tracing

Path tracing is a generalization of distribution ray tracing in that it includes light transport paths of all lengths. A path can be started in any direction and can be represented as a sequence of points $x_0, x_1, \ldots, x_k$ on the scene surfaces. Each x_i can be either be chosen directly on a surface (e.g. sampling a point on a light source), or by choosing a random direction and casting a ray. In either case, each x_i has two degrees of freedom. If we include an integration over time as well, we get a problem of dimensionality $2(k+1)+1$ for a path of length k. In path tracing, k is not bounded, and we need to integrate over an infinite-dimensional space.

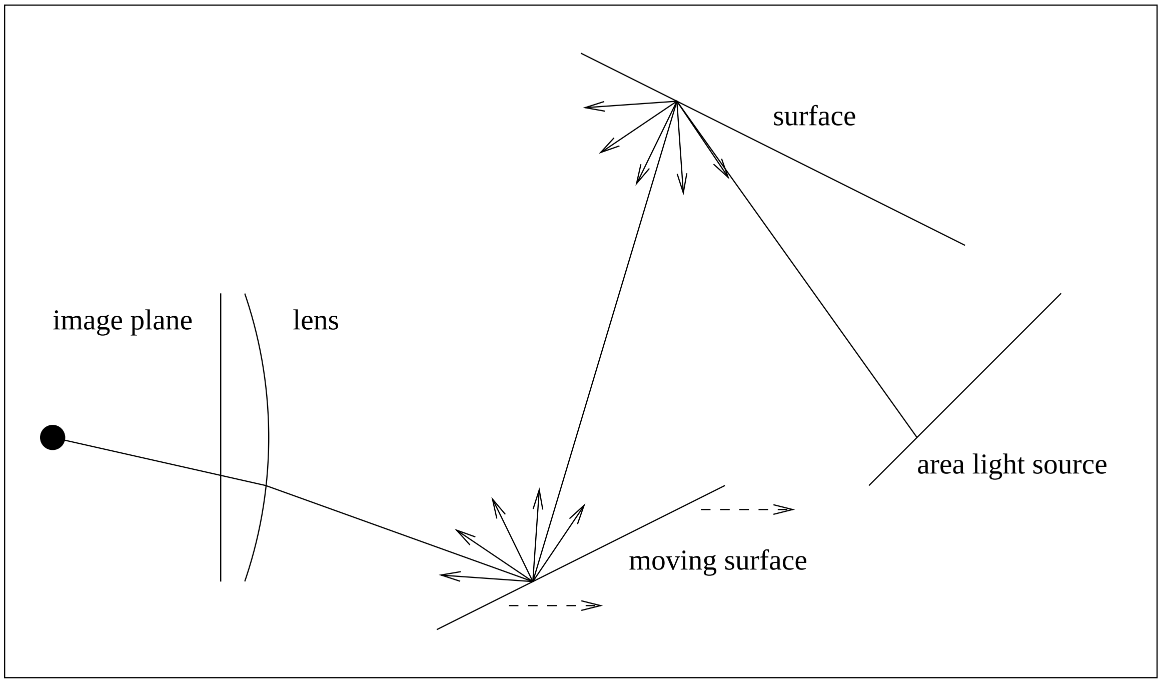


Figure 1: Distribution Ray Tracing.

1.1.3 Radiosity

The form factor computation of radiosity seeks to derive the fraction of light leaving patch A that falls onto patch B, assuming a Lambertian reflectance model. Although an analytic expression has been derived for two arbitrary polygonal patches, this model is slow to evaluate and does not take into account effects of visibility. Thus, this discontinuous 4-dimensional integration problem is amenable to Monte Carlo techniques. Similarly, finite element-based techniques for radiance (rather than radiosity) need to evaluate coefficients that represent the fraction of light that leaves patch A, falls onto patch B, and then gets reflected onto patch C. This is a 6-dimensional integral.

1.2 Finance

Monte Carlo techniques are widely used in the financial world for valuing securities. One class of securities traded is mortgage-backed securities. The mortgages backing these securities have a fixed or variable interest rate and typically have a maturity period of 30 years. The value of a mortgage-backed security depends on a complicated mix of the prime interest rate, the default rate, the remortgaging rate, and a variety of other interdependent factors that vary over time. By modeling these factors over time, a basis for valuation is provided. Solutions to these models involve integrating functions of 20 or 30 or even 360 dimensions. Since these functions are typically smooth, quasi-Monte Carlo techniques work well.

1.3 Physics and Chemistry

The proliferation of Monte Carlo techniques was due to the invention of electronic computers and their application to problems of physics and chemistry. Monte Carlo techniques have been used to solve problems in neutron transport (i.e., bombs), radiation shielding (i.e., reactors), and particle transport (e.g., electron beam lithography).

Another area where Monte Carlo methods are used is to predict the properties of materials. This is the application that originally led to the Metropolis Sampling Algorithm. Problems in this field include the calculation of the density of a liquid at a given temperature, the magnetic properties of alloys, and the modeling phenomena such as superfluidity.

2 A Typical Physics Problem

Before getting into the Metropolis Sampling Algorithm, let us motivate it with a typical physics problem. Imagine a bunch of particles inside a wrap-around box which is at thermal equilibrium, as illustrated in Figure 2. The state of this system can be described by a vector

$$\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m)$$

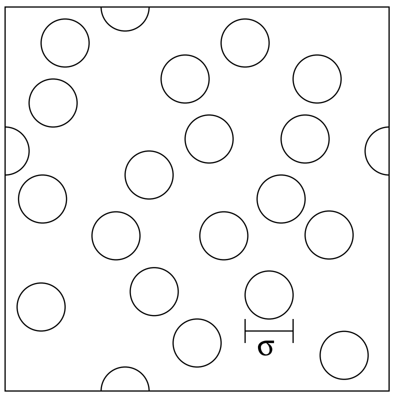


Figure 2: A Typical Physics Problem: Particles with extent σ.

where $\mathbf{x}_i$ is the position of particle i, and we let $\boldsymbol{\Omega}$ represent the space of all possible particle configurations $\mathbf{X}$. (The momenta of the particles are sometimes also required, but we will ignore them in this formulation.) If $\Phi(\mathbf{X})$ represents the energy of the system, then the probability $p(\mathbf{X})$ that the system will be in state $\mathbf{X}$ obeys the Boltzmann distribution:

$$\begin{aligned} p(\mathbf{X}) &= \frac{f(\mathbf{X})}{\int_{\boldsymbol{\Omega}} f(\mathbf{X})\, d\mathbf{X}} \\ \text{where} \qquad f(\mathbf{X}) &= e^{-\Phi(\mathbf{X})/(kT)} \;. \end{aligned}$$

Here T is the temperature, and k is Boltzmann's constant. Note that $f(\mathbf{X})$ is just the unnormalized $p(\mathbf{X})$.

One model for the energy $\Phi(\mathbf{X})$ is to sum up all possible pair potentials:

$$\Phi(\mathbf{X}) = \sum_{1 \le i < j \le m} V(\mathbf{x}_i, \mathbf{x}_j)$$

The hard sphere potential model treats the particles as solid balls of diameter σ that cannot interpenetrate, and any configuration which satisfies this requirement is equally likely. This can be achieved by setting V as follows:

$$V(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} 0 & r_{ij} > \sigma \\ \infty & \text{otherwise} \end{cases}$$

where r_{ij} is distance between particles i and j:

$$r_{ij} = |\mathbf{x}_i - \mathbf{x}_j|$$

Alternatively one could use the Lennard-Jones potential model (illustrated in Figure 3):

$$V(\mathbf{x}_i, \mathbf{x}_j) = 4\varepsilon \left\{ \left(\frac{\sigma}{r_{ij}} \right)^{12} - \left(\frac{\sigma}{r_{ij}} \right)^{6} \right\}$$

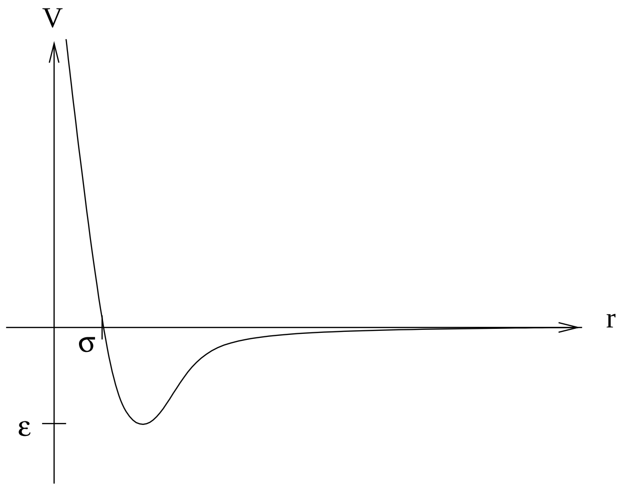


Figure 3: The Lennard-Jones Potential.

In any case, suppose there is some function $g(\mathbf{X})$ in which we are interested. We want to find the expected value of $g(\mathbf{X})$ where $\mathbf{X}$ is distributed proportionally to $f(\mathbf{X})$. In other words, the quantity of interest G is:

$$G = E[g(\mathbf{X})] = \int_{\mathbf{\Omega}} g(\mathbf{X})p(\mathbf{X})\, d\mathbf{X}$$

If we choose $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N$ according to the distribution specified by $p(\mathbf{X})$, an estimator for G is given by:

$$\hat{G} = \frac{1}{N}\sum_{t=1}^{N} g(\mathbf{X}_t)$$

Since $f(\mathbf{X})$ (and consequently $p(\mathbf{X})$) is a complicated function of a huge number of variables (in our case proportional to number of particles), finding an appropriate collection of samples $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N$ poses a formidable task. The Metropolis Sampling Algorithm solves precisely this problem.

3 The Metropolis Sampling Algorithm

The Metropolis Sampling Algorithm was developed in 1953 by Metropolis, Rosenbluth, Rosenbluth, Teller, and Teller (it is also called the M(RT)^2 algorithm for obvious reasons). The algorithm takes a random walk to generate a set of samples that are distributed according to some arbitrary given function. Specifically, suppose want have a domain $\mathbf{\Omega}$ such that $\mathbf{X} \in \mathbf{\Omega}$. Furthermore, let $f(\mathbf{X})$ be some some scalar function. The goal of the M(RT)^2 is to provide a series of samples $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N$ which have a probability density function proportional to $f(\mathbf{X})$. In other words:

Given: A function $f(\mathbf{X})$ where $\mathbf{X} \in \mathbf{\Omega}$.
Goal: Generate $\mathbf{X}_t \sim p(\mathbf{X})$ where $p(\mathbf{X}) \propto f(\mathbf{X})$ and $\int_{\mathbf{\Omega}} p(\mathbf{X})\, d\mathbf{X} = 1$.

3.1 Random Walks

M(RT)^2 generates a set of samples by taking a *random walk* across $\mathbf{\Omega}$. A random walk is simply a sequence of samples $\mathbf{X}_1, \mathbf{X}_2, \ldots \mathbf{X}_N$ where each $\mathbf{X}_t$ is chosen with some probability distribution $p_t(\mathbf{X})$. Furthermore, the random walk generated by M(RT)^2 is a *Markov chain*, meaning that the choice of $\mathbf{X}_t$ depends only on $\mathbf{X}_{t-1}$. If the random walk is independent of the actual value of t, it is called a *time-homogeneous Markov chain*, which can be characterized by a single *transition function* for all t:

$$K(\mathbf{Y} \to \mathbf{X}) = \{\text{density that } \mathbf{X}_t = \mathbf{X} \text{ given that } \mathbf{X}_{t-1} = \mathbf{Y}\}$$

In other words, $K(\mathbf{Y} \to \mathbf{X})$ is the probability density of going from state $\mathbf{Y}$ to state $\mathbf{X}$. Since it is a pdf, it satisfies:

$$\int_{\mathbf{\Omega}} K(\mathbf{Y} \to \mathbf{X})\, d\mathbf{X} = 1 \qquad \text{for all } \mathbf{Y}$$

Thus, for a random walk, we choose an initial state $\mathbf{X}_1$ according to some initial distribution $p_1(\mathbf{X})$. Then $p_2(\mathbf{X} \mid \mathbf{X}_1)$ is set to $K(\mathbf{X}_1 \to \mathbf{X})$, and $\mathbf{X}_2$ is chosen according to that distribution. By repeating this process, $p_t(\mathbf{X} \mid \mathbf{X}_{t-1})$ is set to $K(\mathbf{X}_{t-1} \to \mathbf{X})$ and $\mathbf{X}_t$ is chosen according to that distribution. Now since each $\mathbf{X}_t$ is a random variable, we get the the following distributions for each step of the random walk:

$$\begin{aligned}
p_1(\mathbf{X}) &= \text{initial distribution} \\
p_2(\mathbf{X}) &= \int_{\mathbf{\Omega}} p_1(\mathbf{Y}) K(\mathbf{Y} \to \mathbf{X})\, d\mathbf{Y} \\
p_3(\mathbf{X}) &= \int_{\mathbf{\Omega}} p_2(\mathbf{Y}) K(\mathbf{Y} \to \mathbf{X})\, d\mathbf{Y} \\
&\vdots \\
p_t(\mathbf{X}) &= \int_{\mathbf{\Omega}} p_{t-1}(\mathbf{Y}) K(\mathbf{Y} \to \mathbf{X})\, d\mathbf{Y} \\
&\vdots
\end{aligned}$$

Now if our random walk, which is embodied by $K(\mathbf{Y} \to \mathbf{X})$, is *ergodic*, then there exists a *stationary distribution* $p(\mathbf{X})$ such that:

$$p(\mathbf{X}) = \int_{\mathbf{\Omega}} p(\mathbf{Y}) K(\mathbf{Y} \to \mathbf{X})\, d\mathbf{Y} \tag{1}$$

That means that for some $p(\mathbf{X})$, the application of a step in our random walk will give back the exact same distribution. Furthermore, ergodicity guarantees that the random walk will converge to $p(\mathbf{X})$ for any initial distribution $p_1(\mathbf{X})$:

$$\lim_{t \to \infty} p_t(\mathbf{X}) = p(\mathbf{X})$$

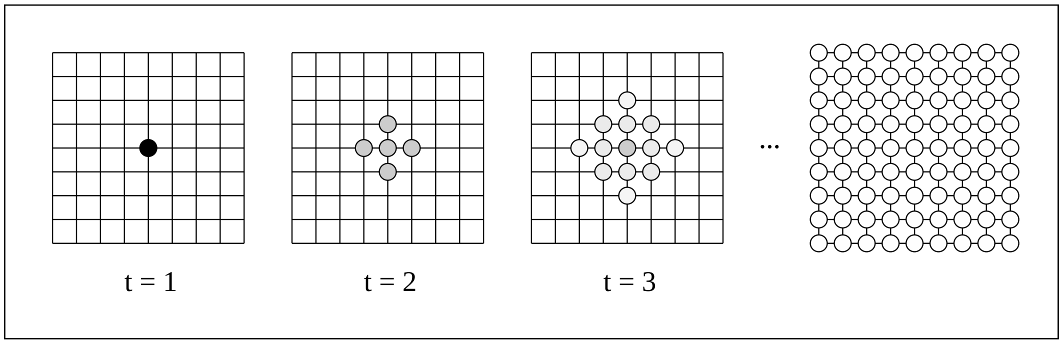


Figure 4: A Random Walk Across a Grid

M(RT)^2 constructs a $K(\mathbf{X} \to \mathbf{Y})$ so that its stationary distribution $p(\mathbf{X})$ is proportional to the desired $f(\mathbf{X})$. Thus, the samples $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_N$ taken along a random walk will eventually obey the desired distribution $p(\mathbf{X})$ no matter what initial distribution $p_1(\mathbf{X})$ is used to pick $\mathbf{X}_1$. Although we will not get into the details of ergodicity, it is a relatively weak condition: there must exist a possibility of getting to any state from any other state in a finite number of steps in non-periodic fashion.

Before getting into the M(RT)^2 construction of $K(\mathbf{Y} \to \mathbf{X})$, we will first give an example of an ergodic random walk for clarity. Imagine a wrap-around grid of size $M \times M$, as illustrated in Figure 4. Each node on the grid represents a distinct point in the space $\mathbf{\Omega}$:

$$\begin{aligned} \mathbf{\Omega} &= \{\mathbf{X} = [i,j] \mid 1 \le i,j \le M\} = \{[1,1], \ldots, [M,M]\} \\ |\mathbf{\Omega}| &= M^2 \end{aligned}$$

We want to generate samples for $\mathbf{\Omega}$ with a uniform distribution:

$$\begin{aligned} f([i,j]) &= 1 \\ p([i,j]) &= \frac{1}{M^2} \end{aligned}$$

We propose a transition function $K(\mathbf{Y} \to \mathbf{X})$ that will keep us at the current node or move us to an adjacent node with equal probability:

$$K([i',j'] \to [i,j]) = \begin{cases} \frac{1}{5} & \text{if } |i'-i| + |j'-j| \le 1 \\ 0 & \text{otherwise} \end{cases}$$

Figure 4 shows what happens with our random walk after a few steps, assuming an initial distribution $p_1([i,j])$ which puts $\mathbf{X}_1$ at the center. The intensity of a location corresponds to the value of the density function $p_t([i,j])$ at step t of the random walk. By observation, we can see that $p_t([i,j])$ converges to $p([i,j])$, and that $p([i,j])$ is indeed a static distribution.

Note that if $\mathbf{\Omega}$ is a discrete domain as in the above example, then $p(\mathbf{X})$ may be replaced by a column vector $\mathbf{p}$ with height equal to $|\mathbf{\Omega}|$. Each entry represents the discrete probability of being in a given state of $\mathbf{\Omega}$. Similarly, $p_t(\mathbf{X})$ may be replaced with a column vector $\mathbf{p_t}$. Furthermore, $K(\mathbf{Y} \rightarrow \mathbf{X})$ may be replaced by a matrix $\mathbf{K}$ of size $|\mathbf{\Omega}| \times |\mathbf{\Omega}|$. Each entry (l, k) represents the probability of the random walk moving from state l to state k, and the sum of any row is equal to 1. Given this matrix notation, the distribution for our random walk becomes:

$$\mathbf{p_t} = \mathbf{K}\mathbf{p_{t-1}} = \mathbf{K}^{t-1}\mathbf{p_1}$$

Now if $\mathbf{K}$ is ergodic, then for any $\mathbf{p_1}$ we get:

$$\lim_{t\rightarrow\infty} \mathbf{K}^t\mathbf{p_1} = \mathbf{p}$$

And the equation satisfied by the stationary distribution becomes:

$$\mathbf{p} = \mathbf{K}\mathbf{p}$$

Thus $\mathbf{p}$ is an eigenvector of $\mathbf{K}$ with eigenvalue 1. Thus in discrete domains, finding a transition function whose stationary distribution is proportional to an arbitrary function is thus equivalent to constructing a matrix with a specific eigenvector (modulo ergodicity).

3.2 The Tentative Transition Function

In order to construct $K(\mathbf{Y} \rightarrow \mathbf{X})$, we will break it up into two parts. A *tentative transition* is first used to propose a transition from the current state into some other state according to some chosen distribution. For example, in the aforementioned m-particle problem, a tentative transition could be to move a single particle by some small random distance. The pdf of this *tentative transition function* is denoted by $T(\mathbf{Y} \rightarrow \mathbf{X})$. Now, once this transition is proposed, we either choose to move to this new state for the next step of the random walk, or else we stay in the same state for the next step of the random walk. This *acceptance probability*, whose pdf is denoted by $A(\mathbf{Y} \rightarrow \mathbf{X})$, is chosen in a manner such that the combination of $T(\mathbf{Y} \rightarrow \mathbf{X})$ and $A(\mathbf{Y} \rightarrow \mathbf{X})$ form a $K(\mathbf{Y} \rightarrow \mathbf{X})$ whose stationary distribution is the desired function $p(\mathbf{X})$.

3.3 Detailed Balance

The question we ask ourselves at this point is as follows: what is the relationship between $T(\mathbf{Y} \rightarrow \mathbf{X})$, $A(\mathbf{Y} \rightarrow \mathbf{X})$, and $p(\mathbf{X})$? To answer this, we borrow an idea from physical equilibria. Imagine a box filled with a gas and another box that contains nothing. A clamped tube connects the two boxes. Once the tube is unclamped, gas begins to flow from the first box into the second box. However, once the second box starts holding some gas, some of that gas flows back into the first box. After some time, the two boxes reach a state of equilibrium. Even though gas flows back and forth between the two boxes and the state of the system is not static, the amount that goes in each direction is equal, and thus the system is at equilibrium.

Let us examine what happens when we apply the notion of *detailed balance* to our problem:

$$p(\mathbf{X})T(\mathbf{X}\to\mathbf{Y})A(\mathbf{X}\to\mathbf{Y}) = p(\mathbf{Y})T(\mathbf{Y}\to\mathbf{X})A(\mathbf{Y}\to\mathbf{X}) \tag{2}$$

This equation says that once a stationary distribution $p(\mathbf{X})$ is reached, then the chance of being in state $\mathbf{X}$ and then taking a proposed transition into state $\mathbf{Y}$ is equal to the chance of being in state $\mathbf{Y}$ and then taking a proposed transition into state $\mathbf{X}$. To see how this provides a solution to stationary distribution, we rewrite Equation (1) in terms of $T(\mathbf{Y}\to\mathbf{X})$ and $A(\mathbf{Y}\to\mathbf{X})$:

$$\begin{aligned} p'(\mathbf{X}) \;=\; & \left[\int_{\mathbf{\Omega}} p(\mathbf{Y})T(\mathbf{Y}\to\mathbf{X})A(\mathbf{Y}\to\mathbf{X})\,d\mathbf{Y}\right] + \\ & \left[p(\mathbf{X})\left(1-\int_{\mathbf{\Omega}} T(\mathbf{X}\to\mathbf{Y})A(\mathbf{X}\to\mathbf{Y})\,d\mathbf{Y}\right)\right] \end{aligned}$$

The first term of the above equation expresses the chance of starting in state $\mathbf{Y}$ and taking a transition into $\mathbf{X}$. The second term expresses the chance of starting in state $\mathbf{X}$ and rejecting a transition into another state $\mathbf{Y}$. These are all the ways that one can end up in state $\mathbf{X}$ after a step of the random walk. Rearranging the terms and moving $p(\mathbf{X})$ into the integral (since it is constant with respect to $\mathbf{Y}$), we get:

$$\begin{aligned} p'(\mathbf{X}) \;=\; & p(\mathbf{X}) + \\ & \int_{\mathbf{\Omega}} p(\mathbf{Y})T(\mathbf{Y}\to\mathbf{X})A(\mathbf{Y}\to\mathbf{X}) - p(\mathbf{X})T(\mathbf{X}\to\mathbf{Y})A(\mathbf{X}\to\mathbf{Y})\,d\mathbf{Y} \end{aligned}$$

Thus in order to get a stationary distribution, the integral in the above equation must evaluate to zero. One easy way to do this is to enforce the detailed balance of Equation (2) in our choice of $T(\mathbf{Y}\to\mathbf{X})$ and $A(\mathbf{Y}\to\mathbf{X})$. Thus, we are left with:

$$p'(\mathbf{X}) = p(\mathbf{X})$$

3.4 The Acceptance Probability

Since $p(\mathbf{X})$ is given to us (by way of $f(\mathbf{X})$), and $T(\mathbf{Y}\to\mathbf{X})$ is some guess for transitions, then our only choice is to set $A(\mathbf{Y}\to\mathbf{X})$ appropriately in order to meet the condition of detailed balance. From Equation (2), we know that we must have:

$$\frac{A(\mathbf{Y}\to\mathbf{X})}{A(\mathbf{X}\to\mathbf{Y})} = \frac{p(\mathbf{X})T(\mathbf{X}\to\mathbf{Y})}{p(\mathbf{Y})T(\mathbf{Y}\to\mathbf{X})} \tag{3}$$

One way to achieve this is:

$$A(\mathbf{Y}\to\mathbf{X}) = min\left(1, \frac{f(\mathbf{X})T(\mathbf{X}\to\mathbf{Y})}{f(\mathbf{Y})T(\mathbf{Y}\to\mathbf{X})}\right) \tag{4}$$

Noting that $f(\mathbf{X})/f(\mathbf{Y}) = p(\mathbf{X})/p(\mathbf{Y})$ since $p(\mathbf{X}) \propto f(\mathbf{X})$, it is clear from substitution that Equation (4) satisfies Equation (3). Note that all of the above scalar values can be readily evaluated for any pair of points. Thus we are now ready for the M(RT)^2 algorithm.

3.5 The M(RT)^2 Algorithm

Assuming we have a way of evaluating $f(\mathbf{X})$ at a point, and a way of evaluating $T(\mathbf{Y} \rightarrow \mathbf{X})$, then the algorithm for generating samples with a pdf proportional to $f(\mathbf{X})$ is as follows:

```
$\mathbf{X}_1 \sim p_1(\mathbf{X})$
for $t = 2$ to $N$
    $\mathbf{X}_{tent} \sim T(\mathbf{X}_{t-1} \rightarrow \mathbf{X})$
    if (random() $< A(\mathbf{X}_{t-1} \rightarrow \mathbf{X}_{tent})$)
        $\mathbf{X}_t = \mathbf{X}_{tent}$
    else
        $\mathbf{X}_t = \mathbf{X}_{t-1}$
```

Note that one may want to reject the first several samples since it takes a while for the random walk to reach the stationary distribution.

Review of Splines and Introduction to Subdivision

Lecture #12:	Thursday, Nov 6 1997
Lecturer:	Denis Zorin
Scribe:	Xiaomei Zhu

In this lecture we will review some simple facts about splines and introduce subdivision. Next few lectures will be about subdivision.

1 The Idea of Subdivision.

Splines and subdivision are closely related. Splines use piecewise polynomials to interpolate curves. There are many ways to evaluate splines. One such method is the de Casteljau algorithm; for quadratic splines it is especially simple. Start with the control polygon of a spline curve; insert two new control points on each edge connecting two old control points. Each new control point divides the edge in the ratio 3:1. Connect the new control points and discard the old.

This algorithm can be generalized: note that each new control point is a linear combination of old control points with weights 1/4 and 3/4. Rather then using these specific weights, we can use other weights. This is the idea of subdivision: use algorithms similar to those that are used for spline evaluation, but modify these algorithms so that they generate curves with different properties (which becomes apparent in the one-dimensional case) and so that they handle arbitrary control meshes for surfaces (which becomes an issue in the two-dimensional case).

2 Subdivision and other modeling methods.

We briefly compare subdivision to several free-form surface and curve representations used in geometric modeling: splines, implicit surfaces, and variational surfaces. All of these representations are primarily used for smooth surfaces.

We can compare these representations from a number of points of view.

1. *Efficiency.* Computational cost is an important aspect of a modeling method. Subdivision is easy to implement and is computationally efficient; the efficiency is a property inherited from splines since the algorithms for evaluation are essentially the same. From the point of view of efficiency, implicit surfaces are not so good. An algorithm such as marching cubes is required to generated the polygonal approximation needed for rendering. Variational surfaces are even worse: a global optimization problem has to be solved each time a control of the surface is changed.

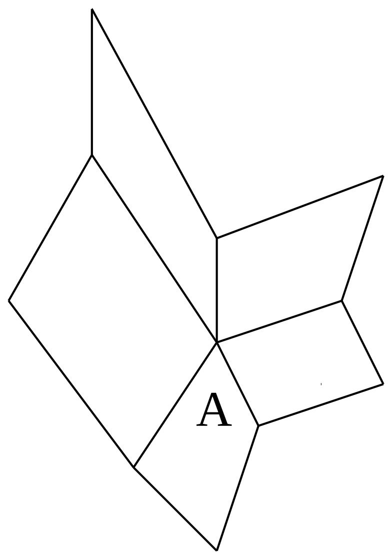


Figure 1: A mesh with an extraordinary vertex: number of faces meeting at A is not 4. Special efforts are required to guarantee smoothness between spline patches meeting at the extraordinary point; subdivision handles such situations in a natural way.

2. *Arbitrary topology.* It is important to be able to produce smooth surfaces with control meshes of arbitrary topology. "Arbitrary topology" means two things: first, the topological genus of the mesh and associated surface can be arbitrary. Second, the structure of the graph formed by the edges and vertices of the mesh can be arbitrary; specifically, each vertex may have an arbitrary degree.

   These two aspects are related: for example, any control mesh for a sphere (topological genus 0) necessarily has vertices of different degrees. Only closed surfaces of genus 1 can be represented using a mesh with all vertices having the same degree.

   From a practical point of view arbitrary control meshes have a considerable advantage: a user of a modeling system would not want to worry about degrees of control vertices when defining a shape.

   When rectangular spline patches are adapted for arbitrary control meshes, enforcing smoothness at extraordinary vertices becomes difficult and considerably increases the complexity of the representation. Implicit surfaces can be of arbitrary topological genus, but the genus, precise location, and even connectivity of a surface are difficult to control typically. Variational surfaces can handle arbitrary topology better than any other representation, but the computational cost has been thus far prohibitive. Subdivision canhandle arbitrary topology quite well without losing efficiency; this is one of its key advantages.

3. *Surface features.* Subdivision allows more flexible controls than is possible with splines. In addition to choosing locations of control points, one can manipulate

the coefficients of subdivision to achieve such effects such as the the sharpness of creases or the control behavior of the boundary curves. Implicit surfaces, on the other hand, are very difficult to control. Variational surface, however, are the best for creating features.

4. *Complex geometry.*

   For interactive applications, efficiency is crucial. Subdivision-based representations for complex geometry are very efficient and have an advantage in situations when the geometry has to be modified (editing, animation). For applications that only require the visualization of the geometry, other representations, such as H. Hoppe's progressive meshes, are likely to be more suitable.

3 Review of Splines

3.1 Piecewise-Polynomial Curves

Splines are piecewise polynomial curves. In the case of quadratic splines, each polynomial segment of a curve can be written as

$$\begin{aligned} x(t) &= a_2t^2 + a_1t + a_0 \\ y(t) &= b_2t^2 + b_1t + b_0 \end{aligned}$$

Where a_2, a_1, a_0, b_2, b_1, and b_0 are constant coefficients. Spline curves are typically specified by *control points* rather then by the coefficients. The curve can be modified by moving the control points. Moving a control point has the greatest effect on the part of the curve near that control point. For B-splines, the effect of changing the position of a single control point is actually local: only a finite portion of the curve is affected.

Specifying the curve with control points amounts to rewriting the equations above in the following form.

$$\begin{aligned} x(t) &= \sum C_x^i B_i(t) \\ y(t) &= \sum C_y^i B_i(t) \end{aligned}$$

$B_i(t)$ are called the basis functions. The collection of basis functions for $i = 0, 1, 2, ..., n$ is the basis for the space of all curves that we can represent. For uniform splines, the basis functions satisfy

$$B_i(t) = B(t - i)$$

We choose the basis function $B_i(t)$ in such a way that the resulting curves are smooth and that the influence of the control points is local. One way to ensure smoothness is to use smooth basis functions. [1] Since polynomials themselves are inifinitely smooth, we only have to make sure that derivatives match at the points where two polynomial segments meet. The higher the degree of the polynomial, the more derivatives we are able to match. We also want the influence of a control point to be maximal at regions of the curve close to it, and for this influence to decrease as we move away along the curve, and for it to disappear completely at some distance. Finally, we would like the basis functions to be piecewise polynomial, and we should be able to represent any piecewise-polynomial curve of a given degree with the apropriate basis. In a moment, we will show how to construct basis functions satisfying all these requirements for polynomial curves of arbitrary degree.

When we talk about curves, it is important to distinguish the curve itself and the graphs of the coordinate functions of the curve (which also can be thought of as curves). For example, a curve can be described by equations $x(t) = sin(t)$, $y(t) = cos(t)$. The curve itself is a circle, but the coordinate functions are sinusoids. For the moment, we are going to concentrate on representing the coordinate functions.

3.2 B-Splines Basis Functions

We start with the simplest case: piecewise constant coordinate functions. Any piecewise constant function can be written as

$$x(t) = \sum c_i^x U_i(t)$$

where $U_i(t)$ is the box function defined as

$$\begin{aligned} U(t) &= 1 \quad \text{if} \quad 0 \le t < 1 \\ &= 0 \quad \text{otherwise} \end{aligned}$$

The functions $U_i(t) = U(t - i)$ are translates of U(t). Furthermore, let us represent The continuous convolution of two functions $f(t)$and $g(t)$ with

$$(f \otimes g)(t) = \int f(s)g(t-s)ds$$

Now a B-spline basis function of degree n can be obtained by convolving the basis function of degree $n-1$ with the box $U(t)$. For example, the B-spline basis function of degree 1 is defined as the convolution $U(t)$ with itself. We need to compute

$$\int U(s)U(t-s)ds$$

Graphically, this convolution can be evaluated by sliding one box function along the coordinate axis from minus to plus infinity while keeping the second box fixed. The value

[1]The smoothness of the basis functions guarantees the smoothness of the coordinate functions of the curve. However, it does not guarantee the geometric smoothness of the curve. We will return to this distinction in our discussion of subdivision surfaces.

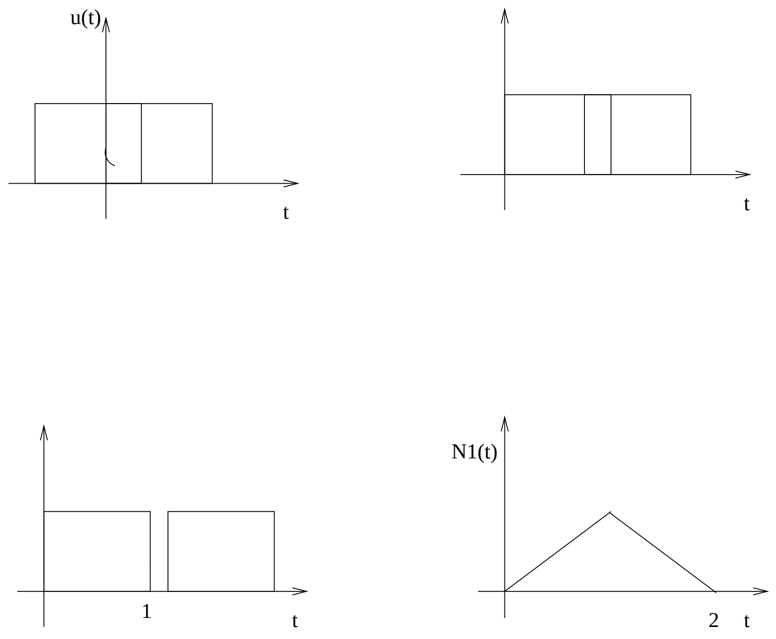


Figure 2: The definition of degree 1 B-Spline basis function through convolution of $U(t)$ with itself.

of the convolution for a given position of the moving box is the area under the product of the boxes, which is just the length of the interval where both boxes are non-zero. At first the two graphs do not have common support. Once the moving box reaches 0, there is a growing overlap between the supports of the graphs. The value of the convolution grows with t until $t = 1$. Then the overlap starts decreasing, and the value of the convolution decreases down to zero at $t = 2$. The function $N_1(t) = \int U(s)U(t-s)ds$ is the linear hat function as shown in Figure 2.

We can compute the B-spline basis function of degree 2 convolving $N_1(t)$ with the box $U(t)$ again.

$$N_2(t) = \int N_1(s)U(t-s)ds$$

In this case, the resulting curve consists out of three quadratic segments defined on intervals $(0,1)$, $(1,2)$ and $(2,3)$. This is illustrated in Figure 3.

In general, by convolving j times, we can get a B-spline basis function of degree j

$$N_j(t) = \int N_{j-1}(s)U(t-s)ds$$

Now convolution has a remarkable property

Theorem 1. *If $f(t)$ is C^k-continuous, then $U(t) \otimes f(t)$ is C^{k+1}-continuous.*

It follows from this theorem that the B-spline basis function of degree n is C^{n-1} continuous because the basis function of degree 1 is C^0-continuous. Defining B-spline basis functions

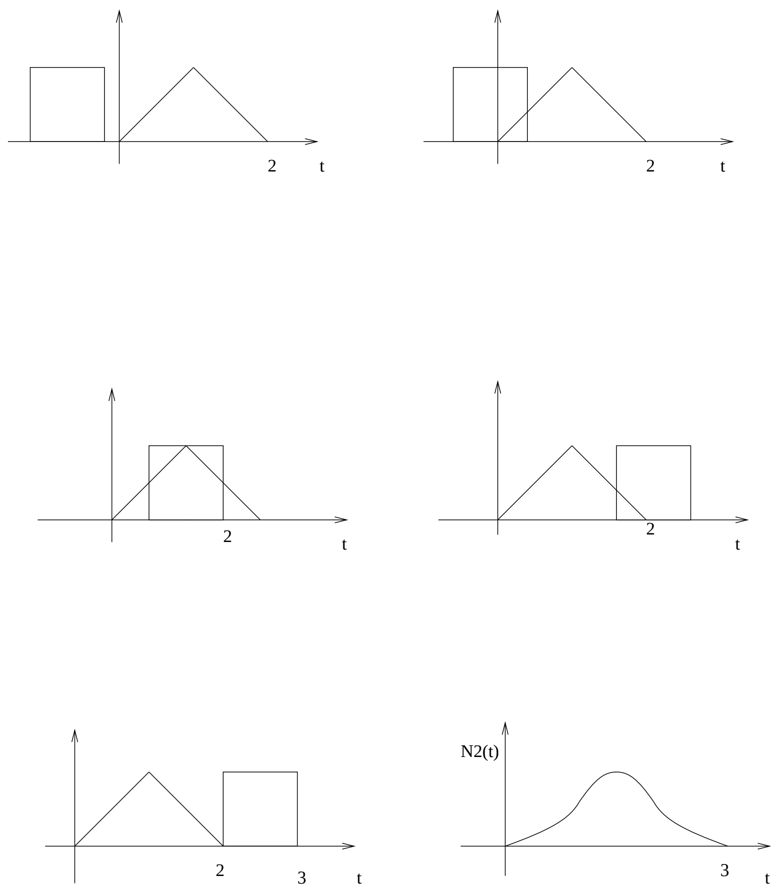


Figure 3: $N_2(t)$ as convolution of $N_1(t)$ with $U(t)$.

through convolution leads us to a scaling equation for splines similar to the dilation equation that we had for scaling functions of wavelet bases. Recall that $U(t)$ is Haar scaling function and that it satisfies the dilation (scaling) equation

$$U(t) = U(2t) + U(2t-1) \tag{1}$$

We have defined the B-spline basis function of degree j as

$$N_j(t) = \otimes_{i=0}^{j} U(t) \tag{2}$$

We can obtain a scaling equation for the basis function $N_j(t)$, which is actually a special case of deBoor's formula. Observe that convolution is a linear operation

$$\begin{aligned} f \otimes (g+h) &= f \otimes g + f \otimes h \\ f \otimes (ag) &= a(f \otimes g) \end{aligned}$$

where f and g are functions and a is a constant. By substituting Equation 1 into Equation 2 and expanding, we obtain an expression of the form

$$N_j(t) = \sum s_k N_j(2t - k)$$

where s_k are some constants. Next time we will show that the coefficients s_k allow us to compute successive piecewise-linear approximations to the limit curve starting with the control polygon.

Analysis of Subdivision Curves

Lecture #13: Tuesday, Nov 11 1997
Lecturer: Denis Zorin
Scribe: Hoofar Razavi

1 Introduction

As we saw in the last lecture, the basis functions used for generating a B-spline of degree n is derived from the convolving the box function with itself n times:

$$N(t) = \bigotimes_{i=0}^{n} U(t)$$

where

$$U(t) = \begin{cases} 1 & \text{if } 0 \leq t < 1 \\ 0 & \text{otherwise} \end{cases}$$

and $\otimes$ denotes convolution. These basis functions all have local support and are piecewise polynomial with degree n. However, more importantly, they also satisfy the dilation equation, which we first encountered in the context of wavelets. The dilation equation for the B-spline basis is simply:

$$N(t) = \sum_k s_k N(2t-k) \tag{1}$$

From before we also know that if a set of basis functions satisfy the dilation equation multiresolution techniques such as successive refinement can be applied to functions in the spaces these basis functions span. We set out to find s_k so that we can apply these techniques to the representation of our curves. If you are familiar with DeBoor's knot insertion algorithm, the above formula can be interpreted as a special case of DeBoor's formula.

2 The Derivation

Recall the following properties of convolution for functions $f(t)$, $g(t)$, and $h(t)$, given that $m(t) = f(t) \otimes g(t)$:

$$\begin{aligned} f(t) \otimes (g(t)+h(t)) &= f(t)\otimes g(t) + f(t) \otimes h(t) && \text{linearity} \\ f(t-i) \otimes g(t-k) &= m(t-i-k) && \text{time shift} \\ f(2t) \otimes g(2t) &= \tfrac{1}{2} m(2t) && \text{time scaling} \end{aligned}$$

Also, recall from last time that the box function can be written in terms of its own dilates

$$U(t) = U(2t) + U(2t-1)$$

Now, looking at the first degree basis and using the above observation we have

$$\begin{aligned} N(t) &= U(t) \otimes U(t) && \text{first degree basis function} \\ U(t) &= U(2t) + U(2t-1) \\ N(t) &= (U(2t) + U(2t-1)) \otimes (U(2t) + U(2t-1)) \end{aligned}$$

Rewriting the above using the convolution properties presented we get

$$\begin{aligned} N(t) &= U(2t) \bigotimes U(2t) + U(2t) \bigotimes U(2t-1) + U(2t) \bigotimes U(2t-1) + U(2t-1) \bigotimes U(2t-1) \\ &= \frac{1}{2}N(2t) + \frac{1}{2}N(2t-1) + \frac{1}{2}N(2t-1) + \frac{1}{2}N(2t-1-1) \\ &= \frac{1}{2}N(2t) + N(2t-1) + \frac{1}{2}N(2t-2) \end{aligned}$$

Comparing the above derivation to Equation 1 we arrive at the coefficients $s_0 = \frac{1}{2}$, $s_1 = 1$, and $s_2 = \frac{1}{2}$. We have managed to rewrite the basis function for a first degree B-spline in terms of *its* own dilates as well. The above derivation generalizes to B-spline basis functions of higher degrees. Now we compute the coefficients s_k for degree $n-1$ splines splines

$$\begin{aligned} N_{n-1}(t) &= \bigotimes_{i=0}^{n} U(t) \\ &= \bigotimes_{i=0}^{n} (U(2t) + U(2t-1)) \end{aligned}$$

With the derivation of the coefficients of the first degree spline, we make the following observation. The coefficients of the translates of the dilate of N(t) in equation 1 are simply the elements of the matrix S. Recall that the binomial expansion derives the coefficients for each term of the exponentiation:

$$(x+y)^n = \sum_{i=0}^{n} a_i x^{n-i} y^i$$

where

$$a_i = \begin{pmatrix} n \\ i \end{pmatrix}$$

Similarly, because the convolution operator is linear, just like multiplication, we can derive our coefficients simply through the binomial expansion by rewriting the above equation.

$$\begin{aligned} N(t) &= \underbrace{(U(2t)+U(2t-1))\bigotimes(U(2t)+U(2t-1))\bigotimes(U(2t)+U(2t-1))}_{n}\ldots \\ &= \sum_{i=0}^{n} s_i \underbrace{U(2t)\bigotimes U(2t)\bigotimes \ldots}_{i} \underbrace{U(2t+1)\bigotimes U(2t+1)\bigotimes \ldots}_{n-i} \end{aligned}$$

From the above it is easily seen that the s_k coefficients are none other than the coefficients of the binomial expansion for $(x+y)^n$ divided by 2^n.

3 Representing Curves

Consider a point in the plane

$$\begin{bmatrix} c_x^i \\ c_y^i \end{bmatrix}$$

to be the i^{th} control point of a curve we would like to represent. The curve, $\gamma(t)$, is thus represented as

$$\gamma(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \sum_i \begin{bmatrix} c_x^i \\ c_y^i \end{bmatrix} N(t-i)$$

As we saw in the last lecture, each of the translates of $N(t)$ has a maximum value at some point during the interval it is nonzero. Writing our curve in this manner ensures that each control point influences the curve as well as guaranteeing that such influence is limited, due to the local support of translates of $N(t)$.

We can rewrite the above more compactly as

$$\gamma(t) = \sum_i c_i N(t-i)$$

Now consider $\mathbf{c}$, the vector of control points of a given curve:

$$\mathbf{c} = \begin{bmatrix} \vdots \\ c_{-2} \\ c_{-1} \\ c_0 \\ c_1 \\ c_2 \\ \vdots \end{bmatrix}$$

and the vector $\mathbf{N}(t)$, which has as its elements the translates of the function N, the desired basis function, defined above:

$$\mathbf{N} = \left[\ \ldots \quad N(t+2) \quad N(t+1) \quad N(t) \quad N(t-1) \quad N(t-2) \quad \ldots \ \right]$$

In this notation we can denote our curve as $\mathbf{Nc}$. Since we know that each of the elements of $\mathbf{N}$ can be represented in terms of its dilates we rewrite the whole vector as such

$$\mathbf{N}(2t) = \left[\ \ldots \quad N(2t+2) \quad N(2t+1) \quad N(2t) \quad N(2t-1) \quad N(2t-2) \quad \ldots \ \right]$$

Where we relate the two vectors with the matrix S as follows:

$$\mathbf{N}(t) = \mathbf{N}(2t)S$$

Rewriting $\gamma(t)$ based on the above representation yields:

$$\gamma(t) = \mathbf{N}(2t)S\mathbf{c}$$

We can think of $\mathbf{N}(2t)$ as our new basis and $S\mathbf{c}$ as our new control points. Since each successive rewriting of the function $\mathbf{N}$ uses basis functions that are twice as narrow as the functions on the previous step, they are shifted by one half of the shift on the previous step. The new control points correspond to values of t which are twice as dense as on the previous level. As we will see below, the new control points are simply linear combination of the old points since they are related through the matrix S. The elements of S are related to the coefficients s_k in Equation 1 by

$$S_{2i+k,i} = s_k$$

Representing our curve, using the above, at each degree gives us a set of successive refinements of the curve:

$$\begin{array}{rcll}
\gamma(t) & = & \mathbf{N}(t)\mathbf{c}^0 & \qquad\qquad\qquad \text{Zeroth degree} \\
& = & \mathbf{N}(2t)\mathbf{c}^{\mathbf{1}} \quad = \mathbf{N}(t)S\mathbf{c}^{\mathbf{0}} & \qquad \text{First degree} \\
& & \vdots & \\
& = & \mathbf{N}(2^j t)\mathbf{c}^{\mathbf{j}} \quad = \mathbf{N}(t)S^j\mathbf{c}^{\mathbf{0}} & \qquad J^{th} \text{ degree}
\end{array}$$

from which we can see the relationship between control points at different levels:

$$\mathbf{c}^{\mathbf{j+1}} = S\mathbf{c}^{\mathbf{j}}$$

where S is our infinite subdivision matrix. Looking more closely at one component, i, of our control points we see that:

$$c_i^{j+1} = \sum_l S_{i,l} c_l^j$$

To find out exactly which s_k is affecting which term, we can divide the above into odd and even entries. For the odd entries we have:

$$c_{2i+1}^{j+1} = sum_l S_{2i+1,l} c_l^j = sum_l s_{2(i-l)+1} c_l^j$$

and for the even entries we have

$$c_{2i}^{j+1} = sum_l S_{2i,l} c_l^j = sum_l s_{2(i-l)} c_l^j$$

From which we essentially get two different subdivision rules one for the new even points of the curve and one for the new odd points. As examples of all of the above, let us consider a couple of concrete cases. For piecewise *linear* subdivision, the basis functions are hat functions. The odd coefficients are $\frac{1}{2}$ and $\frac{1}{2}$, and 1 for the even point. For second degree splines, the odd coefficients turn out to be $\frac{1}{4}$ and $\frac{3}{4}$, and $\frac{3}{4}$ and $\frac{1}{4}$ for the even ones.

4 Discrete Convolution

The coefficients s_k can also be derived from another perspective, namely discrete convolution. More explicitly, using generating functions for a sequence x_k we can consider a polynomial

$$X(z) = \sum_k x_k z^k$$

where $X(z)$ is the z-transform of the sequence x_k; for the case of two coefficients we have

$$(a \bigotimes b)_k = \sum_n a_{k-n} b_n$$

Suppose we have two functions that satisfy the dilation equation:

$$\begin{aligned} f(t) &= \sum_k h_0(k).f(2t-k) \\ g(t) &= \sum_k h_1(k).g(2t-k) \end{aligned}$$

Then $f \otimes g$ also satisfies the dilation equation with coefficients $h'(k) = 1/2 h_0 x h_1$ where

$$h'(k) = \frac{1}{2} \sum_i h_0(k-i) h_1(i)$$

Since $U(t)$ satisfies the dilation equation $U(t) = U(2t) + U(2t-1)$, $N(t) = X_{i=0}^{n-1} U(t)$ satisfies a dilation equation $N(t) = \sum_k s_k N(2t-k)$ with

$$s_k = (1/2)[...0110...]x(1/2)[...0110...]x...x(1/2)[...0110...] \quad (n \text{ times}).$$

The z-transform of [... 0 1 1 0 ...] is 1+z.

We obtain the following generating function for splines:

$$S(z) = \frac{(1+z)^n}{2^{n-1}}$$

Again, the values s_k become apparent: they are simply the coefficients of each of the terms in the generating function above.

5 Interpolation vs. Approximation

Splines are not interpolating, that is, they do not pass through their control points. We can generate interpolating curves by choosing the even rows of the S matrix to be $\ldots 0, 0, 1, 0, \ldots$, so that the control points, once added are never moved. Then we choose the coefficients for the odd points in such a way that the generated curves are smooth. One such choice is the four point scheme, first introduced by Dyn, Levin, and Gregory), which has generating function

$$S(z) = \frac{1}{16}(-z^{-3} + 4z^{-2} - z^{-1})(1 + z)^4$$

Recall that convolving with a box increases smoothness and convolution with $U(t)$ means multiplication of the generating function by $(1 + z)/2$. We have four factors $(1 + z)/2$ in the generating function above, but three of them are required just to get subdivision to converge. One extra term $(1 + z)/2$ ensures that the curve is smooth.

In general, the difficult part is to find a set of coefficients for which subdivision converges. There is no general method to achieve this; once we have a convergent subdivision scheme, we can obtained desired smoothness by convolving with the box.

6 Subdivision vs. Splines

The subdivision coefficients s_k's need not be fixed as we presented them so far. We can vary them, both between levels of subdivision and within one level, between points to obtain different curves. In this regard, splines are just a special case of the more general class of curves, subdivision curves.

There is no sufficiently strong reason for using subdivision in one dimension (in fact none of the commercial line drawing packages do) but the argument becomes much more compelling in higher dimensions.

Classical Subdivision Surfaces

Lecture #14:	Thursday, 13 Nov 1997
Lecturer:	Denis Zorin
Scribe:	Mark Wang
Reviewer:	Hoofar Razavi

In this lecture we discuss the basic ideas of the analysis of subdivision, using subdivision curves as an example, and introduce the a simple subdivision scheme for surfaces – Loop scheme.

1 Subdivision of Curves near Extraordinary Points

Last time we have shown that the uniform B-spline curves can be thought of as a special case of subdivision curves. So far, we have seen only examples for which we use a fixed set of coefficients to compute the control points everywhere. The coefficients define the appearance of the curve, for example, whether it is smooth or has sharp corners. It is possible to control the appearance of the curve by modifying the subdivision coefficients locally.

Consider the four-point curve interpolation algorithm from the previous lecture. Mark a single control point of the curve. Suppose that we change the coefficients that are used to compute new control points near a marked control point on the coarsest level. For example, we can replace the coefficients $-1/16$, $9/16$, $9/16, -1/16$ of the four-point scheme by the coefficients 0, 1/2, 1/2, 0, whenever we compute a new control point adjacent to the marked point. As a result, we introduce a corner into the limit curve at the marked point.

Changing the coefficients of subdivision for computing a fixed number of neighbors of a marked point on each level does not affect the smoothness of the curve outside of a arbitrarily small neighborhood of the control point.

To determine *local* properties of a subdivision curve, we do not need the whole infinite vector of control points or the infinite matrix describing subdivision of the entire curve. Smoothness is a local property of a curve, which means that we can consider an arbitrarily small piece of the curve around a given point, and still be able to determine if the curve is smooth at that point or not.

It turns out that for the four-point algorithm, we need to consider only 7 control points; these 7 points completely define the piece of the curve around a control point. We can consider a set of 7 control points *on any subdivision level*, as we do not care how small our piece of the curve is. Note that we can compute the positions of the seven control points on level $j+1$ from the positions of similar seven control points on level j, using a 7×7 submatrix $\mathbf{S}$ of the infinite subdivision matrix.

The local subdivision matrix for the four-point scheme is:

$$\begin{pmatrix} c_{-3}^{j+1} \\ c_{-2}^{j+1} \\ c_{-1}^{j+1} \\ c_{0}^{j+1} \\ c_{1}^{j+1} \\ c_{2}^{j+1} \\ c_{3}^{j+1} \end{pmatrix} = \begin{pmatrix} -\frac{1}{16} & \frac{9}{16} & \frac{9}{16} & -\frac{1}{16} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{16} & \frac{9}{16} & \frac{9}{16} & -\frac{1}{16} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{16} & \frac{9}{16} & \frac{9}{16} & -\frac{1}{16} & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{16} & \frac{9}{16} & \frac{9}{16} & -\frac{1}{16} \end{pmatrix} \begin{pmatrix} c_{-3}^{j} \\ c_{-2}^{j} \\ c_{-1}^{j} \\ c_{0}^{j} \\ c_{1}^{j} \\ c_{2}^{j} \\ c_{3}^{j} \end{pmatrix}$$

As we iteratively apply the subdivision $\mathbf{S}$ to the initial vector for control points, we see that each of the 7 control points becomes closer to the central control point c_0.

Some properties of the curves generated by subdivision can be inferred from the properties of the subdivision matrix. In particular, smoothness properties of the curve are related to the eigenstructure of the subdivision matrix.

2 Properties of the Subdivision Matrix

2.1 Preliminaries

Recall from linear algebra that an *eigenvector* x of the matrix $\mathbf{M}$ is a non-zero vector such that $\mathbf{Mx} = \lambda \mathbf{x}$, where λ is a scalar; λ is the *eigenvalue* corresponding to $\mathbf{x}$.

Assume $\mathbf{S}$ has real eigenvectors $x_0, x_1, \ldots, x_6$ which form a basis, with corresponding real eigenvalues $\lambda_0 \geq \lambda_1 \geq \ldots \geq \lambda_6$. We can then write any vector x of length 7 as a linear combination of eigenvectors:

$$x = \sum_{i=0}^{n} a_i x_i$$

Similarly, we can write a decomposition of this type for a vector $\mathbf{c}$ of 7 *2-D points* rather than single numbers. In this case each "coefficient" $\mathbf{a}_i$ is a single 2-D point. The eigenvectors $\mathbf{x}_0, \ldots, \mathbf{x}_6$ are simply vectors of 7 real numbers. The expression $\mathbf{c} = \sum_{i=0}^{6} \mathbf{a}_i \mathbf{x}_i$, where $\mathbf{c}$ is a vector of 2D-points can be written more explicitly as

$$\mathbf{c} = \sum_{i=0}^{6} \begin{pmatrix} x_i^0 \mathbf{a}_i \\ x_i^1 \mathbf{a}_i \\ x_i^2 \mathbf{a}_i \\ x_i^3 \mathbf{a}_i \\ x_i^4 \mathbf{a}_i \\ x_i^5 \mathbf{a}_i \\ x_i^6 \mathbf{a}_i \end{pmatrix}$$

where $x_i^j \in \mathbf{R}$ represents the j-th component of eigenvector $\mathbf{x}_i$.

In the basis of eigenvectors we can easily compute the result of application of the subdivision matrix to a vector of control points, that is, the control points on the next level:

$$\begin{aligned}\mathbf{S}c_0 &= \mathbf{S}\sum_{i=0}^{6}\mathbf{a}_i x_i \\ &= \sum_{i=0}^{6}\mathbf{a}_i \mathbf{S} x_i \quad \text{by linearity of } \mathbf{S} \\ &= \sum_{i=0}\mathbf{a}_i \lambda_i x_i\end{aligned}$$

Applying $\mathbf{S}$ n times, we obtain

$$\mathbf{S}^n c_0 = \sum_{i=0}^{6}\mathbf{a}_i \lambda_i^n x_i$$

2.2 Convergence of Subdivision

If $\lambda_i > 1$ for some i, $\mathbf{S}^n c_0$ clearly would diverge as n got larger and larger.

Hence, we can see that in order for the sequence $\mathbf{S}^n c_0$ to converge at all, it is necessary that all eigenvalues are at most 1. It is also possible to show that only a single eigenvalue may have magnitude 1.

2.3 Invariance under Affine Transformations

If we moved all the control points simultaneously by the same amount, we would expect the curve defined by these control points to move in the same way as a rigid object. In other words, the curve should be *invariant under distance-preserving transformations, such as translation and rotation.* It follows from linearity of subdivision that if subdivision is invariant with respect to distance-preserving transformations, it also should be invariant under any affine transformations. The family of affine transformations in addition to distance-preserving transformations, contains shears.

Let $\vec{1} = [1111111]^T$ Then $\vec{1} \cdot \mathbf{a}$ represents a displacement of our seven points by a vector $\mathbf{a}$.

Applying subdivision to the transformed points, we get

$$\begin{aligned}\mathbf{S}(\mathbf{c}^j + \vec{1}\cdot\mathbf{a}) &= \mathbf{S}(\mathbf{c}^j) + \mathbf{S}(\vec{1}\cdot\mathbf{a}) \quad \text{by linearity of } \mathbf{S} \\ &= \mathbf{c}^{j+1} + \mathbf{S}(\vec{1}\cdot\mathbf{a})\end{aligned}$$

We see that for translational invariance we need

$$\mathbf{S}(\vec{1}\cdot\mathbf{a}) = \vec{1}\cdot\mathbf{a}$$

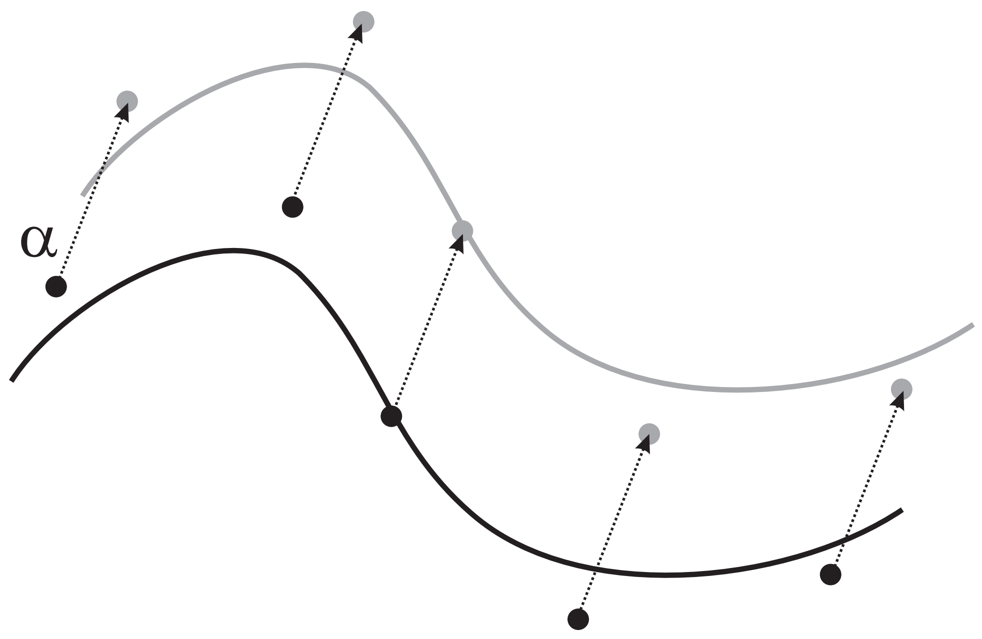


Figure 1: Invariance under translation

Therefore, $\vec{1}$ should be the eigenvector of $\mathbf{S}$ with eigenvalue $\lambda_0 = 1$.

2.4 Geometric Behavior of Repeated Subdivision

If we assume that λ_0 is 1, and all other eigenvalues are less than 1, we can choose our coordinate system in such a way that a_0 is a zero vector; then we have:

$$\mathbf{c}^n = \sum_{i=1}^{6} \mathbf{a}_i \lambda_i^n x_i$$

Dividing both sides by λ_1^n, we obtain:

$$\frac{\mathbf{c}^n}{\lambda_1^n} = \mathbf{a}_1 x_1 + \sum_{i=2}^{6} a_i \left(\frac{\lambda_i}{\lambda_1}\right)^n x_i$$

Assume that $\lambda_2, \ldots, \lambda_6 < \lambda_1$. In this case we see that the sum on the right approaches zero as $n \to \infty$. In other words the term corresponding to λ_1 will "dominate" the behavior of the vector of control points (decrease much slower than the other ones).

In the limit, we get a set of 7 points arranged along the vector $\mathbf{a}_1$. Geometrically, this is the tangent vector of our curve at our center point.

If there were two equal eigenvalues, say $\lambda_1 = \lambda_2$, as n increases, the points in the limit configuration will be linear combinations of two vectors $\mathbf{a}_1$ and $\mathbf{a}_2$, and in general would not be on the same line. This indicates that there will be no tangent vector at

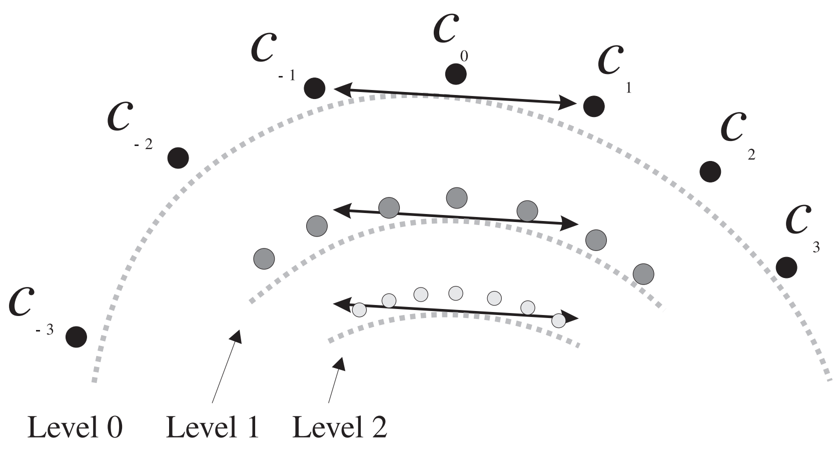


Figure 2: Repeatedly applying the subdivision matrix to our set of seven control points results in the control points converging to a configuration aligned with the tangent vector. The various subdivision levels have been offset vertically for clarity.

the central point. This leads us to the following condition, that, under some additional assumptions, is necessary for existence of the tangent:

All eigenvalues of S except $\lambda_0 = 1$ should be less than λ_1.

2.5 Summary

For our subdivision matrix $\mathbf{S}$ we desire the following characteristics:

- The eigenvectors should form a basis.
- The first eigenvalue λ_0 should be 1.
- The second eigenvalue λ_1 should be less than 1.
- All other eigenvalues should be less than λ_1.

3 Schemes for Subdivision Surfaces

We now focus our attention on the 2-D extension of curves, subdivision surfaces. There are a variety of subdivision schemes for surfaces, including

- Loop
- Catmull-Clark
- Doo-Sabin

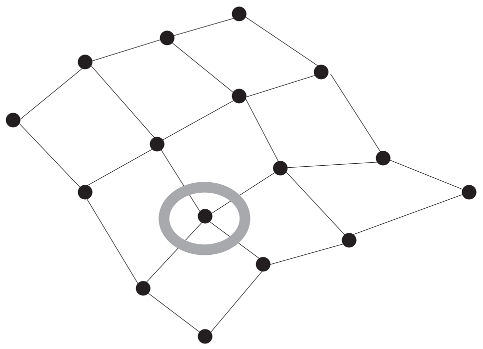


Figure 3: A tensor product B-spline.

- Butterfly

In this lecture, we focus on the Loop scheme.

3.1 Representation of Surfaces with Splines

In the 2-D case, we use a polyhedral surface as a *control mesh* analogous to the control points in the curve case. Additional restrictions may be imposed on the mesh. As an example, consider a *uniform tensor product B-spline* where all points not on the boundary of the control mesh are restricted to having exactly 4 neighbors.

While B-splines may seem like a good general solution at first as a control mesh for surfaces, what if we wanted to represent, say, a sphere? Unlike our B-spline, it is a *closed surface* with no boundary, and it is topologically impossible to represent it using such a mesh. For a mesh in which each vertex has 4 neighbors, the quantity $f - e + v$ (known as the *Euler characteristic*), where f is the number of faces, e is the number of edges, and v is the number of vertices, is 0. On the other hand, for a polyhedral mesh topologically equivalent to a sphere the Euler characteristic has to be 2.

3.2 3-Directional Quartic Splines

The Loop scheme is based on three-directional quartic Box splines, which are defined over regular triangular meshes (each face is a triangle, each vertex has 6 neighbors.) Other schemes are based on other types of splines: tensor product of quadratic B-splines for the Doo-Sabin scheme, and tensor product of cubic B-splines for Catmull-Clark.

The generating function for 3-directional Box spline is

$$f(z_1, z_2) = \frac{1}{16}(1 + z_1)^2(1 + z_2)^2(1 + z_1 z_2)^2$$

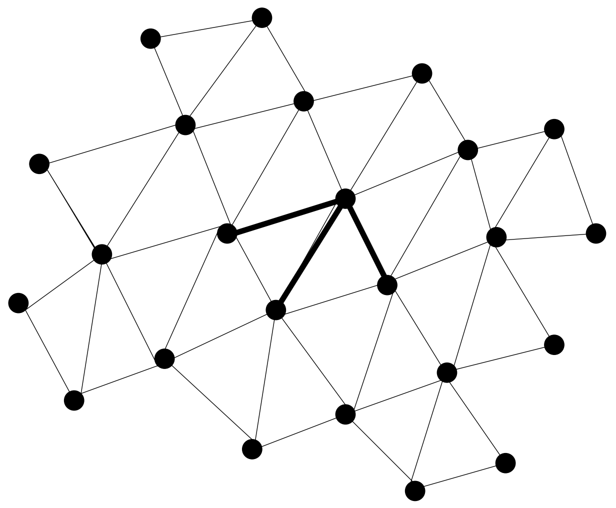


Figure 4: Part of a 3-directional quartic spline, with the three directions of symmetry highlighted.

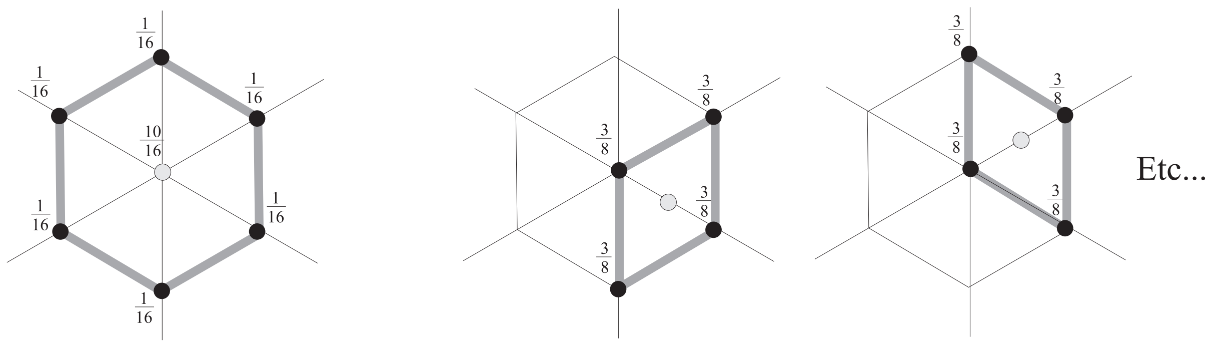


Figure 5: Loop subdivision rules for vertex (left) and edge (right) points at a regular vertex of degree 6.

3.3 Rules for Subdivision

Similar to the one-dimensional case, for the Loop scheme we need 2 rules, a "even" one (for adding new control points) and a "odd" one (for modifying existing ones.) Consider the case where each vertex has six neighbors (has degree 6).

New vertices are created by splitting the edges of the old mesh. Each old edge gives rise to two new edges and three more edges per triangle are created by connecting the new vertices. Each old triangle is subdivided in four.

Our challenge is to make the scheme work for arbitrary triangular meshes. We cannot simply use the spline rules, because they assume that each vertex has 6 neighbors. How do we update the position of vertices with degree other than 6 (known as *extraordinary*

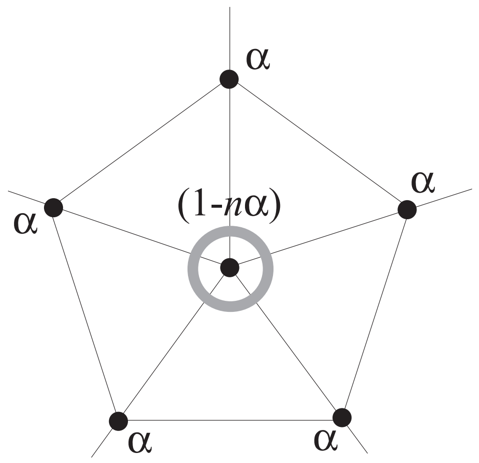


Figure 6: At an extraordinary point, we need to alter our vertex subdivision rule.

vertices)?

Observe that the rule for the spline is

$$c_0^{j+1} = \frac{5}{8}\mathbf{c}_0^j + \frac{1}{16}\sum_{i=1}^{6}\mathbf{c}_i^j$$

where $\mathbf{c}_0$ is the vertex that we recompute, and c_i, $i = 1 \ldots 6$ are the immediate neighbors. This rule can be naturally generalized to the case of n neighbors as

$$c_0^{j+1} = (1 - \alpha n)\mathbf{c}_0^j + \alpha\sum_{i=1}^{6}\mathbf{c}_i^j$$

We need to set the coefficient of our center point to be $(1 - n\alpha)$ in order for the sum of the coefficients to add up to 1 which is required for affine invariance of the scheme. Note that changing a rule at an extraordinary point will only change the characteristics of the generated surface locally, so the analysis in this case is similar to the analysis that we did for subdivision curves with special rules near a point.

The choice of the parameter α in the formula above is not unique, and different considerations can be used to pick an appropriate value. The original value chosen by Loop was

$$\alpha = \frac{1}{n}\left(\frac{5}{8} - \left(\frac{3}{8} + \frac{1}{4}\cos\frac{2\pi}{n}\right)^2\right)$$

We can easily verify that in the case $n = 6$, α is $\frac{1}{16}$ as we would expect. However, the exact value of α does not matter too much: the generated surfaces are smooth for a range of α. For example, for $n > 3$, we can choose α to be simply $\frac{3}{8n}$; the resulting surfaces will be quite similar to those obtained using the original Loop's rules.

Classical Subdivision Surfaces II

Lecture #15:	Tuesday, 18 November 1997
Lecturer:	Denis Zorin
Scribe:	Joey Beheler
Reviewer:	Mark Wang

In the previous lecture, we discussed the Loop subdivision scheme which works for triangular meshes. In this lecture we will look at several other subdivision schemes.

1 Background

Each subdivision scheme can be thought of as two rules: the topological refinement rule that describes how the refined mesh is created from the original mesh, and the subdivision rule, that is used to compute the locations of the vertices of the new mesh. The first rule, which we will call simply the *refinement rule*, describes how many new vertices are added to the mesh and which vertices in the new mesh are connected by edges. For example, the Loop scheme replaces a triangle at subdivision level j with 4 triangles at level $j+1$. The new mesh at subdivision level $j+1$ is created by splitting each edge and connecting the inserted vertices as shown in Figure 1.

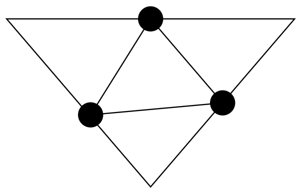


Figure 1: Connecting the 3 new vertices creates 4 new level $j+1$ triangles.

Two of the schemes that we consider in this lecture are defined for quadrilateral rather then triangular meshes. For such meshes we have two main options for refinement rules.

The first refinement rule, used by the *Doo-Sabin* subdivision scheme, first builds a level $j+1$ quadrilateral in the center of each existing level j face quadrilateral. The new mesh is obtained by connecting each vertex of the level $j+1$ quadrilateral with its three neighbors to produce 3 more level $j+1$ quadrilaterals. This refinement rule is illustrated in Figure 2.

The second refinement rule, which is used by the *Catmull-Clark* subdivision scheme subdivides a level j quadrilateral into 4 level $j+1$ quadrilaterals. New vertices are created by splitting each edge into two and creating a new vertex for each face. This method is illustrated in Figure 3.

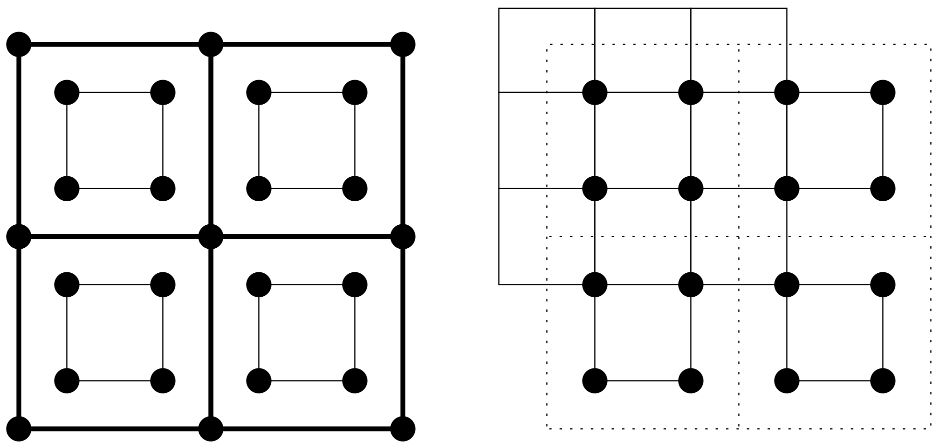


Figure 2: Refinement rule used by Doo-Sabin subdivision scheme. Step 1: New vertices are added to create level *j+1* quadrilaterals in the center of level *j* quadrilaterals. Step 2: The vertices of the center face quadrilaterals are connected to their neighbors. Each level *j* quadrilateral is covered by parts of 9 level *j+1* quadrilaterals, but there are only 4 level *j+1* quadrilaterals created for each level *j* quadrilateral. The vertices of the level *j* quadrilaterals are discarded. This refinement rule works for even degree tensor-product splines.

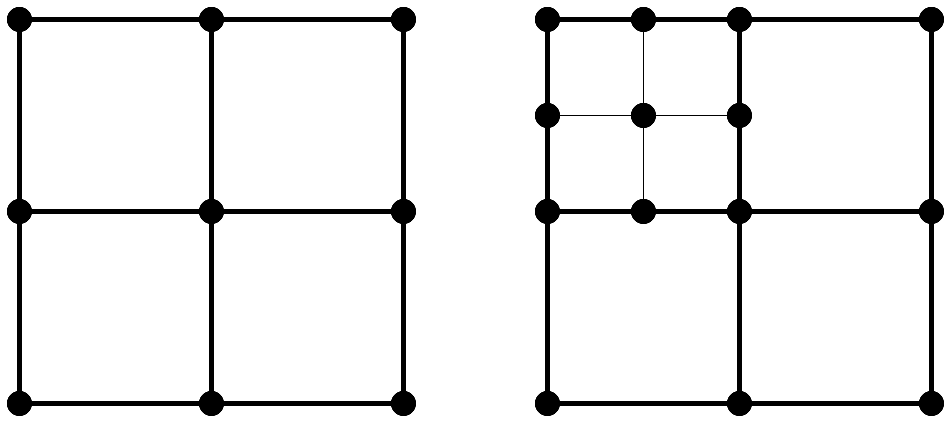


Figure 3: Refinement rule used by Catmull-Clark subdivision scheme. New vertices are added on each edge and in the center. When connected, 4 new level *j+1* quadrilaterals are produced from the single level *j* quadrilateral. This refinement rule works for odd degree tensor product splines.

2 Doo-Sabin

2.1 Quadrilaterals

The Doo-Sabin subdivision scheme is based on quadratic splines. Recall that a generating function for one-dimensional quadratic splines is $(z+1)^3/4$. For a tensor product spline of total degree 4 the generating function is

$$f(z_1, z_2) = \frac{(1+z_1)^3(1+z_2)^3}{2^2 \cdot 2^2}$$

If we multiply this out we end up with:

$$f(z_1, z_2) = \frac{(1+3z_1+3z_1^2+z_1^3)(1+3z_2+3z_2^2+z_2^3)}{16}$$

The coefficients of the monomials of $f(z_1, z_2)$ are the coefficients of the subdivision scheme.

We can arrange the coefficients into a table, with the coefficient of $z_1^i z_2^j$ in position (i, j):

$$\frac{1}{16} \cdot \begin{pmatrix} 1 & 3 & 3 & 1 \\ 3 & 9 & 9 & 3 \\ 3 & 9 & 9 & 3 \\ 1 & 3 & 3 & 1 \end{pmatrix}$$

For quartic (that is, of total degree 4) tensor-product splines we have 4 rules for determining the positions on the vertices on the next subdivision level. They are the Even-Even, Even-Odd, Odd-Even and Odd-Odd rules. Each rule uses 4 coefficients from the above table. The name of each rule determines which 2 rows (Even or Odd) and which two columns (Even or Odd) of the table are selected. The Even-Even rule, for example selects from the even column and even row giving the coefficients $\frac{1}{16}$, $\frac{3}{16}$,$\frac{3}{16}$, and $\frac{9}{16}$. Geometrically, this results in a point EE that is $\frac{1}{16}C_0 + \frac{3}{16}C_1 + \frac{3}{16}C_2 + \frac{9}{16}C_3$ as shown in Figure 4.

2.2 Doo-Sabin subdivision: N-gons

Doo-Sabin subdivision extends the rules for quartic splines described above to general polygonal meshes. For each N-gon at level j, we create a level *j+1* N-gon. This is similar to dealing with extraordinary points in the Loop scheme, but now we cannot use symmetry to compute all coefficients once one of them is known. The rules for the vertices of the new N-gon can be derived using the eigenstructure of the subdivision matrix. Suppose we are computing a new vertex of the N-gon on level $j+1$. This vertex is a linear combination of the vertices of the old N-gon. Suppose these vertices are numbed from 0 to $N-1$ starting with the vertex nearest to the vertex on level $j+1$ that we are computing. Then the coefficients are $\alpha_0 = \frac{1}{4} + \frac{5}{4N}$, $\alpha_k = \frac{3+2cos(\frac{2\pi k}{N})}{4N}$, $k = 1 \dots N-1$ where k is the vertexes number.

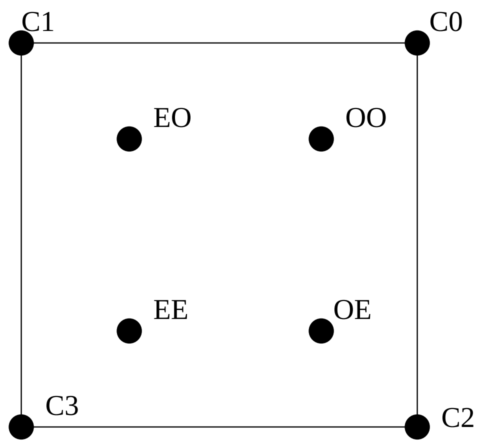


Figure 4: The level *j+1* point determined by the Even-Even rule is at $\frac{1}{16}C_0 + \frac{3}{16}C_1 + \frac{3}{16}C_2 + \frac{9}{16}C_3$. The other points are determined in a similar manner. All rules are the same up to a renumbering of the points.

3 Catmull-Clark

3.1 Quadrilateral Rules

The Catmull-Clark subdivision scheme is based on cubic splines.

Its generating function is:

$$f(z_1, z_2) = \frac{(1+z_1)^4 \cdot (1+z_2)^4}{64}$$

In this case, there are three different rules called the *vertex rule*, the *edge rule* and the *face rule* for computing point positions. These rules also correspond to various choices of columns and rows of the table of coefficients of the generating function: Even-Even, Odd-Even etc. The vertex rule computes a new position of an existing vertex using the vertex itself and 8 immediate neighbors. The edge rule computes the position for a new vertex inserted on an edge between two existing vertices. It uses the closest 6 existing vertices. The simplest rule, the face rule, creates a new point in the middle of a quadrilateral face. All rules are shown in Figure 5.

3.2 Extraordinary Vertices

For extraordinary vertices, Catmull-Clark scheme uses modified rule similar to Loop's scheme. Since there are quadrilaterals meeting at the extraordinary point instead of triangles, we have two sets of vertices that should have equal coefficients by symmetry. We call them *inner* and *outer* vertices (Figure 6). This leaves us two degrees of freedom.

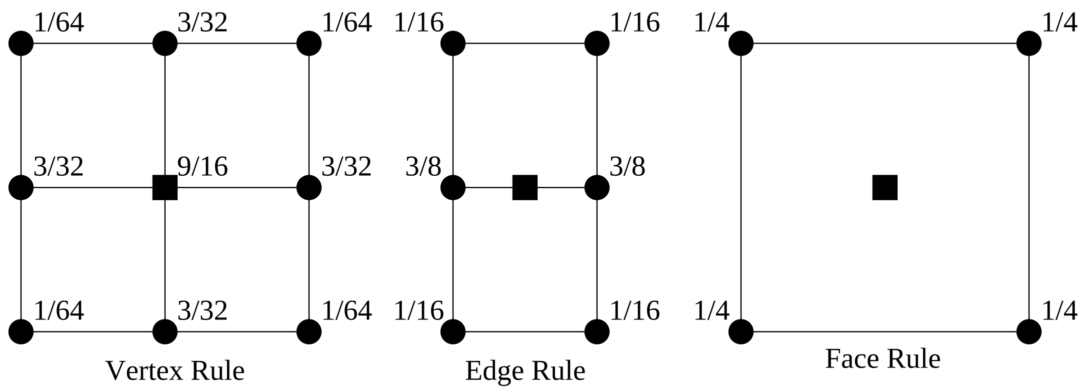


Figure 5: Each black circle represents a vertex at level j; we compute the position of the vertex at level $j+1$ marked by by the black square. Note that for the vertex rule, the control vertex with weight $\frac{9}{16}$ and the new vertex aren't necessarily aligned as they are in the figure.

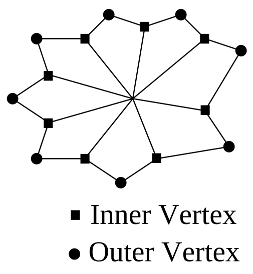


Figure 6: Inner and outer vertices surrounding an extraordinary point in a quadrilateral mesh.

If the coefficients of inner vertices are $\frac{\beta}{N}$ and the coefficients of all outer vertices are be $\frac{\gamma}{N}$, where N is number of inner vertices. The coefficient of the central vertex is $1-\beta-\gamma$ by affine invariance. The values for β and γ which Catmull and Clark came up with are

$$\begin{aligned} \beta &= \frac{3}{2N} \\ \gamma &= \frac{1}{4N} \end{aligned}$$

Note that for $N=4$, we do get the coefficients for the spline vertex rule. Also, note that the weight on an inner or outer point is proportional to $\frac{1}{N^2}$ - we do divide by N twice. These coefficients are by no means unique: there is a range of β and γ that leads to smooth surfaces.

3.3 Arbitrary Meshes

We have defined Catmull-Clark scheme on quadrilaterals; it can be extended to handle arbitrary polygonal meshes. Observe that if we do one step of refinement, splitting each edge into two and inserting a new vertex for each face (Figure 7), we get a mesh which has only quadrilateral faces. Special rules described in Catmull and Clark's paper are used for this first step. On all other steps of subdivision standard rules described above can be applied.

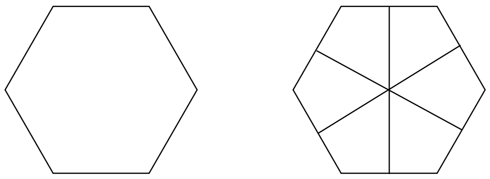


Figure 7: Splitting a hexagon into quadrilaterals.

4 Butterfly Rule : Interpolation on a Triangular Grid

The Butterfly subdivision scheme is similar to the Loop scheme, but it creates an interpolated surface which contains the vertices of the original triangular mesh. The Loop scheme had two rules, one for changing positions of old vertices and one for computing positions of new vertices. Since the Butterfly scheme is interpolating, we don't need a rule for changing the positions of old vertices. If we want the scheme to generate smooth surfaces, we do need wider support for the rule that adds a vertex. To add a new vertex we use the coefficients shown in Figure 8.

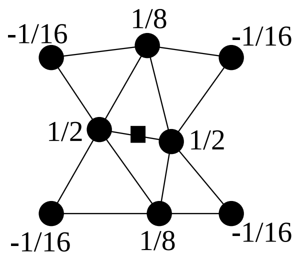


Figure 8: Each black circle is a control point with the corresponding weight. The black square is the new vertex.

This scheme is based on the one-dimensional four-point interpolating scheme with coefficients $\frac{-1}{16}, \frac{9}{16}, \frac{9}{16}, \frac{-1}{16}$. This subdivision scheme generates smooth interpolating surfaces only for regular meshes. Smoothness, however, is not guaranteed at extraordinary points. A modification of this scheme with special coefficients for computing new vertices adjacent to the extraordinary vertices generates surfaces that are smooth everywhere for almost all configurations of control vertices. The coefficients of the scheme are given by

$$s_j = \frac{1}{n}\left(\frac{1}{4} + \cos\frac{2\pi}{n} + \frac{1}{2}\cos\frac{2\pi}{n}\right),\ j = 0 \ldots n-1$$

if the extraordinary vertex has degree $n \geq 5$. The coefficients are $5/12, -1/12, -1/12$ for $n = 3$ and $3/8, 0, -1/8, 0$ for $n = 4$.

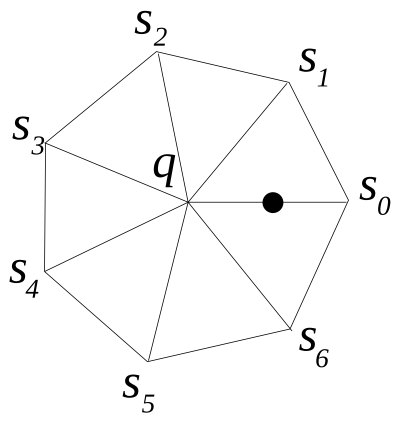


Figure 9: Rule for inserting a new vertex next to an extraordinary vertex for the Modified Butterfly scheme.

5 Creases

The Loop scheme can be modified in such a way that a path of control vertices on the coarsest level can be "marked" meaning the regular Loop scheme rule is replaced with an alternative scheme. To create a crease, the normal weights of $\frac{1}{8}, \frac{3}{8}, \frac{3}{8}, \frac{1}{8}$ are replaced with weights $0, \frac{1}{2}, \frac{1}{2}, 0$ for adding new points on the crease. For changing the position of old points on the crease we weight the current position with $\frac{3}{4}$ and its two neighbors on the crease edge with $\frac{1}{8}$. This rule replaces the normal Loop scheme rule for changing the position of old vertices. The effect is that any vertex on the path won't be influenced by the rest of the surface, it will only be influenced by points already on the path. The downside of this approach is that the surface may behave badly as it approaches this independently defined curve.

CS448: Topics in Computer Graphics Lecture #16
Mathematical Models for Computer Graphics
Stanford University Tuesday, 9 December 1997

More on Subdivision Surfaces and Smoothness

Lecture #16: Tuesday, 20 November 1997
Lecturer: Denis Zorin
Scribe: Szymon Rusinkiewicz
Reviewer: Joey Beheler

Last time, we talked about various subdivision schemes for two-dimensional surfaces. We will now clear up the mystery of where the coefficients in those methods came from, by examining how those coefficients influence the smoothness of subdivision surfaces.

1 What Is a Smooth Surface?

We shall start with an exact definition of a smooth surface. Informally, we would like to express the concept that a smooth surface is "a smooth two-dimensional subset of space." In other words, we need to define some subset of (three-dimensional) space that has a non-degenerate two-variable parameterization that is continuous and differentiable. To make this concept rigorous requires a somewhat complicated definition:

Definition 1. *A* ***simple smooth surface*** *in three-dimensional space is defined as a subset A of space, such that the intersection of A and a neighborhood of a point X in A has a regular parameterization p from the unit disc. Thus, if $A \subset \mathbf{R}^3$, B_X is the neighborhood of some $X \in A$, and $D \subset \mathbf{R}^2$ is the unit disc, A is a simple smooth surface if and only if $D \xrightarrow{p} A \bigcap B_X$. A* ***regular parameterization*** *p is one that is continuously differentiable, one-to-one, and has a Jacobi matrix of maximum rank.*

This definition corresponds directly to the intuition above. We pick a point X on our candidate surface A and examine its immediate neighborhood. This will be some region of space that intersects A. We claim that we have a simple smooth surface only if that region of intersection (which will be some patch of A) has a continuously differentiable one-to-one mapping onto a disc.

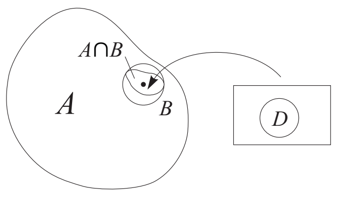


Figure 1: Definition of a simple smooth surface.

The condition that the Jacobi matrix of p have maximum rank is necessary to make sure we have no degeneracies (i.e. that we really do have a surface, not a curve or point). If $p = (p_1, p_2, p_3)$ and the disc is parameterized by x_1 and x_2, the condition is that the matrix

$$\begin{pmatrix} \frac{\partial p_1}{\partial x_1} & \frac{\partial p_1}{\partial x_2} \\ \frac{\partial p_2}{\partial x_1} & \frac{\partial p_2}{\partial x_2} \\ \frac{\partial p_3}{\partial x_1} & \frac{\partial p_3}{\partial x_2} \end{pmatrix}$$

have maximal rank, i.e. two.

Although the above definition is certainly adequate for some applications, it is convenient to have a slightly different definition. The problem arises when we consider surfaces that intersect themselves – no self-intersecting surface can be a simple smooth surface. This becomes obvious when we consider a point on the line of intersection, since its neighborhood in three-dimensional space consists of pieces of *two* smooth surfaces.

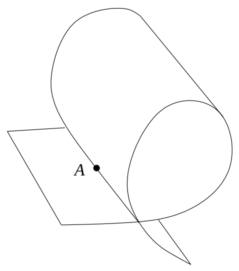


Figure 2: Self-intersecting surface - it is not simply smooth at A.

We would prefer to have a definition of smoothness that is more local, and independent of global properties such as self-intersection. If we perform some topological deformation on a smooth surface, we would like the surface to continue to be classified as smooth, even though the deformation might cause it to intersect itself. Therefore, we introduce a modified definition:

Definition 2. *A surface $A \subset \mathbf{R}^3$ is **smooth** (i.e. C^1) if and only if*

- *There exists a mapping f from a parametric domain $M \subset \mathbf{R}^2$ onto A.*
- *For every point $X \in M$ there exists a regular parameterization of the unit disc onto $f(U)$, where U is the neighborhood in M of X.*

We add an extra mapping f into the definition to provide a way of defining neighborhoods on the surface itself.

This definition is convenient when working with subdivision surfaces, since it can be shown that the mapping f always exists as we subdivide. In particular, consider a triangular mesh in the vicinity of an (ordinary or extraordinary) vertex of degree k. In the original mesh, we just let f be the mapping from a regular k-gon to a portion of

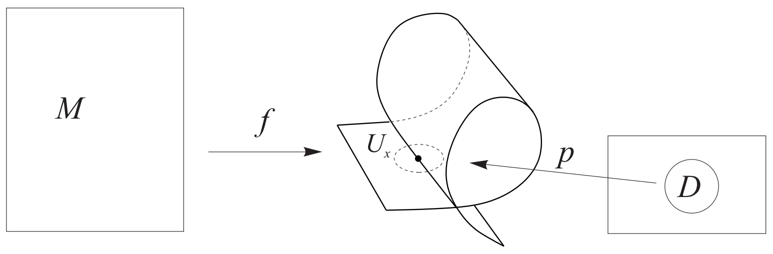


Figure 3: Definition of a C^1 (i.e. smooth) surface.

the mesh. As we subdivide the mesh, we subdivide the k-gon using the midpoint rule. We see that each added vertex in M corresponds exactly to a vertex added to the mesh. Note that although the mapping f defined in this way need not be smooth, this is not a requirement for the definition of a smooth surface – the only requirements are existence and smoothness p.

There is yet another definition of smoothness encountered from time to time. This is the notion of geometric smoothness:

Definition 3. *A surface A is* ***geometrically smooth*** *at $X \in A$ if and only if surface normals are defined in a neighborhood around X and there exists a limit of normals at X.*

This is a useful definition, since it is easier to prove surfaces geometrically smooth (all that is required is to show the existence of a limit), and also since the definition is very intuitive (it captures the notion that a surface is smooth if there exists a tangent plane). Geometric smoothness, however, is weaker than C^1 smoothness. Geometric smoothness at X implies the existence of a parameterization from the unit disc that is regular everywhere except at X. This contrasts with the regular definition of smoothness, which requires the parameterization to be regular everywhere.

As a simple example of a surface that is geometrically smooth but not C^1, consider the shape in Figure 4. Points in the vicinity of the central point are "wrapped around twice", so while there exists a tangent plane at that point, the surface does not "locally look like a plane". Formally, there does not exist a regular parameterization from the unit disc onto the neighborhood of the center point, even though that parameterization is regular in the neighborhood of that point.

From the previous example, we see how the definition of geometric smoothness must be strengthened to become C^1:

Lemma 4. *If a surface is geometrically smooth at a point and the projection of the surface onto the tangent plane at that point is one-to-one, the surface is C^1.*

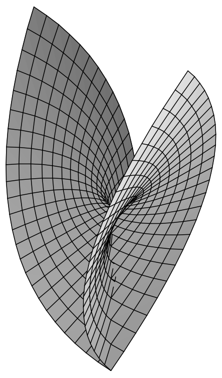


Figure 4: Example of a surface that is geometrically smooth but not C^1.

2 Smoothness of Subdivision Surfaces

We now turn to an examination of the smoothness properties of a subdivision surface. We will first examine the Loop scheme, and show how the eigenvalues of the subdivision matrix determine how smooth the limiting surface will be.

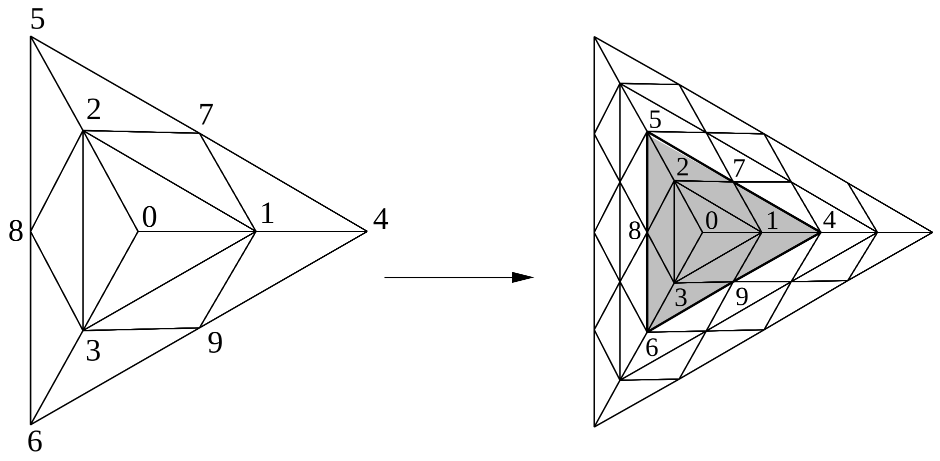


Figure 5: The Loop subdivision scheme near a vertex of degree 3. Note that $3(3)+1 = 10$ points in two rings are required.

Recall that in the Loop scheme only two "rings" of triangles around a vertex that define the surface patch surounding the vertex. So, if we let c_i^k be the coordinate of the point c_i at subdivision iteration k, we can represent the effect of subdivision on a point by

$$\begin{pmatrix} c_0^{k+1} \\ \vdots \\ c_{3n}^{k+1} \end{pmatrix} = \mathrm{S} \begin{pmatrix} c_0^{k} \\ \vdots \\ c_{3n}^{k} \end{pmatrix}$$

In this equation, n is the degree of the vertex, so two rings of triangles around it require $3n + 1$ points. S is the local portion of the subdivision matrix. Note that the coordinates c_i^k are themselves vectors in three-space.

We can now rewrite each of the coordinate vectors in terms of the eigenvectors of the matrix S. Thus,

$$c^0 = \begin{pmatrix} c_0^{0} \\ \vdots \\ c_{3n}^{0} \end{pmatrix} = \sum_i a_i x_i,$$

and

$$c^k = (\mathrm{S})^k c^0 = \sum_i (\lambda_i)^k a_i x_i,$$

where the x_i are the eigenvectors of S, and the λ_i are the corresponding eigenvalues, arranged in nonincreasing order. As stated in an earlier lecture, we require λ_0 to be 1 for all subdivision schemes, in order to guarantee invariance with respect to translations and rotations. Furthermore, all stable, converging subdivision schemes will have all the remaining λ_i less than 1.

It is clear that as we subdivide, the behavior of c^k, which determines the behavior of the surface in the immediate vicinity of our point of interest, will depend only on the largest eigenvalues of S. In the limit as $k \to \infty$, we care only about λ_0 and λ_1 for curves; for surfaces, we also care about λ_2.

To proceed with the derivation, we will assume for simplicity that $\lambda_1 \geq \lambda_2 > \lambda_3$. Furthermore, we let $a_0 = 0$ – recall that this corresponds to choosing the origin of our coordinate system in the limit position of the vertex of interest. Then we can write

$$\frac{c^k}{(\lambda_1)^k} = a_1 x_1 + \left(\frac{\lambda_2}{\lambda_1}\right)^k a_2 x_2 + \cdots,$$

where the higher-order terms disappear in the limit.

We now have two cases to consider – we will first consider what happens if $\lambda_1 = \lambda_2$. In this case

$$\lim_{k \to \infty} \frac{c^k}{(\lambda_1)^k} = a_1 x_1 + a_2 x_2.$$

This means that, up to a scaling by $(\lambda_1)^k$, the control points approach a fixed configuration. This configuration is determined by x_1 and x_2, which depend only on the subdivision scheme chosen, and on a_1 and a_2, which are determined by the initial conditions. Note, however, that given any non-degenerate initial conditions, the result will be the same up to an affine transform. Therefore, in some sense, all surfaces subdivided using a given rule "look the same" in the limit, and the limiting configuration depends on the eigenvectors x_1 and x_2 of the subdivision matrix.

Now recall that a_1 and a_2 are just vectors in $\mathbf{R}^3$, while x_1 and x_2 are $3n+1$-component vectors, one component per control point. Therefore, we see that the expression $a_1x_1 + a_2x_2$ must represent a set of control points that all lie in one plane. So in the limit a subdivision rule with $\lambda_0 > \lambda_1 = \lambda_2 > \lambda_3$ leads to smooth surfaces.

Now consider what happens when $\lambda_1 > \lambda_2$. In this case, the control points still approach a plane, but they "stretch" in one direction more than another. Therefore we still get smooth surfaces. Note that we must be careful to consider both λ_1 and λ_2, and not come to the erroneous conclusion that the control points might approach a line and therefore form a crease.

In the next lecture we will formulate a sufficient condition for smoothness.

Analysis of Subdivision Surfaces, Multiresolution Surfaces

Lecture #17: Tuesday, 25 November 1997
Lecturer: Denis Zorin
Scribe: Michael Malkin
Reviewer: Szymon Rusinkiewicz

In which Denis discusses the smoothness of subdivision surfaces, introduces multiresolution surfaces and brings subdivision to its conclusion.

1 The Smoothness of Subdivision Surfaces

1.1 Introduction

The condition for smoothness of subdivision surfaces can be stated very simply. We could just start by explaining everything at the beginning.

Reif's sufficient condition for smoothness : Suppose the eigenvectors of a subdivision matrix form a basis, the largest eigenvalues three eigenvalues are real and satisfy

$$\lambda_0 = 1 > \lambda_1 \geq \lambda_2 > |\lambda_3|$$

. If the characteristic map is regular and one-to-one, then almost all surfaces generated by subdivision are smooth (C^1-continuous.)

We need to define several concepts mentioned in this criterion.

1.2 Subdivision surfaces as mappings

We will discuss subdivision on triangular meshes; constructions for schemes defined on quadrilateral meshes are similar. Suppose the initial control mesh for the surface has one extraordinary vertex of degree n and the rest of the vertices are regular (degree 6.) If we are interested only in local properties of subdivision surfaces, in fact this is the only kind of mesh we have to consider — any initial mesh after sufficient number of subdivision steps locally is identical to the mesh with one or none extraordinary vertices.

Consider a triangulation of the plane $\mathbf{R}^2$ which has similar structure: one vertex of degree n in the center, and all other vertices of degree 6. We can establish a one-to-one correspondence between the vertices of the triangulation of $\mathbf{R}^2$ and the vertices of the mesh $\mathbf{R}^3$; this defines a mapping from the set of vertices of the triangulation, which is a subset of the points of the plane, to the control vertices of the surface.

Suppose each time we apply the subdivision rules to compute the finer control mesh, we also apply midpoint subdivision to the triangulation of the plane. This means that we leave the old vertices where they are, and insert new vertices in the middle of each edge of the triangulation. Note that each control vertex that we insert in the mesh in $\mathbf{R}^3$ corresponds to a new vertex of the midpoint-subdivided triangulation.

As we repeat this process, we get a mapping from a denser and denser subset of the plane to the vertices of the finer and finer control mesh. In the limit we get a map from an everywhere dense set on the plane to the surface, which can be extended by continuity to the whole plane. As a result, we get a natural parameterization of the subdivision surface over the plane: $f : \mathbf{R}^2 \to \mathbf{R}^3$.

To analyze smoothness, we need to consider arbitrarily small piece of the surface around the extraordinary point. In particular, we can take the part of the surface defined over a n-gon U that consists out of triangles adjacent to the extraordinary vertex: $f : U \to \mathbf{R}^3$. Note that the n-gon can be chosen on any level of subdivision; let U^j be the n-gon after j subdivision steps.

We can construct the map f for $n = 6$; in this case, initially we have a regular triangulation on the plane. It is easy to show that for splines–based subdivision the mapping f in this case will be just the standard piecewise-polynomial mapping defining the spline.

In general, subdivision schemes are typically constructed in such way that the mapping f that we get for a regular initial mesh (with all vertices of degree 6) is C^1-continuous.

An important observation about the mapping f is that after sufficient number of subdivision steps in a neighborhood of any point inside the triangles of the initial triangulation of the plane, the map coincides with the map generated from a suitably chosen regular mesh (this is not true on the boundaries of triangles!)

In particular, f is C^1-continuous inside each triangle of the n-gon U^0 and we can compute the derivatives. To ensure smoothness of the surface, f should have Jacobi matrix of maximal rank.

It turns out that we need not consider all possible maps f to establish smoothness of subdivision. In most cases it is sufficient to consider a single map called the *characteristic map* which is completely defined by the subdivision scheme. To define this map, we need to discuss eigenvalues and eigenvectors of the subdivision matrix.

1.3 Characteristic Map

The part of the subdivision surface on U is completely defined by a set of the control vertices near the extraordinary vertex v. For example, for Loop's scheme the surface is defined by two rings of control vertices around v. For a vertex of degree n these rings contain $3n + 1$ vertices, including the extraordinary vertex itself. Let $c_i = (x_i, y_i)$ be the control vertices for the two rings, forming a vector $\mathbf{c}$ of length $3n + 1$:

$$c_i = (x_i, y_i) \tag{1}$$

$$\mathbf{c} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{3n} \end{bmatrix} \tag{2}$$

$\mathbf{c}^0$ is the initial vector of control vertices and $\mathbf{c}^j$ is the vector after j subdivisions. If we define $\mathbf{S}$ to be the subdivision matrix, then the act of subdividing can be represented as repeated multiplication of $\mathbf{c}$ by $\mathbf{S}$, exactly as in the case of the curves.

$$\mathbf{c}^{j+1} = \mathbf{S}\mathbf{c}^j \tag{3}$$

$$\mathbf{c}^{j+1} = \mathbf{S}^{j+1}\mathbf{c}^0 \tag{4}$$

After each subdivision the $3n+1$ control vertices defining the surface on the n-gon U^j are becoming closer and closer to the central vertex. One more remarkable thing happens: if we scale the control vertices $\mathbf{c}^j$, we will see that the configuration of the control vertices converges to a limit. To describe this more precisely, we have to look at the eigenvalues and eigenvectors of the subdivision matrix.

Suppose λ_i are the eigenvalues of $\mathbf{S}$, and they are numbered in nonincreasing order: $|\lambda_0| \geq |\lambda_1| \geq |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_{3n}|$. For simplicity we assume that there is a basis of eigenvectors of $\mathbf{S}$. λ_0, as usual, must be 1. For convergence we need $\lambda_1 < 1$. In addition, we assume that λ_1, λ_2 are real and $|\lambda_3| < \lambda_2$.

Then similar to the one-dimensional case that we have considered before,

$$\mathbf{c}^{j+1} \cong \lambda_1^n a_1 x_1 + \lambda_2^n a_2 x_2. \tag{5}$$

where λ_1 and λ_2 are the eigenvalues of $\mathbf{S}$, x_1 and x_2 are corresponding eigenvectors, and a_1 and a_2 are vectors in $\mathbf{R}^3$ defining the tangent plane.

We have used the usual trick of moving the origin of our coordinate system to the central point of the subdivision, so that a_0 is 0, and there are no terms containing λ_0. In this expression only a_1 and a_2 depend on the initial control vertices; the eigenvectors x_1 and x_2 depend only on the choice of the subdivision scheme.

Now we are ready to define the *characteristic map.* Note that when we described subdivision as a function from the plane to $\mathbf{R}^3$, we could have use control vertices not from $\mathbf{R}^3$ but from $\mathbf{R}^2$; clearly, subdivision rules can be applied in the plane rather then in space. Then in the limit we obtain a map from the plane into the plane. The characteristic map is the map of this type.

As we have seen, the configuration of control points near an extraordinary vertex approaches $a_1x_1 + a_2x_2$, up to a scaling transformation. This means that the part of the surface defined on the n-gon U^j as $j \to \infty$ approaches the surface defined by the vector of control points $a_1 x_1 + a_2 x_2$. Up to an affine transform which does not affect smoothness, in the limit all surfaces generated by subdivision are identical to the surface generated by the control vertices c_i, such that $c_i = ([x_1]_i, [x_2]_i, 0)$, that is, the first two coordinates are corresponding coordinates of the eigenvectors x_1 and x_2. The control vertices c_i are obtained from $a_1[x_1]_i + a_2[x_2]_i$ applying the affine transform taking a_1 to e_1 and a_2 to e_2, where e_1, e_2, e_3 are the basis vectors. By throwing away the last zero coordinate, we obtain a vector of control points in $\mathbf{R}^2$ completely defined by the eigenvectors of $\mathbf{S}$. This is a map from the n-gon U^0 to $\mathbf{R}^2$.

As we have discussed, inside each triangle of the n-gon U^0 the map is C^1. Moreover, the map has one-sided derivatives on the boundaries of the triangles, except at the extraordinary vertex, so we can define one-sided Jacobians on the boundaries of triangles too. Moreover, it is possible to show that the Jacobian actually has to be continuous away from the extraordinary vertex despite the fact that the derivatives may be not continuous. We will say that the characteristic map is *regular* if its Jacobian is not zero anywhere away from the extraordinary vertex (including the boundaries!)

Now we have all of the tools that we need to understand the condition given at the beginning:

Reif's sufficient condition for smoothness : Suppose the eigenvectors of a subdivision matrix form a basis, the largest eigenvalues three eigenvalues are real and satisfy

$$\lambda_0 = 1 > \lambda_1 \geq \lambda_2 > |\lambda_3|$$

. If the characteristic map is regular and one-to-one, then almost all surfaces generated by subdivision are smooth (C^1-continuous.)

2 Multiresolution Surfaces based on Subdivision

2.1 Problems With Subdivision

Subdivision is great, but there are some problems with it. One problem is that it's often easier to generate or manipulate a model with different levels of detail. For example, when manipulating a model of an armadillo, it's much easier to move an ear as a single object than to move each point that makes up the ear.

One solution for this problem (the one we will be considering) is multiresolution surfaces based on subdivision, We discuss how to use multiresolution surfaces, but will discuss how to generate these surfaces from arbitrary meshes. That is a separate and complicated topic.

2.2 Wavelet-Like Multiresolution Surfaces

2.2.1 Wavelets *A Blast From the Past*

As you recall, wavelets separated low-frequency and high-frequency components of an image. Another way of looking at this is that wavelets generate a signal of averages, and a signal of details.

We will now apply the same strategy to subdivision surfaces.

2.2.2 Comparison of Wavelets and Subdivision

The following figure is a diagram of a wavelet filter. The boxed section is analogous to subdivision, adding more points.

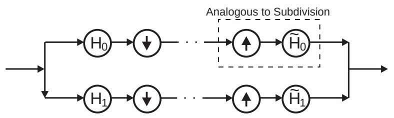


Figure 1: A wavelet filter bank.

If H_0 is known, it is typically possible to choose H_1, in such a way that they form a filter pair.

2.2.3 Another Strategy

If we are doing a hierarchical wavelet transform and we do not subsample the detail part of a signal, the resulting signal is two times larger than if we has subsampled. However, if we skip the subsampling in 2D, the resulting signal is only $\frac{4}{3}$ times larger.

$$1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots = 2 \tag{6}$$

$$1 + \frac{1}{4} + \frac{1}{16} + \frac{1}{64} + \cdots = \frac{4}{3} \tag{7}$$

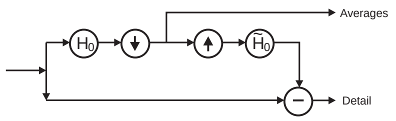


Figure 2: A filter bank that doesn't subsample the detail.

2.3 Performance of Multiresolution Surfaces

With full detail, the computer takes a long time to manipulate the model in the figure. However, by taking out levels of detail and then manipulating the model, everything becomes much faster.

Also, it is much easier to perform gross manipulations of the image when it has coarse detail. For example, if we want to move the ear, we don't have to move every control point, we just have to go to lower detail, move a few points, and then go back to high detail. The details follow the changes made at the coarse level.

Model manipulation is therefore improved in three different ways by using multiresolution surfaces.

1. It is faster because there is much less load on the computer.
2. It is faster because the user needs to manipulate fewer control points.
3. It is more intuitive, because the user can manipulate large sections of the image easily. For example, is is much more intuitive to "move the ear" than it is to "move all of the control points that comprise the ear".

See Figure 3, Figure 4, and Figure 5 for examples.

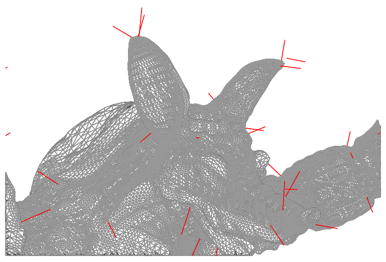


Figure 3: Armadillo Man

3 Improvements of Multiresolution Surfaces based on Subdivision

3.1 Dynamic Subdivision

Another problem with subdivision is that it sometimes gives too much detail. We do not always want the same resolution everywhere; we want more resolution only where

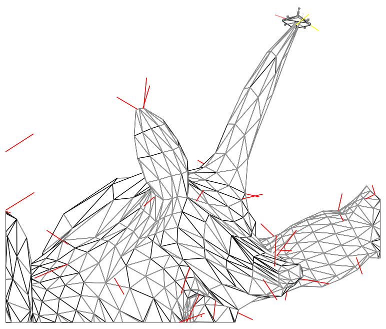


Figure 4: Armadillo Man at coarse resolution, with his ear moved.

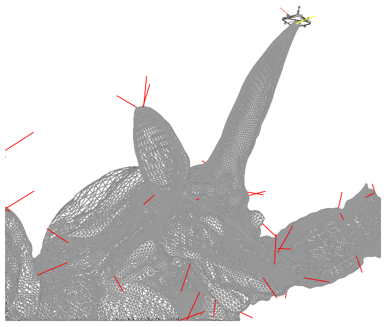


Figure 5: Armadillo Man at high resolution, with his ear moved.

we want a finer representation. In Figure 6, Figure 7, and Figure 8, more resolution is needed where the surface changes rapidly than where the surface remains fairly constant.

This problem can be solved by adaptively subdividing the surface. A triangle will only be subdivided if more subdivision is necessary. A flat plane, for example, should not be subdivided and given more detail, even if an adjacent jagged peak is being subdivided and given more detail.

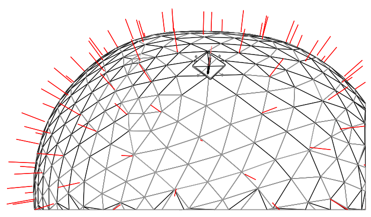


Figure 6: A fairly smooth model.

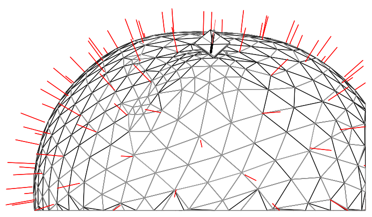


Figure 7: The same model with a slight bump.

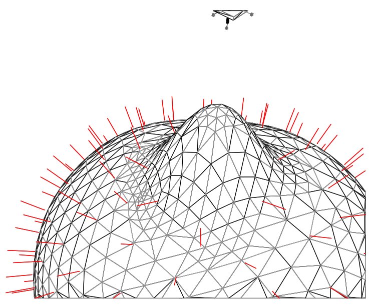


Figure 8: The same model with a large bump.

3.2 Adding Details Correctly

If a coarse version of an image is modified, care must be taken when adding back the details. If they are added blindly, the results may not be what is intended, as in the picture below.

A better strategy is to remember the normal of the surface where a detail is located, and to add the details relative to this normal.

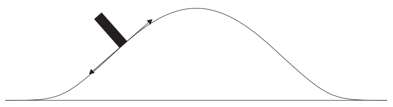


Figure 9: A figure with some detail.

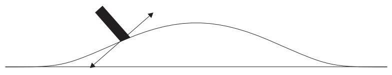


Figure 10: The figure was changed and the detail followed.

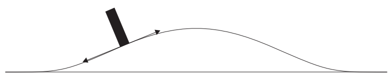


Figure 11: The detail was changed to reflect the new normal.

Surfaces

Lecture #19: Thursday, 4 December 1997
Lecturer: Pat Hanrahan
Scribe: Tamara Munzner
Reviewer: David Koller

1 Plane Curve Review

We first finish up the previous lecture topic of plane curves. The ellipse is used to show some of the properties of the **evolute**.

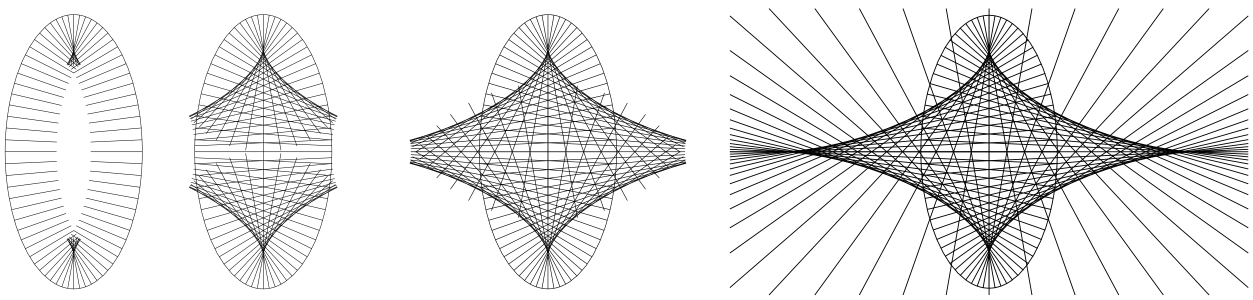


Figure 1: The envelope of the normals is the evolute.

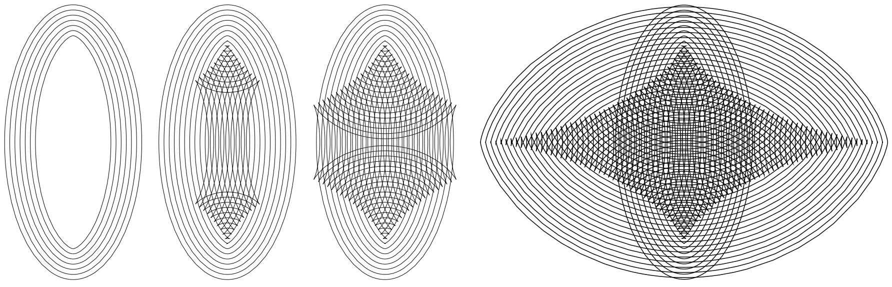


Figure 2: The evolute is the trace of wavefront singularities.

Figure 1 shows the normals to an ellipse drawn at uniform intervals. Recall that the envelope of the normals is the evolute. The figure shows that this evolute is a well defined curve which is the obviously traced out curved diamond shape. Notice that the evolute has cusps. Although it's not shown in the figure, if you think about it you can see that if you drew an osculating circle to the ellipse at any point, its center would fall on the evolute. Another way to think about this is to consider that the differential section of the surface around a point is a tiny arc of the circle centered on the evolute.

You may also recall from the previous lecture that the evolute is the trace of wavefront singularities. Figure 2 shows the offset curves of an ellipse. Again, these curves are drawn at uniform offset intervals. Note that they fold over, and their singularities and cusps lie on the evolute.

These two figures illustrate caustics, a subject we will return to in the next lecture. This one is devoted to curves in space and surfaces.

2 Space Curves

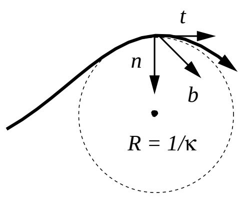


Figure 3: A space curve with a Frenet frame made from the tangent, normal, and binormal. The radius of the osculating circle is proportional to the curvature.

The previous lecture dealt with curves in the plane. Figure 3 shows a curve in space. A **space curve** can be parametrized by arc length with the function $x(s)$. At every point it has a well-defined tangent $t = x'(s)$, which is the derivative of the curve with respect to s. The derivative of the tangent is the curvature times the normal: $t' = \kappa n$. We can once again fit an osculating circle to the curve at the point, whose radius is the inverse of the curvature. In the plane this osculating circle argument was straightforward. However, the extra degree of freedom in space gives us a whole family of circles which fit the curve. We pick the one whose center is pointed to by the normal vector n.

2.1 The Frenet Frame

We know that the tangent is perpendicular to the normal. We can define a coordinate system with the addition of a third orthogonal vector, the **binormal**: $\mathbf{b} = \mathbf{t} \times \mathbf{n}$. This triad of axes is uniquely determined by the curve and is known as the **Frenet frame**.

The Frenet frame has the following properties:

$$\mathbf{b} \cdot \mathbf{b} = \mathbf{n} \cdot \mathbf{n} = \mathbf{t} \cdot \mathbf{t} = 1 \quad (1)$$
$$\mathbf{b} \cdot \mathbf{n} = \mathbf{n} \cdot \mathbf{t} = \mathbf{t} \cdot \mathbf{b} = 0 \quad (2)$$

Equation 1 expresses the fact that the axes have unit length, while equation 2 is a consequence of their orthogonality. The Frenet coordinate frame changes as you move along the curve or surface – it's a function of space. We can express any vector or point with respect to this frame. It's often useful to express changes in this frame:

$$\begin{array}{rcccccc} \mathbf{t}' & = & c_{tt}\mathbf{t} & + & c_{tn}\mathbf{n} & + & c_{tb}\mathbf{b} \\ \mathbf{n}' & = & c_{nt}\mathbf{t} & + & c_{nn}\mathbf{n} & + & c_{nb}\mathbf{b} \\ \mathbf{b}' & = & c_{bt}\mathbf{t} & + & c_{bn}\mathbf{n} & + & c_{bb}\mathbf{b} \end{array}$$

where c_{tt} means $(t' \cdot t)$. We already know that $t' = \kappa n$, which takes care of the first line:

$$\begin{array}{rclclcl} \mathbf{t}' & = & 0\mathbf{t} & + & \kappa\mathbf{n} & + & 0\mathbf{b} \\ \mathbf{n}' & = & (n' \cdot t)\mathbf{t} & + & (n' \cdot n)\mathbf{n} & + & (n' \cdot b)\mathbf{b} \\ \mathbf{b}' & = & (b' \cdot t)\mathbf{t} & + & (b' \cdot n)\mathbf{n} & + & (b' \cdot b)\mathbf{b} \end{array}$$

To simplify $\mathbf{b}'$, we start with the last term. We know from above that $\mathbf{b} \cdot \mathbf{b} = 1$. When we differentiate with the chain rule we find that $\frac{d}{dx}(\mathbf{b} \cdot \mathbf{b}) = \frac{d}{dx}\mathbf{b}^2 = 2\mathbf{b} \cdot \mathbf{b}' = 0$. So $\mathbf{b} \cdot \mathbf{b}' = 0$ and the last term drops away.

We attack the first term of $\mathbf{b}'$ by noting from above that $\mathbf{b} \cdot \mathbf{t} = 0$, which when differentiated gives $\mathbf{b}' \cdot \mathbf{t} + \mathbf{b} \cdot \mathbf{t}' = 0$ and thus $\mathbf{b}' \cdot \mathbf{t} = -\mathbf{b} \cdot \mathbf{t}'$. Substituting the definition of the tangent gives $-\mathbf{b} \cdot \mathbf{t}' = -\mathbf{b} \cdot \kappa\mathbf{n} = -\kappa\mathbf{b} \cdot \mathbf{n}$. Since we know that $\mathbf{b} \cdot \mathbf{n} = 0$, we have found that $\mathbf{b}' \cdot \mathbf{t} = 0$ and can drop the first term.

To find the middle term of $\mathbf{b}'$ we use two of the results from the other terms: $\mathbf{b}' \cdot \mathbf{b} = 0$ and $\mathbf{b}' \cdot \mathbf{t} = 0$. If the dot product of two vectors is 0 they are perpendicular, so $\mathbf{b}'$ is perpendicular to $\mathbf{b}$ and $\mathbf{t}$. The same is true for $\mathbf{n}$, so $\mathbf{b}'$ must be some multiple of $\mathbf{n}$. We will define the **torsion** τ of the curve by the equation $\mathbf{b}' = -\tau\mathbf{n}$. Just as the curvature κ measures the rate of change of the tangent, the torsion τ measures the rate of change of the binormal.

Our remaining unknown is $\mathbf{n}'$. We use the chain rule as above to find that $\mathbf{n}' \cdot \mathbf{n} = 0$ and drop the middle term. For the first term, we differentiate $(\mathbf{n} \cdot \mathbf{t}) = 0$ to find that $\mathbf{n}' \cdot \mathbf{t} + \mathbf{n} \cdot \mathbf{t}' = 0$. Substituting $\mathbf{t}' = \kappa\mathbf{n}$ and $\mathbf{n} \cdot \mathbf{n} = 1$ shows that $\mathbf{n}' \cdot \mathbf{t} = -\mathbf{n} \cdot \mathbf{t}' = -\mathbf{n} \cdot \kappa\mathbf{n} = -\kappa(\mathbf{n} \cdot \mathbf{n}) = -\kappa \cdot 1$. Thus the middle term simplifies to $-\kappa$. We find the last term of $\mathbf{n}'$ using the same process: $\mathbf{n}' \cdot \mathbf{b} = -\mathbf{n} \cdot \mathbf{b}' = -\mathbf{n} \cdot (-\tau\mathbf{n}) = \tau(\mathbf{n} \cdot \mathbf{n}) = \tau \cdot 1 = \tau$. We have thus found that $\mathbf{n}' = -\kappa\mathbf{t} + \tau\mathbf{b}$.

We have just derived what are known as the **Frenet-Serret** formulas:

$$\boxed{\begin{array}{rcccc} \mathbf{t}' & = & & \kappa\mathbf{n} & \\ \mathbf{b}' & = & & -\tau\mathbf{n} & \\ \mathbf{n}' & = & -\kappa\mathbf{t} & & +\tau\mathbf{b} \end{array}}$$

These formulas are just a system of differential equations which tell us about the motion of the Frenet frame. It is plausible that we can integrate them to recover information about the curve itself. In fact, we can completely reconstruct a curve just from curvature and torsion at every point. We state without proof the strong theorem which states this result. It is one of the many theorems in different branches of mathematics known as the Fundamental Theorem.

Theorem 1. *The Fundamental Theorem of Curves*
Given an arc length s, and two arbitrary functions $\kappa(s)$ and $\tau(s)$, we can uniquely determine the shape of curve $x(s)$.

2.2 The Darboux Vector

At the beginning of the last lecture we pointed out that there are many legitimate ways to describe a curve: algebraic, parametric, and intrinsic. The previous section was an example of the intrinsic approach.

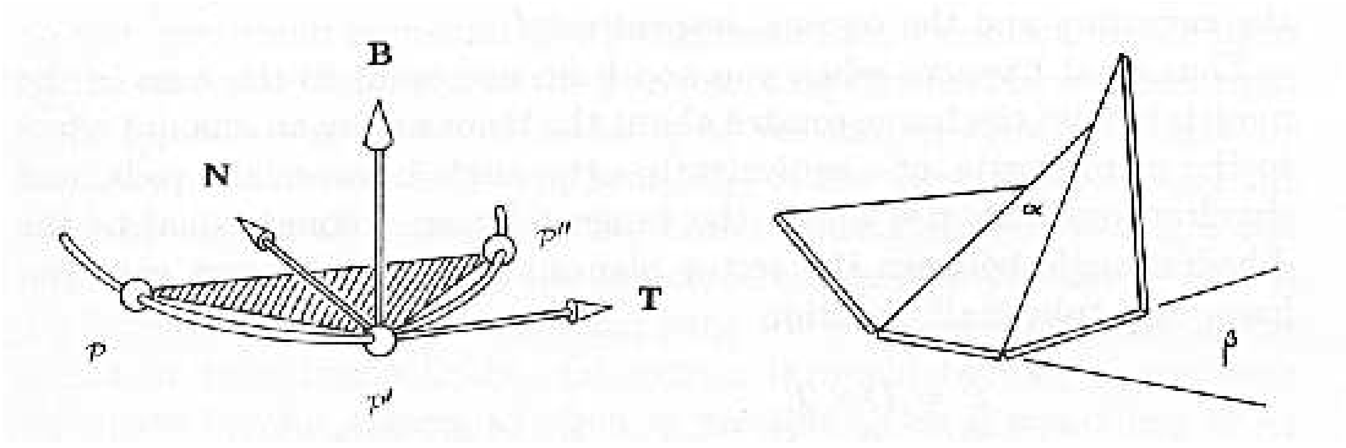


Figure 4: Two geometric interpretations of the Frenet frame from Koenderink [1]. On the left is the continuous case, while the right shows a less common piecewise linear view. Reproduced without permission from [1], Figures 110,111.

Koenderink's extremely readable book [1] builds up the ideas discussed in this lecture by describing several different interpretations in both words and pictures. Figure 4, from the book, shows two different geometrical interpretations of the Frenet frame.

On the left we can clearly see the coordinate frame defined by the normal, tangent, and binormal at point $\mathcal{P}'$. The normal lies in the shaded plane defined by the infinitesmally close points $\mathcal{P}$, $\mathcal{P}'$ and $\mathcal{P}''$. At every point there is a well defined normal – it just changes as you move.

The picture on the right shows a piecewise linear approximation to the smooth curve, as might be constructed by a set of linked rods. The rods themselves show the tangents.

The binormal for the point corresponding to $\mathcal{P}$ is drawn as an additional vertical rod. The osculating planes formed by each two neighboring rods are also shown. As the name implies, the osculating circles lie in these planes. This drawing shows that you can consider the plane as determined by two vectors instead of by a circle.

The normals at each point lie in this osculating plane, and the binormal at that point is perpendicular to this plane. The angle α is the amount of "turn" between the rods, which is the curvature. The binormal is the axis of this rotation. The angle β is the amount of turn around the tangent between two neighboring points – that is, the torsion.

Thus we can think of moving from one endpoint of a rod to the other by locally rotating the curve a little bit around the binormal and a little bit around the tangent. We thus see the motivation for defining the **Darboux vector** as that combined rotation:

$$\mathbf{d} = \kappa\mathbf{b} + \tau\mathbf{t}$$

We can use this quantity to rewrite the Fresnel-Serrat formulas concisely as

$$\begin{aligned} \mathbf{t}' &= \mathbf{d} \times \mathbf{t} \\ \mathbf{b}' &= \mathbf{d} \times \mathbf{n} \\ \mathbf{n}' &= \mathbf{d} \times \mathbf{b} \end{aligned}$$

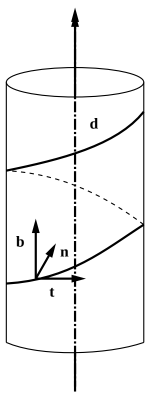


Figure 5: The Darboux vector and coordinate frame on a cylindrical helix.

It is also useful to consider the geometric interpretation of the Darboux vector and the coordinate frame on a cylindrical helix, as shown in Figure 5. The case of the cylindrical helix is unique because both the rate of translation and of rotation are constant. The Darboux vector is the axis of the cylinder, the normal points towards that axis, and the binormal is on the surface of the cylinder. The coordinate frame travels in a screw motion on a helix on the surface of the cylinder as it translates up the instantaneous tangent and rotates around the axis **d**.

3 Surfaces

We now move on from curves to surfaces. Consider a curve C passing through a point P on a surface, represented by the parametric equations u=u(t), v=v(t), so the curve in space is represented as x = x(u(t), v(t)). The tangent vector at a point is the derivative, and can be determined via the chain rule:

$$\mathbf{t} = \dot{\mathbf{x}} = \frac{d\mathbf{x}}{dt} = \frac{\partial \mathbf{x}}{\partial u}\frac{du}{dt} + \frac{\partial \mathbf{x}}{\partial v}\frac{dv}{dt}$$

Thus the tangent to the curve at the point P has direction components

$$\dot{\mathbf{x}} = \mathbf{x}_u \frac{du}{dt} + \mathbf{x}_v \frac{dv}{dt} = \mathbf{x}_u \dot{u} + \mathbf{x}_v \dot{v}$$

The tangent vector to C at P is a linear combination of the vectors $\dot{\mathbf{x}}$ and $\dot{\mathbf{v}}$ tangential to the parametric curves at P. Thus, the tangent to C at P lies in the plane determined by the tangents at P to the parametric curves. If we move in parameter space by (du, dv), we will move by $d\mathbf{x}$ in position along the curve.

3.1 The First Fundamental Form

The differential arc is the magnitude of the change: $ds = |\dot{\mathbf{x}}|$. This differential arc ds is the distance between two points. We can can compute ds^2 as follows:

$$\begin{aligned} ds^2 &= (\mathbf{x}_u \cdot \mathbf{x}_u)du^2 + 2(\mathbf{x}_u \cdot \mathbf{x}_v)dudv + (\mathbf{x}_v \cdot \mathbf{x}_v)dv^2 \\ &= Edu^2 + 2Fdudv + Gdv^2 \\ &= I \end{aligned}$$

This expression for ds^2 is known as the **first fundamental form** of the surface. Its constituent elements $E = \mathbf{x}_u \cdot \mathbf{x}_u$, $F = \mathbf{x}_u \cdot \mathbf{x}_v$, and $G = \mathbf{x}_v \cdot \mathbf{x}_v$ characterize the surface metrically. For instance, we can express the change in area as $dA = (EG - F^2)^{1/2}$ and the area of a surface patch is $\int\int (EG - F^2)^{1/2} dudv$.

The direction of the normal to the surface is $\mathbf{x}_u \times \mathbf{x}_u$. We normalize by its magnitude to find the unit normal:

$$\mathbf{N} = \frac{\mathbf{x}_u \times \mathbf{x}_v}{|\mathbf{x}_u \times \mathbf{x}_v|}.$$

We can also express the magnitude of the normal using E, F, and G:

$$\begin{aligned} (\mathbf{x}_u \times \mathbf{x}_v) \cdot (\mathbf{x}_u \times \mathbf{x}_v) &= (\mathbf{x}_u \cdot \mathbf{x}_u)(\mathbf{x}_v \cdot \mathbf{x}_v) - (\mathbf{x}_u \cdot \mathbf{x}_v)^2 \\ &= EG - F^2 \end{aligned}$$

resulting in

$$N = \frac{xu \times xv}{sqrt(EG - F^2)}.$$

3.2 The Second Fundamental Form

The first fundamental form allowed us to understand arc length and areas. We now move onward to curvature. Recall that we are discussing a curve on a surface. We now need to distinguish between the normal to the curve, which we have been writing as **n**, and the normal to the surface, which we shall call **N**. These two quantities are often different. An example is any circle on the sphere which is not a great circle: the normal to the circle is different from the normal to the sphere.

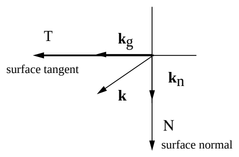


Figure 6: The total curvature can be decomposed into geodesic and normal components.

The tangent **t** is of course perpendicular to the normal and proportional to some vector which we will call **k**. We can decompose **k** into two components:

$$\begin{aligned} \mathbf{k} &= \mathbf{k}_n + \mathbf{k}_g \\ &= \kappa_n \mathbf{N} + \kappa_g \mathbf{T} \end{aligned}$$

These components are shown in Figure 6. The first component is proportional to the surface normal and is the **normal curvature $\mathbf{k}_n$**. The second is known as the **geodesic curvature $\mathbf{k}_g$** or **tangential curvature**, and is intrinsic – it depends on the characteristics of the surface itself, not its embedding in three dimensional space. We focus for now on the normal curvature.

Note that the normal curvature of a curve on a surface is not the same as the normal curvature of a space curve. Here we discuss the former. The normal curvature vector $\mathbf{k}_n$ of all curves on a surface through a given point which share a tangent vector is the same. The normal curvature is the projection of **k** onto **N**: $\kappa_n = \mathbf{N} \cdot \mathbf{k} = \mathbf{N} \cdot \mathbf{t}'$. Since $\mathbf{N} \cdot \mathbf{t} = 0$, we can differentiate along the curve to find that $(\mathbf{N} \cdot \mathbf{t})' = \mathbf{N} \cdot \mathbf{t}' = -\mathbf{N}' \cdot \mathbf{t}$. Thus

$$\kappa_n = \mathbf{N} \cdot \mathbf{t}' = -\mathbf{N}' \cdot \mathbf{t} = -\frac{d\mathbf{N}}{ds} \cdot \frac{d\mathbf{x}}{ds} = \frac{-d\mathbf{N} \cdot d\mathbf{x}}{ds \cdot ds} = \frac{\mathbf{II}}{\mathbf{I}} = \frac{edu^2 + 2fdudv + gdv^2}{Edu^2 + 2Fdudv + Gdv^2}$$

Recall the ds^2 is **I**, the first fundamental form. The quantity $-d\mathbf{N} \cdot d\mathbf{x}$ is the second fundamental form, also called **II** or the shape operator. It describes how the tangent (or normal) plane turns as you move around the surface. There are direct analogs to the elements E,F, and G found in first fundamental form, namely $e = -\mathbf{N}_u \cdot \mathbf{x}_u$, $2f = -(\mathbf{N}_u \cdot \mathbf{x}_v) + \mathbf{N}_v \cdot \mathbf{x}_u)$, and $g = -\mathbf{N}_v \cdot \mathbf{x}_v$. The properties of the shape operator are explored in great detail in classical differential geometry.

3.3 Meusnier's Theorem

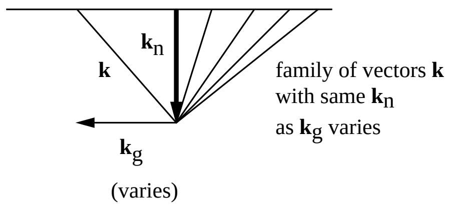


Figure 7: There is an entire family of curvature vectors which all share the same normal component.

Above we discussed the decomposition of **k** into the geodesic and normal components, as shown in Figure 6. Consider all the vectors **k** with the same normal component. These can be thought of as the intersection between the surface and a normal plane. Figure 7 shows that there is a family of such vectors which result from changing the geodesic component but leaving the normal component fixed. Thus all curves through the same point have the same normal curvature vector.

Actually, there are two different but interesting ways to move a plane about a normal vector: it can rock or roll. Figure 8 shows both cases. First, a family of planes which share a tangent can rock around the normal. All osculating circles to curves with the same tangent direction describe a sphere, sometimes known as Meusnier's sphere.

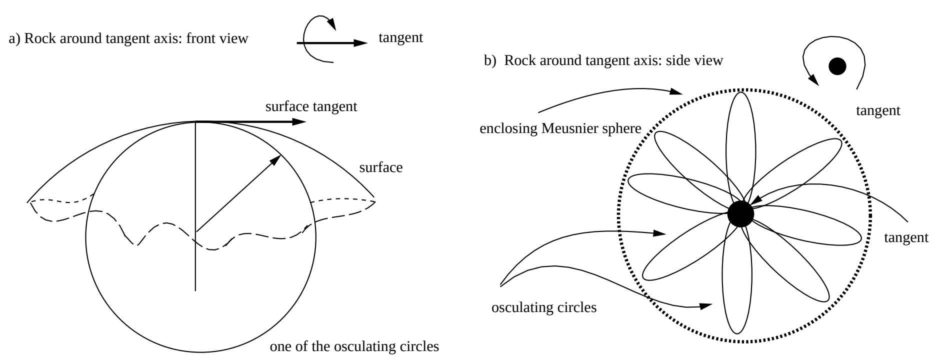
We can roll around the tangent vector to gain other degree of freedom In this case the osculating circles will change size as we rotate (unless the surface is a perfect sphere). The above discussion has been an informal statement of Meusnier's Theorem, which we will not prove.

The family of osculating circles that result from rocking around the tangent vector at a point has a well-defined minimum and maximum. The minimum and maximum circles are perpendicular to each other. The radii κ_1 and κ_2 of these two special circles are called the **principal radii of curvature** of the surface. The tangents are called the **lines of curvature**, and $t_1 \cdot t_2 = 0$. At every point on the surface we can compute these characteristic numbers which define the local curvature properties.

The radii of curvature are used to define the Gaussian and the mean curvature. The **Gaussian curvature** K is defined as $K = \kappa_1 \cdot \kappa_2$. The motivation for this definition is not immediately obvious, but we will discuss it in the next lecture.

$$H = \frac{\kappa_1 + \kappa_2}{2}$$

is the appropriately-named **mean curvature**.


a) Rock around tangent axis: front view
tangent
surface tangent
surface
one of the osculating circles
b) Rock around tangent axis: side view
enclosing Meusnier sphere
tangent
tangent
osculating circles

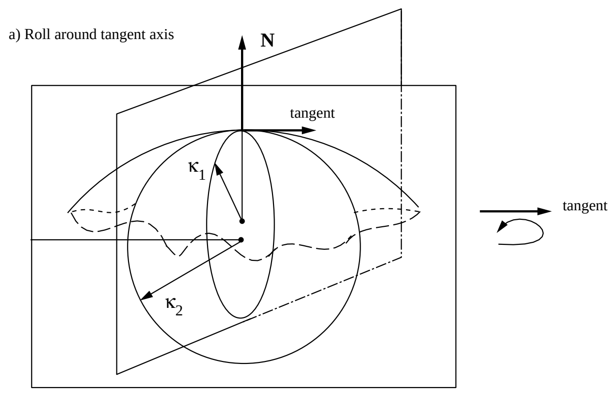


Figure 8: Meusnier's "Rock and Roll" Theorem. a) Rocking around the tangent vector on the normal plane: there are a whole family of circles which share the same tangent on the surface. b) A side view of the rocking, looking down onto the tangent axis, showing the rotating osculating circles which form Meusnier's sphere. c) Rolling around the tangent vector on the normal plane. The size of the osculating circle changes, and the maximum and minimum sizes appear in perpendicular directions. The radii of curvature κ_1 and κ_2 of these extremal circles are shown.

3.4 Ellipsoid

The curvature vectors on a surface are uniquely determined. Figure 9 shows the lines of curvature on an ellipsoid. In this case κ_1 and κ_2 are both positive: that is, they are both on the same side of the tangent plane. The lines of curvature form a net: they always intersect at right angles, and thus form a nice natural parametrization for the surface. These lines can be computed by starting at some point and integrating. The dot in the figure is called an **umbilic point** and is the place where minimimum and maximum curvatures are the same. At this point the local neighborhood of the surface can be approximated by a perfect sphere.

We can also define a series of concentric ellipsoids. Consider the 3D net we could define by connecting up the lines of curvature on these offset ellipsoids. These new

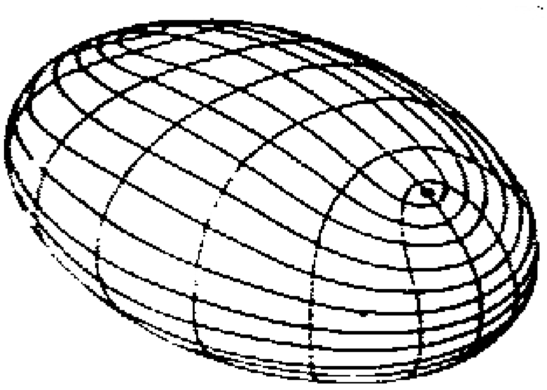


Figure 9: Lines of curvature meet at right angles and form a net on an ellipsoid. The minimimum and maximum curvatures are the same at the dot, which is an umbilic point. Reproduced from [3], Figure 2.22

surfaces would all meet at right angles and thus form a nice 3D curvilinear coordinate system.

3.5 Cuboid

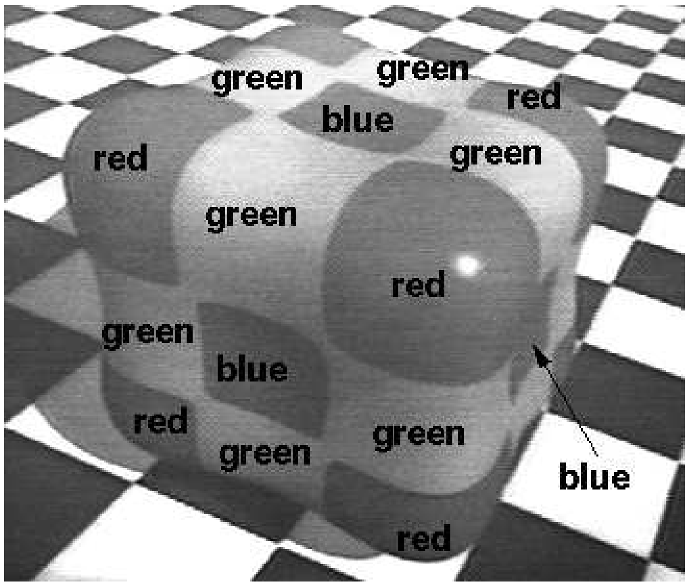


Figure 10: A cuboid surface colored by Gaussian curvature. Reproduced from [2], Plate 1

Figure 10 shows a cuboid surface colored by Gaussian curvature. The figure is annotated with a color legend since these notes are distributed in black and white. The red parts at the “corners” of the cuboid are elliptic – they are locally convex and both the

Gaussian curvature K and the principal radii κ_1 and κ_2 are positive. The blue patches in the center of the faces are concave: the Gauss curvature is still positive although the radii κ_1 and κ_2 are both negative. The center of curvature is on the opposite side of the tangent plane. The green parts of the surface along the edges have negative Gauss curvature. They are saddle points where the principle radii of curvature have different signs. Two colors meet along **parabolic lines**. These lines of inflection separate the positive and negative Gaussian curvature regions.

3.6 The Apollo of Belvedere

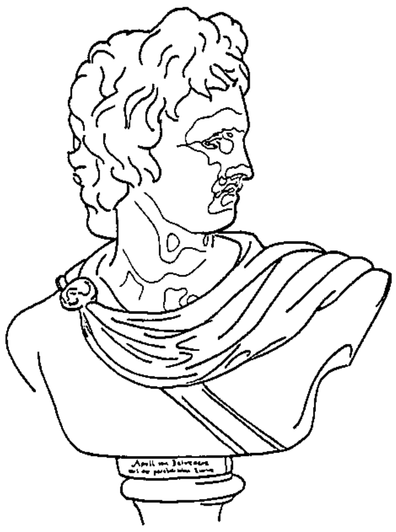


Figure 11: The Apollo of Belvedere with parabolic lines drawn in by Felix Klein. Reproduced from [1], Figure 440.

Figure 11 shows a reproduction of the Apollo of Belvedere. Parabolic lines were hand-drawn on its surface at the behest of mathematician Felix Klein. He hoped that decomposing a complex shape along parabolic lines would illustrate its fundamental aesthetics. Supposedly he abandoned this theory after seeing the resulting picture. Harvard mathematics professor David Mumford has looked at the problem more recently. Finding the true parabolic lines is tricky and is rather sensitive to noise. Stanford graphics professor Marc Levoy will be scanning the real Apollo during his Digital Michelangelo Project in Italy next year, at which time the parabolic lines could be computed directly to satisfy the curiosity of any aspiring aestheticians among us.

All theories of aesthetics aside, parabolic lines have many nice properties. Some of these pertain to caustics, which we will hear about in the next lecture.

References

[1] Jan J. Koenderink. *Solid Shape*. MIT Press, 1990.

[2] Don Mitchell and Pat Hanrahan. Illumination from curved reflectors. In *Computer Graphics (Proc. SIGGRAPH 1992)*, July 1992.

[3] Dirk J. Struik. *Lectures on Classical Differential Geometry*. Dover, 2nd edition, 1950.

Caustics

Lecture #20: Tuesday, 9 December 1997
Lecturer: Pat Hanrahan
Scribe: Christopher Stolte
Reviewer: Tamara Munzner

In this lecture, we discuss applications of differential geometry within the field of computer graphics. We will see how concepts discussed in earlier lectures can be used to solve problems involving the geometry of optics. Specifically, we will look at Fermat's Principle, rays and wavefronts, and caustics.

1 Fermat's Principle

In geometrical optics, we assume that the wave-like behaviour of light is insignificant and thus model the behaviour of light using rays. Light emitted from a point is assumed to travel along such a ray through space. In an effort to explain the motion through space taken by rays as they pass through various media, Fermat developed his *Principle of Least Action.*

> The path of a light ray connecting two points is the one for which the time of transit, not the length, is a minimum.

At the time that Fermat developed this principle, his justification was more mystical than scientific. His justification can be summarized by the statement that nature is essentially lazy, and these rays are simply doing the least possible work.

We can however develop a more useful formulation of the principle. We know from earlier lectures that the time along a curve through space can be calculated as

$$S(t) = \int dt = \int \frac{ds}{ds/dt}$$

We also know that ds/dt is velocity, which for light is know to be $\frac{ds}{dt} = \frac{c}{\eta}$ where η is the refractive index of the medium. Therefore, we have

$$S(t) = \int \frac{ds}{\frac{c}{\eta(s)}} = \frac{1}{c}\int \eta(s)ds \;\propto \int \eta(s)ds$$

We can thus define the optical path length from one point on a ray to another as the geometric path length weighted by the refractive index of the media. Furthermore, we can now restate Fermat's Principle as

> Light travels along paths of stationary optical path length, where the optical path length is a local maximum or minimum with respect to any small variation in the path.

Determining the path taken by a light ray between two points then becomes a simple matter of optimizing $S(t)$ between the points.

2 Applications of Fermat's Principle

We can make several observations as a result of Fermat's Principle which will prove useful as we explore the realm of geometric optics:

1. In a homogenous medium, light rays are rectilinear. That is, within any medium where the index of refraction is constant, light travels in a straight line.
2. The angle of reflection off of a surface is equal to the angle of incidence. This is the *Law of Reflection.*

We can also make some interesting and useful observations about conic surfaces. Conic surfaces are particulary useful in mirror optics - for example, the design of telescopes. We consider two conjugate points - two points that are perfect images of each other. A salient property of these conjugate points is that the optical path length of all rays connecting them is equal.

Consider a conic surface such as an ellipse. An ellipse is defined as the locus of all points such that the sum of the distances from each point to two fixed points (the foci) is constant, as in Figure 1. The two foci of a mirrored ellipse must then be optically conjugate points. A point source located at one focus must be imaged perfectly at the other focus.

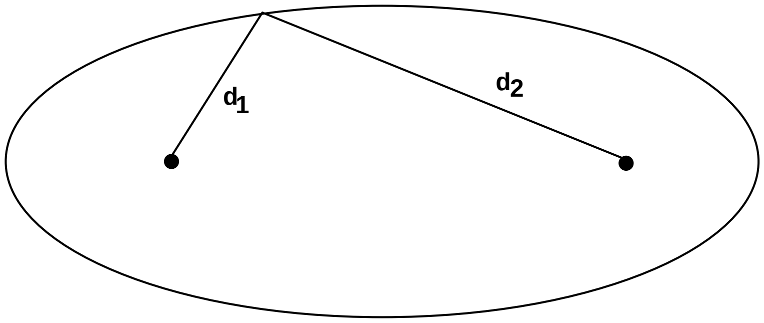


Figure 1: *An ellipse.* The ellipse is defined as the locus of all points such that the sum of the distances from each point to two fixed points is constant: $d_1 + d_2 = c$.

In the case of a parabola, one focus has become infinite. This can be interpreted by saying that an aggregate of rays, parallel to one another and to the axis of a paraboloid after being reflected by the paraboloid, will pass through the focus of the paraboloid. The Newtonian telescope leverages this fact in its design to collect and focus light from distant objects.

In general, a conic surface can be thought of as having two foci and these foci will be optically conjugate points. Figures 2 and 3 illustrate how this property of conic surfaces and geometrical optics has been applied in the design of the Cassegrainian telescope and the Gregorian telescope.

As a side note, a construction called a "Cartesian Oval" is similar to an ellipse, but has weighted distances. That is, rather than being constrained by the equation $d_1 + d_2 = c$, as in Figure 1, the oval is constrained by the equation $n_1 d_1 + n_2 d_2 = c$. The resulting non-elliptical shape will nevertheless have two points of perfect focus.

3 Virtual Light Sources

We now turn our focus to virtual light sources. In traditional ray tracing, a visual ray that encounters a reflective surface is bounced off of that surface and cast in the direction of reflection. Similarly, visual rays are refracted through volumes. Eventually these visual rays reach a non-reflecting surface and the shading calculation is calculated at this point of intersection. This traditional model is depicted in Figure 4.

Although this traditional ray tracing model does allow us to simulate the effect of seeing a scene through a reflective or refractive surface, it does not extend to the simulation of refracted or reflected illumination. In other words, the shading calculation at the point of intersection is limited to the direct components of illumination.

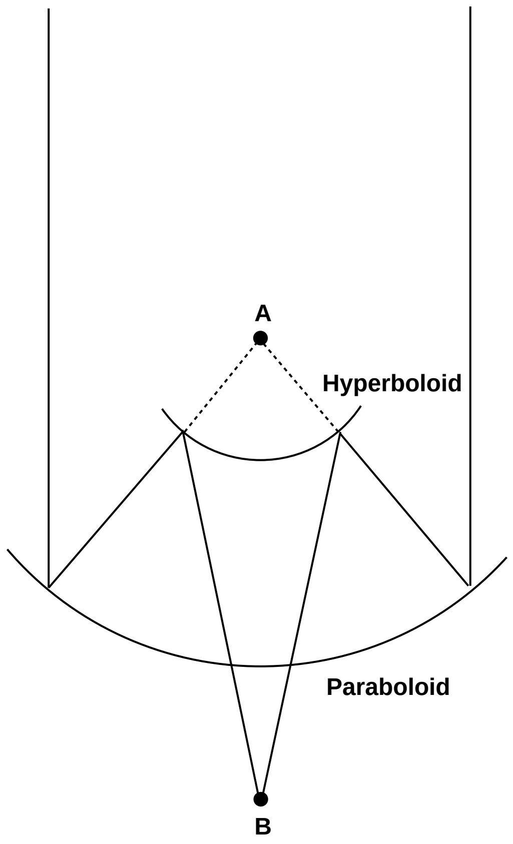


Figure 2: *The Cassegrainian telescope configuration* Point A is where the focus of the paraboloid and the virtual focus of the hyperboloid coincide. Point B is the real focus of the hyperboloid.

We can utilize differential geometry to allow us to solve this more complex problem - determining the reflected illumination at a point on a surface. To do this, we must cast rays from light sources so that they will reflect off of the mirrored surfaces and intersect the point being illuminated. When the light is reflected off the mirrored surfaces it is possible that the light rays may diverge or converge depending on the curvature of the reflective surface at the point of reflection. Thus there are two key problems that must be solved - determining which light rays will intersect the point being illuminated and calculating the proper irradiance at that point. The problem of reflected illumination is depicted in Figure 5.

Finding the paths from the light source to the point **P** that reflect off of the mirrored surface is not as complex as might be assumed. A possible path from the light source to the point **P** is depicted in Figure 5. The optical path length is

$$d(\mathbf{x}) = \sqrt{(\mathbf{s} - \mathbf{x})^2} + \sqrt{(\mathbf{p} - \mathbf{x})^2}$$

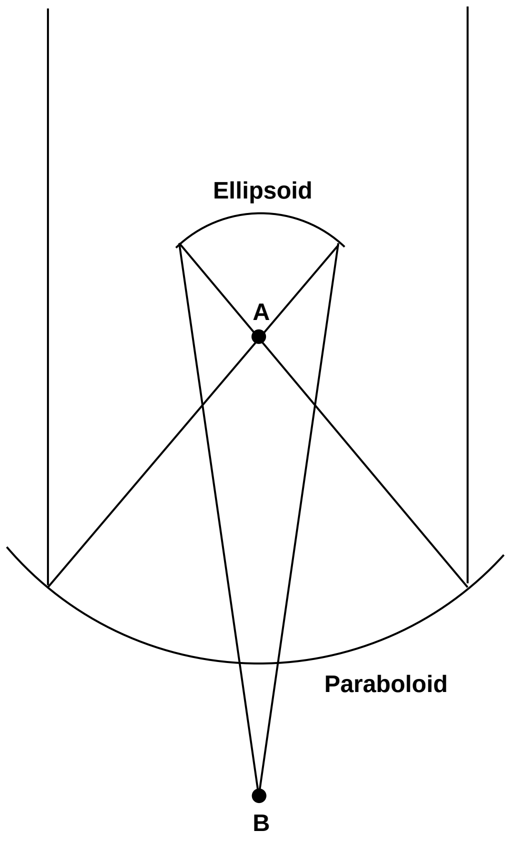


Figure 3: *The Gregorian telescope configuration* Point A is where the focus of the paraboloid and one of the foci of the ellipsoid coincide. Point B is the other focus of the ellipsoid.

According to Fermat's Principle, we want to optimize $d(\mathbf{x})$ in order to determine the path of the light rays. If the mirrored surface is defined by $g(\mathbf{x}) = 0$ then we can optimize $d(\mathbf{x})$ subject to the constraint that $\mathbf{x}$ lie on the surface defined by $g(\mathbf{x})$ using the technique of Lagrange multipliers:

$$\begin{aligned} \nabla d(\mathbf{x}) + \lambda \nabla g(\mathbf{x}) &= 0 \\ g(\mathbf{x}) &= 0 \end{aligned}$$

Solving these equations yields paths of locally extremal length.

Recall from our earlier discussion of conic figures that the two focal points of an ellipse are perfect images of each other - the optical path lengths of all reflected rays connecting

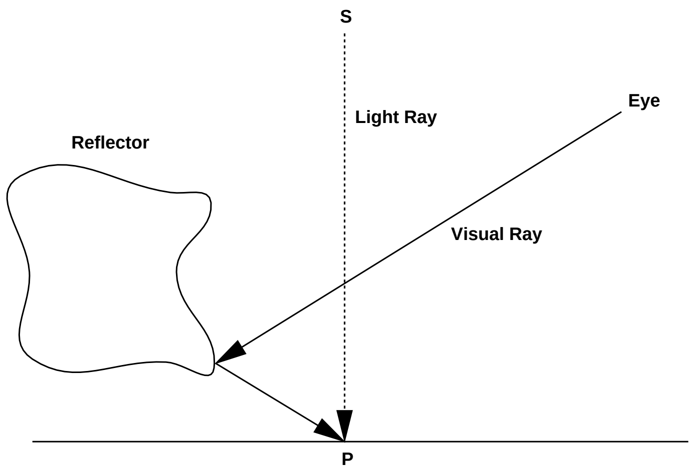


Figure 4: *Reflected Visual Rays*

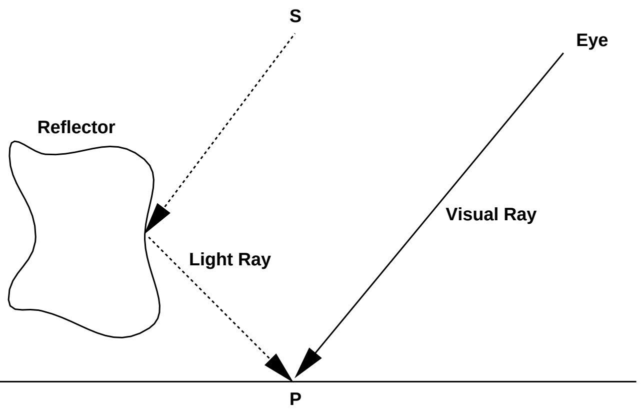


Figure 5: *Reflected Ilumination*

them are equal. If we select **s** and **p** as our foci of a family of ellipsoids and vary the optical path length, we get a family of confocal ellipsoids.

The system of equations produced by the Lagrange multipliers have a simple geometric interpretation. The extremal points must not only lie on the surface defined by $g(\mathbf{x}) = 0$, but they must also lie on the surface of one of these confocal ellipsoids and the ellipsoid must be tangent to the surface at the point of contact. Figure 6 depicts this geometric

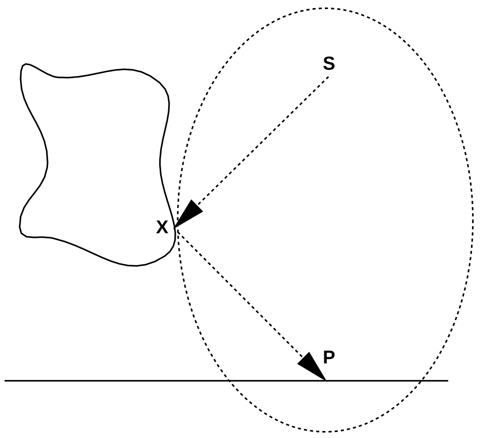


Figure 6: Osculating Ellipsoid

interpretation.

4 Rays and Wavefronts

In order to be able to compute the proper irradiance at a point being illuminated, we will need to determine if the rays of light from the light source are converging or diverging at the point. A given light will be the source of many rays, and the paths of the rays emitted are determined by the following equation:

$$S = \int \eta(s) ds$$

We will define a wavefront W to be the surface defined by the points on each ray at a constant s. Alternatively, the wavefront can be described as the locus of points at a given optical path length. We will not go into the details of wavefront properties, but one important property that should be noted is that the wavefront surfaces are orthogonal to the rays. You can think of wavefronts as isosurfaces in space.

This section is focused on intuitive concepts rather than formal derivations. In this entire discussion, light sources are assumed to be point light sources, although similar concepts and methods can be extended to the area light source case. Figure 7 depicts three simple types of wavefronts: those emitted by a single local point light, those emitted by an infinitely distant point light and a set of converging wavefronts.

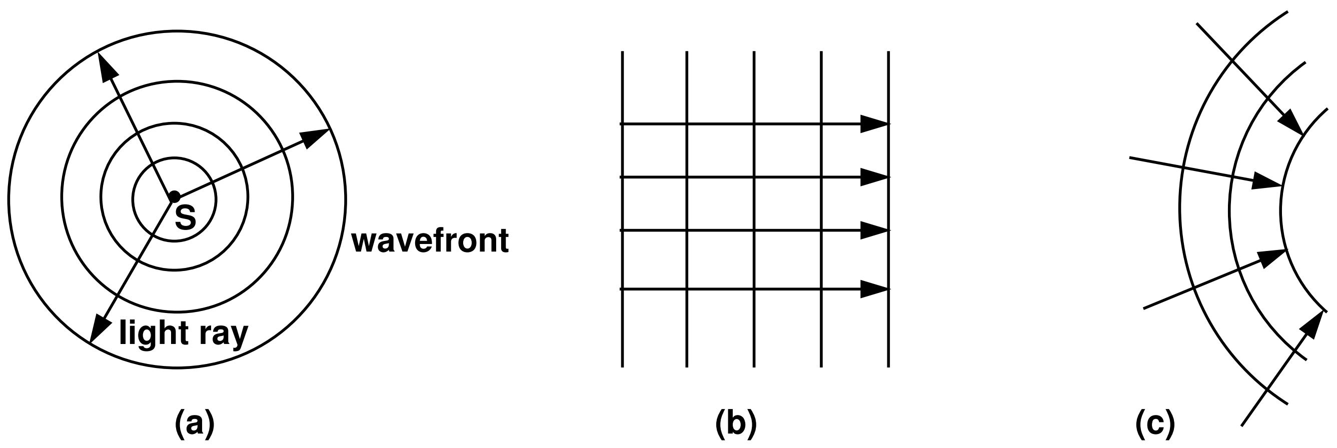


Figure 7: *Three different wavefronts:* (a) those emitted by a single local point light source, (b) those emitted by an infintely distant point light and (c) a set of converging wavefronts.

We are interested in the convergence and divergence of the rays because we need to be able to calculate the intensity of the light at a point being illuminated, and the intensity of the light can be shown to be equal to the radiant power of the light divided by the wavefront area.

Intuitively, this makes sense. Consider a set of rays which represent light moving forward over time, and two wavefronts defined by these rays at different points in time. Furthermore, consider the two patches of area on these wavefronts, shown in Figure 8, which are defined by this set of rays. The situation in the figure is a divergent wavefront, so the area dA' is greater than the area of dA. If the wavefront were converging, the opposite would hold.

We can think of these two patches of area as the ends of a tube containing the set of rays. Although the area of the two patches is different, the total power transmitted through the tube is a constant. Thus the intensity, which can be thought of as the number of rays per unit area, decreases as it passes through the tube. Note again that the intensity would increase in the case of a convergent wavefront.

We can formalize this intuition. We consider the general situation of the neighborhood of a point on a rectilinear ray. There is some orientation of a cutting plane at this point that will yield the maximum radius of curvature r_1, and another orientation of a different cutting plane which will yield the minimum radius of curvature r_2. Furthermore, we know the planes associated with these two radii of curvature are orthogonal from our earlier lectures on differential geometry. These radii of curvature are depicted in Figure 8.

Let dA be this element of area on the wavefront. All rays passing through dA will intersect some subsequent wavefront with area dA'. Let $d\theta_1$, $d\theta_2$ be the elements of angle

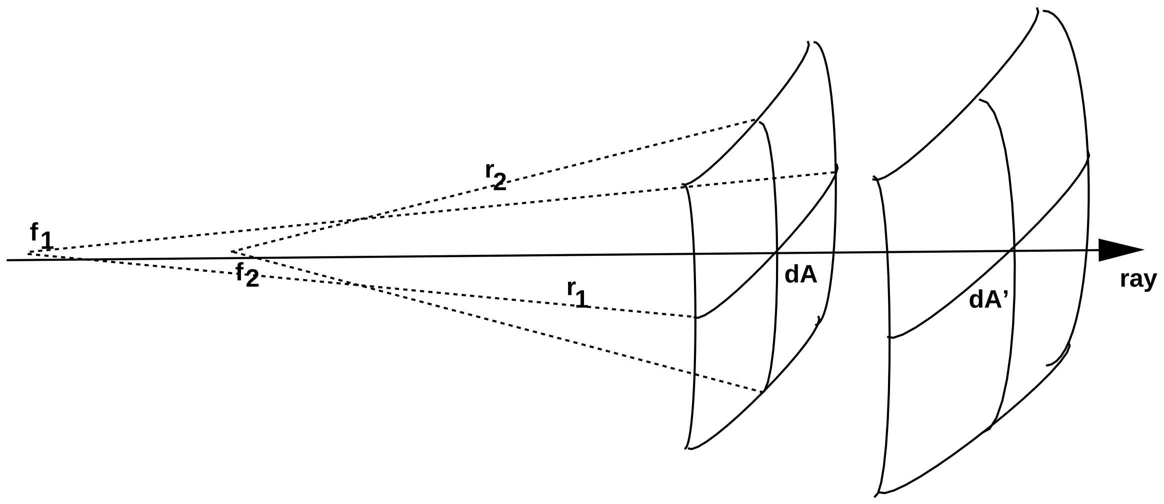


Figure 8: A ray and two small patches of area, dA and dA' on two wavefronts associated with the ray. For small enough patches, the area of the patch is defined by the radi of curvature and the angle subtended at the center of curvatures as $dA = r_1 d\theta_1 r_2 d\theta_2$.

subtended at the centers of curvature by these point areas. Because of the conservation of energy, we must have

$$d\Phi = I' dA' = I dA$$

where $d\Phi$ is power and I is intensity. Therefore

$$\frac{I}{I'} = \frac{dA}{dA'} = \frac{r_1 r_2 d\theta_1 d\theta_2}{r_1' r_2' d\theta_1 d\theta_2} = \frac{r_1 r_2}{r_1' r_2'} = \frac{\kappa'}{\kappa}$$

This illustrates the important point that the intensity is not only related to the area of the wavefront, but also to the inverse of the Gaussian curvature of the wavefront.

We know also that as a wavefront evolves forward according to the principles of optics, that the new wavefront will be an offset surface from the original wavefront. Thus, if the wavefront is diverging we can express the new radii of curvature as

$$\begin{aligned} r_1' &= r_1 + d \\ r_2' &= r_2 + d \end{aligned}$$

Alternatively, if the wavefront is converging, we can express the radii of curvature as

$$\begin{aligned} r_1' &= r_1 - d \\ r_2' &= r_2 - d \end{aligned}$$

Since intensity is inversely proportional to the radi of curvature, this means that at some point there must be infinite brightness. This point of infinite brightness is called a *caustic*, from the dimunitive form of the Greek word for "burning iron".

The caustic is thus the evolute, the locus of the centers of curvature. In the three dimensional case, there will be two caustic surfaces, one for each of the principal directions of curvature. What we colloqially call "caustics" are the curves formed by the intersections of these surfaces with a ground plane or object.

5 Orthotomics

We have seen from above that when a wavefront converges, a caustic is created. We are interested specifically in the case where a point light source shines upon a curved reflector, and the reflected light converges to a caustic. The *orthotomic curve* is an intermediate curve that we will use to find the caustics in this reflected case.

Because of the complexities of the three dimensional case, where the caustic is a curve and there are two caustic surfaces, we will focus our discussion on the orthotomic in the two dimensional case.

We will construct the orthotomic for an arbitrary light source and reflective surface. In doing so, we will show that the orthotomic corresponds to the reflected wavefront.

We construct the orthotomic as follows. Given a source s and a curve c, pick a point x on the curve and find its tangent. Then the locus of reflections of s about such tangents form the orthotomic curve, also known as the secondary caustic. This construction is depicted in Figure 9.

We now explain the construction of the orthotomic more rigorously. In this explanation, let $\mathbf{n}$ be the normal to the curve at point x and $\mathbf{t}$ be the tangent. Let $\mathbf{x}$ be the vector from s to the point x.

We know that if we reflect $\mathbf{x}$ about the tangent at x, this will define the point:

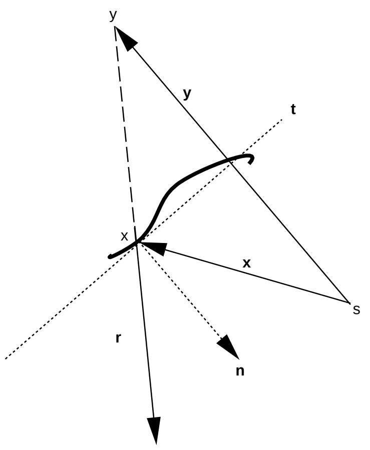


Figure 9: *Construction of the Orthotomic Curve*

$$y = 2[\mathbf{x} \cdot \mathbf{n}]\mathbf{n}$$

Looking at Figure 9 we can understand the formula for the point y. When we reflect the vector $\mathbf{x}$ across the tangent to the curve at x, it defines this new point. We can then note that the projection of $\mathbf{x}$ onto the normal $\mathbf{n}$ multiplied by 2 also gives us this same point. We must multiply by two to account for the fact that we have reflected the vector $\mathbf{x}$ across the tangent. To find $\mathbf{y}'$ we simply differentiate:

$$\begin{aligned}
\mathbf{y}' &= 2[\mathbf{x}' \cdot \mathbf{n}]\mathbf{n} + 2[\mathbf{x} \cdot \mathbf{n}']\mathbf{n} + 2[\mathbf{x} \cdot \mathbf{n}]\mathbf{n}' \\
&= 2[\mathbf{t} \cdot \mathbf{n}]\mathbf{n} - 2\kappa[\mathbf{x} \cdot \mathbf{t}]\mathbf{n} - 2\kappa[\mathbf{x} \cdot \mathbf{n}]\mathbf{t} \\
&= -2\kappa[(\mathbf{x} \cdot \mathbf{t})\mathbf{n} + (\mathbf{x} \cdot \mathbf{n})\mathbf{t}] \\
&\propto (\mathbf{x} \cdot \mathbf{t})\mathbf{n} + (\mathbf{x} \cdot \mathbf{n})\mathbf{t}
\end{aligned}$$

We use identities from the previous lectures: $\mathbf{t} \cdot \mathbf{n} = 0$, $\mathbf{n}' = -\kappa\mathbf{t}$, $\mathbf{x}' = \mathbf{t}$.

Consider the vector $[\mathbf{y} - \mathbf{x}]$. We would like to show that the normal to the orthotomic curve at the point y lies in the same direction as this vector. The normal at the point y must be perpendiculer to the tangent at y (which we will call $\mathbf{t}_y$). If the vector $\mathbf{r}$ (which is the reflection of $\mathbf{x}$ across the tangent to the curve at x) is in the same direction, it too must be perpendiculer to $\mathbf{t}_y$. Since $\mathbf{t}_y = \mathbf{y}'$, it is sufficient to show that $[\mathbf{y} - \mathbf{x}] \cdot \mathbf{y}' = 0$:

$$\begin{aligned}
[\mathbf{y}-\mathbf{x}]\cdot\mathbf{y}' &\propto [\mathbf{y}-\mathbf{x}]\cdot[(\mathbf{x}\cdot\mathbf{t})\mathbf{n}+(\mathbf{x}\cdot\mathbf{n})\mathbf{t}] \\
&= \mathbf{y}\cdot[(\mathbf{x}\cdot\mathbf{t})\mathbf{n}-(\mathbf{x}\cdot\mathbf{n})\mathbf{t}]-\mathbf{x}\cdot[(\mathbf{x}\cdot\mathbf{t})\mathbf{n}-(\mathbf{x}\cdot\mathbf{n})\mathbf{t}] \\
&= (\mathbf{x}\cdot\mathbf{t})(\mathbf{n}\cdot\mathbf{y})-(\mathbf{x}\cdot\mathbf{n})(\mathbf{t}\cdot\mathbf{y})-(\mathbf{x}\cdot\mathbf{t})(\mathbf{x}\cdot\mathbf{n})+(\mathbf{x}\cdot\mathbf{n})(\mathbf{t}\cdot\mathbf{x}) \\
&= (x\cdot\mathbf{t})(\mathbf{n}\cdot 2(x\cdot\mathbf{n})\mathbf{n})-(x\cdot\mathbf{n})(\mathbf{t}\cdot 2(\mathbf{x}\cdot\mathbf{n})\mathbf{n}) \\
&= 2(\mathbf{x}\cdot\mathbf{t})(\mathbf{n}\cdot\mathbf{n})(\mathbf{x}\cdot\mathbf{n})-2(\mathbf{x}\cdot\mathbf{t})(\mathbf{n}\cdot\mathbf{n})(\mathbf{x}\cdot\mathbf{n}) \\
&= 0
\end{aligned}$$

We know that the normal to the point y must be perpendicular to the tangent at y, which we calculated above. Since the dot product of the vector $[y-x]$ with this tangent is zero, this vector must lie in the same direction as the normal at y. Furthermore, from the figure and the law of congruent triangles, we can see that the reflected light ray **r** from the source must also travel in the direction of $[y-x]$.

Therefore the normal to the orthotomic at y is along the direction of **r** and passes through x. In more detail, light from s is reflected by the curve at x, according to the the Law of Reflection. Thus the incident ray and the reflected ray make equal angles on opposite sides of the normal to x. By congruent triangles, the reflected ray is along the line from y to x. From above, this is the normal to y.

It follows then that light rays having the orthotomic as their intial wavefront (i.e light rays starting simultaneously at all points on the orthotomic and then propagating down the normals) are the same as light incident from s and reflected by x. Thus the caustic by reflection of s is the caustic of the orthotomic.

Now, let's sum up intuitively what we have just formally explained. Given a light source and a curved reflector, we want to be able to find the caustics that would be formed. An easy way to compute these caustics is to use the orthotomic curve. The orthotomic curve has the property that its wavefronts will evolve to the same caustics as the wavefronts from the true light source will after being reflected.

6 The Gauss Map

Every point on a surface has some normal $n(u,v)$. The Gauss map is a mapping of every point on a surface to the point on the unit sphere with the same normal. This map is not one-to-one. Figure 10 shows an intuitive sketch of this construction for a small portion of the gauss map. The 3D case is too complicated to draw, so we show the 2D analog.

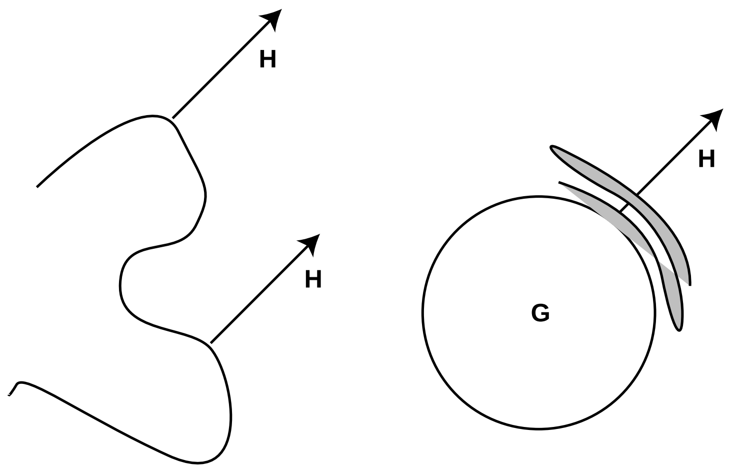


Figure 10: *Construction of the Gauss map:* the multiple normals on the original surface that are in the same direction as the vector N map to the single point on the unit sphere with normal in the direction of N. When the multiplicty of points being mapped to a single point on the unit sphere is greater than one, folds develop in the Gauss map.

The resulting Gauss map may have folds. These folds correspond to inflection points on the original surface, that is, the bottom of a concave valley, the top of a convex hill or a saddle point. At these points, the map which we are tracing out on the unit sphere changes direction because of the change in curvature at the inflection point on the original surface. If the original surface is smooth, the Gauss map will be continuous.

Consider a small patch of area S on the original surface. There will be a corresponding area patch w on the Gauss map. The Gaussian curvature is the differential ratio of the two areas: $\kappa = \lim_{S \to 0} \frac{S}{w}$.

We can formally define the Gauss map:

$$G(x(u,v)) = f(u,v)$$

as a map $G : s = S^2$ from the surface patch S to the unit sphere S^2.

When we are dealing with infinitely distant point light sources, the Gauss map can be used to tell how many virtual lights will be created by a reflective surface. Consider Figure 11. For every position of the viewer and the light source there is a vector H:

$$H = \frac{L+E}{|L+E|}$$

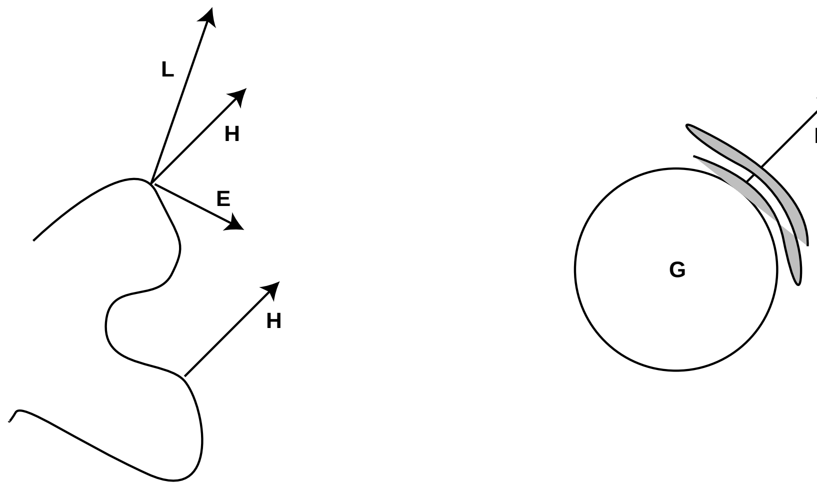


Figure 11: *Specularities* The number of virtual lights is the multiplicity of the points on the Gauss map with normal equal to H.

The number of virtual lights created by a reflective surface is the multiplicity of points on the Gauss map with normal equal to H, that is how many layers exist on the Gauss map.

We are thus interested in the folds created on the Gauss Sphere when the mapping is performed. When the reflective object is deformed or moved, the virtual lights on its surface move, and may be created or destroyed. This creation or destruction of virtual lights occurs at the parabolic points on the Gauss map.

7 Implementation Issues

When trying to implement shading calculations using virtual lights, we can utilize some of the properties we have learned to optimize our calculation. A brief overview of the techniques that can be used when implementing virtual lights is provided here. For a more detailed exposition, see [1].

Most importantly, we can leverage the observation made above that the intensity of light is proportional to the Gaussian curvature of the wavefront associated with that point. Thus, rather than keep track of all of the geometry associated with the wavefronts we can simply track the Gaussian curvature as the wavefronts evolve forward.

Propogating curvature through free space is trivial - we just need to add distance to the radius of curvature. The difficulty lies in efficiently calculating the change in curvature that occurs when the wavefront is reflected off a curved reflector.

We derive the equations necessary to calculate the reflected curvature. Recall the equation giving the directions of the reflected ray:

$$\mathbf{n}^{(r)} = \mathbf{n}^{(i)} + 2\cos i \mathbf{n}^{(s)}$$

In these equations, $\mathbf{n}^{(i)}$ and $\mathbf{n}^{(s)}$ are the normals to the incident wavefront and surface respectively; $\mathbf{n}^{(r)}$ is the normal to the reflected wavefronts. i is the angle of incidence of the rays.

We calculate the vector $\mathbf{u} = \mathbf{n}^{(i)} \times \mathbf{n}^{(s)}$. This vector must be tangent to both the incident wavefront and the surface since it is perpendicular to both normals. We can calculate the curvatures of the incident wavefront in the direction of $\mathbf{u}$ by rotating the principal curvatures using the angle between $\mathbf{u}$ and the line of curvatures and Euler's Formula. The curvatures of the surface in this direction can be computed using the curvature tensor of the surface.

The curvatures of the new wavefronts are computed by taking the directional dervvatives of $\mathbf{n}^{(r)}$ in the direction $\mathbf{u}$. These derivatives can be computed from the formulae for the reflected vectors and the directional derivatives of the normals on the incident wavefront and the surface.

For reflection:

$$\begin{aligned}
\kappa_u^{(r)} &= \kappa_u^{(i)} + 2\cos i \kappa_u^{(s)} \\
\kappa_{uv}^{(r)} &= -\kappa_{uv}^{(i)} - 2\kappa_{uv}^{(s)} \\
\kappa_v^{(r)} &= \kappa_v^{(i)} + (2/\cos i)\kappa_v^{(s)}
\end{aligned}$$

For refraction:

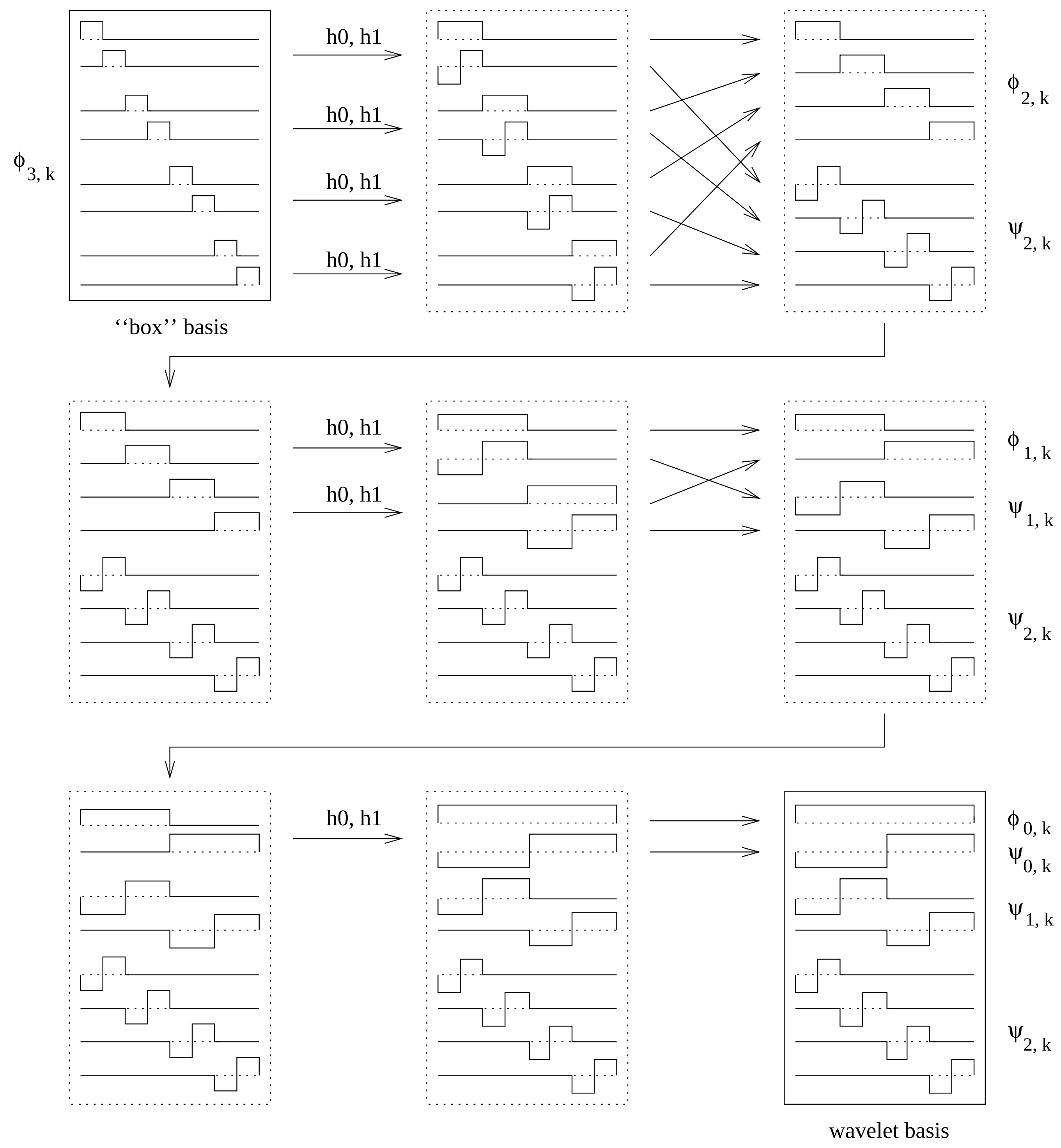
$$\begin{aligned}
\kappa_u^{(t)} &= \eta\kappa_u^{(i)} + \gamma\kappa_u^{(s)} \\
\kappa_{uv}^{(t)} &= \eta\kappa_{uv}^{(i)} + \gamma(\cos i/\cos t)\kappa_{uv}^{(s)} \\
\kappa_v^{(t)} &= \eta\kappa_v^{(i)} + \gamma(\cos i/\cos t)^2\kappa_v^{(s)}
\end{aligned}$$

Remember that the curvature of a plane is 0. Therefore, the curvatures of an outgoing wavefront reflected from a planar surface will be the same as the incoming wavefront (the fact that κ_{uv} switches sign is a result of the change in orientation of the coordinate system due to reflection). This is as expected, since a perfectly reflected wave does not change its shape. Note also that a planar wavefront incident onto a a reflecting surface essentially inherits the curvature of the surface. Thus if the surface is convex, the reflected wavefront will be diverging; whereas if the surface is concave, the wavefront will be converging, eventually forming a caustic.

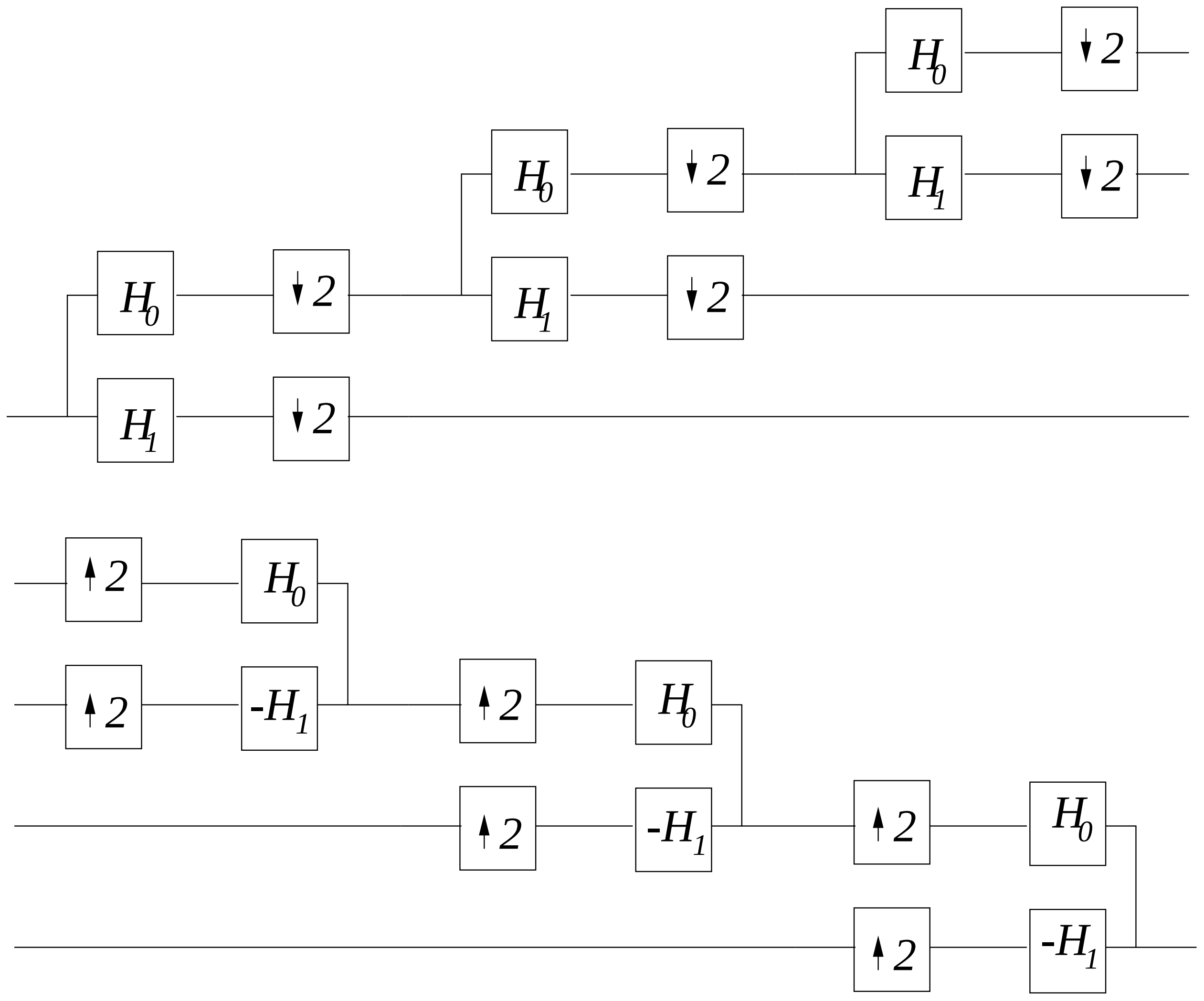
References

[1] Mitchell,D. and Hanrahan,P., Illumination from Curved Reflectors, *Computer Graphics* 26, 2 (1992), 283-291.

[2] Stavroudis, O. N. *The Optics of Rays, Wavefronts and Caustics*,Academic, 1972.



$\phi_{3,k}$
h0, h1
h0, h1
h0, h1
h0, h1
$\phi_{2,k}$
$\psi_{2,k}$
''box'' basis
h0, h1
h0, h1
$\phi_{1,k}$
$\psi_{1,k}$
$\psi_{2,k}$
h0, h1
$\phi_{0,k}$
$\psi_{0,k}$
$\psi_{1,k}$
$\psi_{2,k}$
wavelet basis



H_0
H_1
$\downarrow 2$
$\uparrow 2$
$-H_1$