Como seleccionar aleatoriamente una dirección utilizando la BRDF como distribución de probabilidades

Rosana Montes

14 de febrero de 2003

1. Introducción

El problema con el que nos encontramos es el de la selección de la direcció más probable de salida que se considera que se producirá al evaluar la BRDF para un vector incidente conocido. Sabemos que la BRDF es una función dependiente principalmente de dos direcciones, expresadas mediante cuatro valores correspondientes a las coordenadas polares de dichos vectores. Ya que conocemos la dirección incidente, tan solo deberemos de ocuparnos de obtener un valor para (θ_o, ϕ_o). En el contexto de un trazador de rayos se considerarán además la energía entrante en la dirección incidente, y se determinará la energía que saldrá desde el punto en la dirección w_o.

Puntos a tratar:

- Cambio de parámetros: proyección de un vector en el circulo
- Subdivisión de la BRDF en volumenes adaptados a la superficie.
- Muestreado por importancias en base a la selección aleatoria de un dominio.
- Cambio de parámetros: del circulo la dirección de salida w_o.

2. Muestreado de la BRDF

El muestreo de la BRDF podría realizarse de forma uniforme sobre su superficie, pero esto nos deja una varianza elevada. La eficacia de un estimador se mide por su varianza y por el tiempo que es necesario emplear para evaluarlo. Es lógico que se hayan desarrollado métodos que nos permiten reducir la varianza:

- Uso de valores esperados (*expected values*)
- Muestreado por importancias (*importance sampling*)
- Variables aleatorias de control (*control variates*)

Vamos a utilizar el muestreado por importancias, para encontrar una función de densidad p que es proporcional a aquella que queremos muestrear f. Es una técnica que se utiliza en informática gráfica para muestrear funciones que no son uniformes. En este caso se encuentra la BRDF, ya que suele presentarse con un pico en la dirección especular.

Supongamos que queremos aproximar una integral utilizando Monte Carlo. Se escoje N variables aleatorias uniformes X_i sobre el dominio, obteniendo N estimandores primarios $< I >_i$

$$I = \int f(x)dx$$

$$< I >_i = f(X_i) \; con \; X_i \in [0,1]$$

$$< I > \;=\; \frac{1}{N}\sum_{i=1}^{N} < I >_i \approx \int f(x)dx$$

El muestreado por importancias supone una pequeña transformación en la función estimada. Consiste en tomar X_i siguiendo una funcion de densidad $p(x)$, definiendo una nueva variable aleatoria Y_i:

$$< I >_{imp} \;=\; \frac{1}{N}\sum_{i=1}^{N} Y_i \quad donde \quad Y_i \;=\; \frac{f(X_i)}{p(X_i)}$$

Se demuestra que el valor esperado de esta nueva variable aleatoria aproxima adecuadamente la integral:

$$E[Y_i] \;=\; \int \frac{f(x)}{p(x)} p(x)dx \;=\; \int f(x)dx$$

En nuestro caso $p(x)$ será la probabilidad de seleccionar una zona con mucha *energía* o con capacidad de dar como salida una dirección en la que se condense de alguna forma la capacidad de reflexión del objeto en función de las características locales del material que lo compone.

2.1. Subdivisión Adaptativa

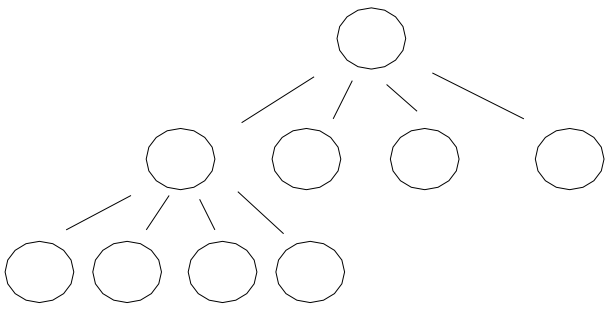
El muestreo puede ser más eficiente (de menor varianza) añadiendo la subdivisión adaptativa de la BRDF, de forma que en aquellas zonas donde la función es más variable, se utilice mayor resolución que cuando es más uniforme.

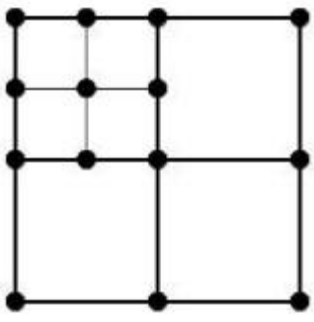
Esta subdivisión se realiza en el plano XY, y se toma como convenio, el uso de la proyección del vector de dirección en un circulo unidad embebido en un rectángulo cuyo dominio es $[-1, 1)$ x $[-1, 1)$. La BRDF es una función definida sobre la semiesfera, por lo que no es de estrañar que se eliga como representación del vector, su proyección en el circulo. Este vector w, se puede expresar como (x, y, z) y estará mapeado en dos valores (u, v). La proyección se realiza descartando la coordenada z (ver `Codificador.deCartesianasAlCirculo(w)`). La dimensión Z será la utilizada para el cálculo del volumen de un nodo, dato que participará en el criterio de subdivisión.

El dominio cuadrado original D será particionado en una serie de regiones $R_1, R_2, \ldots, R_n$, disjuntas que cubren el completamente el dominio. Esto es:

$$\begin{array}{rcl} R_i \cap R_j & = & \emptyset \quad \forall\, i \neq j \\ \cup_{i=1}^{n} R_i & = & D \end{array}$$

Esto se puede realizar construyendo un árbol (*quadtree*), de forma que en cada paso recursivo se subdivide el dominio en cuatro partes iguales, como se puede ver en las siguientes figuras.





Finalmente cada nodo hoja del *quad tree* corresponderá a una región R_i (con $i = 1 \ldots n$).
El árbol se construye dada una instancia de BRDF y conociendo la normal en un punto, y la dirección incidente sobre este. Asociado a éste, existirán otras clases, como son:

```
constructor Quadtree(Direccion entrante, Direccion normal, BRDF modeloBrdf)

        coder = construir Codificador(entrante, normal)
        root = construir Nodo(0, 0, 1, nulo, coder, modeloBrdf)
        alturaMaxima = evaluar modeloBrdf para direccion incidente en la normal
        root.particionar( alturaMaxima )


constructor Nodo(real u, real v, real s, Nodo ancestro, Codificador coder, BRDF fr)

        asignar los parametros a las variables miembro


clase Codificador

    constructor Codificador (Direccion entrante, Direccion normal)
            asignar los parametros a las variables miembro
            construir un sistema local con respecto a la normal

    metodo deCartesianasAlCirculo( Direccion w)
        u = w.x()
        v = w.y()
        devolver (u,v)

    metodo deCirculoADireccion( real u, real v)
        x = u
        y = v
        z = sqrt(1 - x^2 - y^2)
        direccion = construir Direccion(x,y,z)
        devolver direccion
```

Figura 1: Pseudocódigo de creación del quadtree

1. La clase *Direccion* que representa un vector en el espacio, con valores (x, y, z) y que consideramos normalizados.

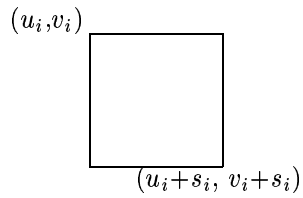
2. Un codificador de direcciones *Codificador* que sea el encargado de pasar una dirección a un punto en el circulo base, y viceversa.

3. La clase *Nodo* del árbol, en donde el nodo sin descendientes es nodo hoja, y el nodo raíz es aquel que no tiene ancestro. Cada nodo contiene la siguiente información:

   - puntero al nodo ancestro
   - puntero a los cuatro nodos descendientes
   - puntero a la clase *Region*

4. La clase *Region* que contiene la información específica a nuestro problema dentro del *quadtree*. Sea R_i la región considerada, se tienen los diguientes datos:

   - dominio $R_i = [u_i, u_i + s_i)$ x $[v_i, v_i + s_i)$, donde (u_i, v_i) es la posición del nodo en la región total y s_i, es el lado de la región cuadrada. Gráficamente:



- altura máxima del paralepípedo, M_i
- volumen encerrado por la superficie del nodo i:

$$v_{1i} = \int_{x,y \in R_i} f(x,y)\, dxdy$$

- volumen total del paralepípedo de base R_i y altura M_i:

$$v_{2i} = s_i * s_i * M_i$$

- probabilidad asociada a la región, P_i

El proceso de subdivisión es realizado por cada nodo en el método denominado particionamiento. Aquí el volumen de la función v_{1i} se calcula como la aproximación I_i de la integral de fr, asumiendo que $v_{1i}/I_i \approx 1$. Esta aproximación se realiza integrando en el dominio D en lugar de la semiesfera. Las zonas del rectangulo que caen fuera del circulo base, corresponden a zonas en las que la BRDF se evalua a cero, con lo que su probabilidad también es cero y no introducen ningún error en nuestros cálculos.

Así mismo calculando el volumen total del nodo v_{2i}, se define la probabilidad de seleccionar la región R_i como:

$$P_i = \frac{v_{1i}}{v_{2i}}$$

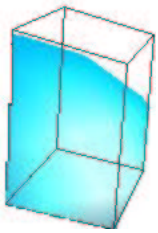
Cada nodo será el encargado de subdividirse cuando se cumpla la siguiente condición:

$$Si\ \ P_i < 0{,}5\ \ entonces\ \ subdividir$$

El criterio empleado para subdividir nos asegura que para todo nodo en el *quadtree*, se cumple para la probabilidad que $P_i \geq 0{,}5$. A continuación, se reparte la región actual del nodo entre sus cuatro descendientes, de forma que la unión de los cuatro descendientes pueda proporcionarnos la información del nodo ancestro.
Posteriormente se utilizará la probabilidad de cada nodo para recorrer el arbol hasta alcanzar un nodo hoja. Sea r un valor aleatorio uniformemente distribuido, P_i la probabilidad del nodo actual, y F_i la probabilidad acumulada de dicho nodo, se tiene que partiendo del nodo raíz podemos recorrer el *quadtree* con la intencion de seleccionar el nodo que cumpla:

$$\text{si } F_{i-1} < r \leq F_i \text{ se selecciona el nodo } i$$

Finalmente nos quedamos con una región en la que se recoge parte de la función.



```
clase Nodo
    metodo particionar(real m)

        //aproximar la integral de fr con incrementos dx y dy
        para x = u hasta u+s
            para y = v hasta v+s
                v1 += fr(x,y) * dx * dy
                calcular nuevo maximo
            fin_para
        fin_para
        v2 = s * s * m;

        p = v1 / v2
        si (p < 0.5)
          para i= 1 hasta 4
              nodoHijo[i] = construir Nodo
              nodoHijo[i].particionar(nuevoMaximo)
          fin_para
        fin_si
```

Figura 2: Método de subdivisión

2.2. Muestreado por Importancias

Una vez que tenemos seleccionada de forma aleatoria una región R_i, el siguiente paso consiste en muestrear siguiendo la técnica de *rejection sampling*, tal y como se muestra:

1. Generar un punto uniformemente distribuido en el dominio de la base, que denominamos (x, y). Observamos que el punto $(u_i + x,\ v_i + y)$ me define una dirección que llamaremos w_s (*sampled direction*).
2. Generamos una altura aleatoria independiente en $[0,M_i)$ y lo llamamos z.
3. Sea w_i la dirección incidente dada en la construcción del *quadtree*, aceptamos el punto $(u_i + x,\ v_i + y)$ si $f_r(w_i, w_s) \geq z$, es decir, si el punto está por debajo de la función.
4. Rechazamos el punto $(u_i + x,\ v_i + y)$ y repetimos desde el paso 1, si $f_r(w_i, w_s) < z$.

Este proceso se muestra ahora en pseudocódigo:

3. Obtención de Resultados

El ultimo paso es simple y de hecho ya se ha mostrado en el método anterior. Consiste en realizar la transformación inversa al inicial cambio de sistema (recordemos que se utilizó la normal para crear un sistema de referencia local).

Tan solo resta conocer el modo de utilización de nuestro *quadtree*, para obtener una dirección aleatoria utilizando la BRDF como distribución de probabilidades, como planteábamos al inicio del informe.

```
clase Quadtree
    metodo seleccionarRegion()
        devolver root.seleccionarRegion()

clase Nodo
    metodo seleccionarRegion()
        si (nodo actual es hoja)
          devolver nodo actual
        si_no

        real r = aleatorio(0,1)
        Nodo r1 = nulo
        real S = sumatoria nodoHijo(i).v1 (para i=1...,4)
        real F = 0
        para i=0 hasta 4
            Nodo r2 = nodoHijo(i)
            si (F < r) Y (r ≤ F + r2.v1/S) entonces
              r1 = r2
            sino
              F += r2.v1 / S
            fin_si
        devolver r1.seleccionarRegion()
```

Figura 3: Pseudocódigo del recorrido

4. Comprobación de los resultados obtenidos

En esta sección se pretende evaluar los resultados obtenidos por el método anterior de selección de las direcciones importantes, en base al principio fundamental de Monte Carlo: el *teorema de los números largos*. Básicamente este quiere decir que con un número elevado de muestras, el método converge a la solución esperada.
Utilizando esta idea se ha diseado un test que aplicado al método `refleja` de la clase `BRDF`, evalúa la convergencia.
Los modelos de BRDFs con los que se trabaja son:

- Modelo difuso perfecto de Lambert
- Modelo especular perfecto
- Modelo de Phong y sus derivados: Blinn-Phong y Lewis-Phong
- Modelo basado en *microfacets* de Cook-Torrance
- Modelo basado en la física de He-Torrance-Sillion-Greenberg
- Modelo anisotrópico basado en cilindros de Poulin-Fournier
- Modelo de Oren-Nayar
- Modelo anisotrópico de Ward
- Modelo empírico de Schlick
- Modelo de Strauss

```
clase Quadtree
    metodo muestrear(Nodo region)
        dirLocal = region.muestrear()
        dirGlobal = codificador convertir al sistema de referencia original
        devolver dirGlobal

clase Nodo
    metodo muestrear()
        hacer
            x = aleatorio(0,s)
            y = aleatorio(0,s)
            z = aleatorio(0,M)
            Direccion wi = codificador.entrante
            Direccion ws = codificador.deCirculoADireccion(u+x, v+y)
            sample = fr(wi, ws)
        mientras (sample < z)
        devolver ws
```

Figura 4: Selección aleatoria de una dirección de salida

```
clase BRDF
    metodo refleja( Direccion win, Direccion normal, Direccion wout)
        qtree = construir Quadtree(win, normal, this)
        Nodo region = qtree.seleccionarRegion()
        wout = qtree.muestrear( region )
```

Figura 5: Método de la superclase BRDF para la selección de direcciónes importantes

4.1. Método de Evaluación

El *principio de los números largos* indica que a mayor número de muestras utilizadas en los cálculos, mejores serán los resultados obtenidos. Esto nos va a servir para evaluar la bondad del método `BRDF.refleja` propuesto. El algorítmo que se comenta en este apartado, propone la repetición de dicho método un número elevado de veces, para comprobar si la proporción de rayos de salida sigue la función de densidad utilizada, la BRDF.

Sea m el número de muestras tomadas y sea Ω el conjunto de direcciones posibles de la semiesfera centrada en la normal de un punto en la superficie. El conjunto de todas las direcciones se representan (ver apdo) en un círculo inscrito en un cuadrado cuyo dominio D es [-1,1)x[-1,1).

Esto es, para todo $w_i \in \Omega$ con x,y,z reales se tiene que:

$$w_i = (x, y, z) \equiv w_i = (u, v), \quad u, v \in [-1, 1)^2$$

El dominio cuadrado original D será particionado uniformemente en una serie de celdas $C_1, C_2, \ldots, C_n$, disjuntas que cubren el completamente el dominio. Esto es:

$$\begin{aligned} C_i \cap C_j &= \emptyset \quad \forall\, i \neq j \\ \cup_{i=1}^{n} C_i &= D \end{aligned}$$

Cada celda equivale a un conjunto de direcciones en la semiesfera Ω, y va a intervenir en dos cálculos:

1. La estimación de la integral numérica de la BRDF en dicha zona.

2. La estimación estocástica de la fracción de direcciones que alcanza dicha zona utilizando el método `refleja` de la clase `BRDF`.

Conceptualmente todos los cálculos se realizan tomando direcciones en el plano XY, y solo cuando se utiliza la interfaz de la clase BRDF se realiza una conversión a direcciones cartesianas. La clase encargada de ello es similar a la ya comentada `DirCoder` salvo porque en este caso estamos en un sistema de referencias local y no hay que hacer ninguna transformación en las direcciones.

Se toman todas las direcciones de la esfera y se calculan numéricamente la integral de la BRDF en la celda i, con $i = 0, ..celres$ x $celres$. Llámese I_i. Para cada BRDF se fija aleatoriamente una dirección incidente cuyo origen es el centro de coordenadas del sistema local de referencia, y se obtienen m direcciones de salida. El test continua contando el número de veces que se ha dado como dirección de salida una dirección perteneciente a la celda i. Llámese n_i. El método `contiene` detectará a qué celda le corresponde la dirección generada.

A continuación se muestra la interfaz de la clase `Celda`

```
clase Celda
    constructor Celda(u, v, size)
            inicializa los datos miembro

    metodo init( )
        reinicia el numero de impactos
        calcula la integral de la BRDF

    metodo impacto()
        incrementa en uno el numero de impactos
```

La base de la evaluación del método `refleja` estriba en encontrar que las muestras generadas siguen como función de probabilidad la propia BRDF. Si esto es así el número de impactos de una zona de direcciones debería se proporcional a la integral de la BRDF en la misma zona.

$$I_i \propto \frac{n_i}{m}$$

El ratio entre los dos valores será claramente una constante.

$$\frac{n_i/m}{I_i} \;=\; a_i$$

Cuando calculamos la integral en toda la semiesfera de direcciones de salida, vemos que las proporciones anteriormente definidas se siguen manteniendo:

$$I = \sum_{i=1}^{N} I_i = \frac{n_1/m}{a_1} + \frac{n_2/m}{a_2} + \ldots + \frac{n_N/m}{a_N} \;=\; \frac{1}{a}$$

Una de las propiedades de las funciones de densidad es que están acotadas por uno, por lo que se cumple que:

$$\textstyle\sum_{i=1}^{N} \frac{n_i}{m} \;=\; 1 \quad \Rightarrow \quad a_1 = a_2 = \ldots = a_N = a$$

Finalmente el método `testBrdf()` nos mostrará si se produce la convergencia al valor constante a, esto es:

$$lim_{m\to\infty} \frac{n_i/m}{I_i} \;=\; a$$

```
clase SampleTree
    constructor SampleTree(BRDF instancia, real nmuestras, real ntest, entero celres)

            inicializar variables
            construir celres*celres Celdas


    metodo testBrdf()

            repetir desde 1 hasta ntest
                seleccionar direccion incidente aleatoria
                construir SampleQuadtree(incidente, normal, instancia)

                repetir para todas las celdas
                    calcular el valor de la integral en la celda
                    acumular el valor total de la integral
                fin
                cte = inversa de la integral total

                repetir para el numero de muestras
                    dirSalida = llamada a refleja()
                    celda = celdaQueContiene( dirSalida )
                    celda.impacto()
                    m' = sumatorio dirSalida
                fin

                repetir para todas las celdas
                    si hay impactos en celda
                      n = numero de impactos
                      I = integral en la celda
                      a = (n/m) / I
                    fin
                fin

                comprobar que los todos los a' convergen a cte
            fin


    metodo celdaQueContiene(real u, real v)

        size = tamaño de la celda (uniforme)
        i = parte entera de u+1/sz
        j = parte entera de v+1/sz
        devolver celda(i,j)
```

Figura 6: Algorítmo para la evaluación de `refleja`