3
Amostragem de Pontos em Superfícies Trianguladas

Neste Capítulo serão apresentadas as técnicas de amostragem de pontos por discos de Poisson em superfícies trianguladas, bem como as suas extensões para múltiplas classes de amostras. Inicialmente descreveremos a técnica para uma única classe e em seguida as suas extensões para múltiplas classes.

3.1
Descrição do Problema

Dada uma superfície triangulada M procuramos um conjunto de pontos S distribuídos aleatoriamente e uniformemente sobre M de modo que cada ponto deste conjunto seja distribuído por discos de Poisson com um raio r definido,

$$d(s_m, s_n) \geq r, \tag{3-1}$$

para todo $s_m, s_n \in S$, sendo d uma métrica.

Se procuramos apenas um conjunto S que satisfaça ao critério da distância mínima (raio do disco), então dizemos que S constitui uma única classe de pontos. Nosso problema se estende a encontrar diversos conjuntos de pontos disjuntos $S_0, \ldots, S_{c-1}$ sobre M de modo que para cada $i,\ 0 \leq i < c$ o conjunto S_i e a união deles $S_0 \cup \ldots \cup S_{c-1}$ contenham pontos distribuídos aleatoriamente e por discos de Poisson. Neste caso teremos múltiplas classes, onde cada conjunto S_i constitui a classe i. A solução que encontramos para este problema é uma extensão para superfícies não planares da técnica para geração de pontos no plano distribuídos por discos de Poisson em múltiplas classes, descrita em Wei (22).

Uma importante expectativa para este método de amostragem é obter a distribuição dos pontos em cada classe e na união das classes com características de amostragem de ruído azul, o que aponta para uma boa qualidade da distribuição.

3.2 Única Classe

O *dart throwing* (Cook, (4)) é um processo clássico de amostragem, onde são sorteados uniformemente pontos p sobre uma superfície S, tal que cada novo ponto p é aceito em S caso não viole o critério de distância mínima em relação aos pontos já amostrados. É considerado um processo de alto custo computacional para grandes conjuntos de pontos, pois quanto maior o conjunto de pontos a ser gerado, maior será o número de testes com o critério da distância mínima e o número de tentativas para inserção de um novo ponto pode ser elevado, o que aumenta o custo computacional do algoritmo.

Na Figura 3.1 é fácil perceber que muitas tentativas terão que ser feitas para se acrescentar um novo ponto nas regiões brancas interiores ao retângulo, de maneira a cobrir este retângulo por discos cumprindo a maximalidade desta distribuição. Quando for amostrado um novo ponto o grau de dificuldade para as próximas novas amostragens aumentará.

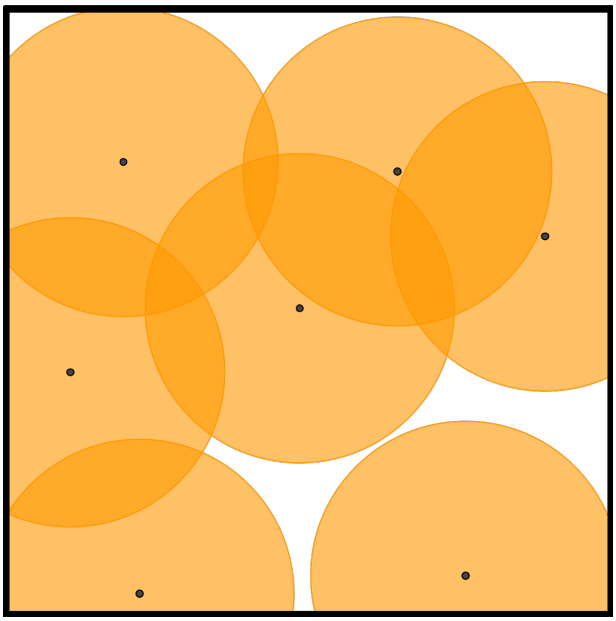


Figura 3.1: *Dart throwing* em uma região bidimensional

Seguimos algumas propostas encontradas em Bowers *et al.* (2) e Medeiros *et al.* (17) que apresentam um algoritmo mais eficiente para realização do *dart throwing* sobre superfícies em uma única classe. Este método inicia gerando um conjunto de pontos aleatórios O amostrados uniformemente sobre toda a superfície e construindo uma estrutura de dados que calcula a *bounding box* de O e particiona este conjunto armazenando as suas partições em uma grade celular limitado pela *bounding box*, de maneira que cada célula desta grade tenha o lado medindo $\frac{r}{\sqrt{3}}$, para um dado raio r.

O conjunto das células criado é randomizado, para reduzir enviesamentos na distribuição, e para cada célula sorteada neste conjunto, o algoritmo faz até k tentativas dentre os pontos aleatórios de O localizados nesta célula. O objetivo é amostrar um ponto que satisfaça a distância mínima com todos os pontos já amostrados por discos de Poisson e armazenados nas células imediatamente vizinhas.

Se os k pontos selecionados estiverem em conflito com os seus respectivos vizinhos, então a célula permanecerá vazia e sortearemos outra célula na tentativa de se armazenar um ponto com a propriedade requerida.

Testar somente as células imediatamente vizinhas é suficiente pois para cada candidato s numa dada célula $cell_i$, a região esférica centrada em s e com raio r conterá no máximo as células imediatamente vizinhas a $cell_i$, já que cada célula tem a diagonal medindo r. A coleção de todos os pontos aceitos forma o conjunto $S \subset O$.

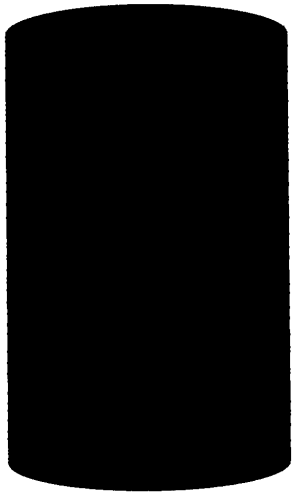
Na Figura 3.2 esquematizamos este processo. Primeiro geramos o conjunto de pontos aleatórios O (Figura 3.2(a)), em seguida, particionamos este conjunto em uma grade celular (Figura 3.2(b)). Sorteamos aleatoriamente uma célula, tomamos o primeiro ponto de O localizado nesta célula e checamos a sua distância com todos os pontos de S pertencentes as células vizinhas. Após verificarmos o t-ésimo ponto com os seus respectivos vizinhos pertencentes a S, este ponto será aceito ou não para $1 \leq t \leq k$, dependendo do critério da distância mínima. Se na t-ésima tentativa encontramos o ponto então o armazenamos nesta célula e partimos para outra célula dando início aos testes para a amostragem nesta nova célula. A Figura 3.2(c) exibe as bolas centradas nos pontos aceitos e demonstra o critério de mínima distância. O conjunto S é representado pela Figura 3.2(d), onde temos uma amostragem por discos de Poisson com o raio das bolas $r = 0.023$ e com 1285 pontos.

Neste exemplo visualizamos a amostragem dos pontos aceitos em uma única classe. Podemos observar pelo *zoom* da Figura 3.2(c) que cada bola contém no máximo um ponto como o esperado, o que equivale a cada ponto amostrado satisfazer a distância mínima com os seus vizinhos.

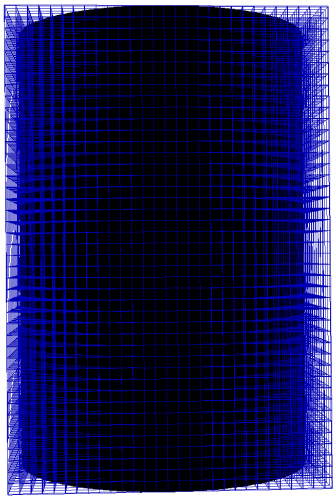
Nas próximas subseções vamos detalhar cada uma das etapas, descrevendo a geração do conjunto de pontos O sobre a malha e a estrutura usada para armazenamento e gerenciamento destes pontos.

3.2.1 Geração dos Pontos sobre a Malha

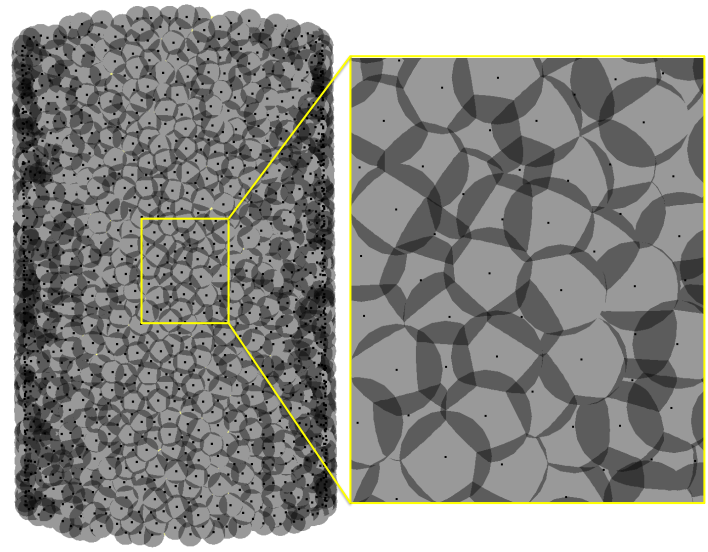
O processo é iniciado com a geração de um conjunto aleatório O de pontos uniformemente distribuídos sobre toda a superfície, de maneira que $|O| = \delta N$,



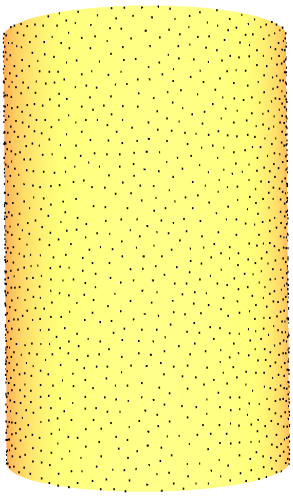
3.2(a): Conjunto O



3.2(b): Grade



3.2(c): Bolas



3.2(d): Conjunto S

Figura 3.2: Amostragem dos pontos em uma única classe

onde $|O|$ é quantidade de pontos em O, δ é um parâmetro de densidade da amostragem e N é a quantidade de pontos para um dado raio.

A quantidade de pontos para um dado raio é calculado de acordo com Lagae e Dutré (14), que define o maior raio possível para os discos em função da quantidade de pontos, com adaptação para superfícies, dada por,

$$r_{max} = 2\sqrt{\frac{\acute{A}rea(M)}{2\sqrt{3}N}}. \tag{3-2}$$

O raio r para os discos de Poisson de uma distribuição de pontos pode ser especificado como uma fração ρ do raio máximo,

$$r = \rho r_{max}. \tag{3-3}$$

Segundo Lagae e Dutre (14), se o parâmetro $\rho \in [0.65, 0.85]$ então este raio relativo r define uma boa medida para a distância mínima entre os pontos e neste caso é esperado que a distribuição deles mantenha características de ruído azul. Portanto, o valor do raio r fica estimado pela Equação 3-3, para um dado N.

O algoritmo para amostragem dos pontos sobre a malha triangular seleciona repetidamente triângulos com probabilidade proporcional a área de cada triângulo e amostra um ponto uniformemente sobre este triângulo por meio das suas coordenadas baricêntricas. A coleção de todos estes pontos gerados forma o conjunto O.

Amostragem de Pontos em Triângulos

As coordenadas de amostragem em triângulos são apresentadas em Pharr *et al.* (18). Para a simplificação do problema assumiremos a amostragem em um triângulo isósceles medindo 0.5 unidades de área, contudo, este método é bem definido para qualquer triângulo independente desta simplificação. Esta técnica de amostragem retornará as coordenadas baricêntricas do ponto amostrado.

Suponhamos que x e y sejam as coordenadas baricêntricas do ponto a ser amostrado. De fato, a função distribuição de probabilidade conjunta (f.d.p.) $p(x, y)$ deve ser uma constante igual ao inverso da área formada por x e y, ou seja, $p(x, y) = 0.5^{-1} = 2$.

Para calcular x e y é necessário encontrar as funções de distribuição acumulada (f.d.a.) $P(x)$ e $P(y)$ e em seguida inverter estas funções. Para isso, basta descobrir as f.d.p. $p(x)$ e $p(y|x)$ e integrá-las ao longo de x e y, respectivamente:

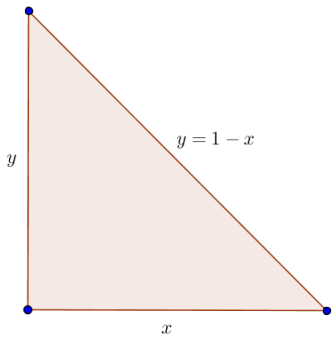


Figura 3.3: Triângulo isósceles

$$p(x) = \int_0^{1-x} p(x,y)dy = 2\int_0^{1-x} dy = 2(1-x), \tag{3-4}$$

é a densidade marginal de $p(x, y)$;

$$p(y|x) = \frac{p(x,y)}{p(x)} = \frac{2}{2(1-x)} = \frac{1}{1-x}, \tag{3-5}$$

é a densidade condicional de $p(x, y)$ dado $p(x)$. Tendo calculado $p(x)$ e $p(y|x)$, vamos encontrar $P(x)$ e $P(y)$:

$$\begin{aligned} P(x) &= \int_0^x p(x')dx' = 2x - x^2, \\ P(y) &= \int_0^y p(y'|x)dy' = \frac{y}{1-x}. \end{aligned}$$

Invertendo $P(x)$ e $P(y)$ teremos, respectivamente, as duas variáveis,

$$\begin{aligned} x &= 1 - \sqrt{\phi_1}, \\ y &= \phi_2\sqrt{\phi_1}, \end{aligned}$$

onde ϕ_1 e ϕ_2 são dois números aleatórios e uniformes em [0, 1].

Podemos obter a terceira coordenada baricêntrica z a partir de x e y já que

$$x + y + z = 1 \Leftrightarrow z = 1 - x - y. \tag{3-6}$$

Com isso temos que $p = xv_0 + yv_1 + zv_2$, donde p será um ponto aleatório e uniformemente distribuído no interior do triângulo e v_0, v_1 e v_2, são os vértices do triângulo.

Este passo da geração de O é o único que depende diretamente da malha triangular, sugerindo que o nosso algoritmo de amostragem de pontos distribuídos por discos de Poisson em múltiplas classes sobre superfícies pode ser utilizado sobre qualquer malha com topologia bidimensional, desde que se tenha um bom método para gerar pontos aleatórios e uniformemente

distribuídos sobre esta malha.

3.2.2 Estrutura de Dados

Após construir um conjunto de pontos O aleatórios e uniformemente distribuídos sobre toda a superfície triangulada o algoritmo usa uma estrutura de dados que permite armazenar e gerenciar estes pontos. Para isso, particionamos o conjunto O em uma grade celular e uma *hash table* é utilizada para o acesso eficiente às células desta grade.

Particionamento de O em uma grade

Particionar o conjunto de pontos O em uma grade celular é basicamente construir uma grade celular limitado pela *bounding box* de O usando $\frac{r}{\sqrt{3}}$ como o tamanho de cada célula e para cada ponto de O armazena-se a sua célula correspondente, associando o índice da célula aos seus pontos. Em seguida, basta ordenar este conjunto de pontos pelo índice da célula correspondente, de modo que os pontos pertencentes à mesma célula fiquem juntos. Esta ordenação não interfere na aleatoriedade da distribuição espacial dos pontos, pois eles permanecem armazenados dentro das suas células correspondentes na mesma posição em que foram gerados inicialmente.

As células que contêm pelo menos um ponto dentro são denominadas como válidas, caso contrário ficam denominadas como inválidas. Como consideramos malhas com topologia bidimensional, então o número de células inválidas supera bastante o número de células válidas, como mostrado na Figura 3.4. Não estamos interessados nas células inválidas, pois permanecerão vazias durante todo o processo. Portanto, percorrer o conjunto das células válidas equivale a percorrer um conjunto esparço em relação a grade.

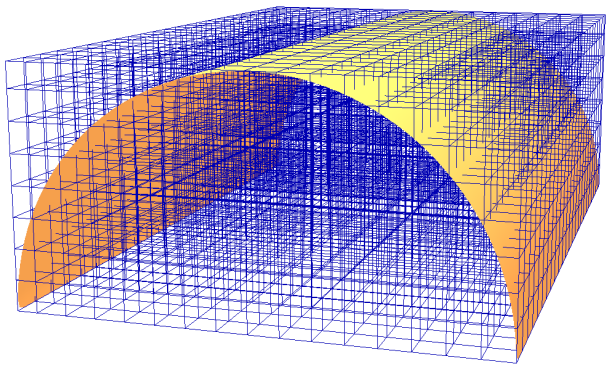


Figura 3.4: Células válidas e inválidas

Armazenamento das células

Como temos um conjunto esparso de células válidas, torna-se conveniente utilizar um método mais eficiente para percorrer este conjunto, visto que as células inválidas podem ser descartadas. A *hash table* é a estrutura usada para o armazenamento das células válidas e do seu respectivo ponto amostrado por disco de Poisson. Esta estrutura permite acessar as células válidas da grade com eficiência e é criada após a geração e a ordenação dos pontos aleatórios.

O índice global de cada célula é definido como uma chave (*hash_key*), que identifica, a partir da função *hash*, a posição (*hash_idx*) das células armazenadas dentro da *hash table*. Esta função *hash* foi definida como:

$$hash\_idx = hash\_key \ \% \ hash\_table\_size$$

O tamanho da *hash table* (*hash_table_size*) foi definido como quatro vezes o número de células válidas. Para avaliar as colisões causadas por células com o mesmo *hash_idx* foram armazenadas pequenas estruturas definidas como *buckets*, onde para cada *bucket* teremos:

bucket {
- índice da célula (*hash_key*)
- índice do primeiro e do último ponto randômico armazenado na célula
- variável para um possível ponto amostrado por disco de Poisson (ADP)

Os *buckets* com o mesmo *hash_idx* são armazenados sequencialmente em uma lista encadeada criada nesta posição dentro da *hash table*. Esta posição é mapeada pela função *hash*, a partir da *hash_key*.

A Figura 3.5 exemplifica uma *hash table* com o armazenamento das listas de *buckets* nas posições correspondentes.

Após criada a *hash_table*, a pesquisa das células será feita pelos seus índices. Para isso é encontrada a posição (*hash_idx*) na *hash_table* pela função *hash* usando o índice da célula (*hash_key*) e então é feita uma busca linear dentre os *buckets* existentes nesta posição com o objetivo de se encontrar aquele que armazena o índice da célula correspondente. A célula permanece inválida, caso não haja correspondência encontrada. De outro modo, a pesquisa retorna imediatamente o *bucket* com o índice da célula correspondente.

Exemplo: Supondo que em uma grade haja 1750 células válidas ($hash\_table\_size = 4 \times 1750 = 7000$). Desejamos encontrar a célula 347 ($hash\_key = 347$), que a princípio não sabemos se é válida ou não. Então, descobrimos a posição em que o *bucket* correspondente estaria armazenado, ou seja, $hash\_idx = 7000 \ \% \ 347$. O *bucket* correspondente só poderia estar armazenado na posição 60 ($hash\_idx = 60$). Daí fazemos uma busca linear dentre os *buckets*

que foram armazenados nesta posição na tentativa de encontrar aquele cujo índice da célula é 347. Caso haja algum correspondente ele será retornado, caso contrário esta célula não é válida.

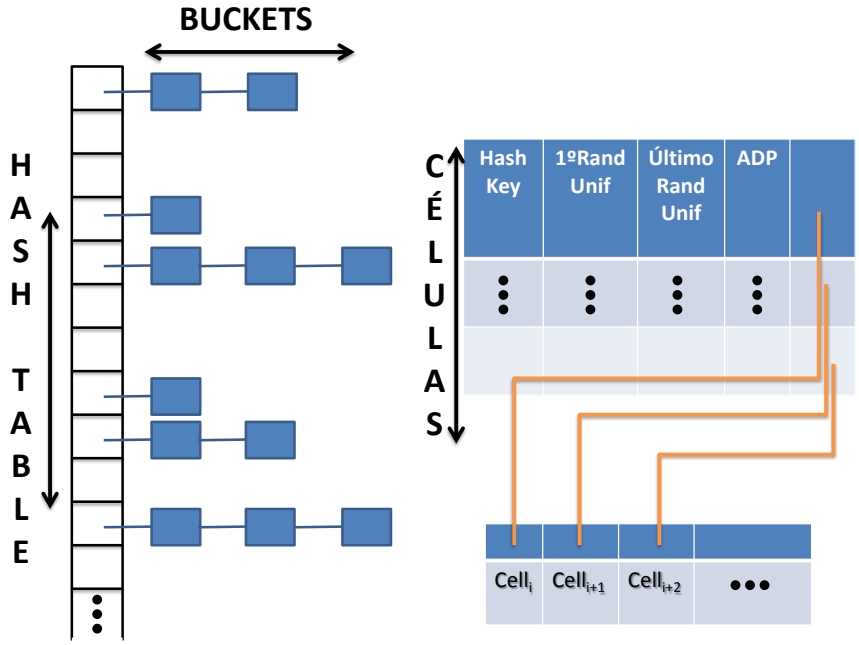


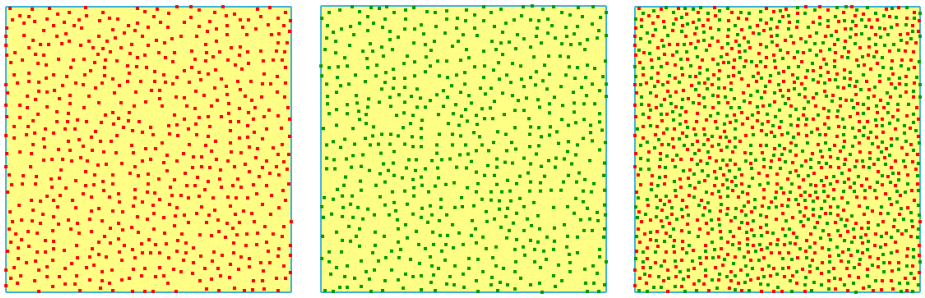
Figura 3.5: *Hash table* com as células válidas

3.3 Múltiplas Classes

Para a amostragem de pontos por discos de Poisson em múltiplas classes desejamos encontrar c classes de pontos (conjuntos de pontos disjuntos) de modo que cada classe S_i e a união delas, $S_0 \cup \ldots \cup S_{c-1}$, sejam amostrada por discos de Poisson, como na Figura 3.6. Neste caso espera-se características de ruído azul em ambas as classes e na união delas. Além da boa qualidade da amostragem, garantida pelas propriedades de ruído azul em cada classe e na união, a independência entre os pontos de classes diferentes permite que cada conjunto de pontos S_i tenha atributos independentes por classe, o que torna este método mais interessante.

Wei (22) propõe um algoritmo para domínios planares, onde a principal ideia é estender a técnica do *dart throwing* em única classe para o *dart throwing* em múltiplas classes.

Como visto na Seção 3.2, com amostragem em uma única classe, um raio fixado era suficiente como distância mínima para garantir a distribuição por discos de Poisson. Para amostrar diversas classes de pontos S_i sobre um mesmo



3.6(a): Classe 0 3.6(b): Classe 1 3.6(c): Múltiplas classes

Figura 3.6: Amostragem planar

domínio espacial de modo a preservar a distribuição por discos de Poisson em cada classe S_i e em $S_0 \cup \ldots \cup S_{c-1}$ é necessário um método eficiente para o armazenamento e acesso de cada distância mínima r_i da classe S_i e de cada r_{ij}, distância mínima entre os pontos de S_i e S_j, para todo $0 \leq i, j < c$.

Neste sentido, Wei (22) propõe outro algoritmo robusto que reúne o espaçamento mínimo entre os pares de pontos amostrados. Este algoritmo consiste em substituir o raio dos discos por uma matriz simétrica $(r_{ij})_{c \times c}$ (r_matriz) que contenha na sua diagonal principal os raios definidos para as c classes, onde, $r_{ii} = r(i,i) = r_i$ é o raio definido para S_i (classe i) e fora da diagonal principal, em $r_{ij} = r(i,j)$, temos o espaçamento mínimo entre os pontos de S_i e S_j (classe j), para $i \neq j$. Deste modo, teremos a seguinte matriz:

$$r = \begin{pmatrix} r_{00} & r_{01} & \cdots & r_{0(c-1)} \\ r_{10} & r_{11} & \cdots & r_{1(c-1)} \\ \vdots & \vdots & \ddots & \vdots \\ r_{(c-1)0} & r_{(c-1)1} & \cdots & r_{(c-1)(c-1)} \end{pmatrix}$$

Vale ressaltar que $r_{ij} = r_{ji}$, $\forall i, j$, pela simetria da distância entre dois pontos.

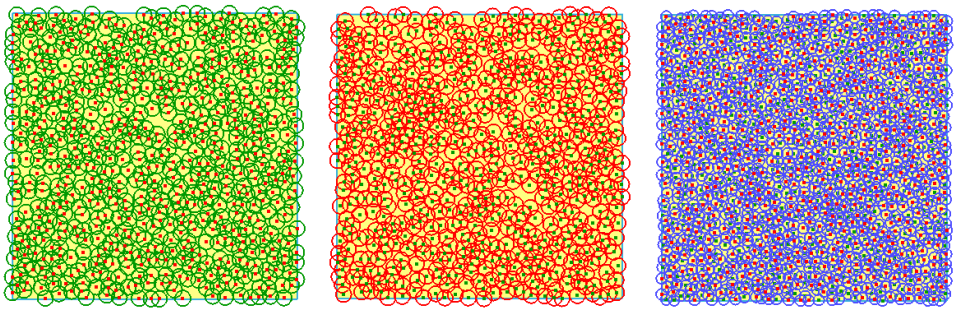
O critério para a distância mínima entre os pontos de S_i e S_j é estabelecido como:

$$d(s_i, s_j) \geq r(i,j),$$

para todo $s_i \in S_i$ e $s_j \in S_j$, sendo d uma métrica e $0 \leq i, j < c$ ($i = j \Rightarrow S_i = S_j$).

Na Figura 3.7 amostramos duas classes no plano com discos cujos raios representam a distância mínima em cada classe (Figura 3.7(a) e Figura 3.7(b)) e com discos que representam a distância mínima entre as classes (Figura 3.7(c)). A construção completa da matriz de raios será detalhada na Subseção 3.3.2.

Para garantir uma boa amostragem em cada classe e durante todo o



3.7(a): Classe 0, raio r_i 3.7(b): Classe 1, raio r_j 3.7(c): Múltiplas classes, raio entre classes r_{ij}

Figura 3.7: Discos representando as distâncias mínimas

processo, a escolha do próximo ponto é sempre feita a partir da classe menos preenchida. Este preenchimento é medido por uma taxa definida como:

$$taxa\_de\_preenchimento(S_i) = \frac{\#S_i}{N_i}, \tag{3-7}$$

onde N_i é o número de pontos estimados para classe i e calculado pela Equação 3-3 aplicada a cada classe i. Já a cardinalidade de S_i ($\#S_i$)é sempre aumentada em uma unidade após a inserção de um novo ponto nesta classe. As taxas de preenchimento devem ser mantidas equilibradas entre as classes, pois o caso contrário pode implicar em não uniformidade da distribuição dos pontos.

3.3.1 Extensão para Superfícies Não Planares

Nossa proposta estende este algoritmo planar do *dart throwing* em múltiplas classes proposto por Wei (22) para domínios não planares como as superfícies trianguladas, a partir das técnicas descritas na Seção 3.2 que apresenta um método mais eficiente do *dart throwing* para as superfícies trianguladas em uma única classe.

Na extensão do algoritmo para múltiplas classes em superfícies é necessário garantir que para cada s_i pertencente a uma célula da grade e para cada s_j pertencente a vizinhança da célula $d(s_i, s_j) \geq r(i, j)$, para qualquer que seja i e j com $0 \leq i, j < c$. Deste modo, o lado de cada célula desta grade deverá medir $\frac{max\{r_0,\ldots,r_{c-1}\}}{\sqrt{3}}$, para que a medida da sua diagonal seja a maior possível, $max\{r_0, \ldots, r_{c-1}\}$, garantindo que seja sempre possível medir as distâncias entre pontos de mesma classe com o maior raio.

Nos *buckets* a variável que antes, no caso de uma única classe, armazenava um único ponto com a propriedade de distância mínima é substituída por um vetor com até c dimensões para o armazenamento dos pontos que satisfazem esta propriedade. Com esta alteração permitimos o armazenamento de até c

pontos aceitos dentro de uma mesma célula.

Do mesmo modo, como apresentado na Seção 3.2, para cada célula válida da grade o nosso algoritmo faz até k tentativas, dentre os pontos aleatórios localizados nesta célula pela partição de O, para amostrar na classe menos preenchida um ponto que satisfaça o critério da distância mínima, como descrito em 3.3, com todos os possíveis pontos de outras classes já amostrados nesta própria célula e com todos os pontos já amostrados por discos de Poisson nas células vizinhas.

A Figura 3.8 mostra uma porção de superfície com pontos aceitos após verificação com o critério da distância mínima. A diagonal das células, neste caso, mede $\frac{r(0,0)}{\sqrt{3}}$.

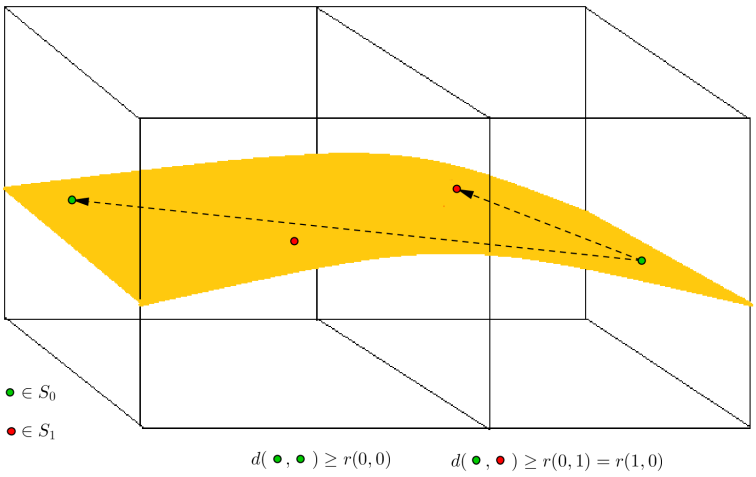


Figura 3.8: Verificação da distância mínima na vizinhança de uma célula

Observe nesta Figura que o critério da distância mínima só será totalmente satisfeito se a diagonal das células for de fato $\frac{max\{r_0,r_1\}}{\sqrt{3}}$, caso contrário o teste nas células vizinhas poderá ser insuficiente.

O gerenciamento da amostragem de pontos fica condicionado a dois casos:

Caso 1 - Uniformidade dos Raios:

O caso em que $r(i,i) = r$ é um número real constante para todas as c classes, $0 \leq i < c$, mantém a uniformidade dos raios. Neste caso, as taxas de preenchimento tendem a ser constantes, pois é acrescentado um ponto após o outro respeitando a preferência da classe menos preenchida, ou seja, inicialmente todas as classes estarão vazias, então é acrescentado um ponto na classe 0, em seguida será inserido na classe 1 respeitando a distância mínima, e assim por diante até chegar na classe $c-1$. Neste passo a taxa de preenchimento

é igual nas c classes e portanto, após a classe $c-1$ é escolhida a classe 0 e o processo recomeça sempre testando as distâncias permitidas para amostragem. Como todas as classes têm o mesmo raio então a quantidade de pontos ao término do processo e a taxa de preenchimento ficam equilibradas.

Caso 2 - Não-Uniformidade dos Raios:

O caso em que $r(i,i) = r_i \neq r_j = r(j,j)$ para algum i e j, com $0 \leq i,j < c$ e $i \neq j$ é dito possuir raios não-uniformes. Neste caso, preencher a classe com a menor taxa de preenchimento pode não ser suficiente para garantir uma distribuição uniforme dos pontos. O processo pode ser incapaz de detectar um novo ponto sem conflito com os existentes e isso pode acontecer bem no início do processo estando ainda longe da uniformidade da distribuição.

Para resolver o problema da insuficiência da taxa de preenchimento Wei (22) propôs um método adicional que consiste na remoção do conjunto de pontos já distribuídos por discos de Poisson, mas que implicam no desequilíbrio das taxas de preenchimento e da quantidade de pontos esperados ao término do processo.

Este conjunto de pontos é definido como $R_{s_j} = \{s_i \ : \ d(s_i, s_j) < r(i,j)\}$ e será removido, para algum $0 \leq i,j < c$, se, e somente se:

1. cada $s_i \in R_{s_j}$ pertence a uma classe i com um raio menor que o da classe j, ou seja, $r(i,i) < r(j,j)$;

2. cada S_i é, pelo menos, tão preenchido quanto S_j, ou seja, a taxa de preenchimento de S_j é menor ou igual a taxa de preenchimento de S_i.

Somente serão removidos os pontos das classes com pequenos raios e que já estão tão preenchidas quanto a classe atual.

Algoritmos

Os algoritmos a seguir resumem as estratégias utilizadas para a amostragem de pontos em múltiplas classes sobre superfície com distribuição por discos de Poisson. O Algoritmo 1 descreve a condição de remoção de pontos, proposta por Wei (22) e o Algoritmo 2 descreve a nossa extensão para superfícies.

Algoritmo 1: Removível

Entrada: N_s, s, r
Saída: *verdadeiro*
inicio
  enquanto *cada* $s' \in N_s$ **faça**
    se $r(cl_{s'}, cl_{s'}) \geq r(cl_s, cl_s)$ *ou*
    $taxa\_de\_preenchimento(S_{cl_{s'}}) < taxa\_de\_preenchimento(S_{cl_s})$
    então
      retorne $falso$;
  fim
fin

3.3.2 Construção da r_matriz

Os raios definidos para cada classe são armazenados na diagonal principal da r_matriz, mas ainda é necessário definir como serão as entradas fora da diagonal principal ($r(i, j)$, para $i \neq j$), ou seja, como calcular a distância mínima entre pontos de classes distintas. Nesta Subseção apresentaremos como é realizado este cálculo e o algoritmo para a criação da r_matriz proposto por Wei (22). Observamos que este algoritmo é bem aplicado à nossa extensão em superfícies.

Inicialmente é importante notar que assim como para domínios planares, não é suficiente garantir que cada classe individualmente tenha a propriedade de distribuição por discos de Poisson, isto é, $r(i, j) = 0$ para $i \neq j$, pois a união de todas as classes não manterá a propriedade como mostrado na Figura 3.9.

Do mesmo modo, se fixarmos uma distância mínima entre as classes como a média entre os raios da classe i e da classe j, isto é, $r(i, j) = \frac{r(i,i)+r(j,j)}{2}$, para $i \neq j$ e $0 \leq i, j < c$ as amostragens não preservarão a propriedade requerida nas classes individualmente, apesar de ser mantida na união das classes como mostrado na Figura 3.10.

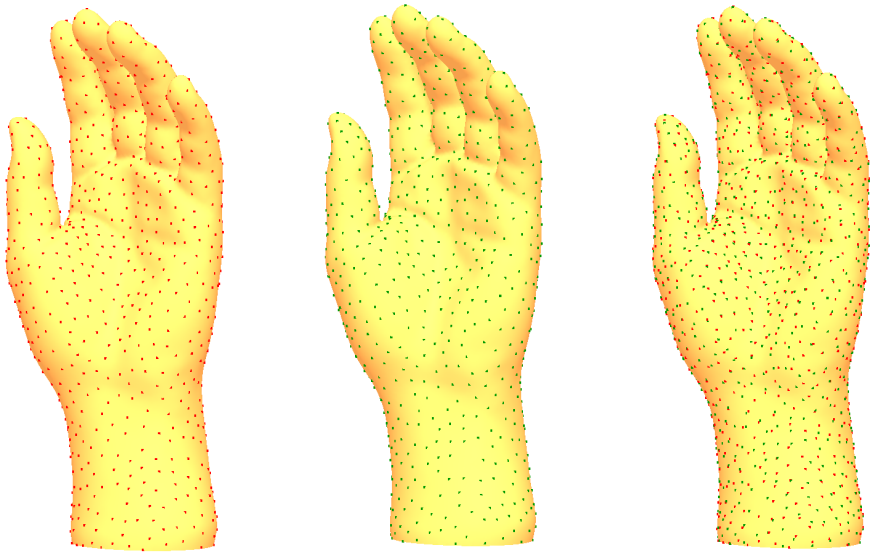
Portanto, é proposto uma distância mínima entre as classes baseada na densidade esperada da amostragem. Para c classes com raios uniformes, o raio entre as classes ($r_{entreClasses}$) pode ser calculado como:

$$\frac{1}{(r_{entreClasses})^2} = \sum_{i=0}^{c-1} \frac{1}{r(i,i)^2} \iff r_{entreClasses} = \frac{1}{\sqrt{\sum_{i=0}^{c-1} \frac{1}{r(i,i)^2}}} \tag{3-8}$$

pois a amostragem em cada classe tem densidade proporcional a $\frac{1}{r(i,i)^2}$, donde

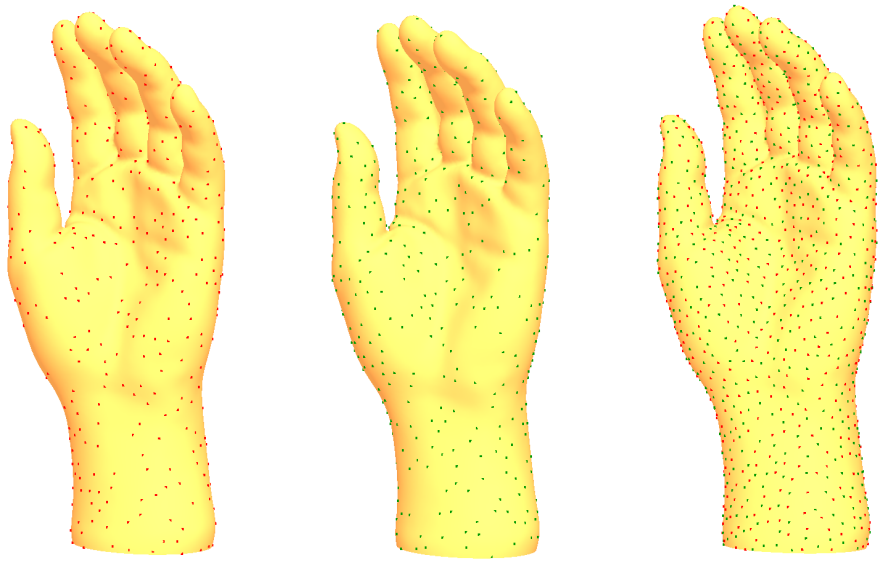
Algoritmo 2: Múltiplas classes por discos de Poisson em superfícies

Entrada: O, r, k
Saída: S
inicio
  $\alpha \longleftarrow max\{r(i,i) : \ \forall i,\ 0 \leq i \leq c-1\}$;
  $\mu \longleftarrow \frac{\alpha}{\sqrt{3}}$;
  $S \longleftarrow \{\}$;
  $N_s \longleftarrow \{\}$;
  $box \longleftarrow BoundingBox(O)$;
  $\{C\} \longleftarrow \left[\frac{box.max - box.min}{3}\right]^3$;
  $hashtable \longleftarrow InicializaHashtable(O, C)$;
  $t \longleftarrow 1$;
  enquanto *id* $v \in V$ **faça**
    //cada celula valida *id* v ;
    $celula \longleftarrow hashtable.busca(v)$;
    enquanto $t \leq k$ **faça**
      se $O[celula.P\_UnifRand\_id + t].celula.id \neq celula.id$
      então
        pare ;
      $s \longleftarrow O[celula.P\_UnifRand\_id + t]$;
      //classe menos preenchida
      $cl_s \longleftarrow min\_taxa\_de\_preenchimento(S)$;
      $conflito \longleftarrow falso$;
      enquanto *id* $v_j \in V$ **faça**
        //cada celula vizinha *id* $v_j \ \in \ V$;
        se $celula_j \longleftarrow hashtable.busca(v_j) \neq NULA$ **então**
          se $\forall s' \in celula_j,\ s' \neq s,\ d(s,s') < r(cl_s, cl_{s'})$ **então**
            $conflito \longleftarrow verdadeiro$;
            $N_s \longleftarrow N_s \cup \{s'\}$;
      fim
      se *conflito* == *verdadeiro* **então**
        se *Removivel(N_s, s, r)* **então**
          remove N_s *de* S ;
          $S \longleftarrow S \cup \{s\}$;
          pare ;
      senão
        $S \longleftarrow S \cup \{s\}$;
        pare ;
      fim
      $t++$;
    fim
  fim
fin



3.9(a): $r(1,1) = 0.041$ 3.9(b): $r(2,2) = 0.041$ 3.9(c): $r(1,2) = r(2,1) = 0$

Figura 3.9: Discos de Poisson somente por classes (3.9(a) e 3.9(b))



3.10(a): $r(0,0) = 0.041$ 3.10(b): $r(1,1) = 0.041$ 3.10(c): $r(0,1) = r(1,0) = 0.041$

Figura 3.10: Discos de Poisson somente na união das classes (3.10(c))

a densidade da união de todas as classes será $\sum_{i=0}^{c-1} \frac{1}{r(i,i)^2}$. A Equação 3-8 define um bom método para calcular $r_{entreClasses}$ no caso em que os raios das classes são uniformes. Foi verificado experimentalmente para a nossa amostragem em superfícies que outros valores fixados menores ou maiores que

esse, recaem, respectivamente, nos problemas demonstrados nas Figuras 3.9 e 3.10. Assim, para raios uniformes aplicamos $r(i,j) = r_{entreClasses}$, para todo $i \neq j,\ 0 \leq i,j < c$.

Exibiremos o algoritmo com a construção da r_matriz que satisfaz a ambos os casos, uniforme ou não-uniforme. Para começar, inicializamos a r_matriz preenchendo a sua diagonal principal com os raios de todas as c classes. Em seguida, iremos preencher as entradas fora desta diagonal observando que as classes com raios maiores interferem no resultado final das classes de raios menores. Agrupamos todas as classes com raios idênticos resultando em p grupos G_a $(0 \leq a < p)$ que serão ordenados, em ordem decrescente, pelos raios representantes de cada grupo. Cada grupo G_a será armazenado num grupo C e estes grupos serão visitados nesta ordem decrescente, onde as classes com raios pequenos terão prioridade. Juntamente ao armazenamento de cada G_a armazenamos a densidade total D acumulada e calculada pela Equação 3-8.

Deste modo, para $i \neq j$ onde $i \in G_a$ e $j \in C$ teremos $r(i,j) = \frac{1}{\sqrt{D}}$, onde D é a densidade acumulada. Com isso, calculamos as entradas fora da diagonal principal dando prioridade as classes com raios menores e observando a densidade total acumulada, o que garante em cada grupo G_a e no conjunto dos grupos uma boa amostragem.

A Figura 3.11 esquematiza uma organização feita pela r_matriz. Neste caso acumulamos em D a densidade no grupo G_0 e para cada classe i em G_0 percorremos todas as classes j do conjunto C armazenando $\frac{1}{\sqrt{D}}$ em $r(i,j)$. Em seguida, atualizamos D acumulando a densidade total dos grupos G_0 e G_1 e para cada classe i do grupo G_1 percorremos todas as classes j do grupo C armazenando $\frac{1}{\sqrt{D}}$ em $r(i,j)$. Continuamos assim até chegarmos no grupo G_{p-1}.

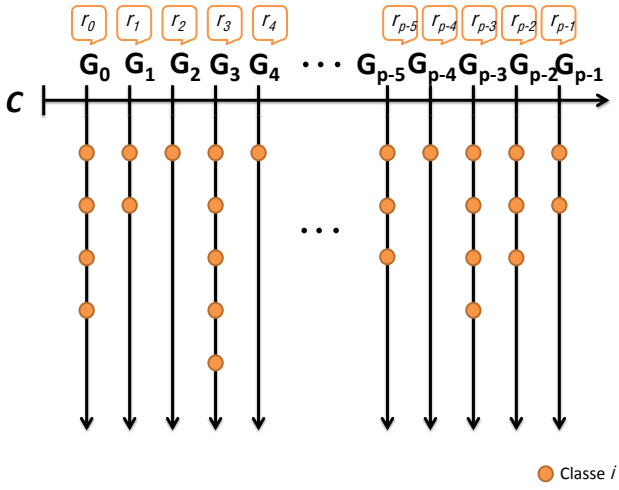


Figura 3.11: r_matriz

Algoritmo

Segue abaixo o Algoritmo 3 que sintetiza a construção da r_matriz proposto por Wei (22). Este algoritmo retorna uma matriz simétrica que armazena o espaçamento mínimo entre pares de pontos a serem amostrados.

O uso desta matriz de raios aliado as técnicas descritas na Seção 3.3, caso 1 e caso 2, nos fornece bons resultados com características no padrão de ruído azul. Estas características serão exploradas no Capítulo 5 onde mostraremos que o nosso método preserva as estatísticas esperadas para este padrão de visualização.

Algoritmo 3: r_matriz

Entrada: $\{r_i\}_{i=0:(c-1)}$
Saída: r
inicio
  //$\{r_i\}$ raios das c classes
  //r é a r_matriz
  enquanto $i < k$ **faça**
    $r(i,i) \longleftarrow r_i$;
    //diagonal da r_matriz
    $i++$;
  fim
  ordene as classes em p grupos $\{G_a\}_{a=0:(p-1)},\ r_i < r_j,\ \forall i,j$;
  //classes de mesmo raio permanecem num mesmo grupo
  $C \longleftarrow \{\}$;
  $D \longleftarrow 0$;
  enquanto $a < p-1$ **faça**
    $C \longleftarrow C \cup G_a$;
    enquanto $i \in G_a$ **faça**
      $D \longleftarrow D + \frac{1}{r_i^2}$;
      //expoente 2 refere-se a dimensão do espaço de amostragem
    fim
    enquanto $i \in G_a$ **faça**
      enquanto $j \in C$ **faça**
        se $i \neq j$ **então**
          $r(i,j) \longleftarrow r(j,i) \longleftarrow \frac{1}{\sqrt{D}}$;
          //r é simétrica
      fim
    fim
    $a++$;
  fim
fin