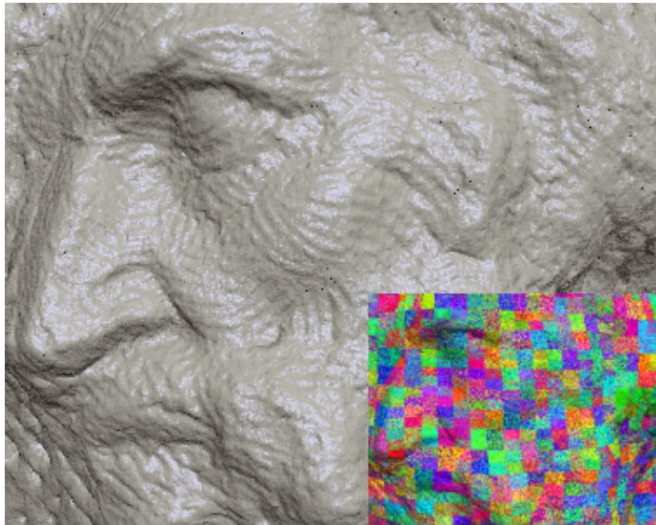


3D GEOMETRIC MODELING & PROCESSING Sampling

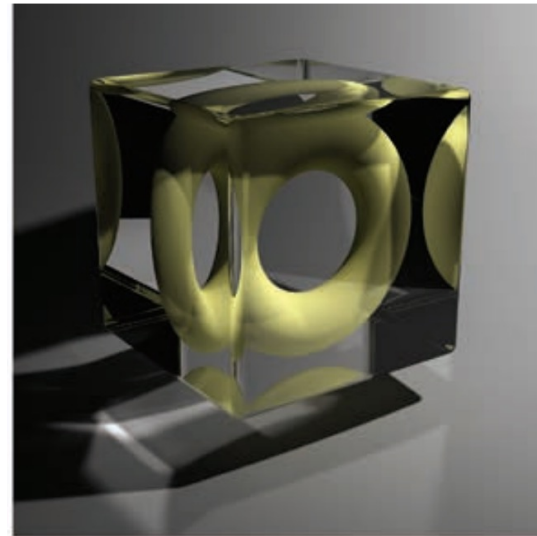
(thanks to Massimiliano Corsini for slides)

Motivations: Rendering

[Layered Point clouds, Gobbetti 2004]

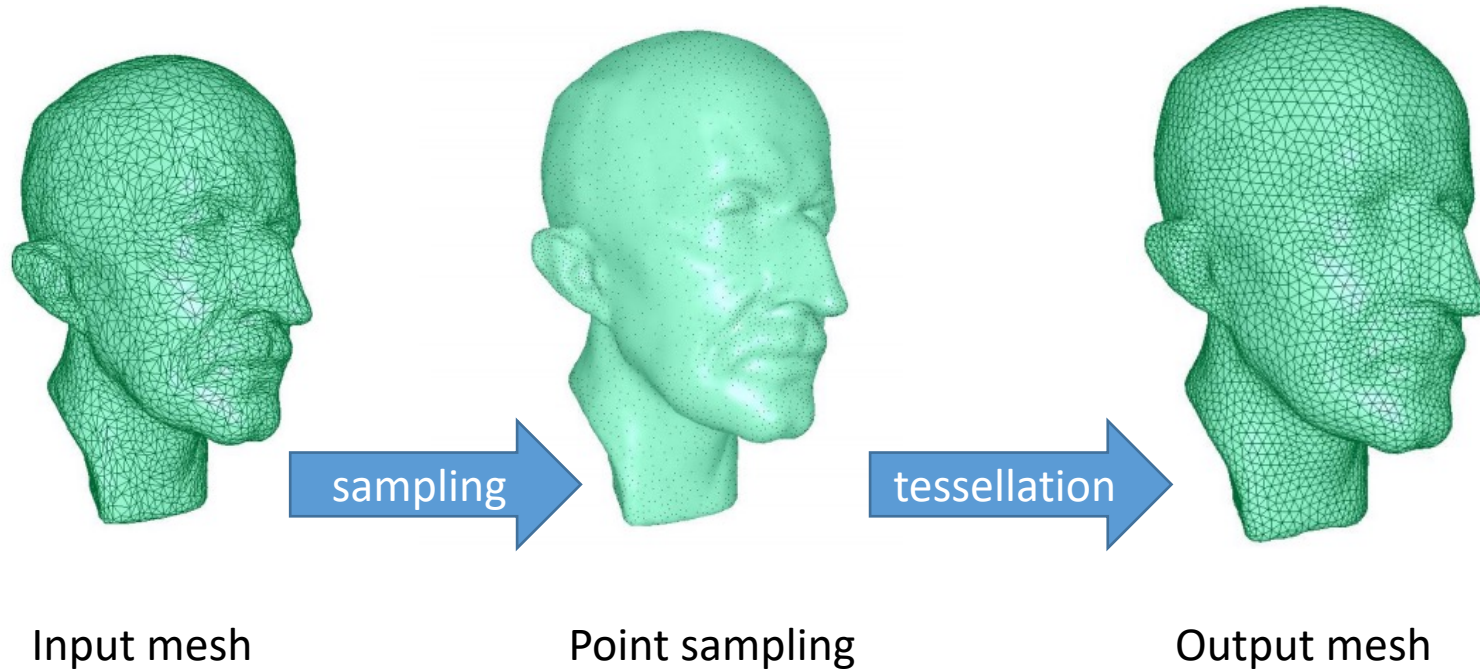


point based rendering



Photorealistic rendering

Motivations: Remeshing

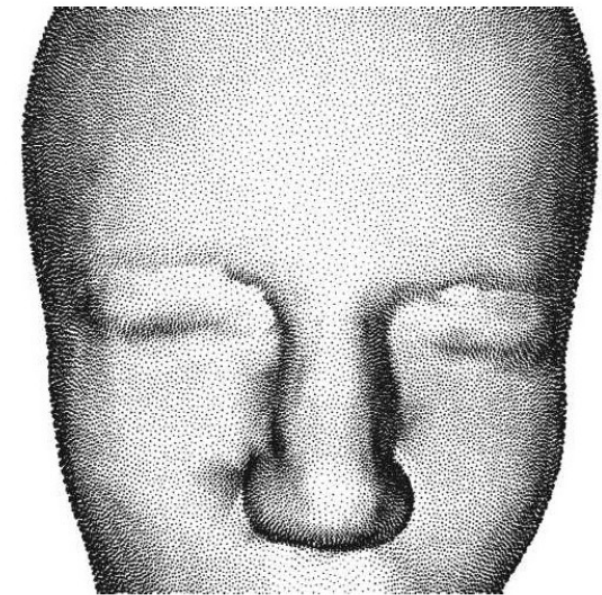
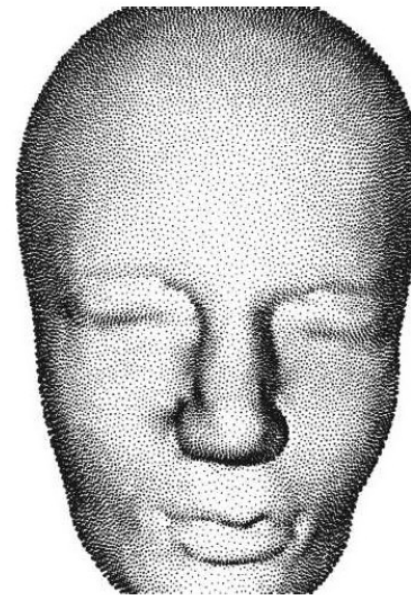
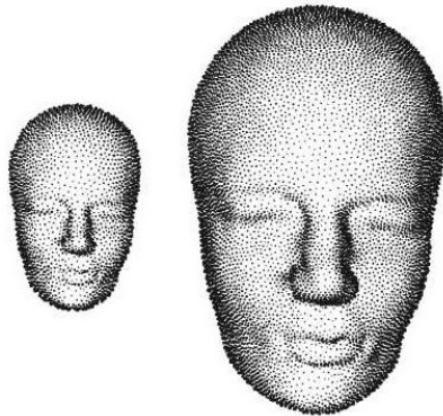


Motivations: Image/Video stippling

[PixelPie: Maximal Poisson-disk Sampling with Rasterization]



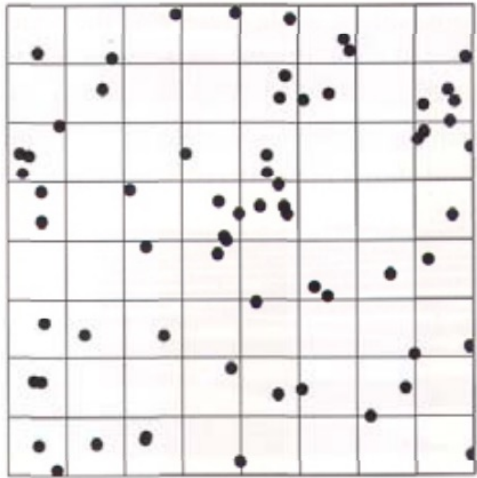
[Instant Stippling]



[Bilateral blue noise sampling: Additional algorithms and applications]

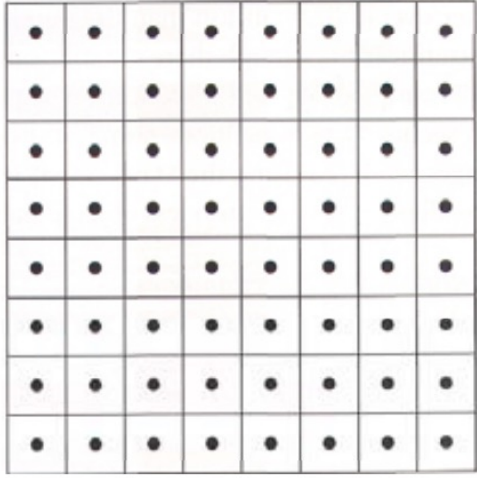
Jittering

Random



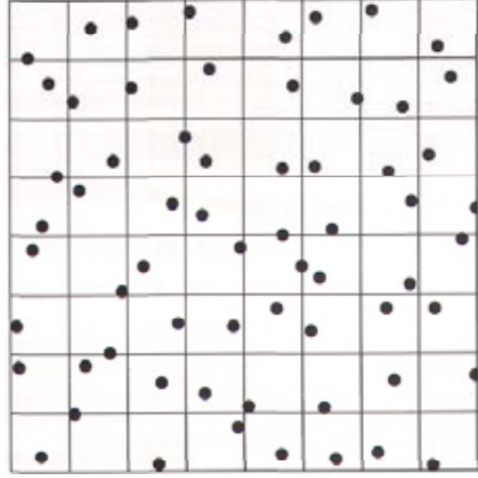
Domain not
uniformly
sampled

Uniform



Domain
uniformly
but not in a
random way

Jittered



Uniform &
random.
Trading
aliasing for
noise

```
S JitteredSampling(){  
  for each Cell in GRID  
    S = S + RandomPointInThe Cell()  
  return S  
}
```

Jittering: from aliasing to noise



256 samples per pixel as reference



1 sample per pixel (no jitter)



1 sample per pixel (jittered)



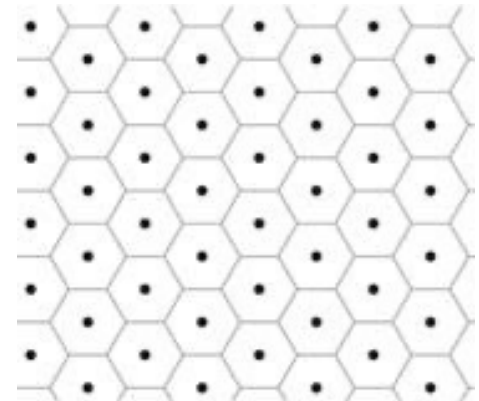
4 samples per pixel (jittered)

How to do sampling

- Characterization
 - Domain: 2D,3D, surfaces..
 - Metric: geodesic, euclidean
 - Shape of the primitive: points, lines, balls
 - Type of algorithms: jittering, dart Throwing, relaxation, Tiling

Poisson Disk Sampling

- A Poisson Disk Sampling is a sampling $X = \{(x_i, r_i) \mid i = 1, \dots, n\}$ such that
 1. Minimal distance: $\forall (x_i, x_j) \in X, \|x_i - x_j\| > \min(r_i, r_j)$
 2. Unbiased sampling: The probability of a region to be covered is proportional to its size
 3. Maximal sampling property: $\Omega \subseteq \cup disk(x_i, r_i)$
- Q: is any PDS a «good» sampling?
- A: No, a maximal PDS sampling (for a given radius) is obtained placing samples at the centers of cells of a hexagonal lattice



PDS: Dart Throwing [cook86]

DartThrowingPDS

```
n_miss = 0
```

```
do{
```

```
    x_cand = RandomPoint()
```

```
    if not x_cand  $\subseteq$  covered(X) // no disk in X contains x_cand
```

```
        X = X + x_cand // hit
```

```
    else
```

```
        n_miss = n_miss +1 // miss
```

```
} while(n_miss / (n_miss +#X) < threshold )
```

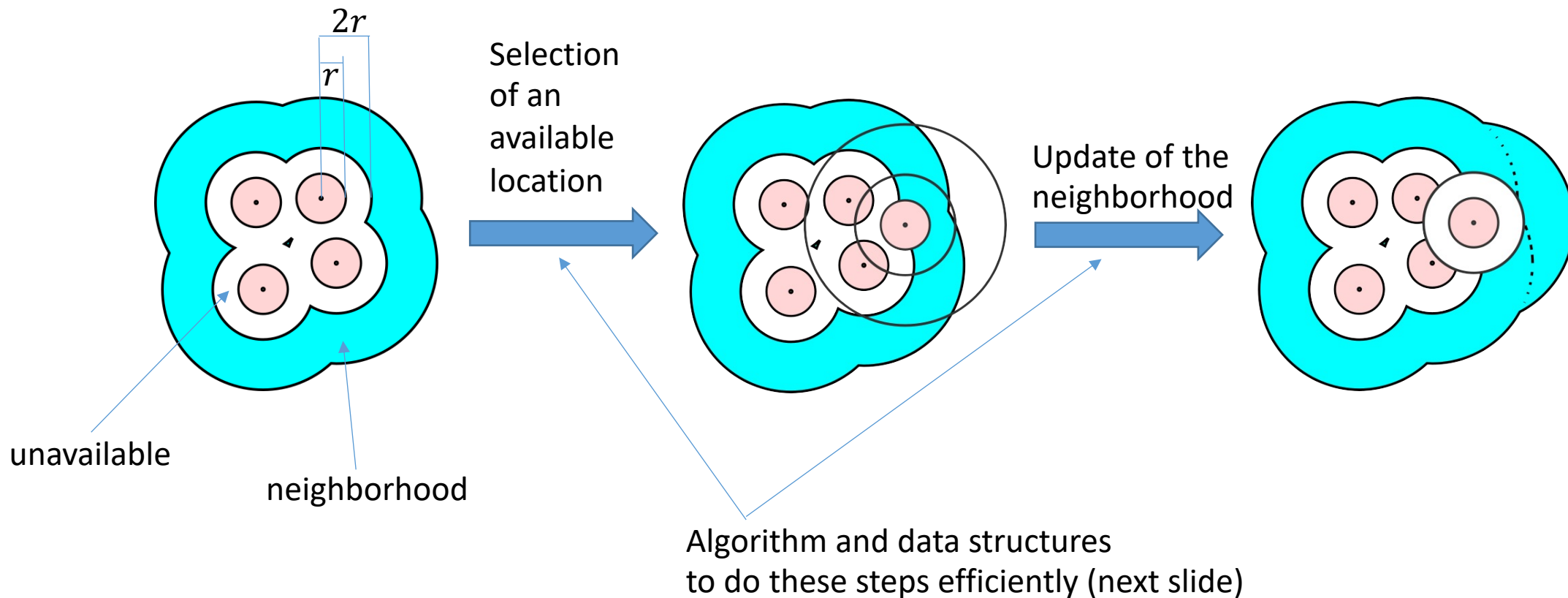
- Works on any domain provided a metric
- Very slow convergence rate, $O(n^2)$ asymptotic complexity
- Maximality not guaranteed in given time (likelability of «hit» tends to 0)
- Many approaches devoted to *efficiency* and *maximality*

Efficiency in PDS algorithms

- Two basic operations that determine convergence speed of a PDS algorithms
 1. choosing a sample location with unbiased probability
 2. Testing if a new location is not already covered

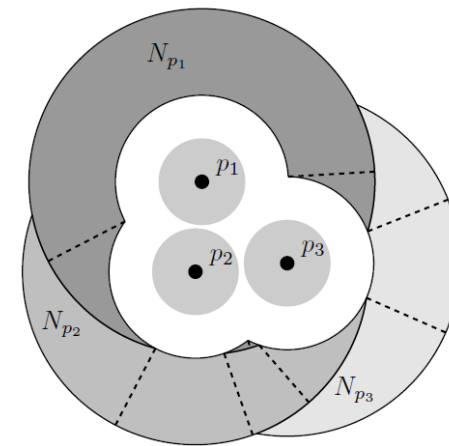
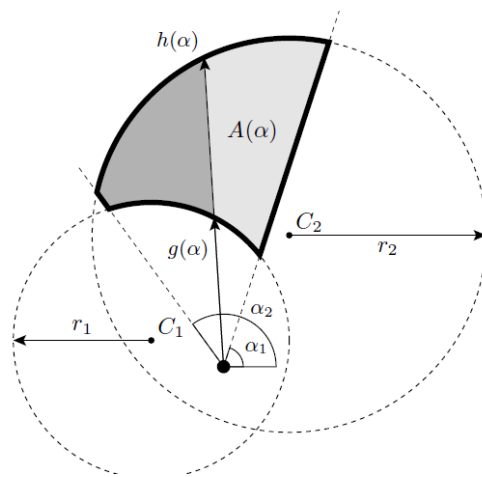
Scalloping [Dunbar06](1/4): Dart Throwing in $O(n \log n)$

- Core idea: if a sampling is not maximal, there must be an *available* location in the neighborhood of the *unavailable* region



Scalloping (2/4): Dart Throwing in $O(n \log n)$

- *Scalloped regions:*
- *Def. Scalloped sector:* a region of the domain bounded by two circular arcs
- *Def: Scalloped region:* a union of scalloped sectors

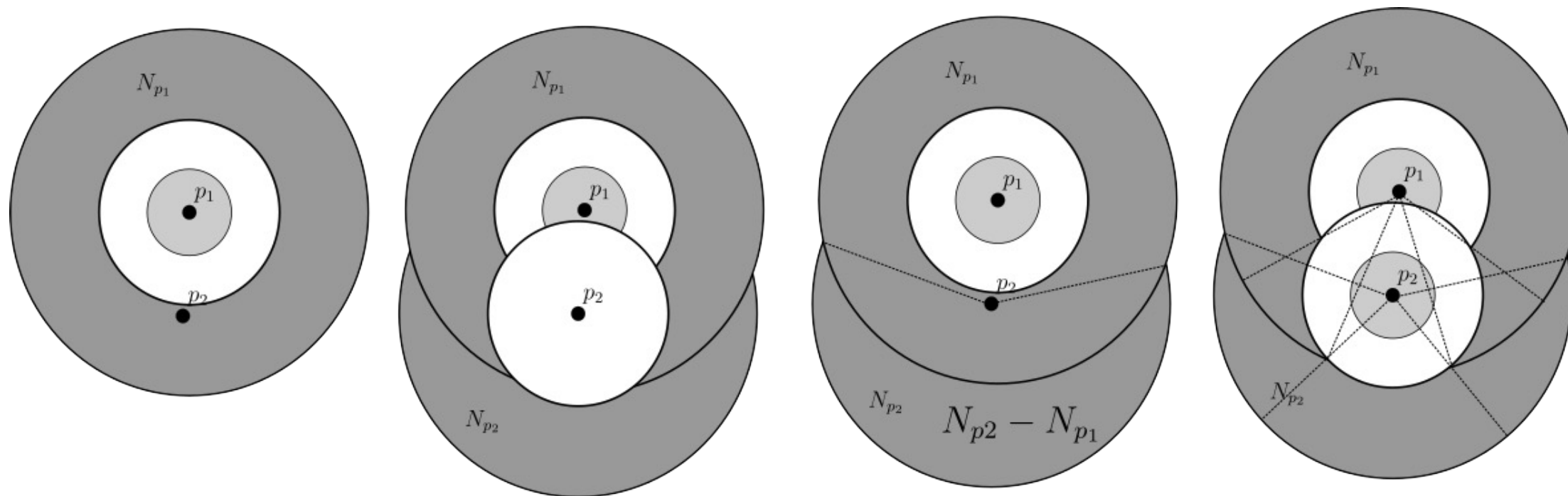


Scalloping (3/4): adding a new disk

Order of insertion

$$N_p = D(p, 4r) - \bigcup_{p' \in P} \begin{cases} D(p', 4r), & p' < p \\ D(p', 2r), & p' \geq p \end{cases}$$

Available neighborhood disk

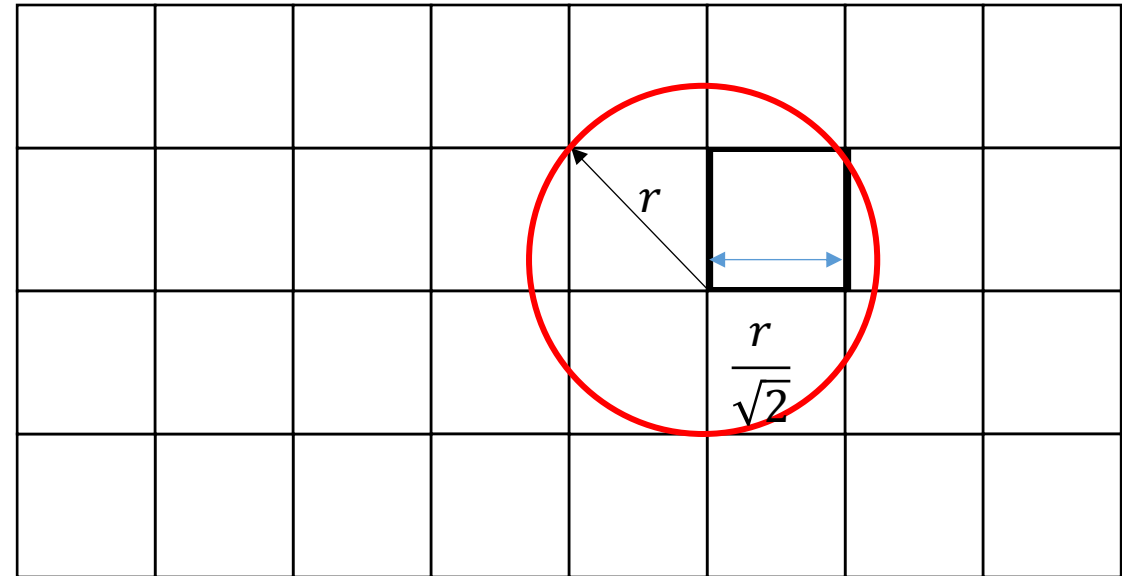


Scalloping (4/4): Speed

- The maximum number of scalloped sectors of r neighborhood is bounded by a constant
- The update of neighborhood is limited to the $4r$ radius from the sample and can be done in $O(1)$
- Unbiased sampling. All available neighborhoods are stored in a balanced tree $O(\log N)$

Hierarchical Dart Throwing [White07] (1/3)

- Regular grid where each cell is the root of a quadtree
- Size of each cell so that it is completely covered by a disk which center is inside the cell
- **def:** *Active cell*
Cell not yet entirely covered by a disk
- **def:** *Active list with index i*
List of active cells at level i
- *Initial condition: Active list L_0 contains all the cells of the grid*



Hierarchical Dart Throwing (2/3)

Hierarchical Dart Throwing

Put base level squares on active list 0 (the base level).

Initialize the point set to be empty.

While there are active squares

 Choose an active square, S , with prob. proportional to area.

 Let i be the index of the active list containing S .

 Remove S from the active lists.

 If S is not covered by a point currently in the point set

 Choose a random point, P , inside square S .

 If P satisfies the minimum distance requirement

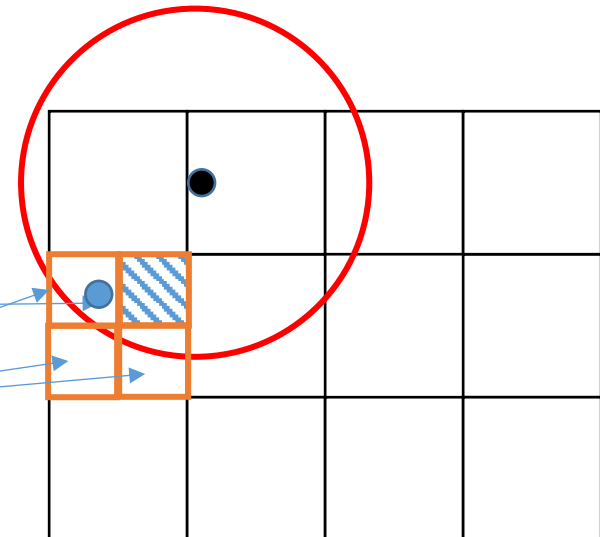
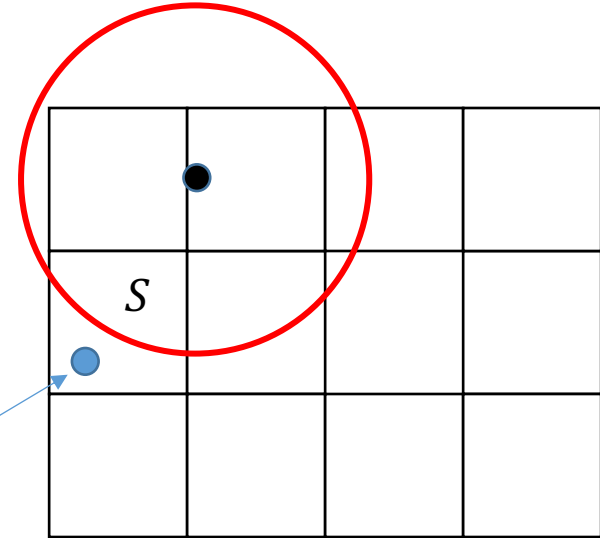
 Add P to the point set.

 Else

 Split S into its four child squares.

 Check each child square to see if it is covered.

 Put each non-covered child of S on active list $i + 1$.

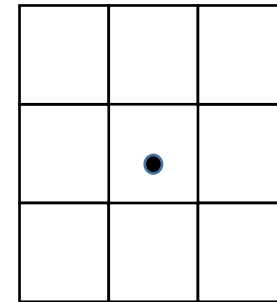


HDT (3/3): Cost

- Coverage test: a secondary uniform grid of cells with size r stores a copy of the point set. By construction, no cell can contain more than 4 points. Therefore, the cost for coverage is constant

- Choice of sample:

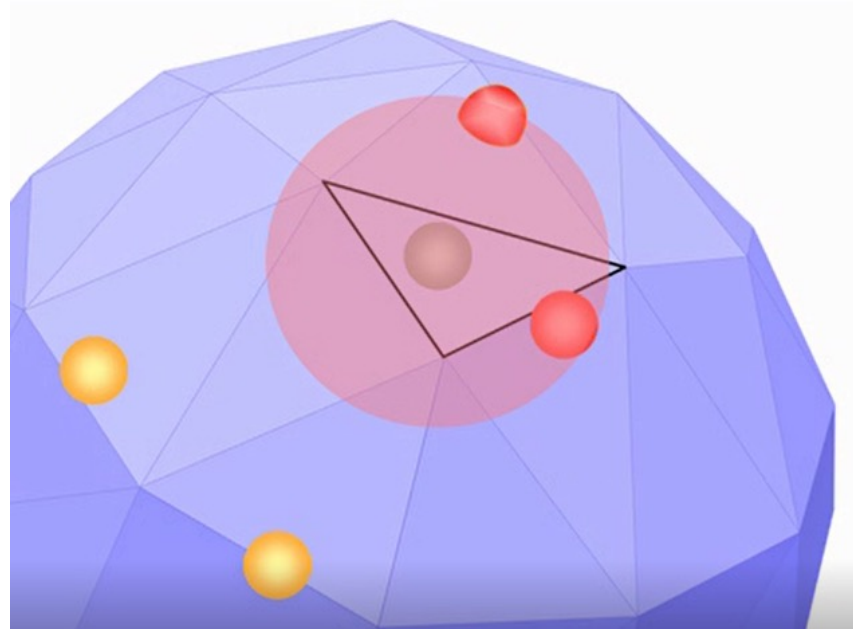
1. keep the sum of areas for each list a_0, \dots, a_k
2. Generate a random number in $[0,1]$
3. Find m s.t. $\sum_{i=0}^{m-1} a_i \leq a_{tot}x < \sum_{i=0}^m a_i$.
4. Pick a random square in the list



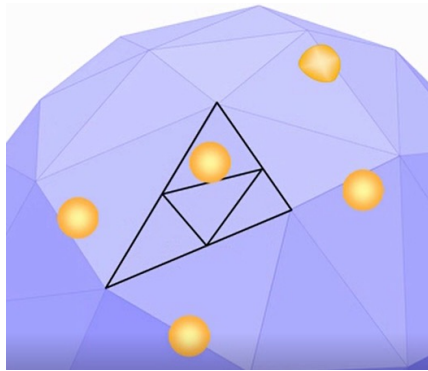
1. Asyntoptic cost: how many cells are created? Authors claim $O(N)$:
«While we cannot provide a rigorous proof, we are convinced that hierarchical dart throwing is $O(N)$ in both space and time on average»

PDS on surfaces [Cline09] (1/6)

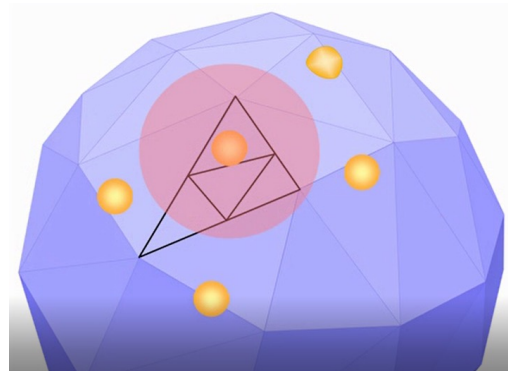
- «Essentially» the same as HDT
 - Replace «uniform grid» with «triangulation»
 - Replace «quadtree» with 1-4 triangle subdivision



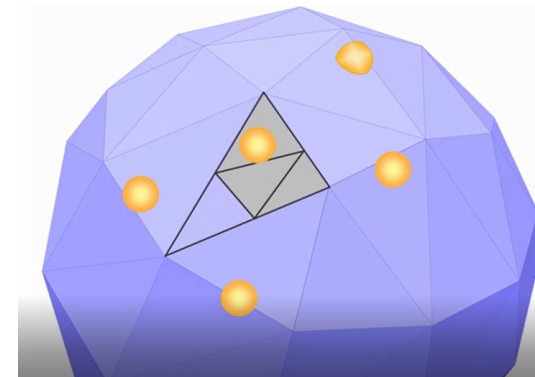
PDS on surfaces [Cline09] (2/6)



Pick a point randomly,
check that it is not closer
than r to any other
point



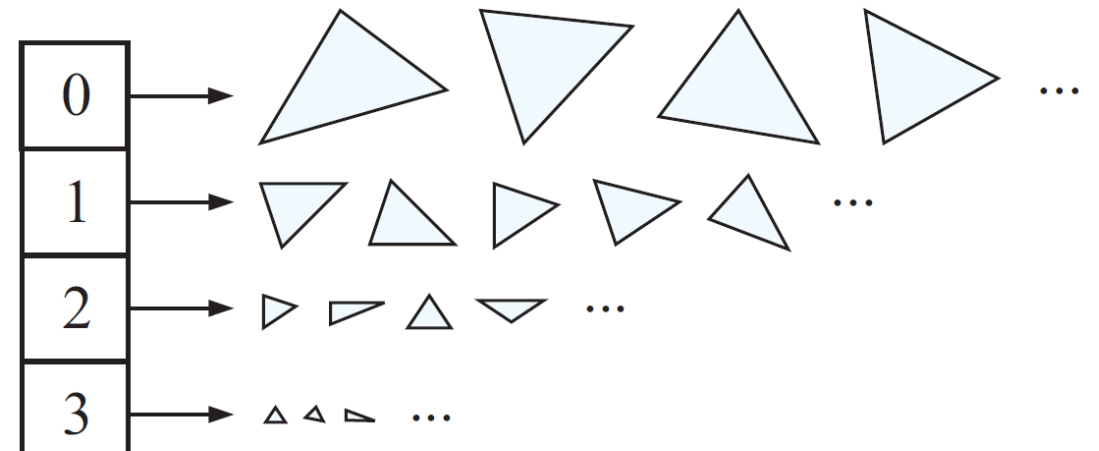
until the triangle
containing the point is
not entirely within
distance r , iterate 1-4
splitting



Remove entirely
covered triangles from
the active list.
End when the list is
empty.

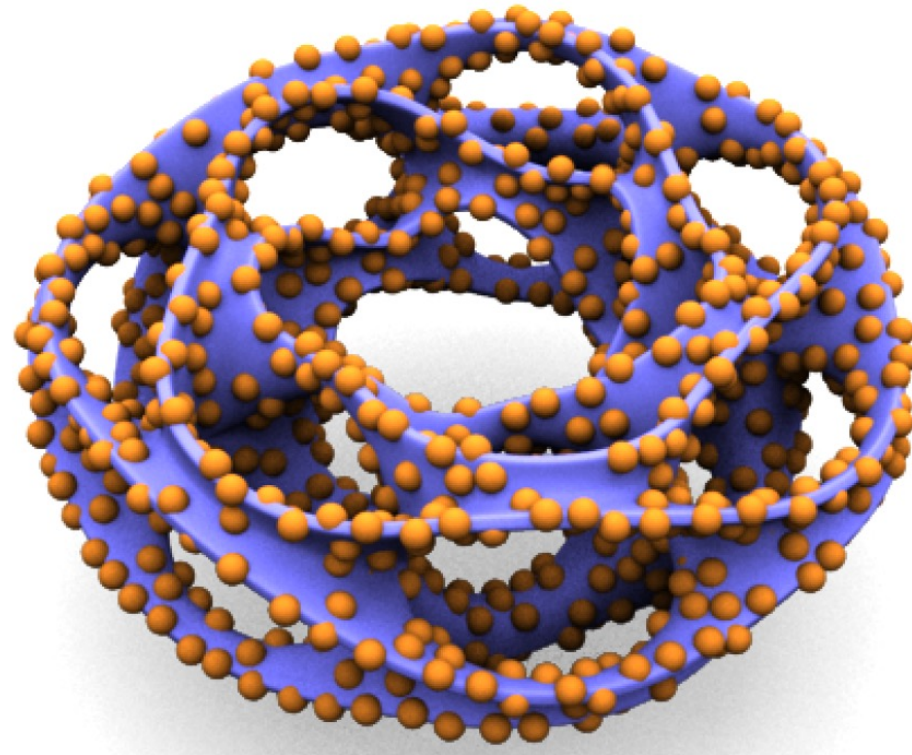
PDS on surfaces [Cline09] (3/6): Cost

- Coverage test. Constant like in [White07]
- Unbiased sampling: Logarithmic binning
 - each bin store a list of triangles within a range area
 1. $a = \text{random value in } [0, \text{total_area}]$
 2. Linear search on the bins until $\text{sum_b} > a$
 3. Pick a triangle and accept with $\text{prob} = \text{triangle_area} / \text{bin_area}$. Repeat until accept
 4. Pick a random point in the triangle



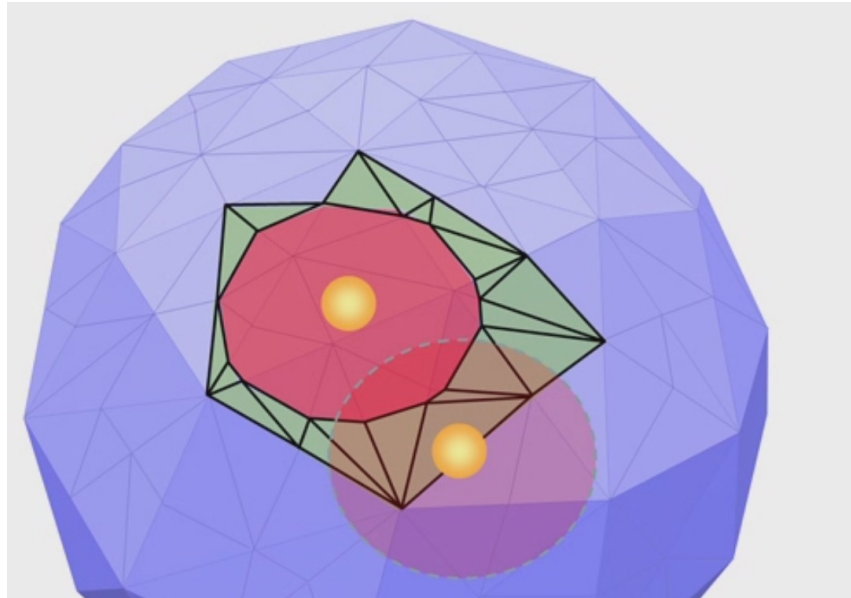
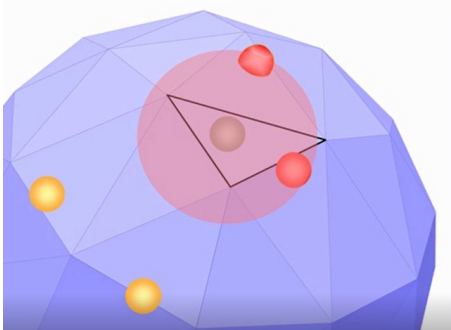
PDS on surfaces [Cline09] (4/6)

- Does it work well?



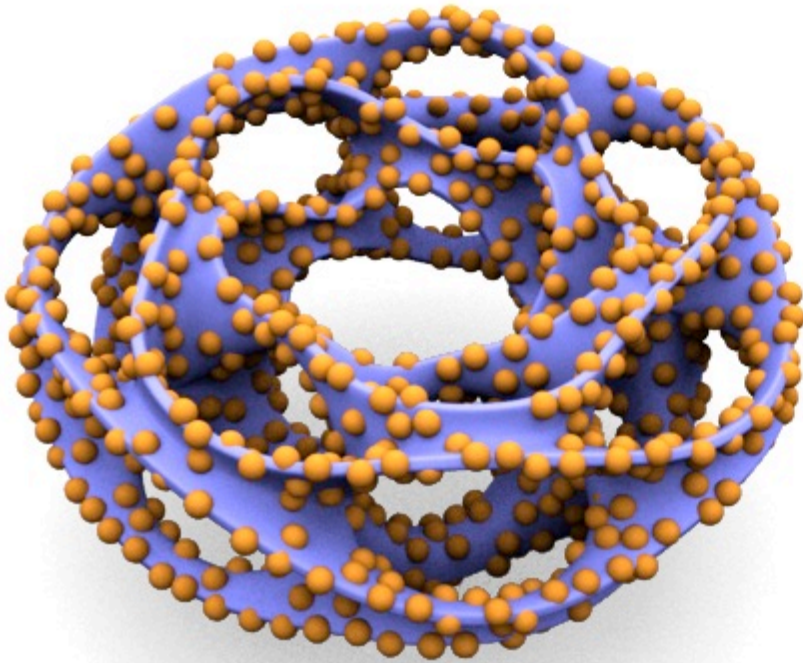
PDS on surfaces [Cline09] (5/6)

- «Essentially» the same as HDT
 - Replace «uniform grid» with «triangulation»
 - Replace «quadtree» with 1-4 triangle subdivision
 - Replace «Euclidean distance» with «geodesic distance»

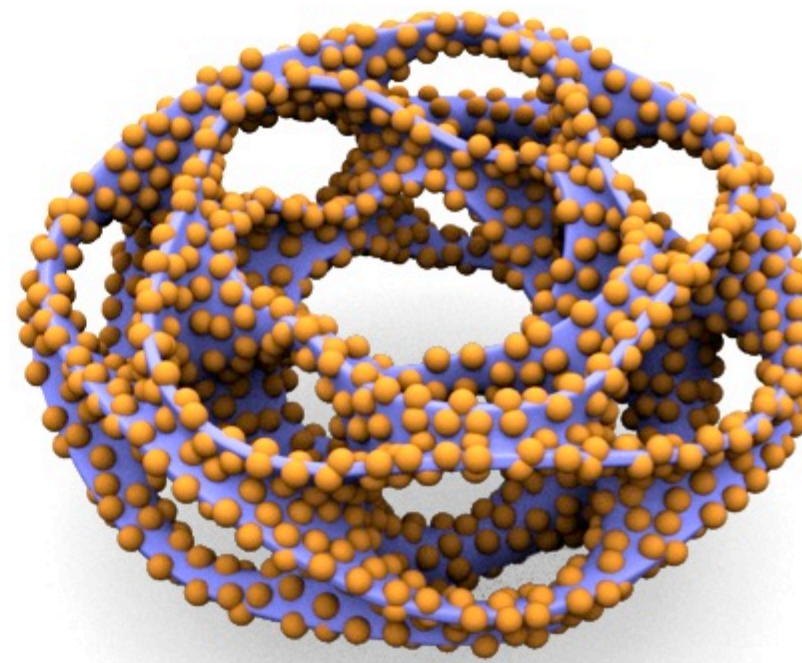


PDS on surfaces (6/6)

Euclidean distance

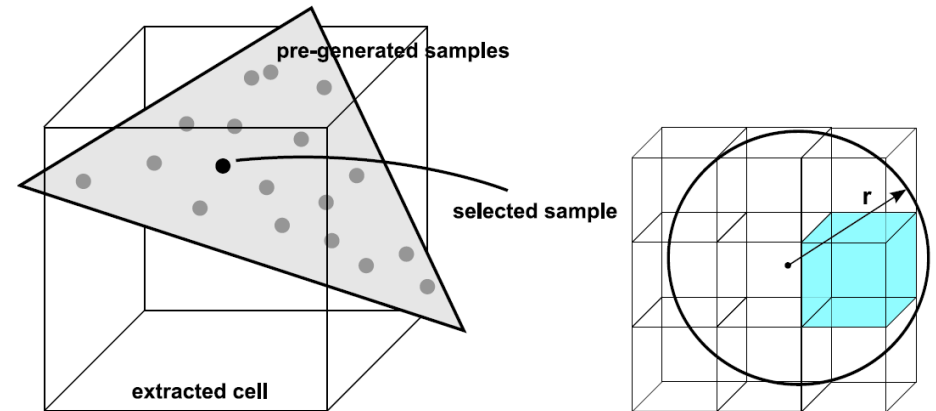


geodesic distance



HDT in 3D space [Corsini12]

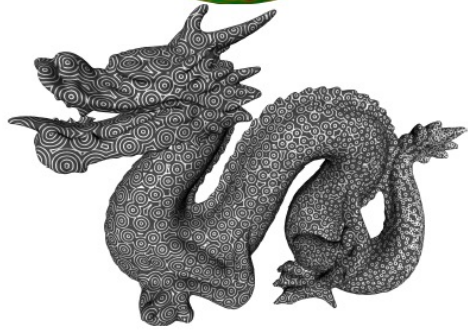
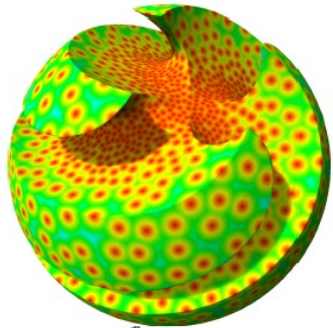
- Surface immersed in a 3D uniform grid where each cell is the root of a oct-tree
- The data structure is a direct extension of HDT except that the cell is **not the** domain but it **contains** the domain (the triangles)
- Basic operation: once an active cell is chosen, pick a point on the contained surface
- Use of *pregenerated samples*



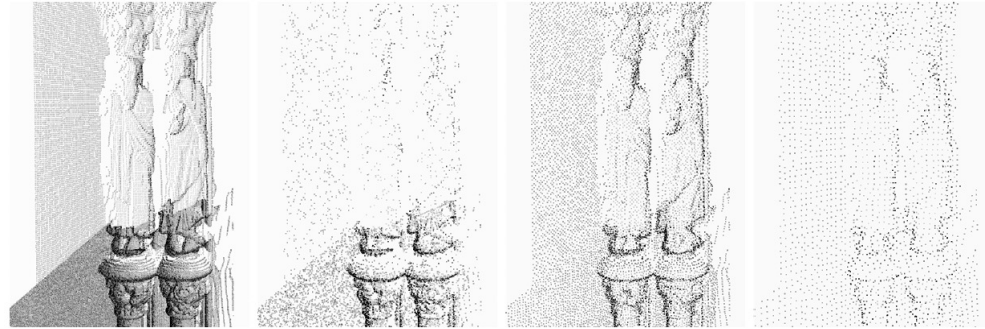
Samples generation

- Variant: just shuffle the initial overampling and iterate point removal. No hierarchy needed anymore!
 1. Generate the pool: a dense sampling of the surface
 2. Until there are available samples
 1. Pick a sample randomly
 2. Remove the sample closer than the disk radius from the pool
- Q: Both versions work well: where is the trick?
- A: the creation of the pool essentially converts the initial «continuous» domain (faces) in a point sampled one. The algorithm cannot guarantee maximality of the sampling up to the intersample distance of the initial pool

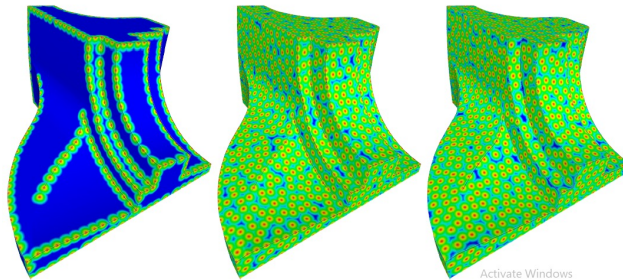
Example uses



Importance sampling / variable radius



Subsampling Point clouds

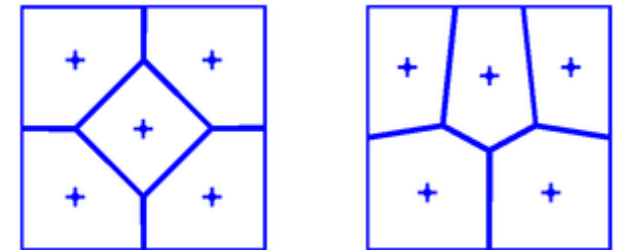


Edge preserving sampling

Relaxation methods

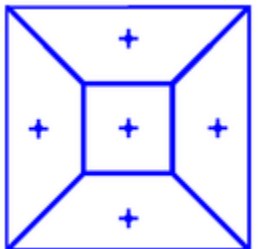
- Methods that iteratively optimize the position of the samples w.r.t. some energy function
- **def:** Centroidal Voronoi Tessellation (Diagram).
A CVT is a VT where each site (point) lies in the centroid of its region:

$$p_i = \frac{\int_{V_i} x \rho(x) dx}{\int_{V_i} \rho(x) dx} \quad \rho(x) \text{ Some density function}$$



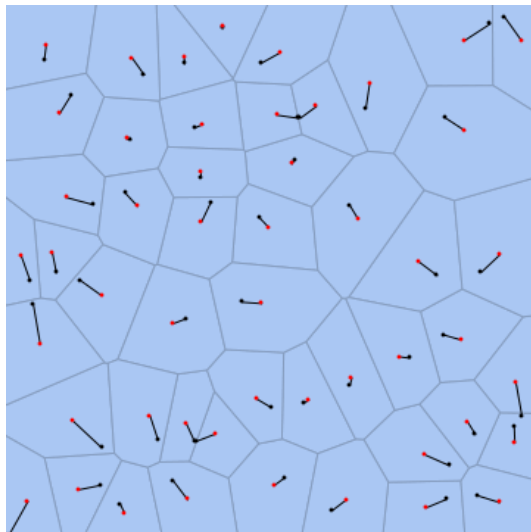
- The minimum of the energy function below is on a CVT

$$\mathcal{F}(S, \mathcal{V}) = \sum_{i=1}^n \int_{V_i} \rho(x) |x - s_i|^2 dx,$$

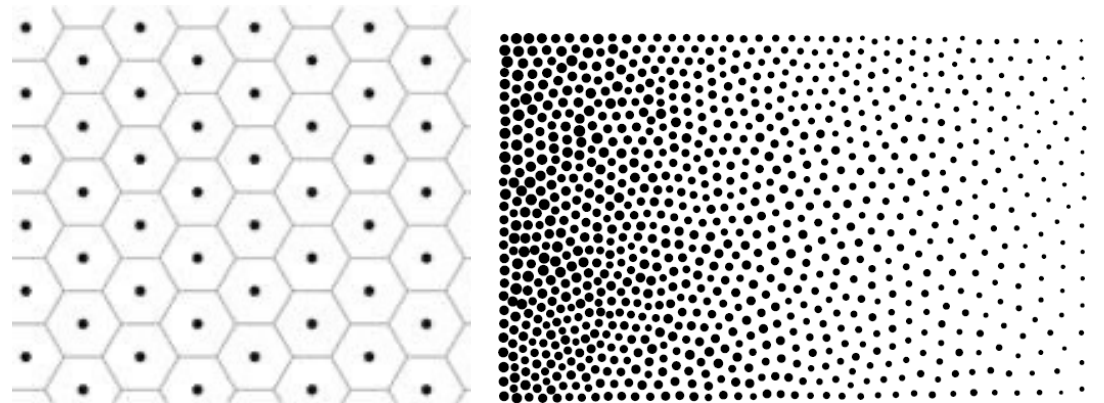


Lloyd's method

1. generate the Voronoi tessellation $V(S)$ in Ω
2. move each site $s_i \in S$ to the centroid p_i of the corresponding Voronoi region $V_i \in V$;
3. if the new sites in S meet some convergence criterion, then terminate; otherwise return to step 1.

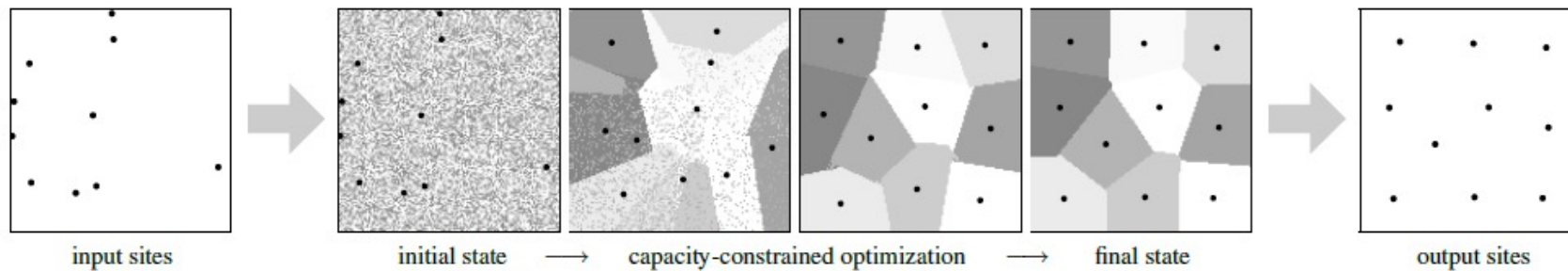


Do CVT necessary lead to a good sampling?

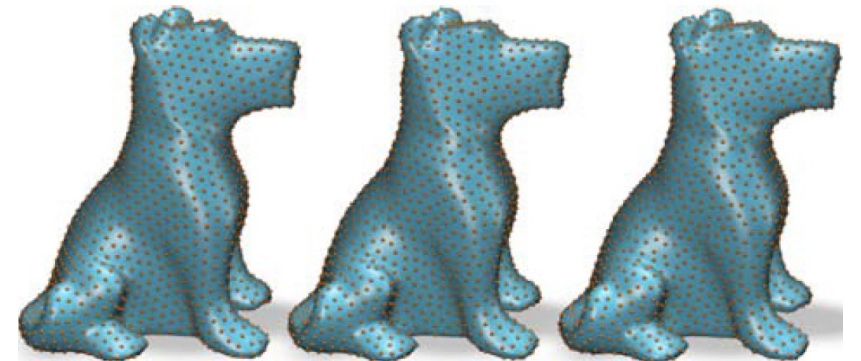


CVT methods

- Methods that modify CVT to obtain better sampling properties
- Capacity Constrained CVT [balzer09] : CVT plus the constraint that each region of the VT has the same area



Extension to surfaces. Capacity constrained Delaunay Triangulation [chen12]



References

- [cook86] R. L. Cook, "Stochastic sampling in computer graphics," ACM Trans. on Graphics, vol. 5, no. 1, 1986
- [Lagae08] A. Lagae and P. Dutre, "A comparison of methods for generating Poisson disk distributions," Computer Graphics Forum, vol. 27, no. 1, 2008
- [Dunbar06] D. Dunbar and G. Humphreys, "A spatial data structure for fast Poisson-disk sample generation," ACM Trans. on Graphics (Proc. SIGGRAPH), vol. 25, no. 3, 2006
- [white07] K. B. White, D. Cline, and P. K. Egbert, "Poisson disk point sets by hierarchical dart throwing," in Proceedings of the IEEE Symposium on Interactive Ray Tracing 2007.
- [cline09] D. Cline, S. Jeschke, A. Razdan, K. White, and P. Wonka, "Dart throwing on surfaces," Computer Graphics Forum (Proc. EGSR), vol. 28, no. 4, 2009.
- [corsini12] M. Corsini, P. Cignoni, and R. Scopigno, "Efficient and exible sampling with blue noise properties of triangular meshes," IEEE Trans. on Vis. and Comp. Graphics, vol. 18, no. 6, 2012
- [balzer09] M. Balzer, T. Schlömer, and O. Deussen, "Capacity-constrained point distributions: A variant of Lloyd's method," ACM Trans. on Graphics (Proc. SIGGRAPH), vol. 28, no. 6, 2009
- [chen12] Z. Chen, Z. Yuan, Y.-K. Choi, L. Liu, and W. Wang, "Variational blue noise sampling," IEEE Trans. on Vis. and Comp. Graphics, vol. 18, no. 10, pp. 1784. 2012.
- A Survey of Blue-Noise Sampling and Its Applications Dong-Ming Yan, Jian-Wei Guo, Bin Wang, Xiao-Peng Zhang & Peter Wonka Journal of Computer Science and Technology volume 30, pages439–452(2015)