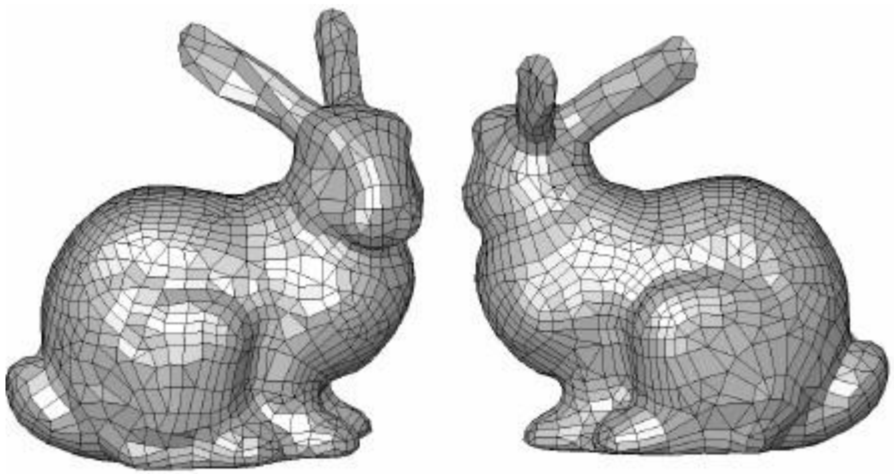
Point Processing & Modeling

Source cours de Jean-Marc Thiery
https://perso.telecom-paristech.fr/jthiery/

Representations de surfaces



Opérations sur les points

- Recherche de voisinage
- Filtrage
- Rendu
- Définition de surface implicite
- Conversion en maillage triangulaire
- Enrichissement
- Recalage
- Édition / manipulation
- Coloriage
- ...

Sources

LIDAR

Batiments, etc

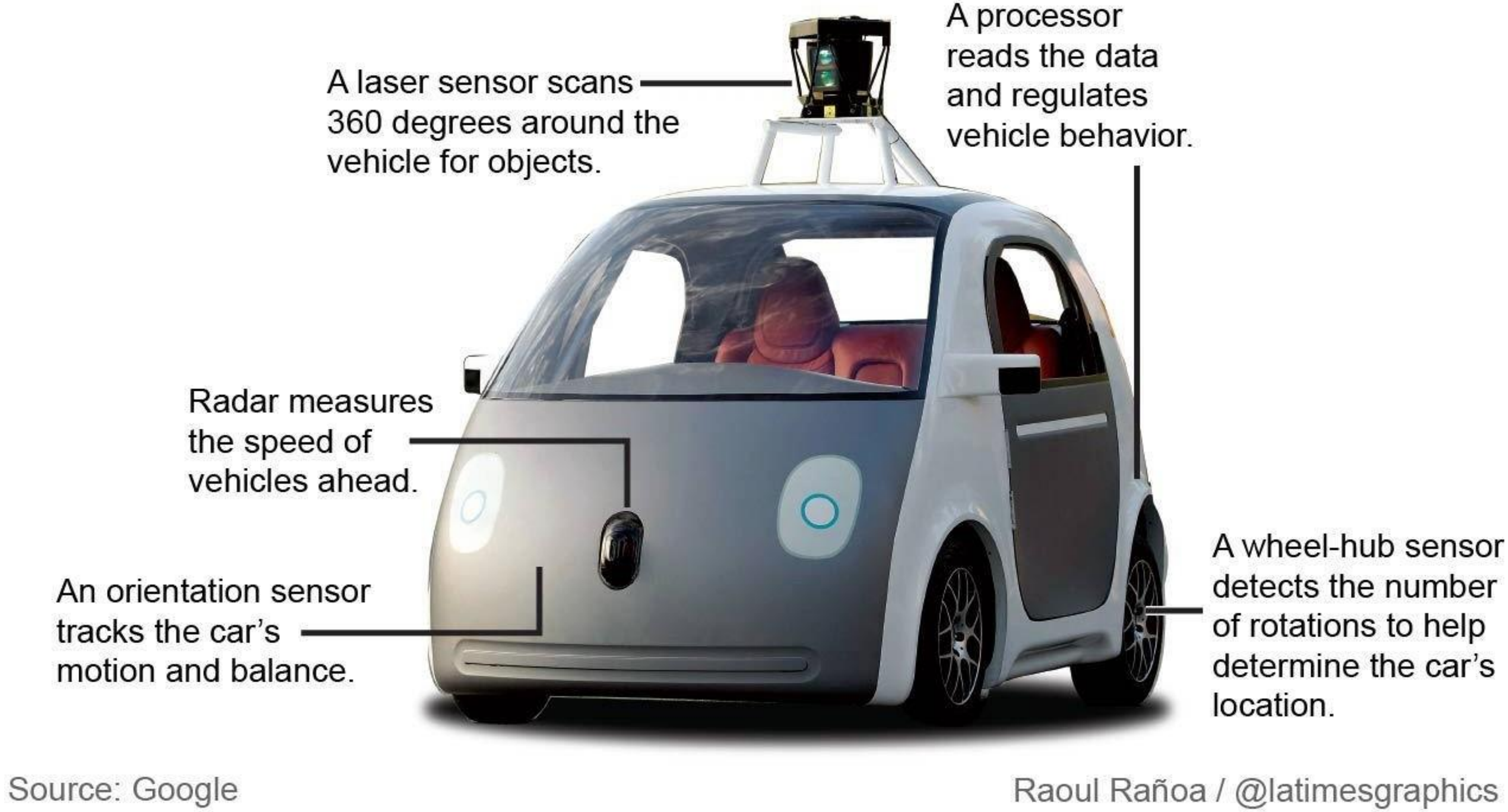
Sources

Table tournantes

Sources

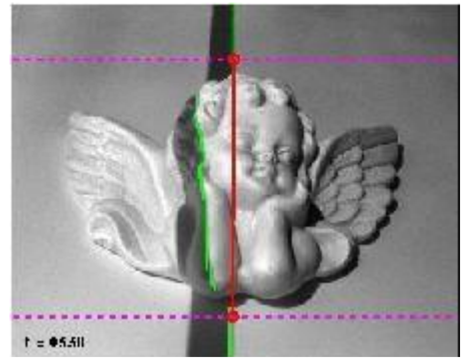
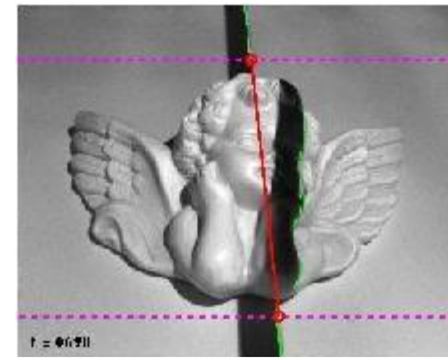
Kinect

Sources

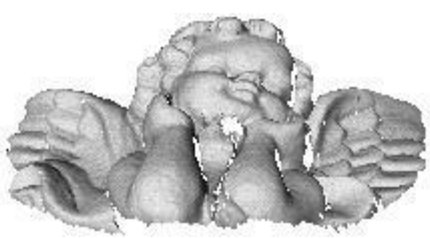


Google car

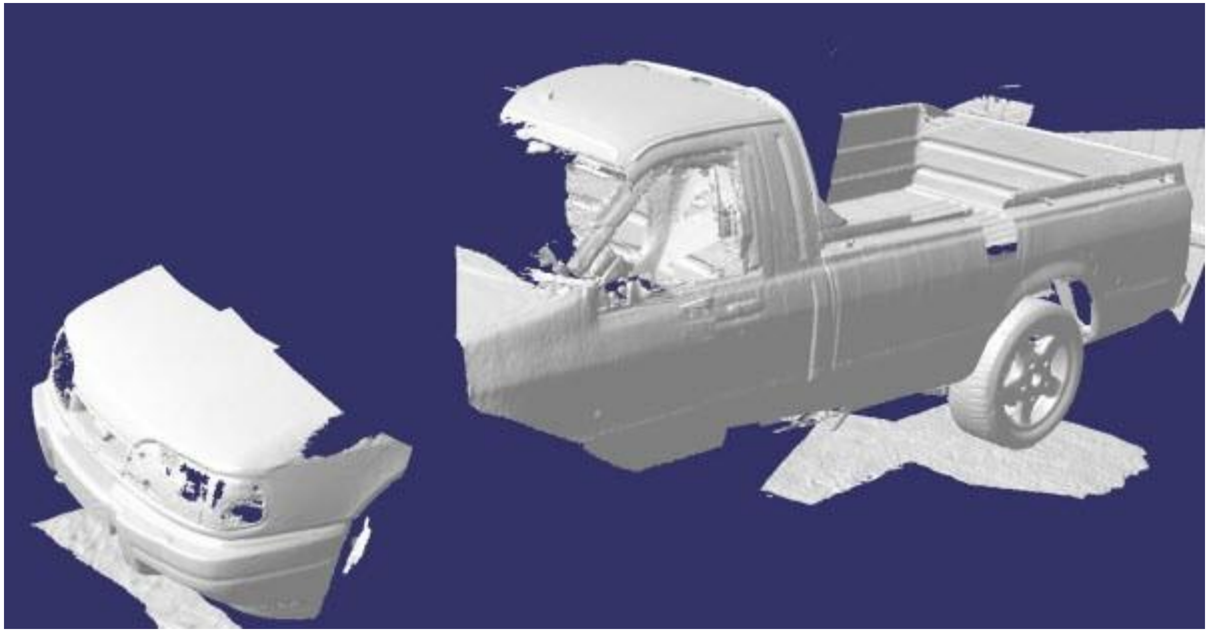
Sources

Système fait avec une lampe et un crayon…
http://www.vision.caltech.edu/bouguetj/ICCV98/



Bruits et propriétés géométriques

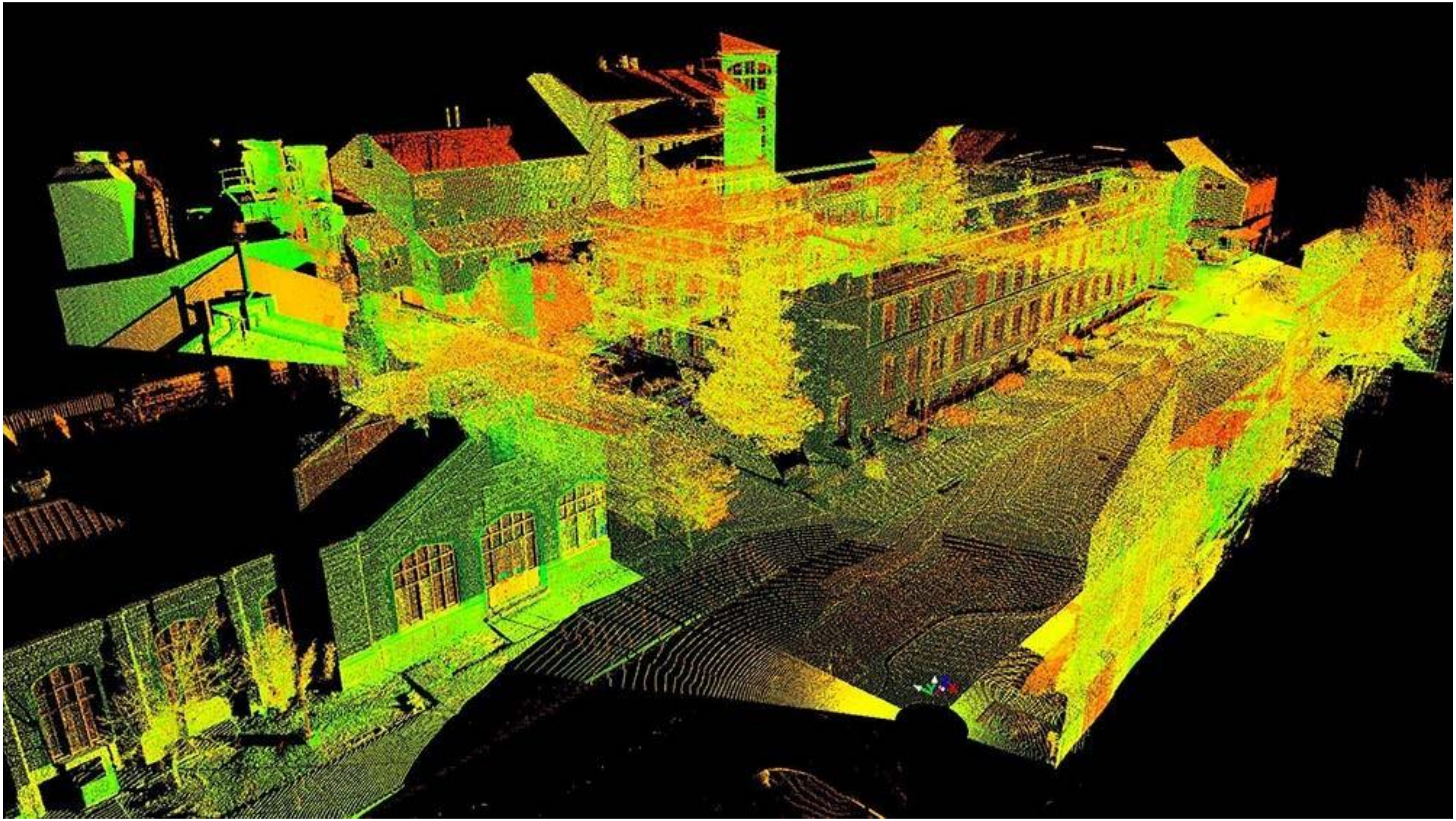


Occlusions, parties transparentes ou réfléchissantes

Bruits et propriétés géométriques

Quantification, popping

Bruits et propriétés géométriques



Direction du scan
http://www.adamdealva.com/

Bruits et propriétés géométriques

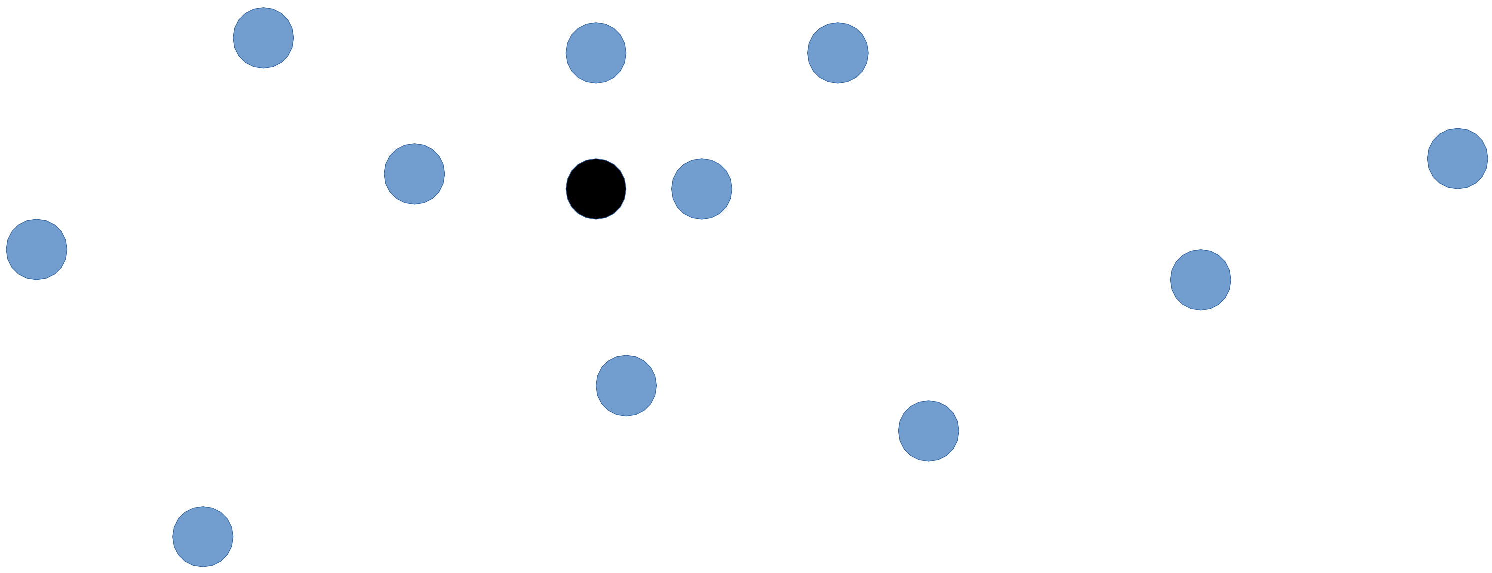
Bruits et propriétés géométriques

Différents types de bruit, différents artifacts

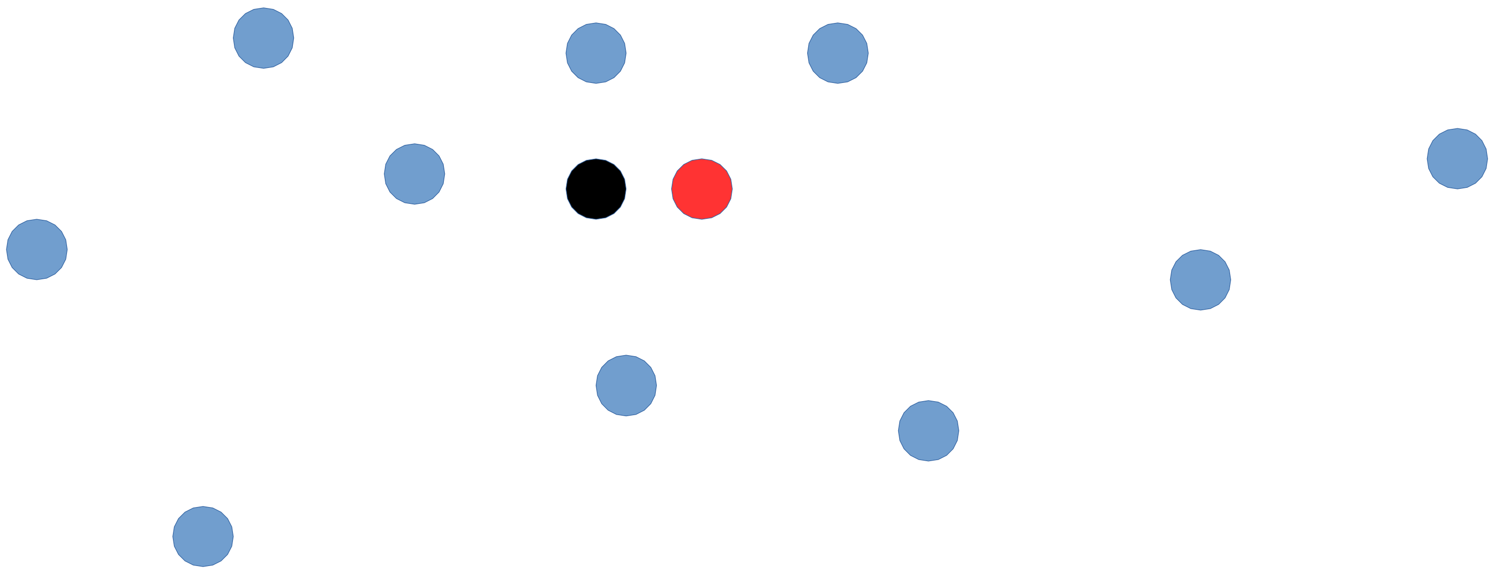
$\longrightarrow$

Différents algorithmes de traitement

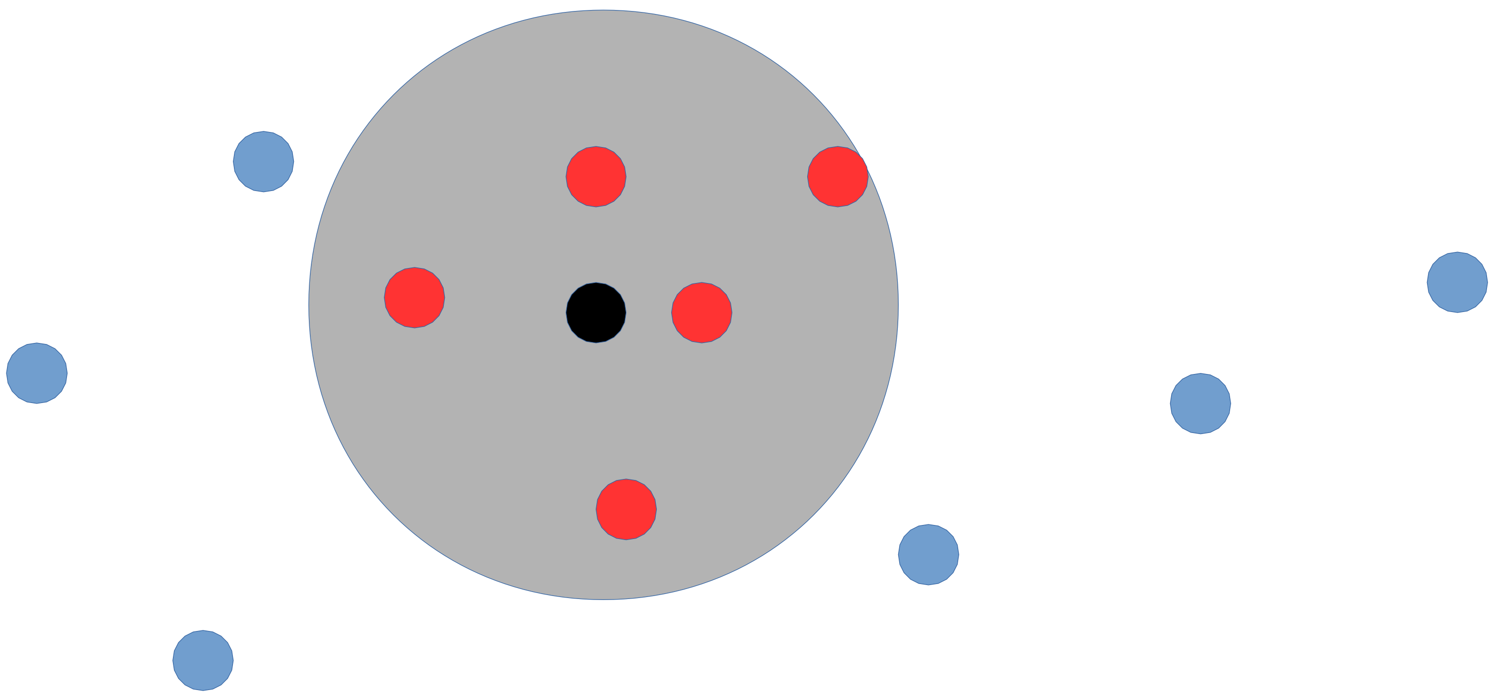
Recherche de voisinages Euclidiens



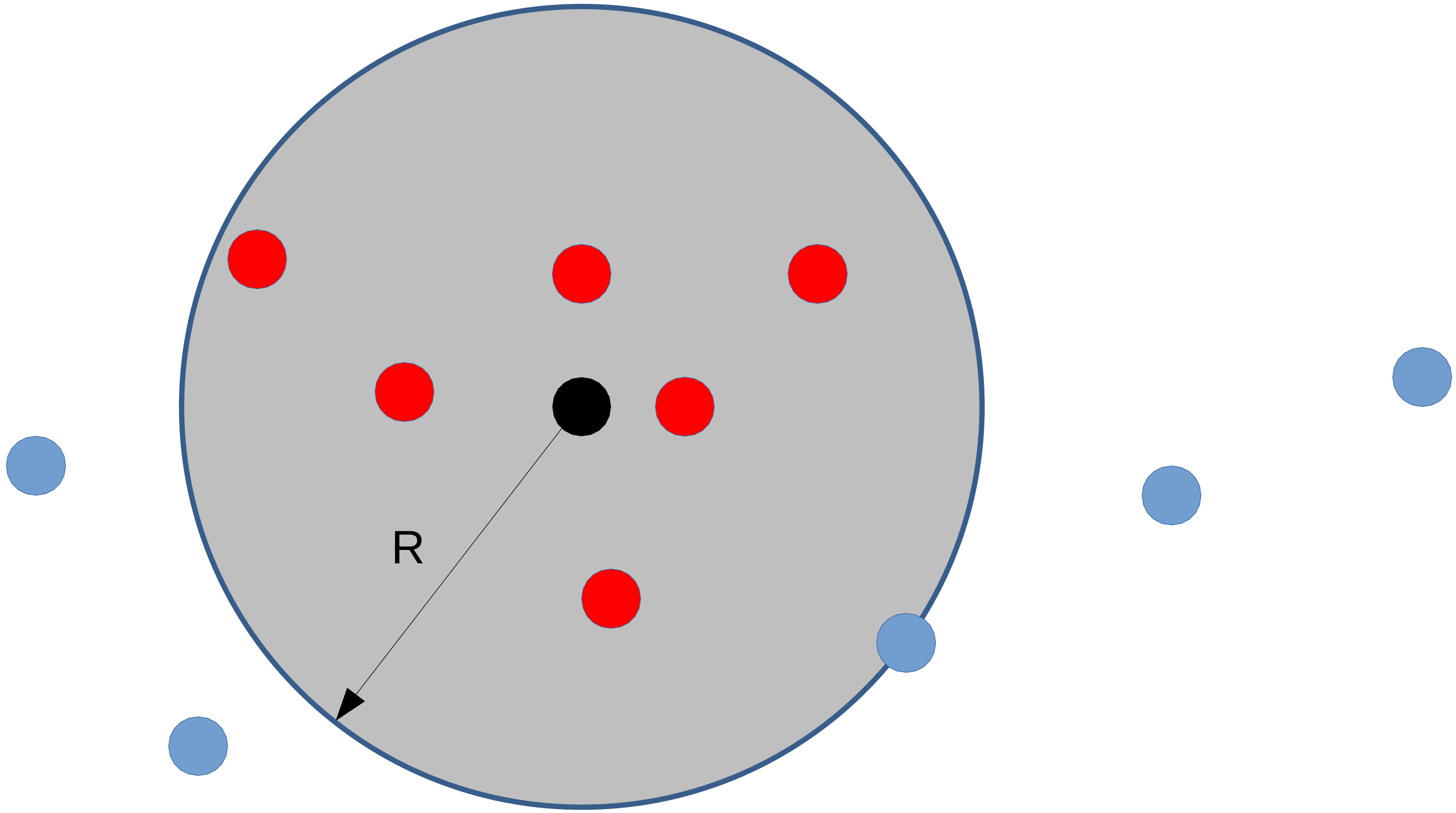
Plus proche voisin



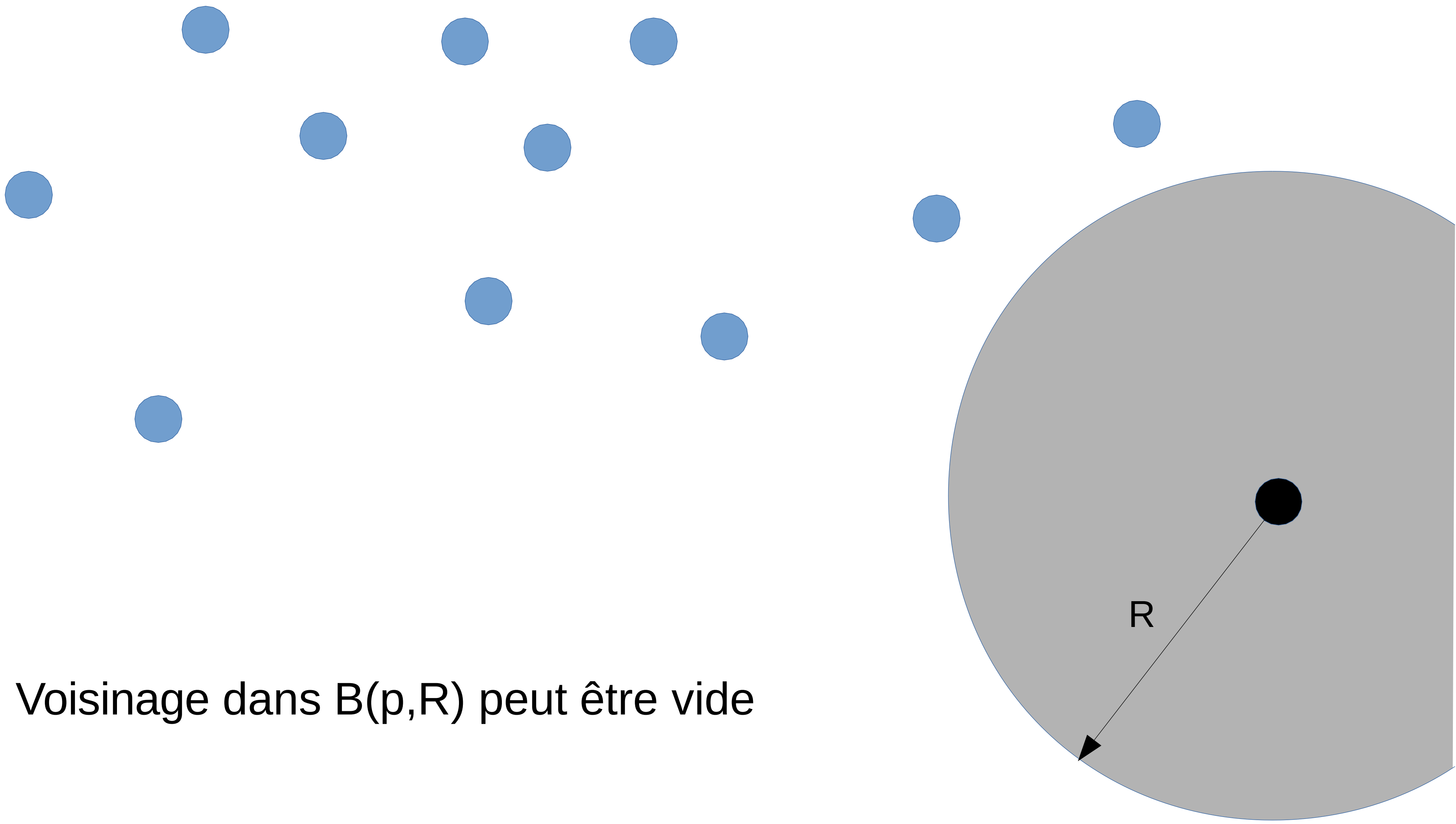
K (ici, 5) plus proches voisins



Voisins a distance < R

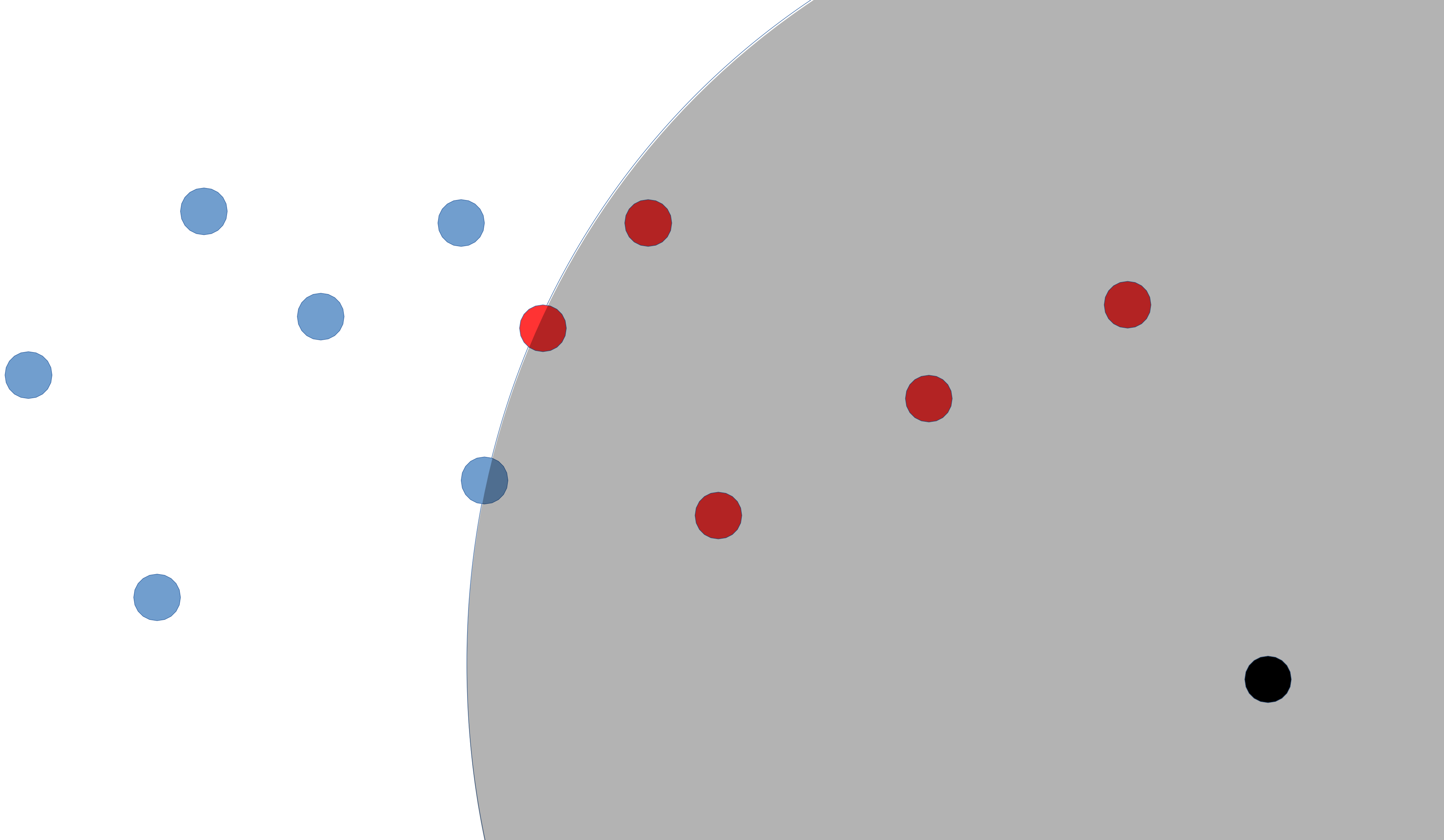


Voisins a distance < R



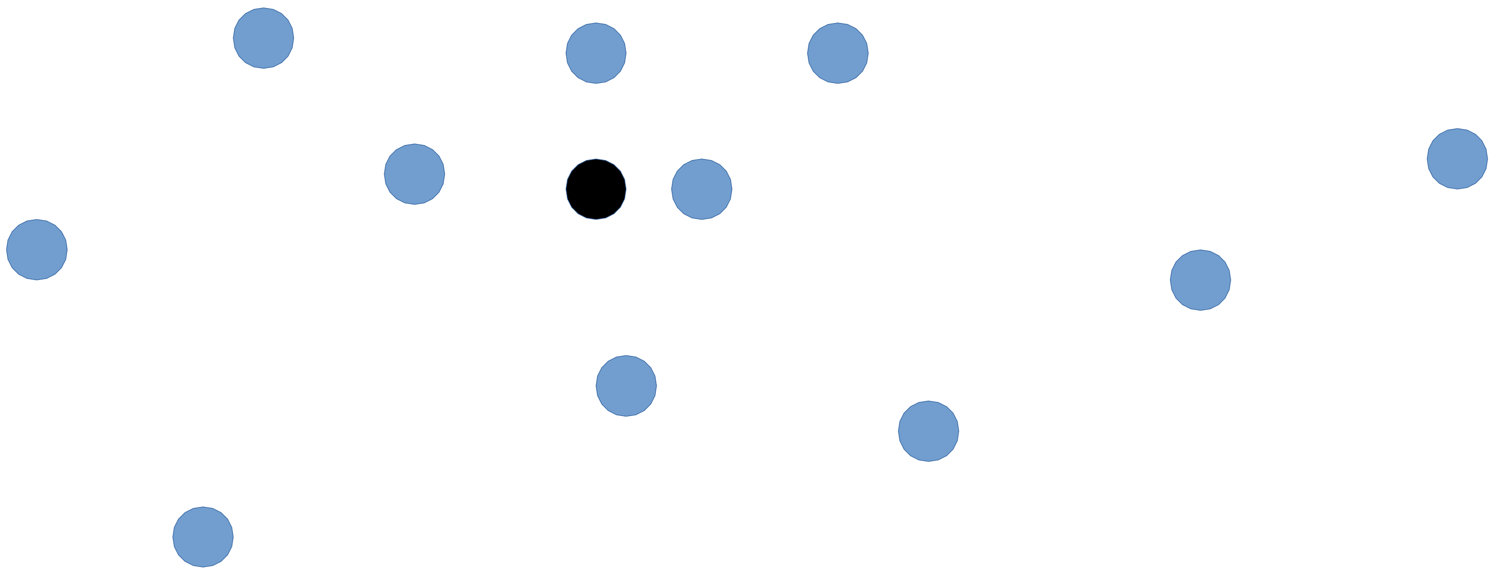
Voisinage dans B(p,R) peut être vide

K (ici, 5) plus proches voisins



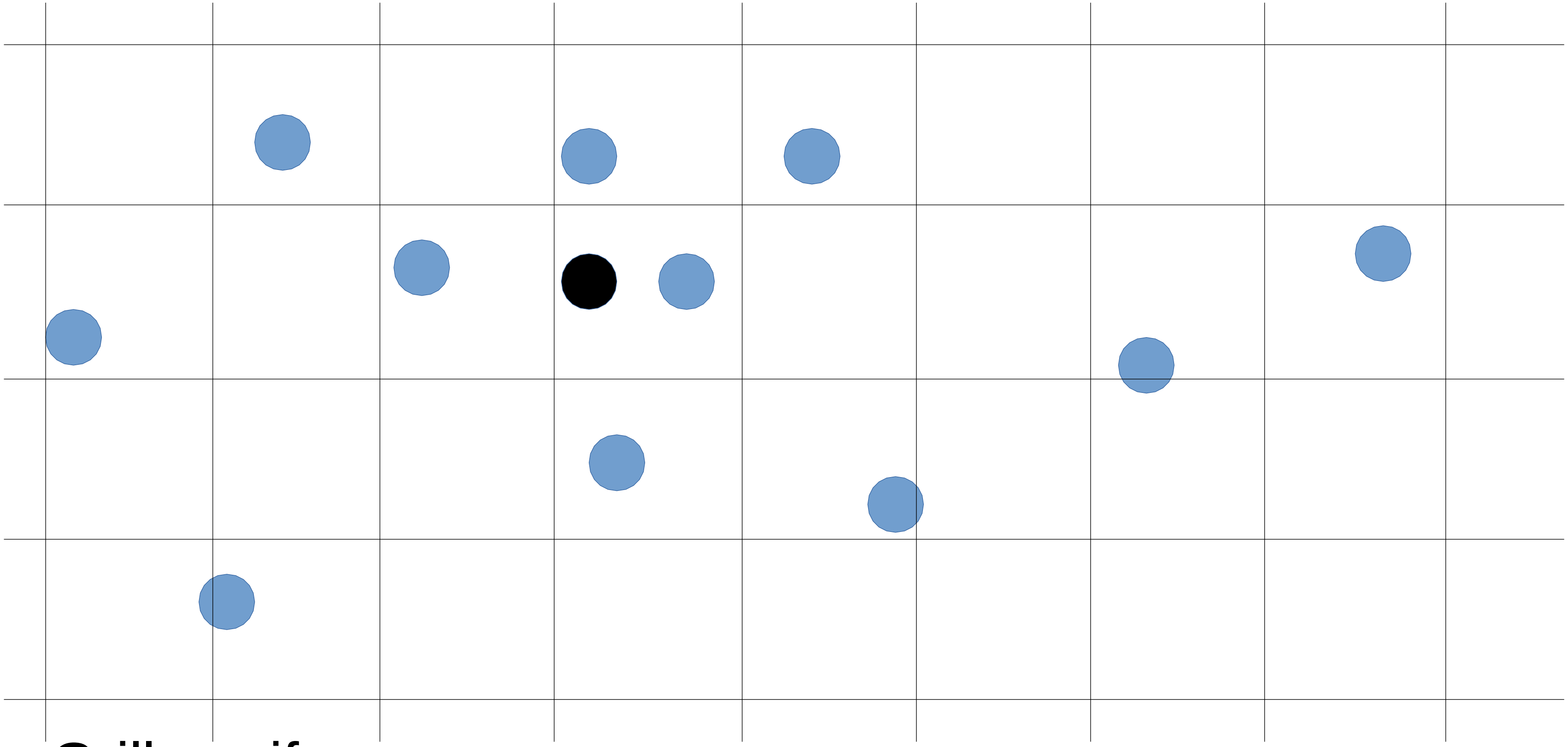
K plus proches voisins peuvent être dans un voisinage très éloigné, et mal distribués !

Recherche de voisinages Euclidiens : Structures



Parcours linéaire : le plus simple, aucune structure à stocker.

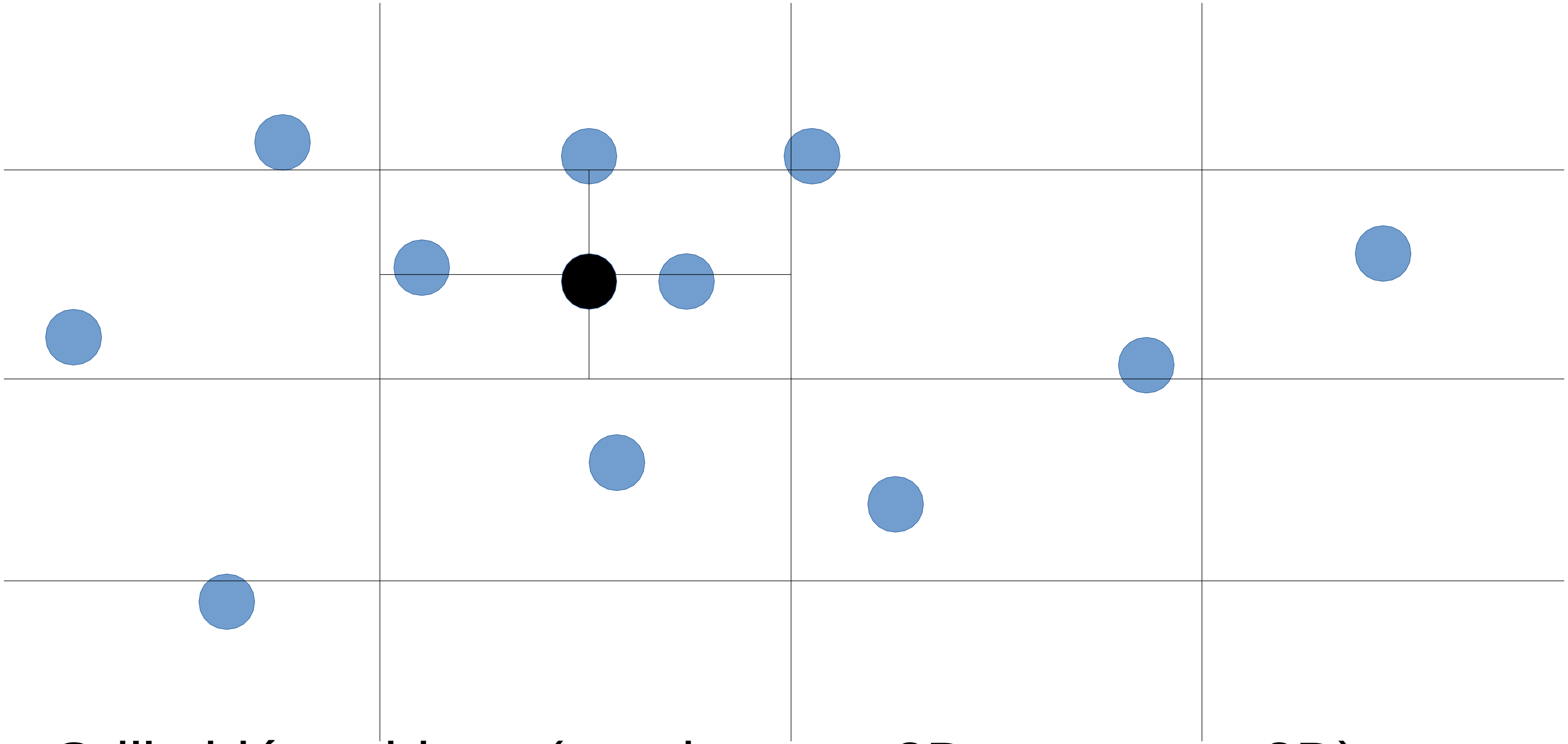
Recherche de voisinages Euclidiens : Structures



Grille uniforme :

- très simple,
- dépend de la taille de la cellule,
- gâche des espaces vides → pas adaptée à des points distribués sur une surface

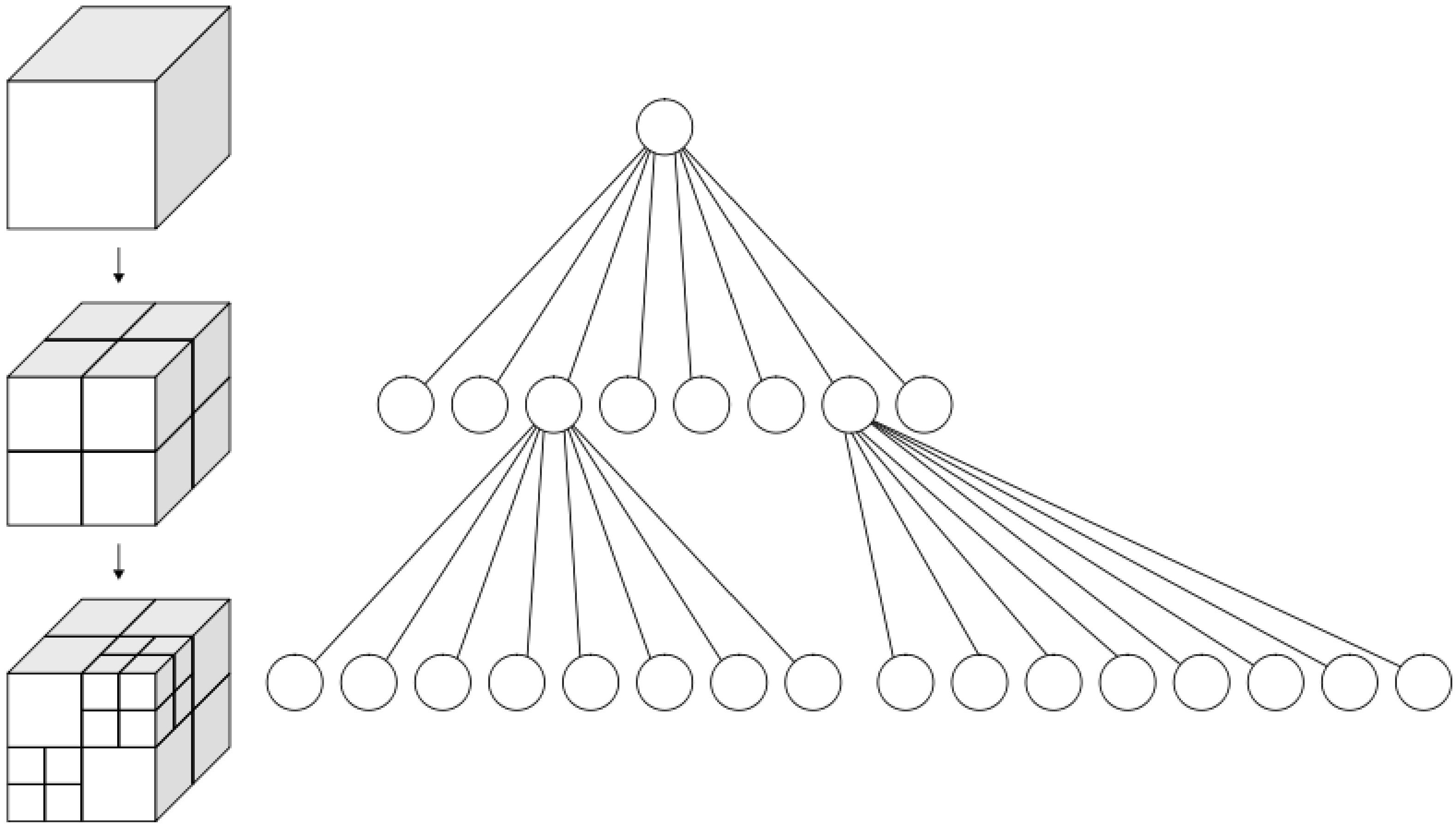
Recherche de voisinages Euclidiens : Structures



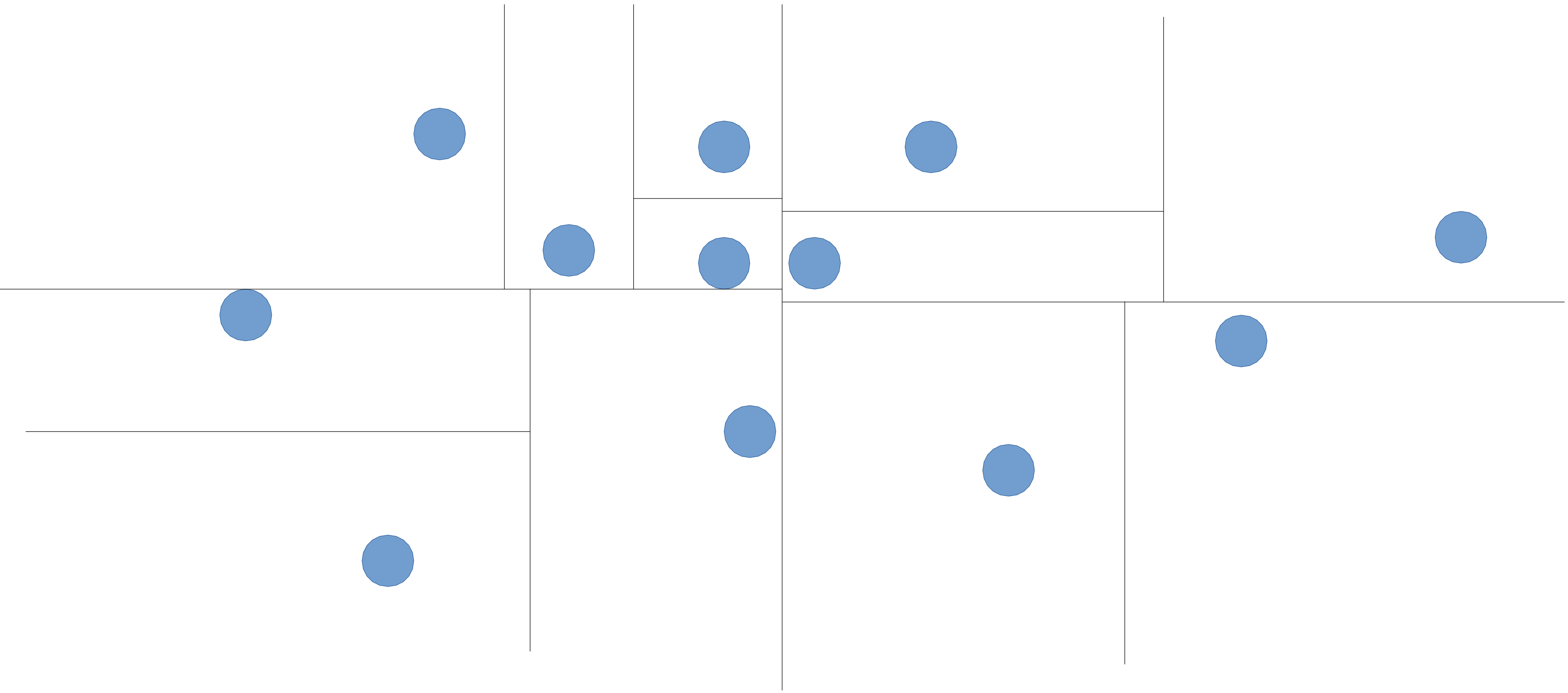
Grille hiérarchique (quadtree en 2D, octree en 3D) :

- construction simple,
- query simple,
- adaptatif → adapté a des points distribués sur une surface

Représentation « sparse » des octrees

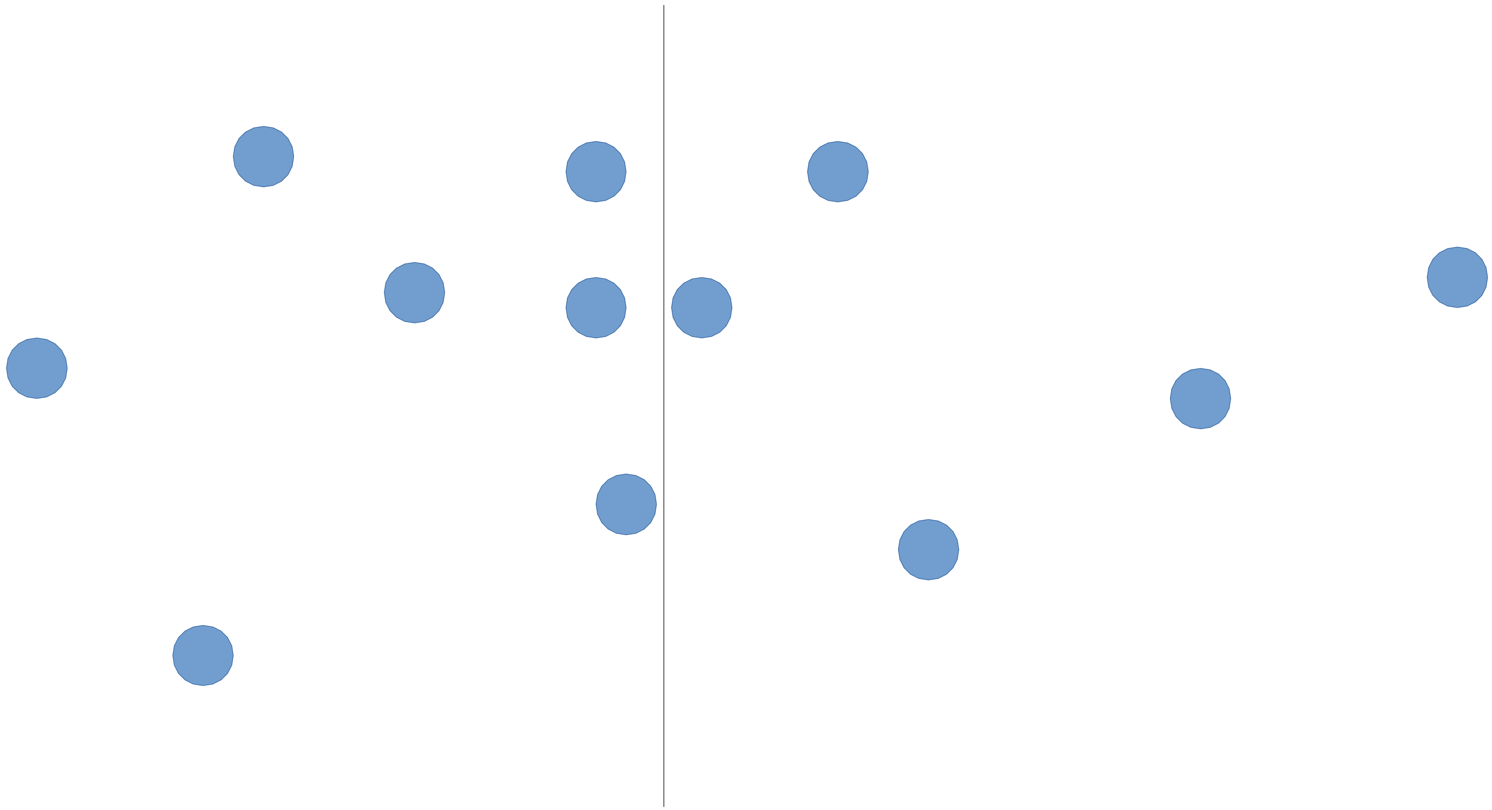


kd-tree



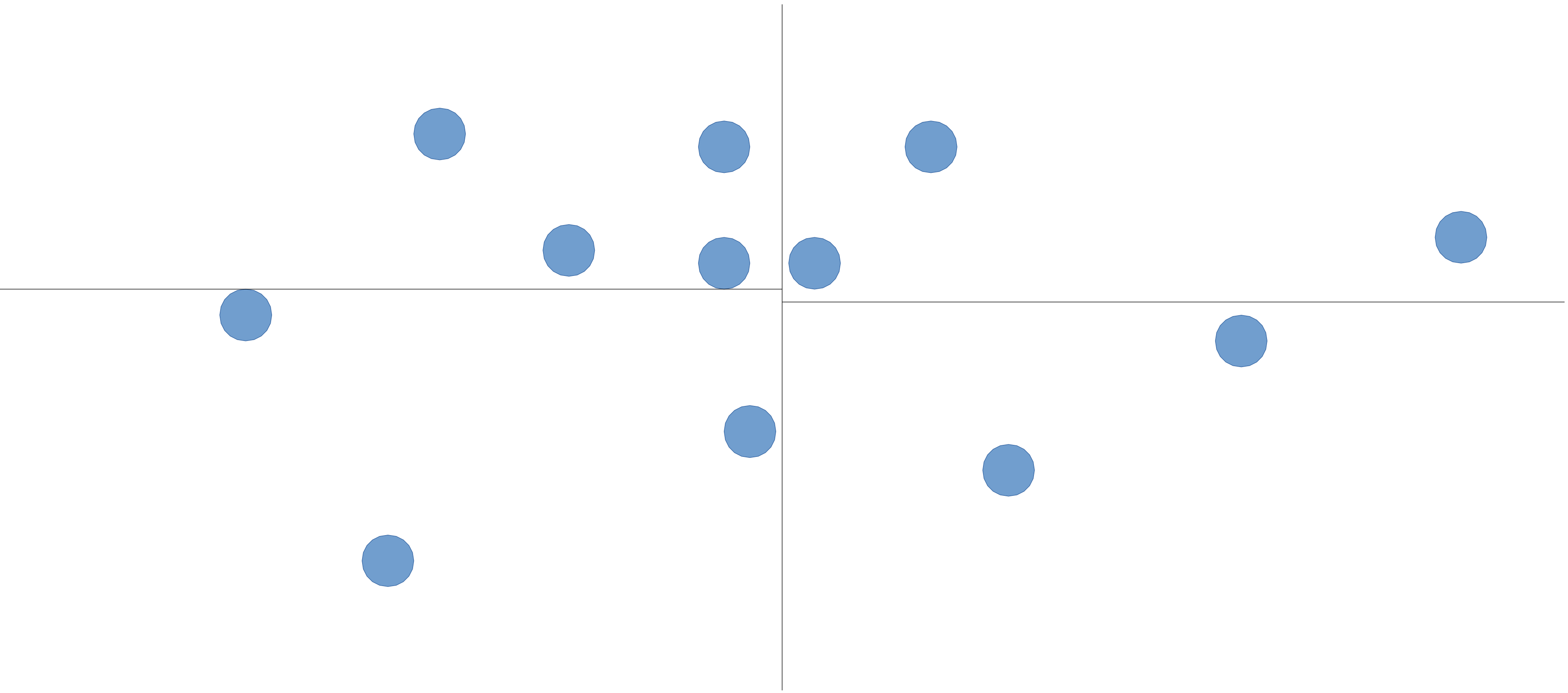
kd-tree: construction simple, query un peu compliquée mais efficace, adaptatif → adapté a des points distribués sur une surface

kd-tree



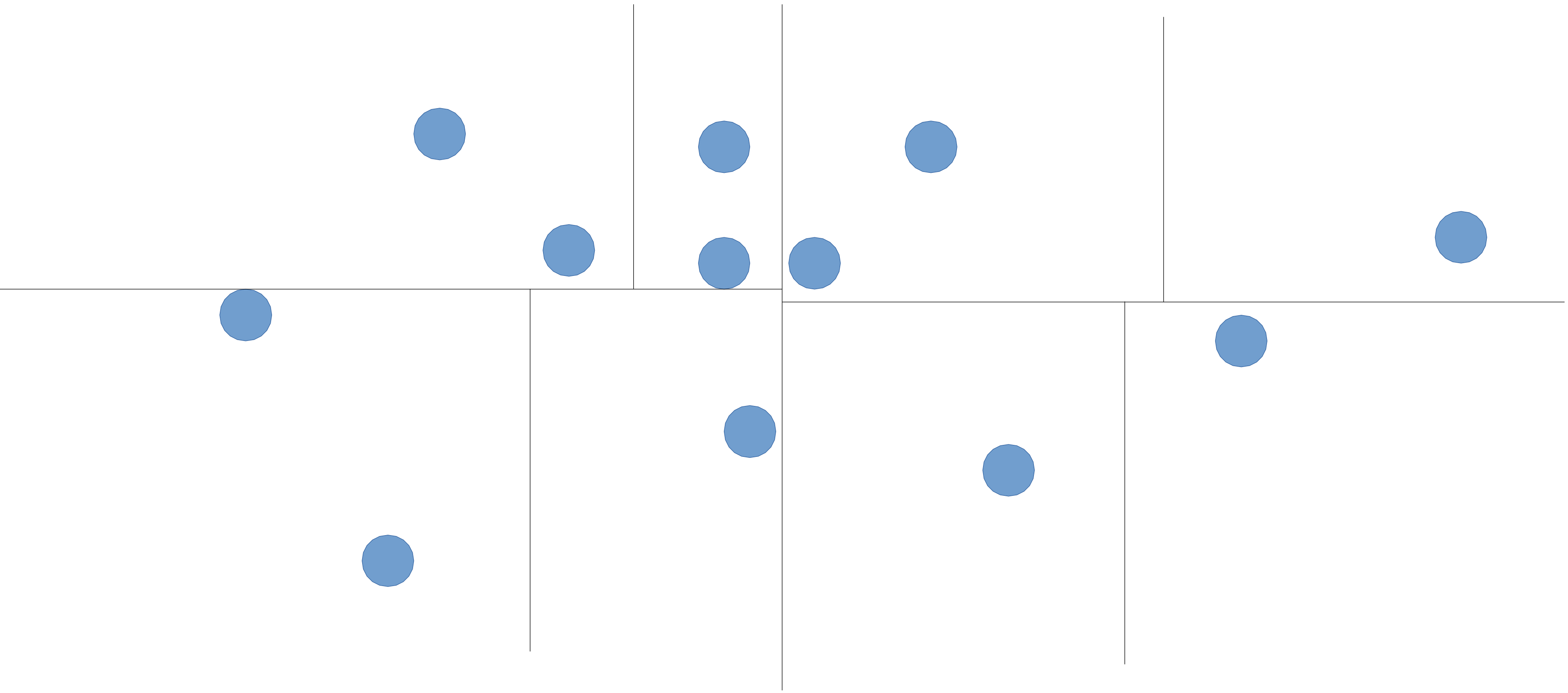
kd-tree: construction simple

kd-tree



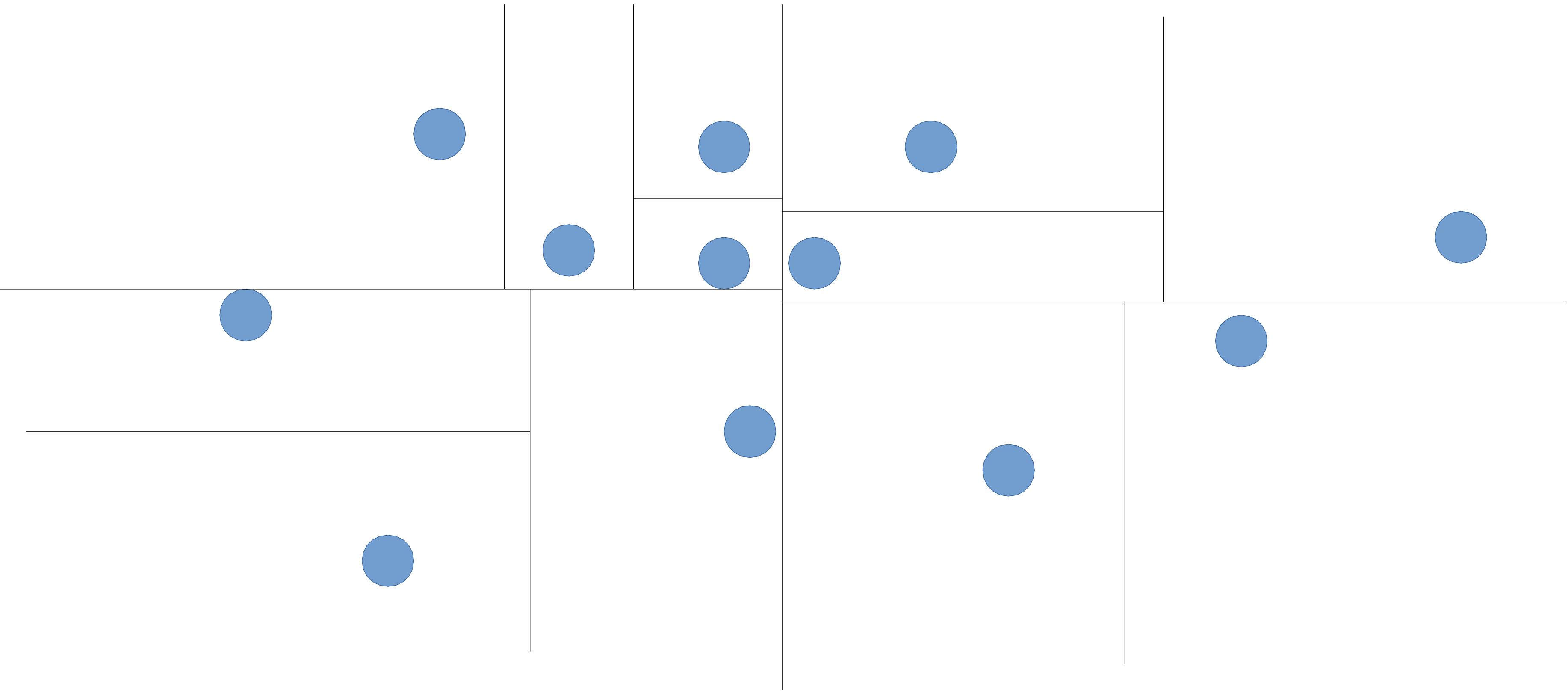
kd-tree: construction simple

kd-tree



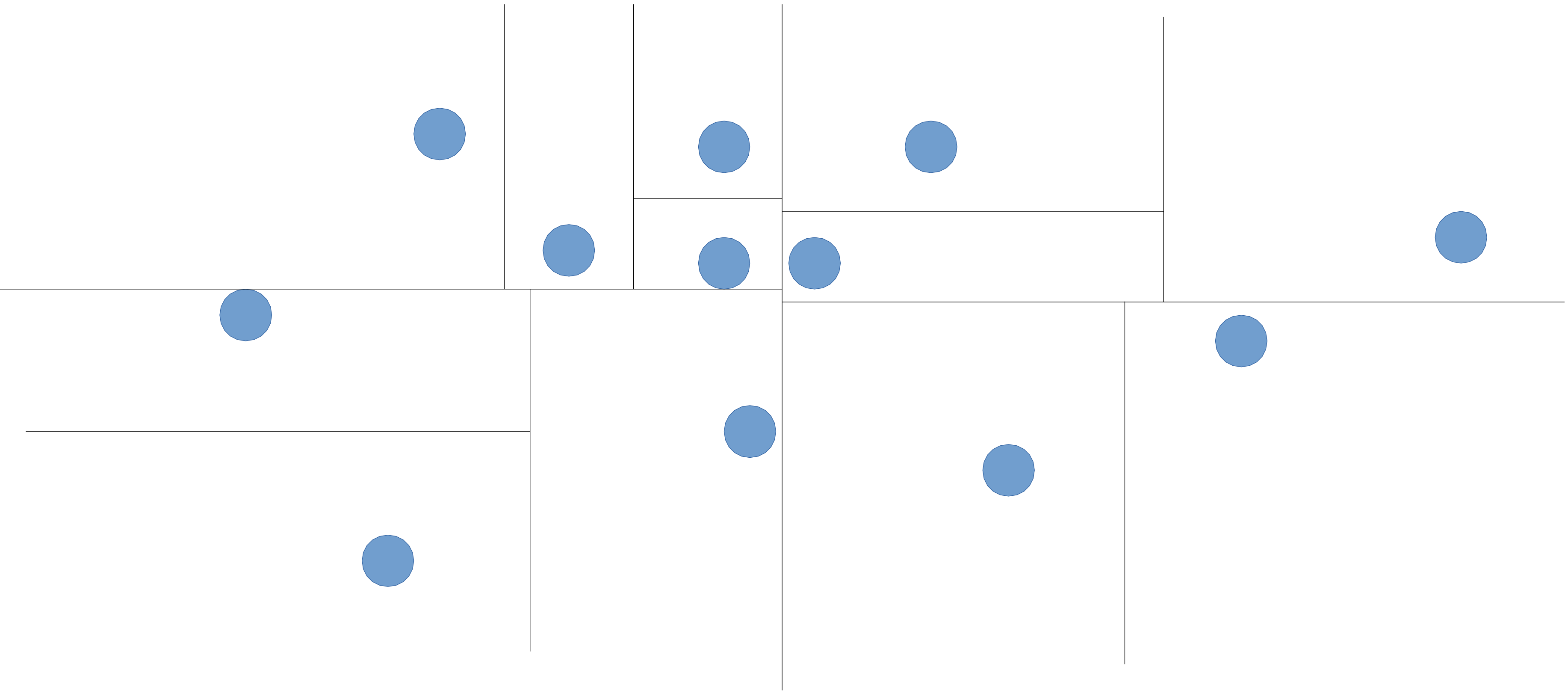
kd-tree: construction simple

kd-tree



kd-tree: construction simple

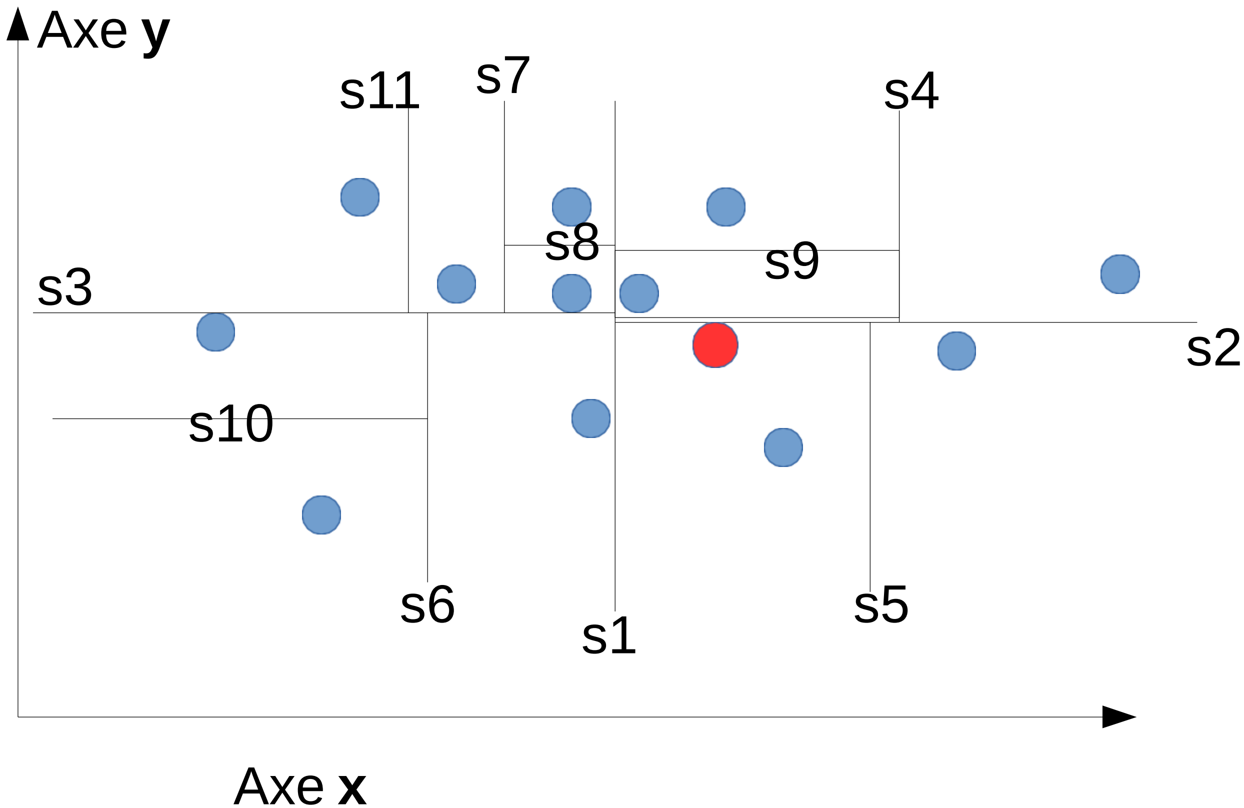
kd-tree

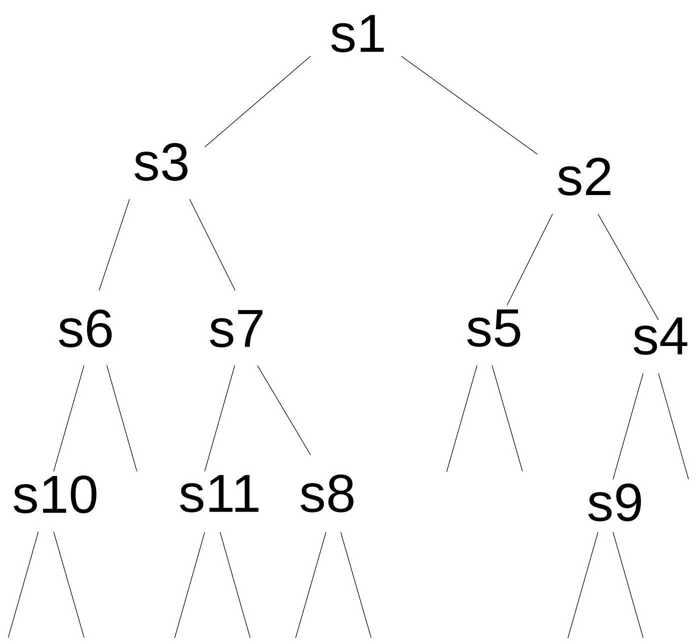


kd-tree: chaque nœud stocke l’axe de subdivision (par ex, « x »), une équation décrivant sa géométrie (par ex, « x=4 »), et des pointeurs vers ses deux nœuds fils

Kd-tree : recherche du plus proche voisin

- Identifier la feuille dans laquelle tombe le point
- En remontant l'arbre, verifier si les cellules voisines peuvent contenir des points plus proches





Kd-tree : recherche du plus proche voisin

```
NNS(q: point, n: node, p: point, w: distance) : point {
if n.left = null then {leaf case}
   if distance(q,n.point) < w then return n.point else return p;
else
   if w = infinity then
     if q(n.axis) ≤ n.value then
        p := NNS(q,n.left,p,w);
        w := distance(p,q);
        if q(n.axis) + w > n.value then p := NNS(q, n.right, p, w);
     else
        p := NNS(q,n.right,p,w);
        w := distance(p,q);
        if q(n.axis) - w ≤ n.value then p := NNS(q, n.left, p, w);
   else //w is finite//
     if q(n.axis) - w ≤ n.value then
     p := NNS(q, n.left, p, w);
     w := distance(p,q);
     if q(n.axis) + w > n.value then p := NNS(q, n.right, p, w);
   return p
}
```

Ressources extérieures utilisées :
https://courses.cs.washington.edu/courses/cse373/02au/lectures/lecture22l.pdf

Kd-tree

- Permet la recherche du plus proche voisin
- Permet la recherche des k plus proches voisins
- Permet la recherche des voisins dans une boule
- Complexite de construction : n log(n)
- Complexite de query : k log(n)

- Librairies open source existantes
- Code fourni pour le TP

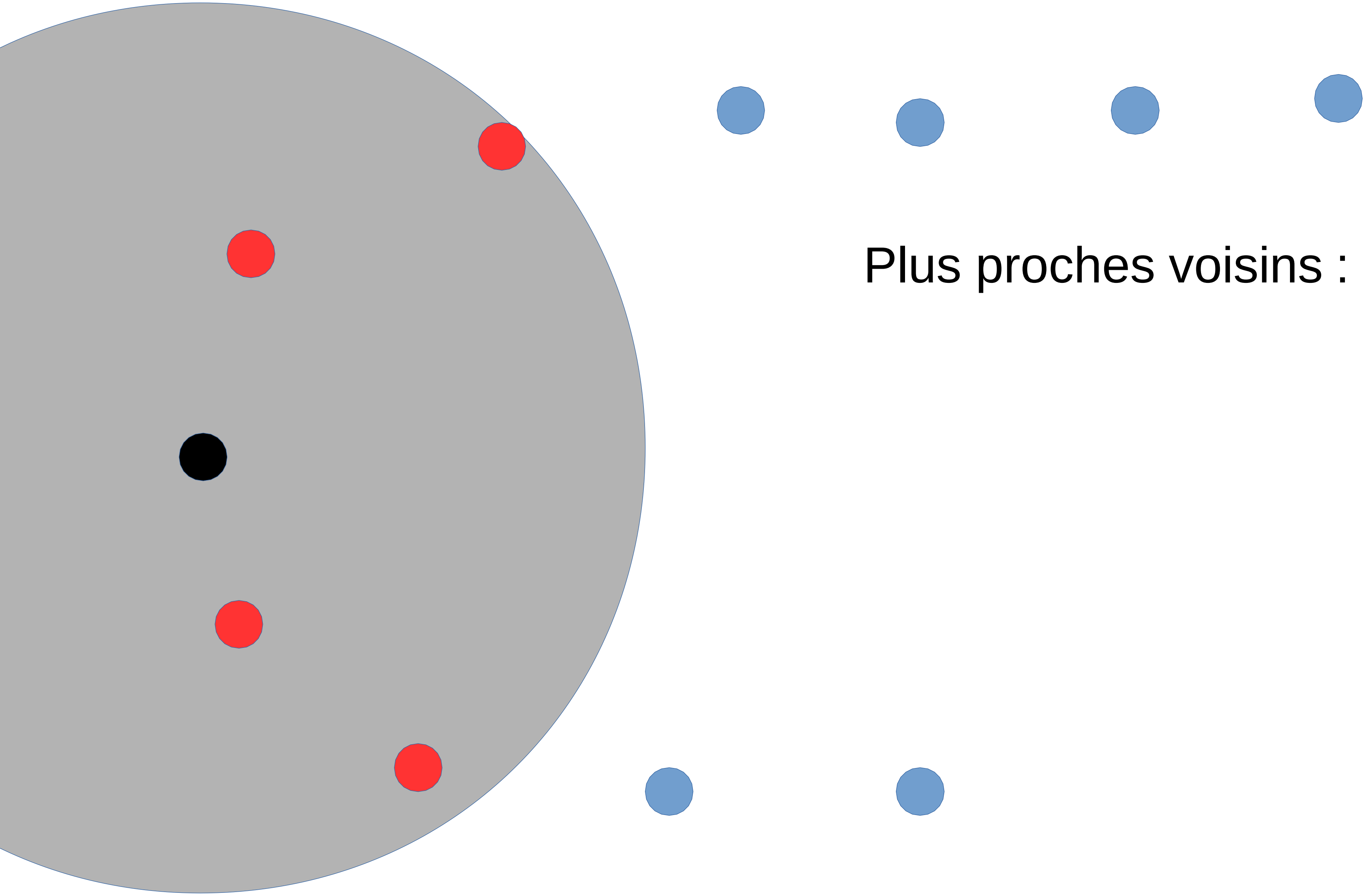
Questions ?

?

Estimation des normales

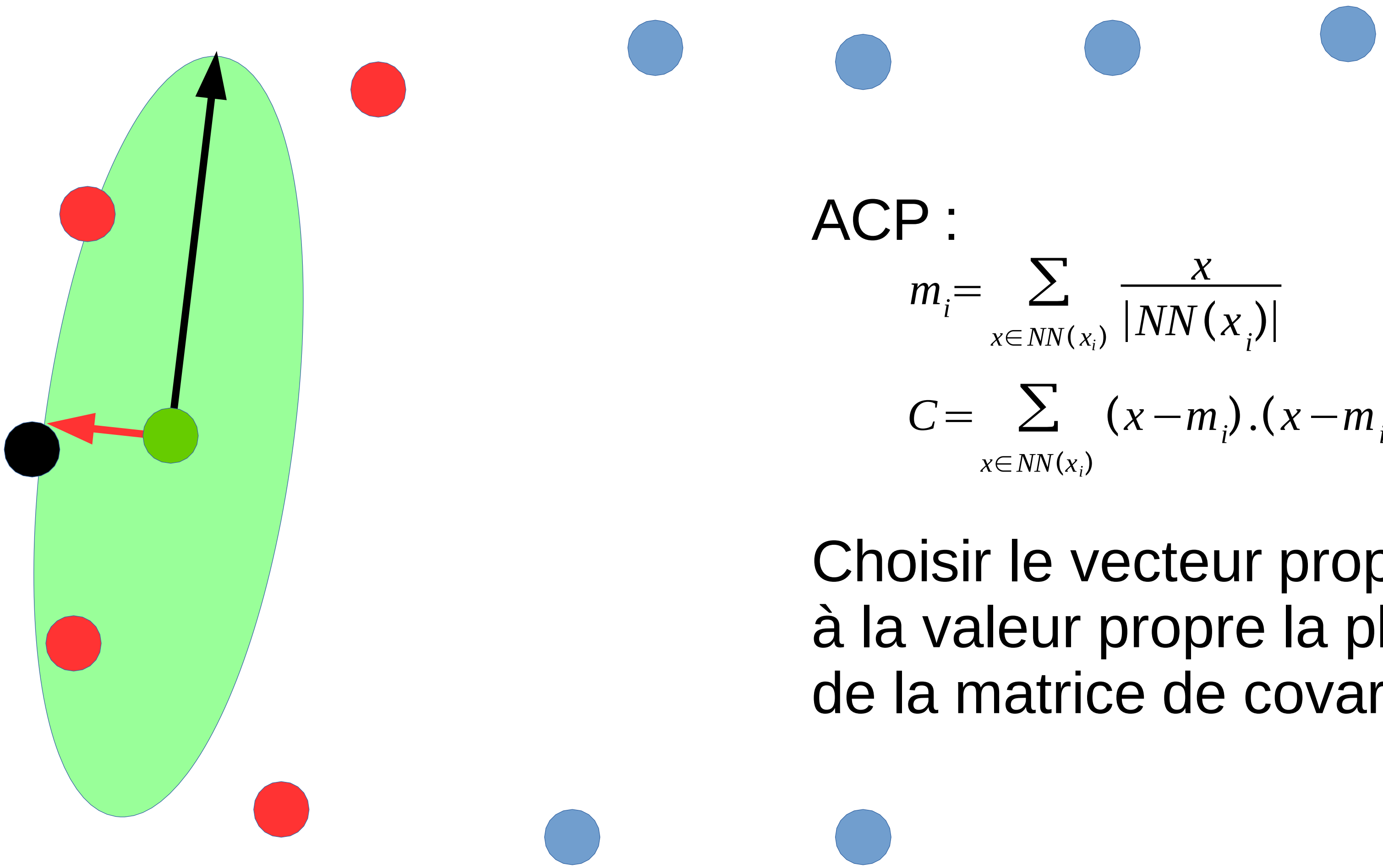
- Parfois, on ne dispose que de points 3D, sans leur vecteur normal.
- Certaines définitions de surface en ont besoin.
- Utile pour le rendu de pointsets.
- Définit l’intérieur et l’extérieur de manière consistante.

Estimation des normales



Plus proches voisins : $NN(x_i)$

Estimation des normales



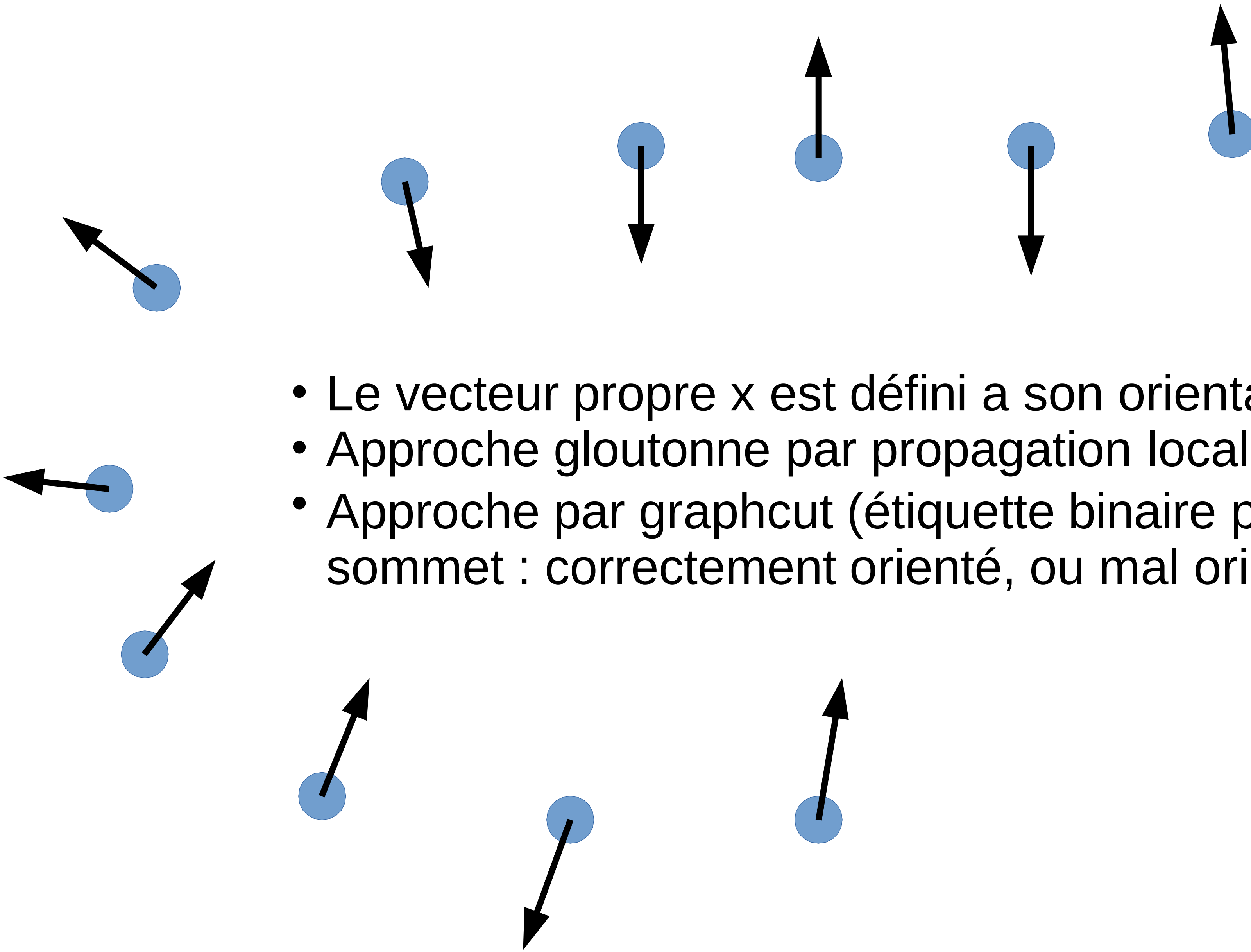
ACP :

$$m_i = \sum_{x \in NN(x_i)} \frac{x}{|NN(x_i)|}$$

$$C = \sum_{x \in NN(x_i)} (x - m_i).(x - m_i)^T$$

Choisir le vecteur propre associé à la valeur propre la plus petite de la matrice de covariance.

Orientation consistente



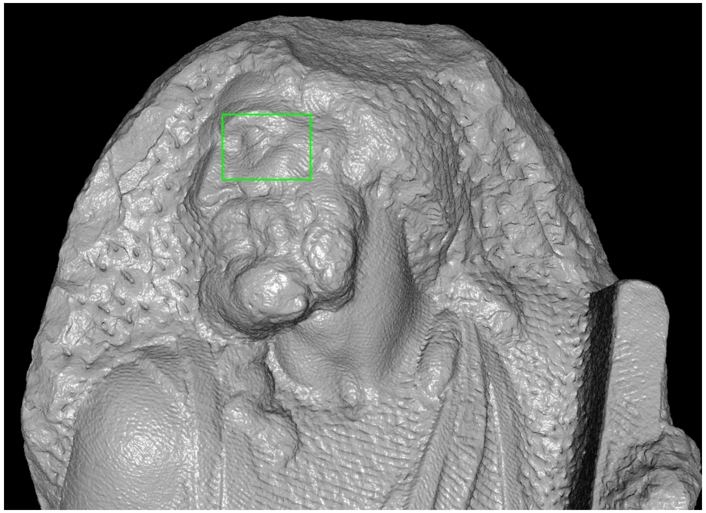
- Le vecteur propre x est défini a son orientation près
- Approche gloutonne par propagation locale
- Approche par graphcut (étiquette binaire pour chaque sommet : correctement orienté, ou mal orienté)

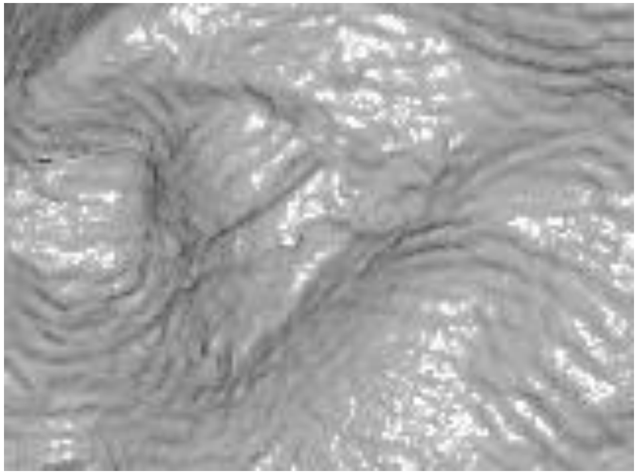
Questions ?

?

Visualisation de pointsets massifs

Rendu de 127 millions de points





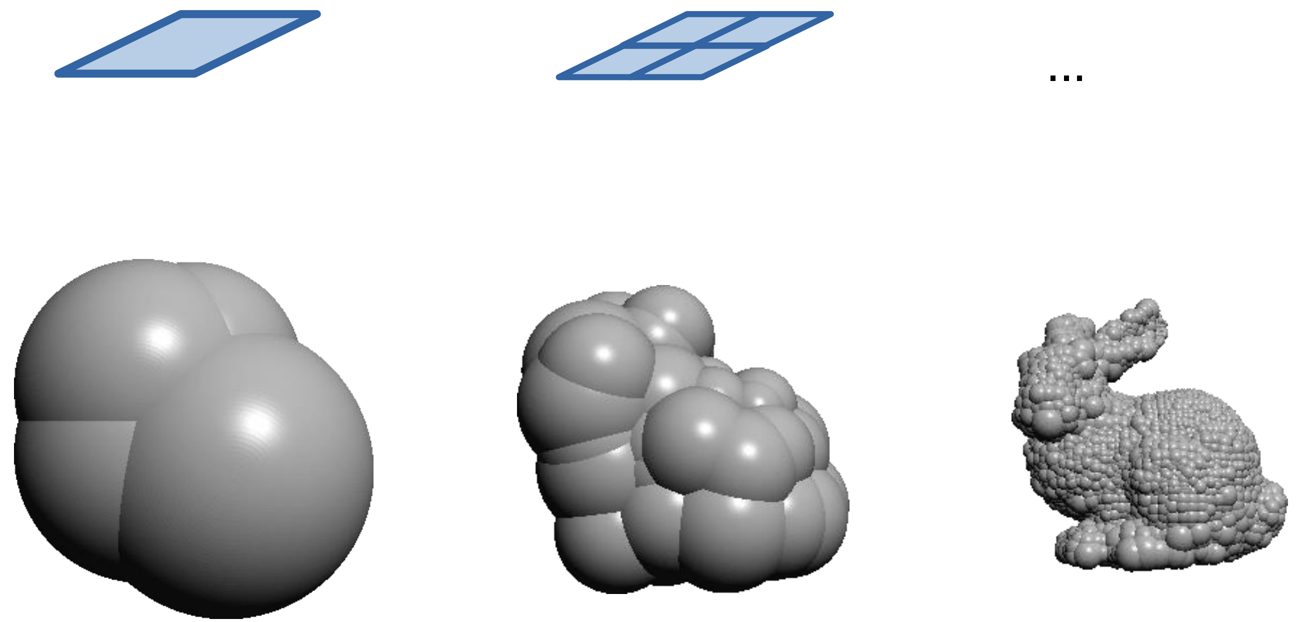
[Rusinkiewicz & Levoy 200] QSplat: A Multiresolution Point Rendering System for Large Meshes

Visualisation de pointsets massifs

- S'appuie sur une hiérarchie de sphères englobantes
- On n'affiche que ce que l'on doit dans un pixel
- Construction faite avec un (par exemple) un kdtree ou un BSPtree

Pixel :

Niveau adapte :

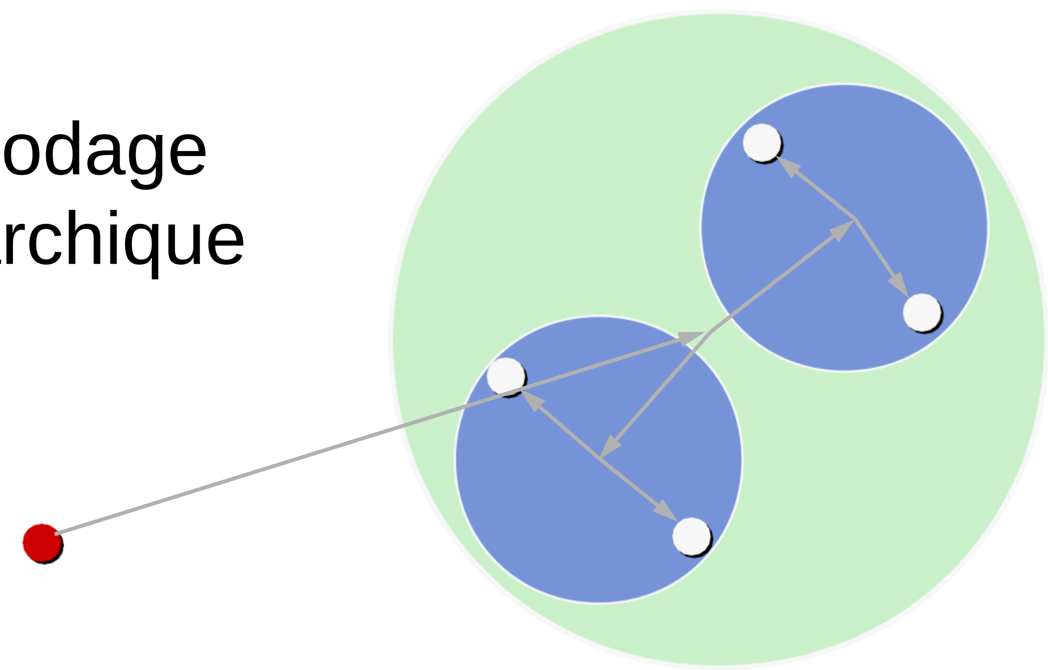


http://web.cs.wpi.edu/~emmanuel/courses/cs563/S05/talks/matt_wk12_QSplat.pdf

Visualisation de pointsets massifs

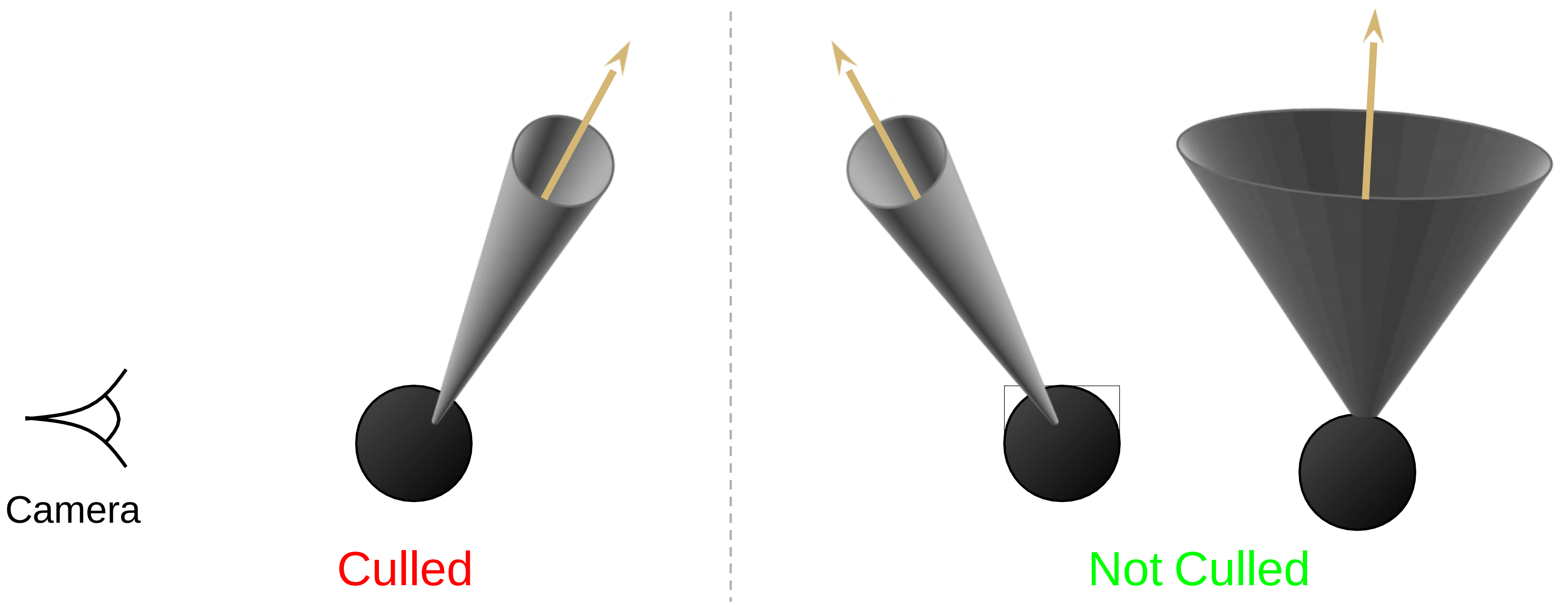
- Chaque nœud contient un pointeur vers ses deux sphères filles
- Ainsi que l'ouverture du cône des normales contenues dans le noeud
- La géométrie de chaque sphère est encodée de manière hiérarchique (des petits nombres encodés sur peu de bits)

Encodage hiérarchique



Visualisation de pointsets massifs

- L’ouverture du cône des normales est utilisée pour le cull des sphères contenant les points qui ne font pas face a la caméra.



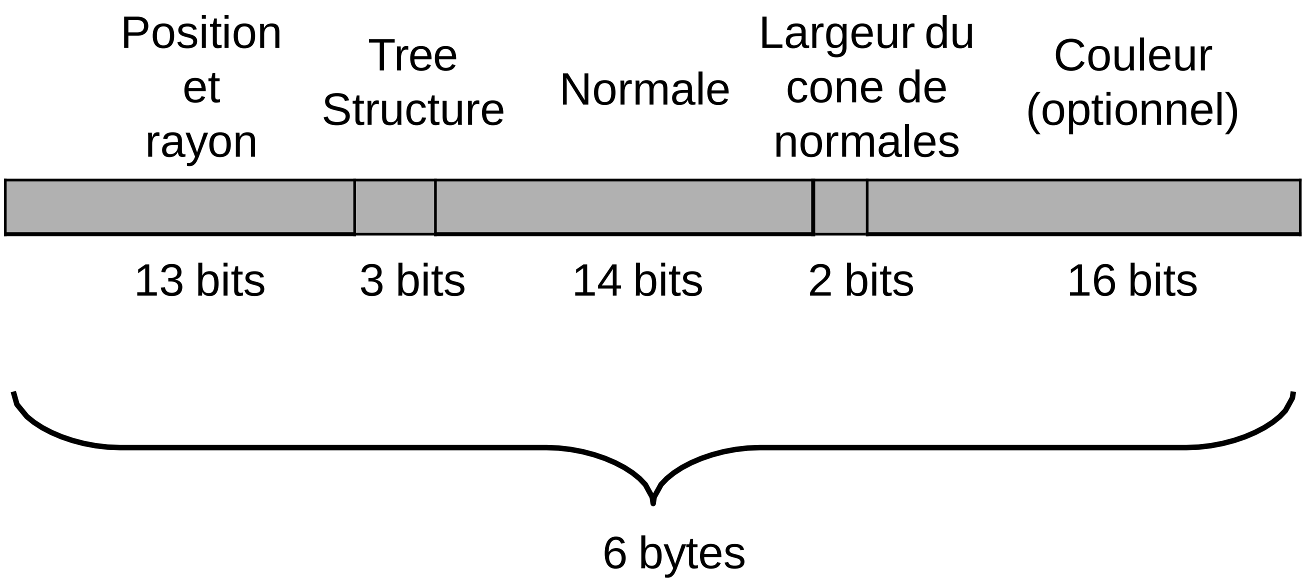
Visualisation de pointsets massifs

Pseudo code :

```
if (node not visible)
        Skip this branch
else if (leaf node)
        Draw a splat
else if (size on screen < threshold)
        Draw a splat
else
        Traverse children
```

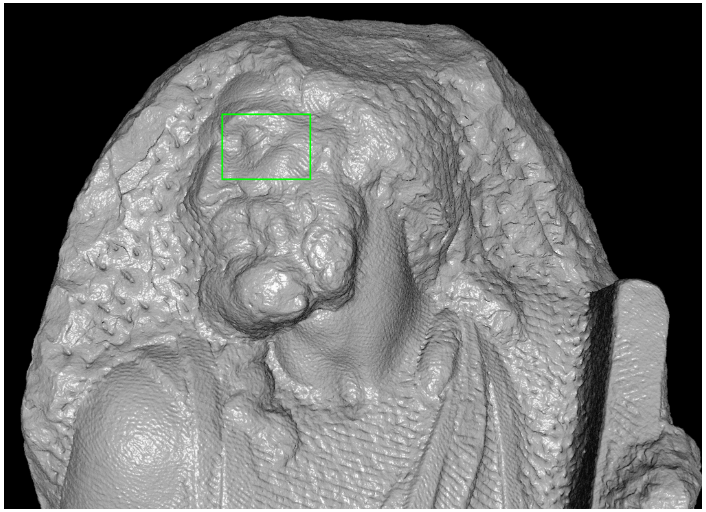
Visualisation de pointsets massifs

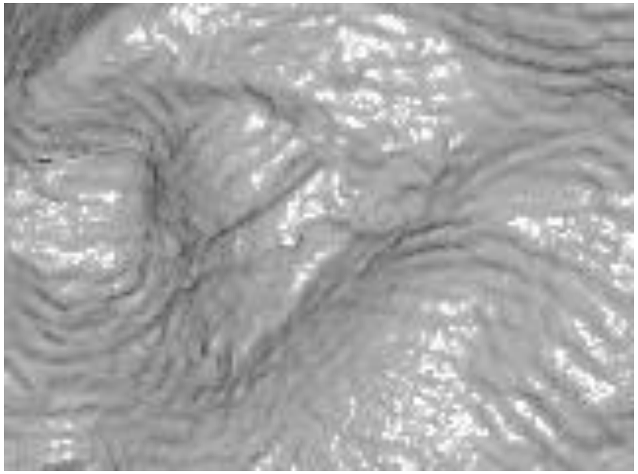
Encodage compresse



Visualisation de pointsets massifs

- 3D scan, statue de 2m70 a 0.25 mm
- 102,868,637 points
- Fichier: 644 MB
- Demo sur portable (PII 366, 128 MB), sans carte graphique (Vivent les annees 2000!)
- Preprocessing: 1h

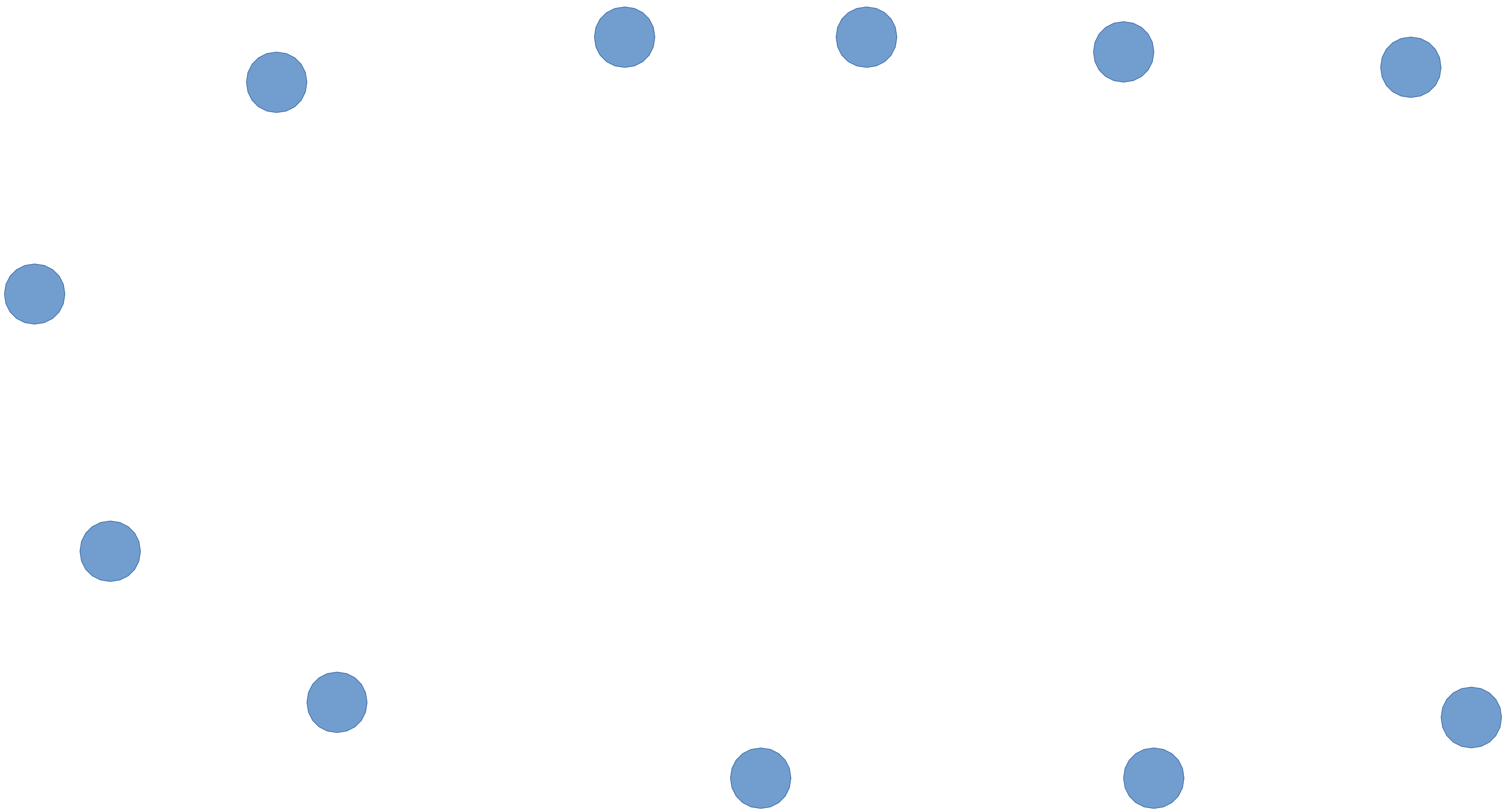




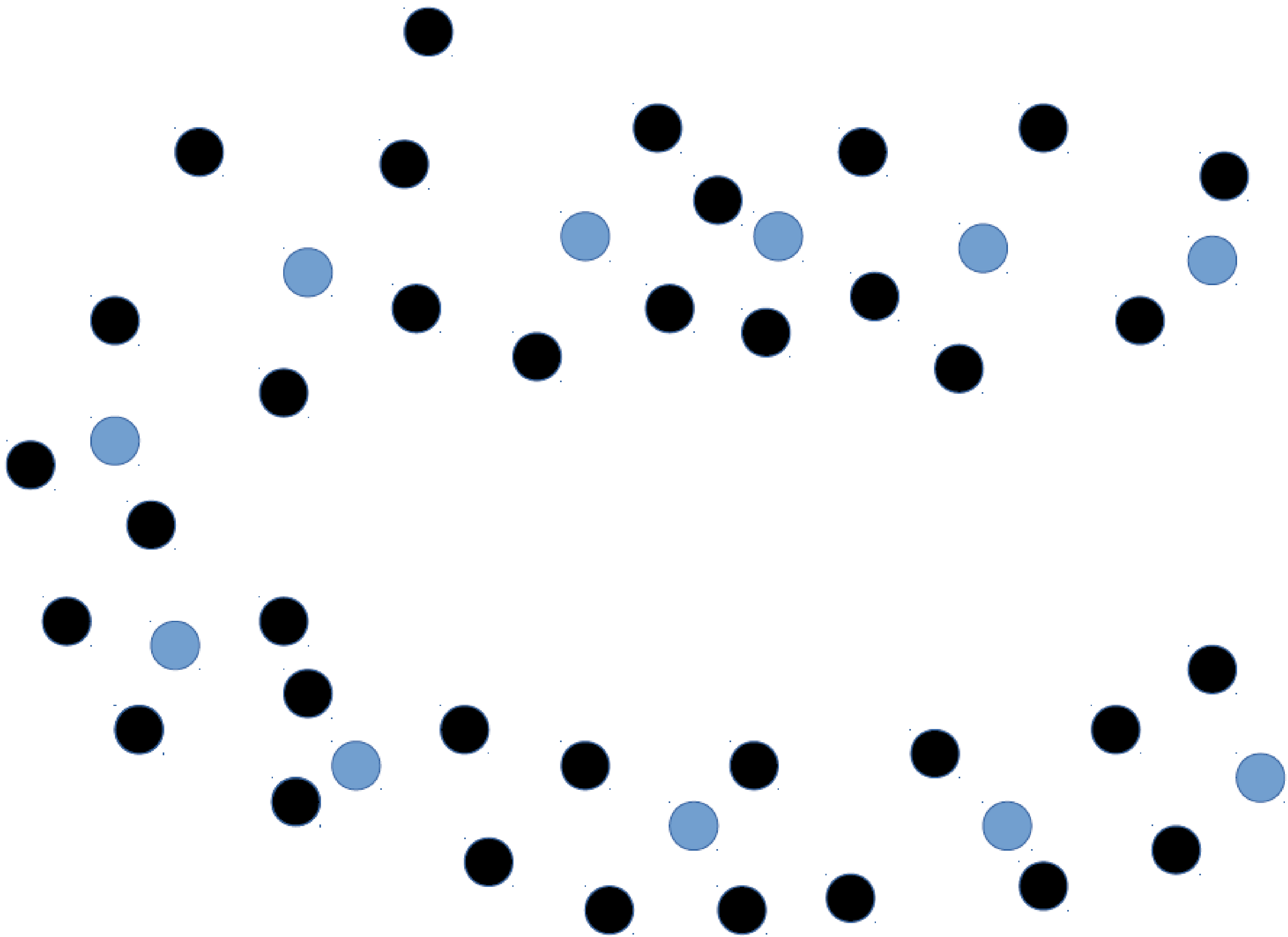
Questions ?

?

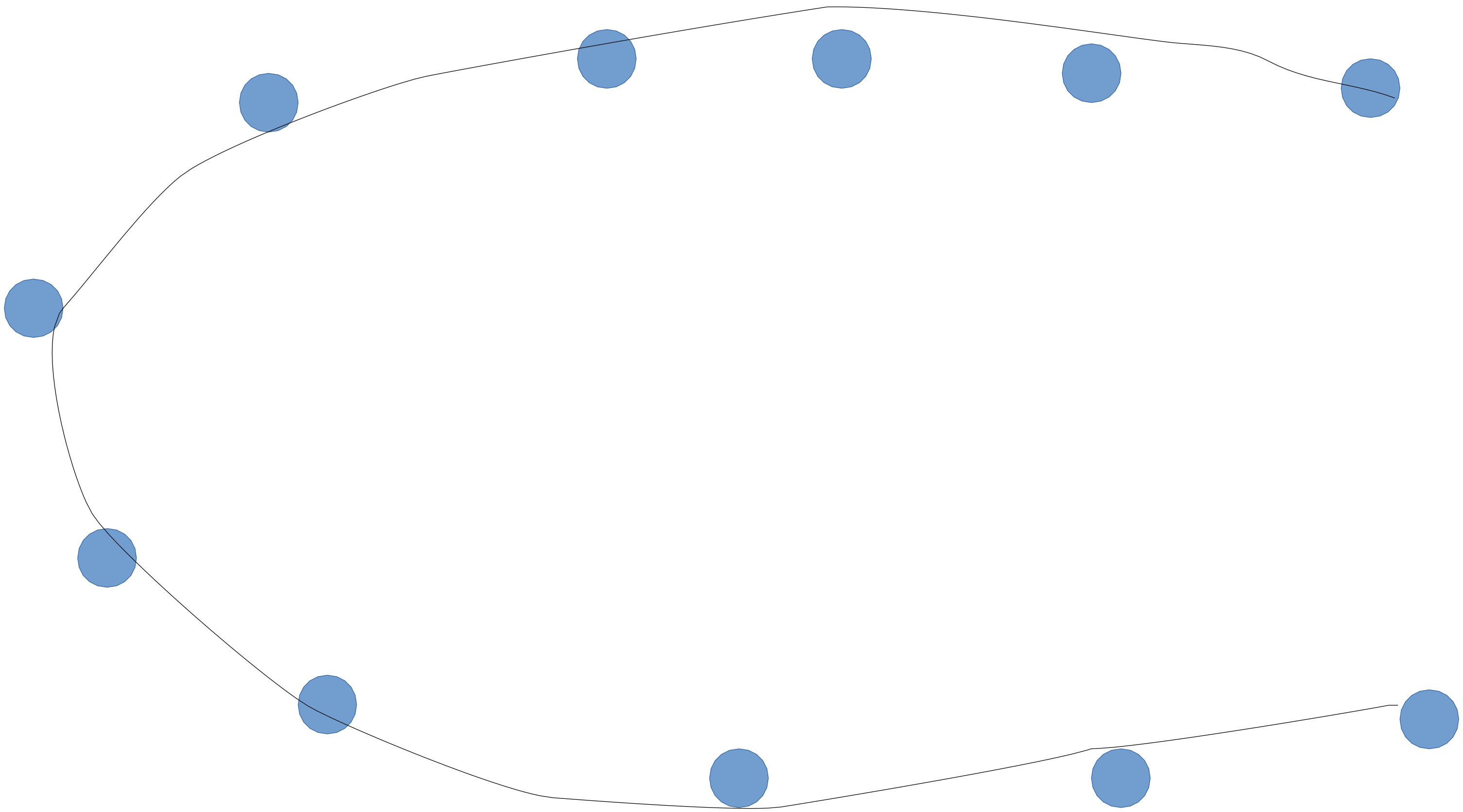
Modèles de surfaces de points



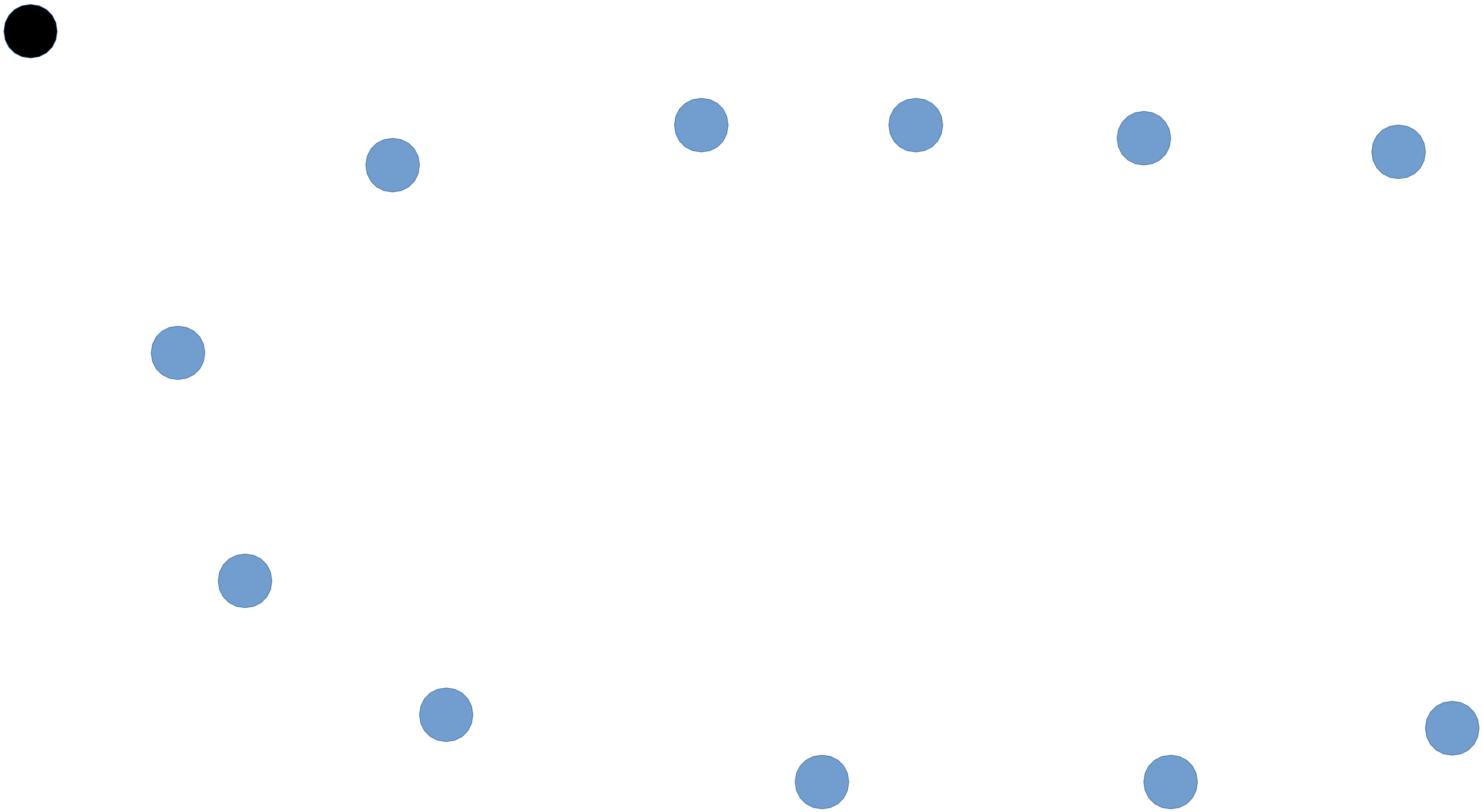
Définition par projection



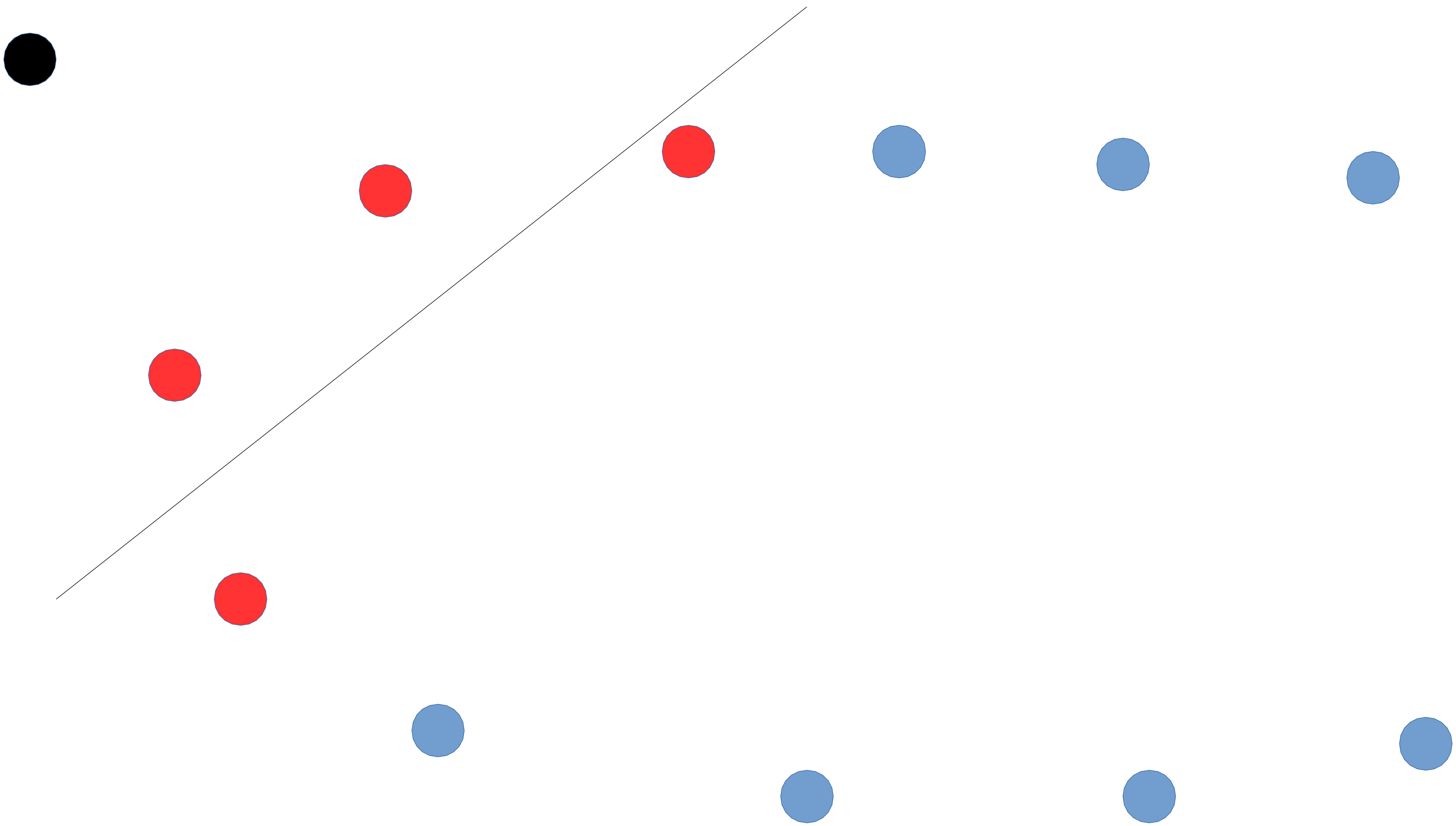
Définition par projection



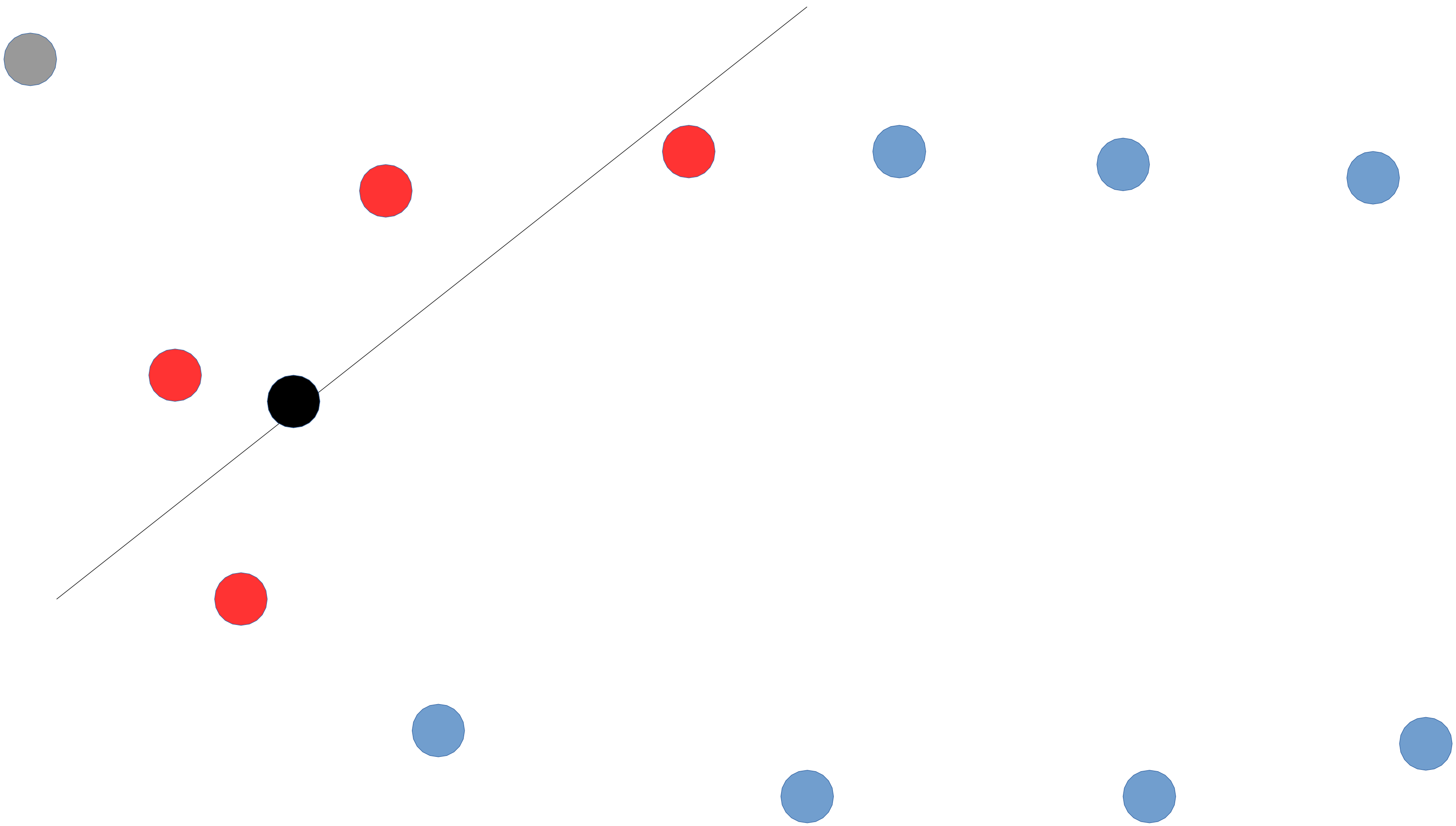
« Moving Least Squares »



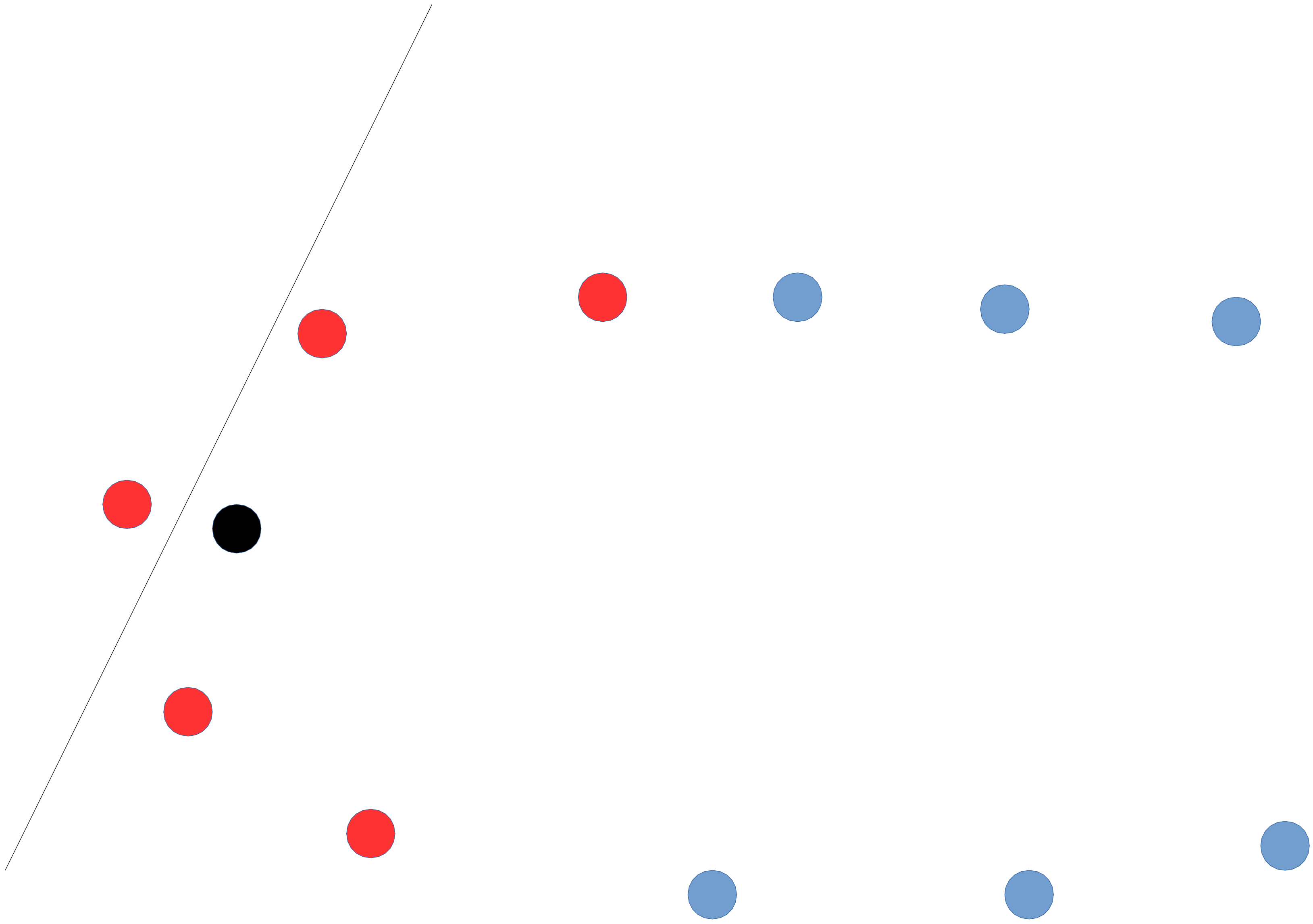
« Moving Least Squares »



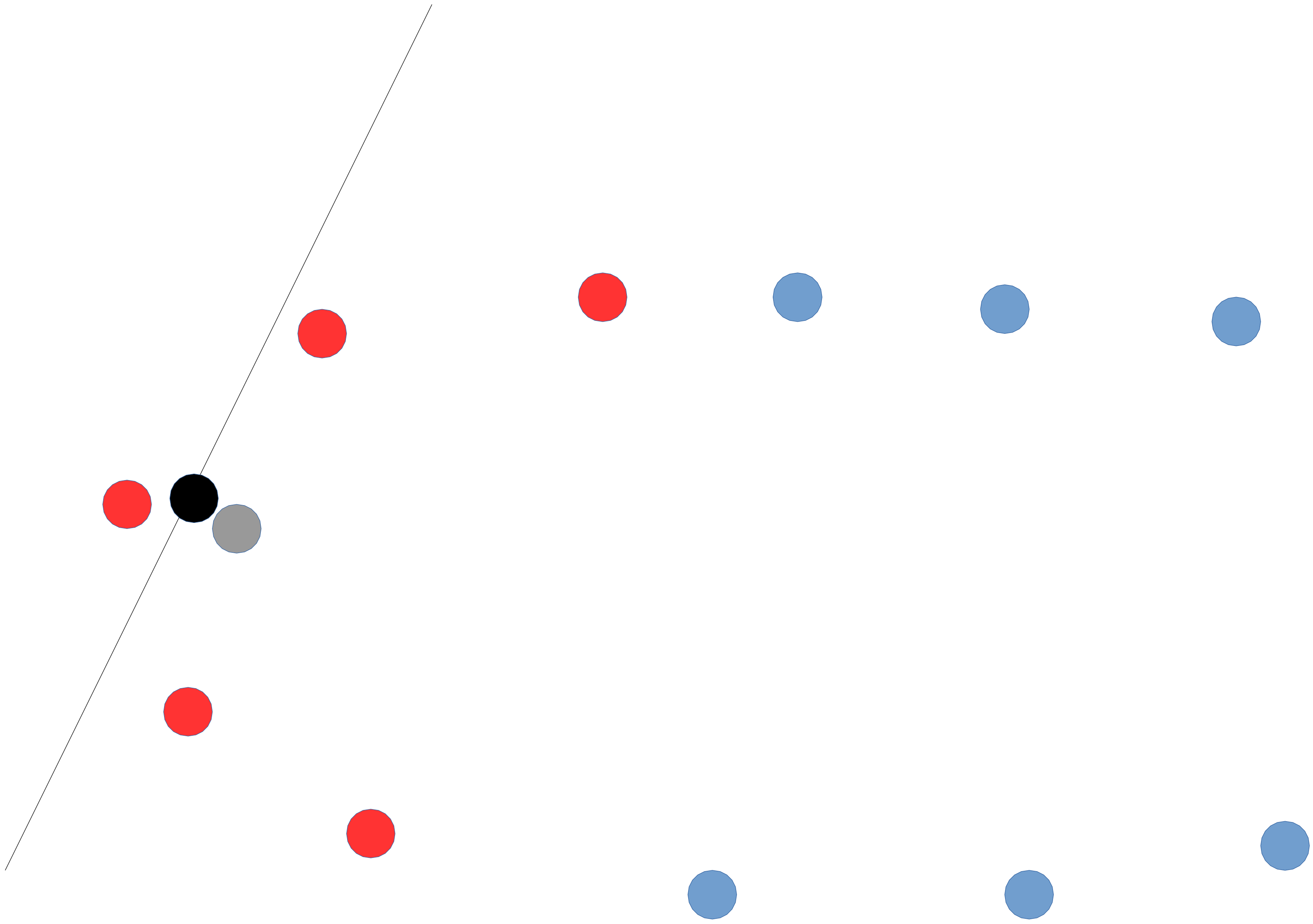
« Moving Least Squares »



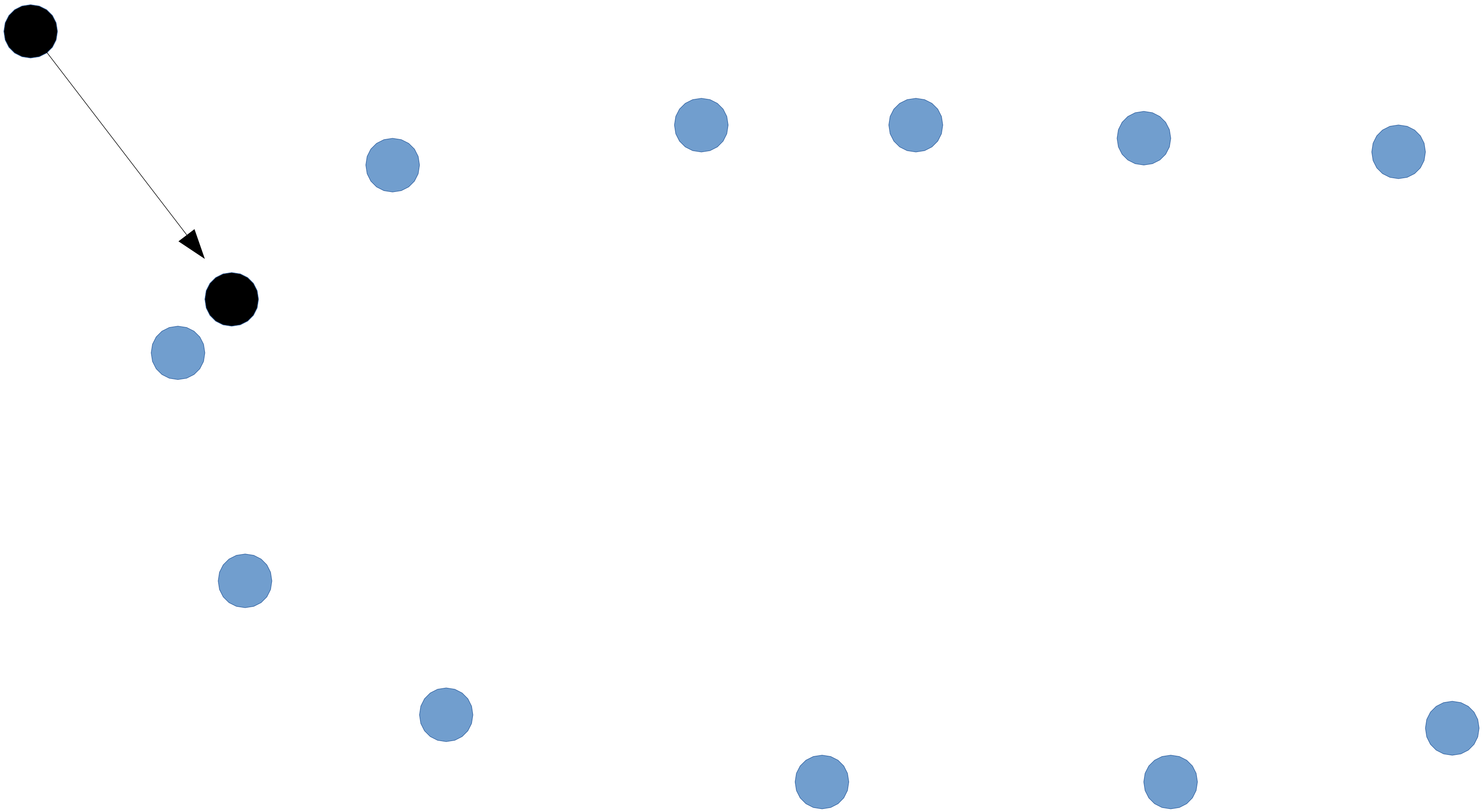
« Moving Least Squares »



« Moving Least Squares »



« Moving Least Squares »



« Moving Least Squares »

- A chaque itération k, on cherche les plus proches voisins, et on définit des poids vis a vis d'eux

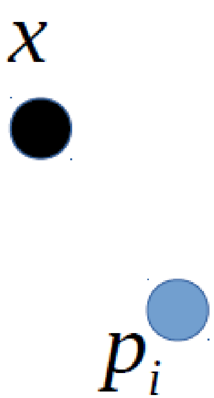
- On trouve le plan qui passe au mieux par ces points (ACP pondérée)

$$\begin{cases} c = \sum_{p_i \in NN(x)} w_i p_i / \sum_{p_i \in NN(x)} w_i \\ Cov = \sum_{p_i \in NN(x)} w_i (p_i - c).(p_i - c)^T \in \mathbb{R}^{3x3} \end{cases} \qquad n = eig_3(Cov)$$

- On projette sur ce plan $x_{k+1} = c - \frac{(x_k - c)^T.n}{\|n\|} n$

- Appelé « Moving Least Squares » car les poids varient en fonction de la position du point au fil des itérations {k}

Définition des poids



$$w_i = \phi\left(\left\|x - p_i\right\|\right)$$

:Influence du point d'entrée sur x

Gaussien : $\phi(r) = e^{\frac{-r^2}{h^2}}$

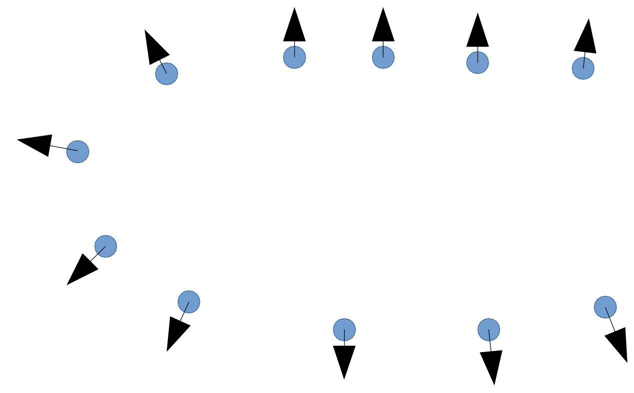
Wendland : $\phi(r) = \left(1 - \frac{r}{h}\right)^4 . \left(1 + 4\frac{r}{h}\right)$

→ Surface approximant les points d'entrée

Singulier : $\phi(r) = \left(\frac{h}{r}\right)^s$

→ Surface interpolant les points d'entrée

« Point set surfaces »



- Modernes définitions de surfaces de points considèrent que le nuage d'entrée doit être équipé de normales.

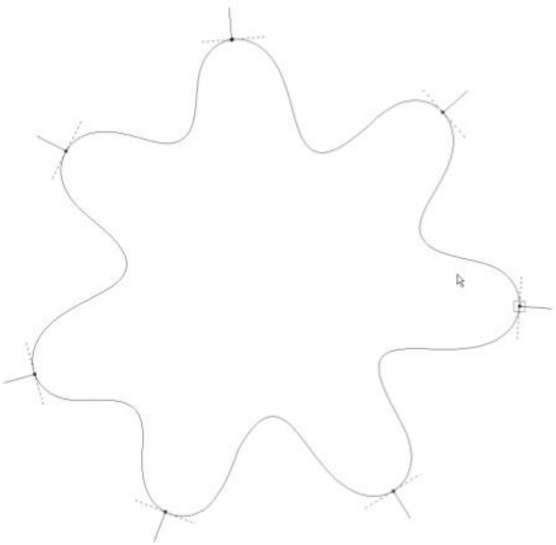
- On utilise $\begin{cases} c(x) = \sum_{p_i \in NN(x)} w_i\, p_i \Big/ \sum_{p_i \in NN(x)} w_i \\ n(x) = \sum_{p_i \in NN(x)} w_i\, n_i \Big/ \sum_{p_i \in NN(x)} w_i \end{cases}$ a la place de la PCA.

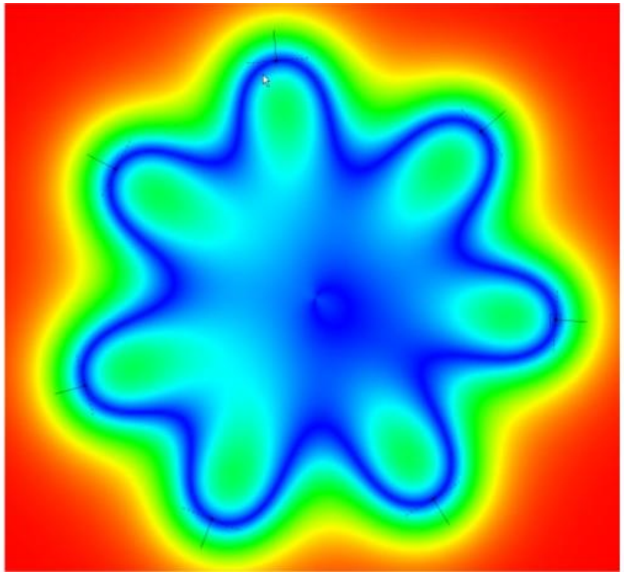
- On peut definir une fonction implicite $f(x) = (x - c(x))^T . n(x)$ (la surface est alors le « 0-set » de cette fonction).

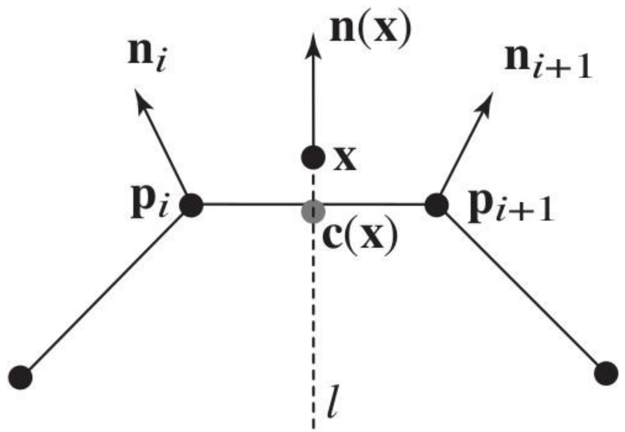
- Avec un noyau tres singulier, on s'attend a pouvoir interpoler points ET normales.

- Ce schéma de PSS est généralement appelé SPSS (Simple Point Set Surface)

« Point set surfaces » : problèmes du schéma standard

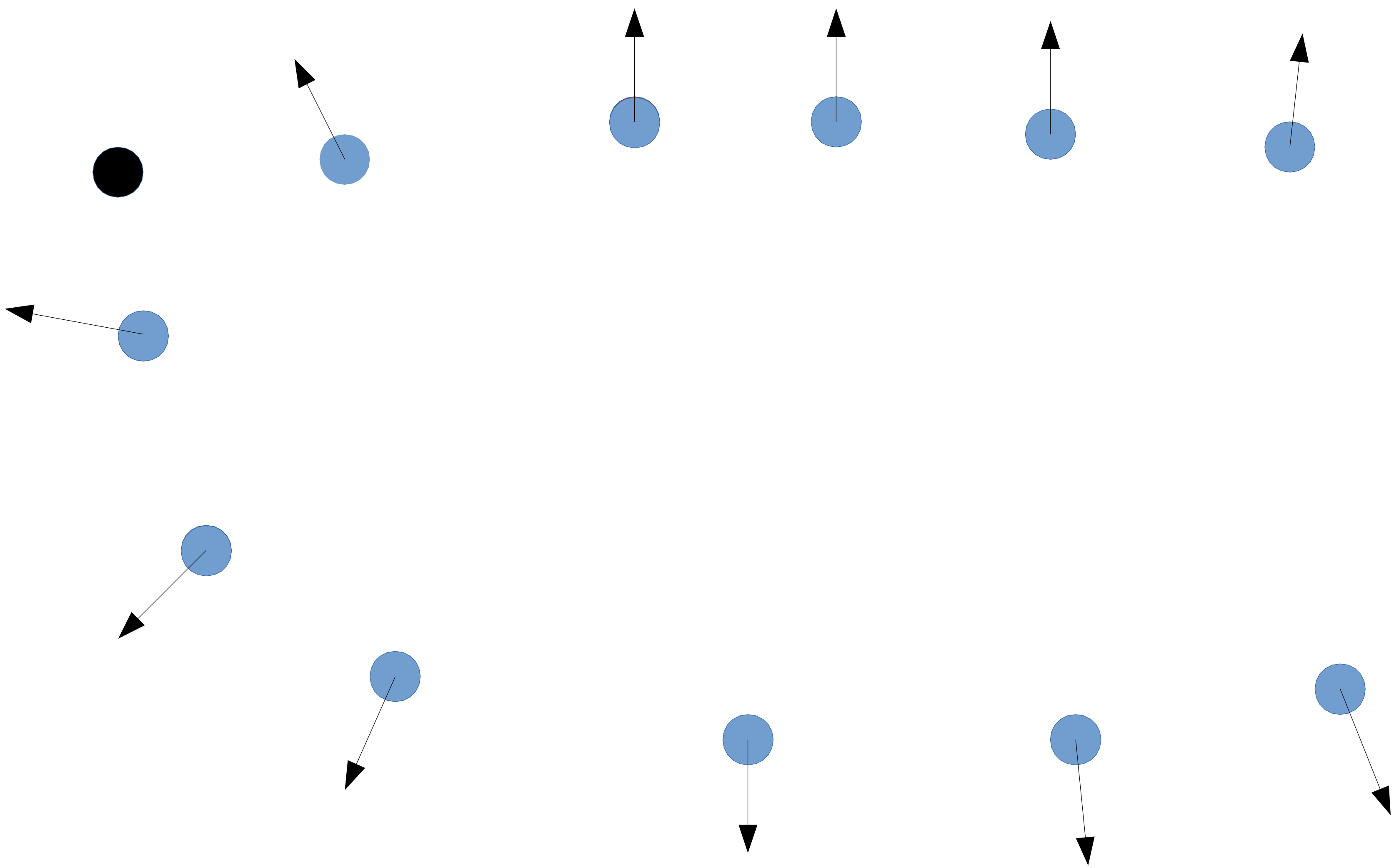




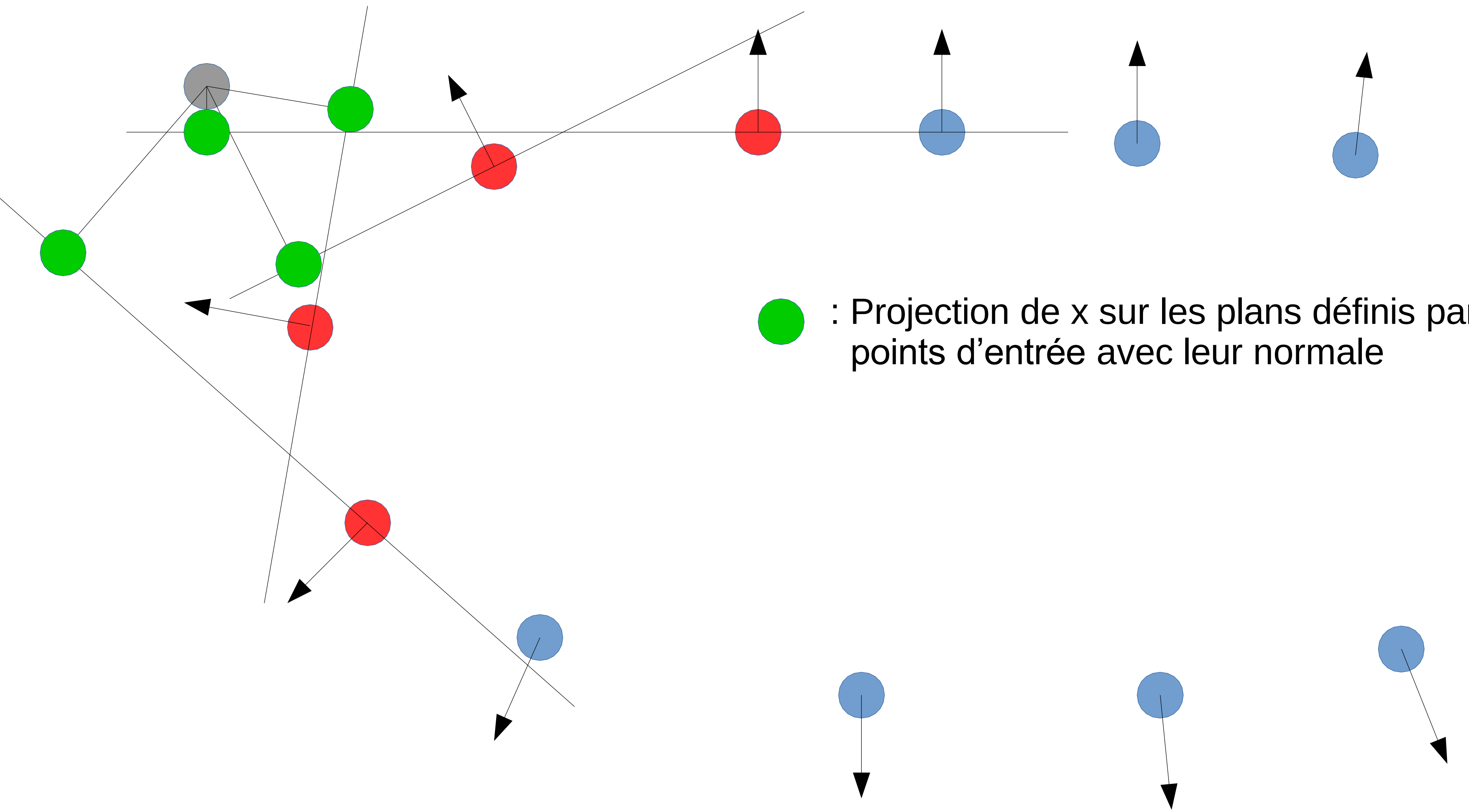


Impossible d'obtenir des surfaces interpolantes convexes.

« Hermite point set surfaces »

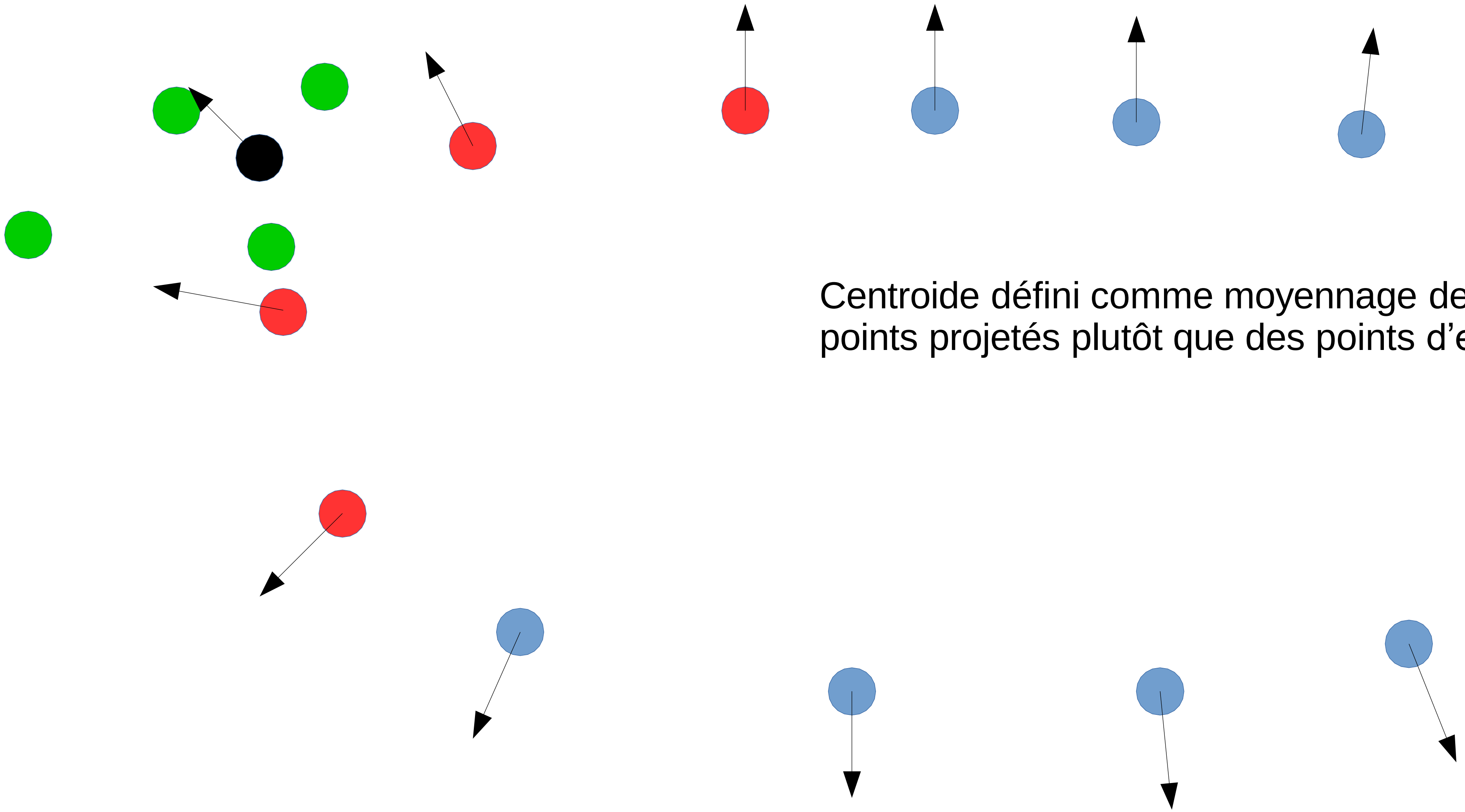


« Hermite point set surfaces »



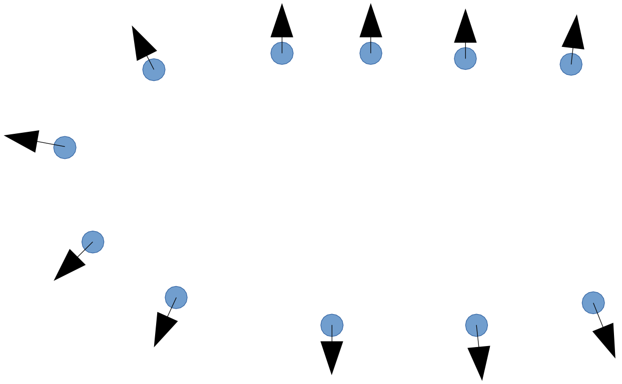
: Projection de x sur les plans définis par les points d’entrée avec leur normale

« Hermite point set surfaces »



Centroide défini comme moyennage des points projetés plutôt que des points d’entrée

« Hermite point set surfaces »

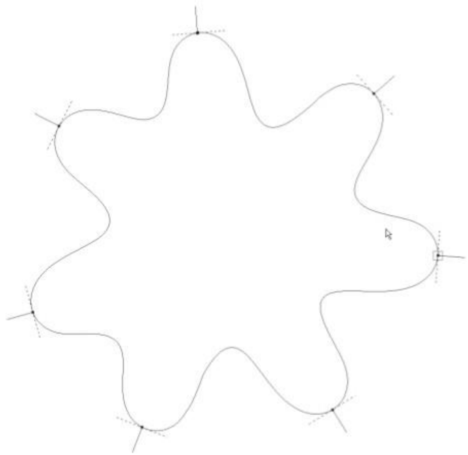


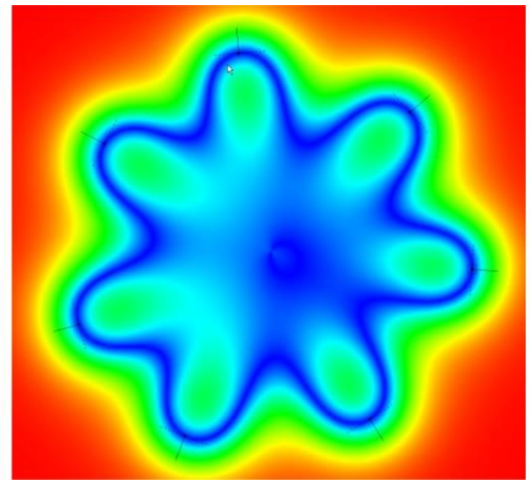
- On utilise $$\begin{cases} \tilde{p}_i(x) = x - ((x-p_i)^T . n_i) n_i \\ c(x) = \sum_{p_i \in NN(x)} w_i \tilde{p}_i(x) / \sum_{p_i \in NN(x)} w_i \\ n(x) = \sum_{p_i \in NN(x)} w_i n_i / \sum_{p_i \in NN(x)} w_i \end{cases}$$

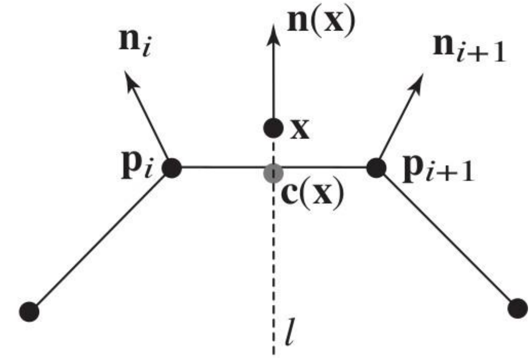
- Schémas populaires de projection itérative :
  - Simple : $x_{k+1} = project(x_k, (c(x_k), n(n_k)))$
  - « Presque orthogonal » : $x_{k+1} = project(x, (c(x_k), n(n_k)))$

« Hermite Point set surfaces »

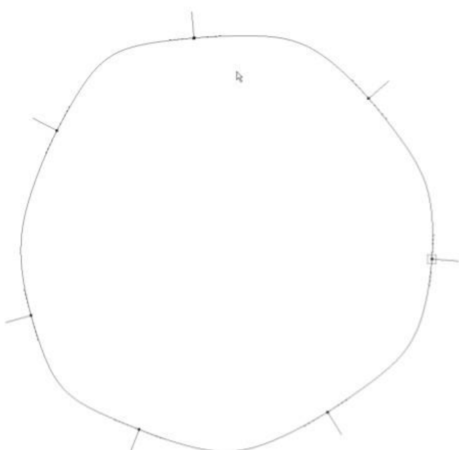
PSS

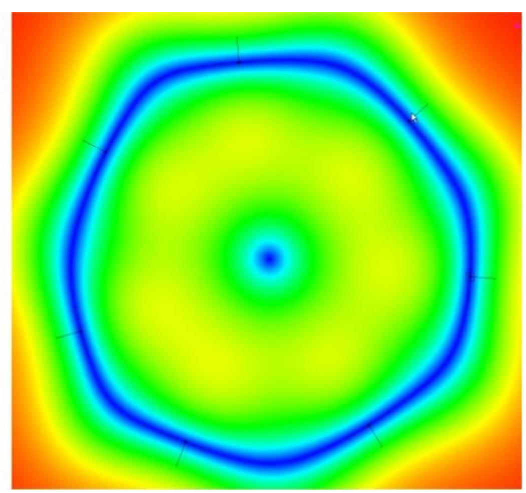


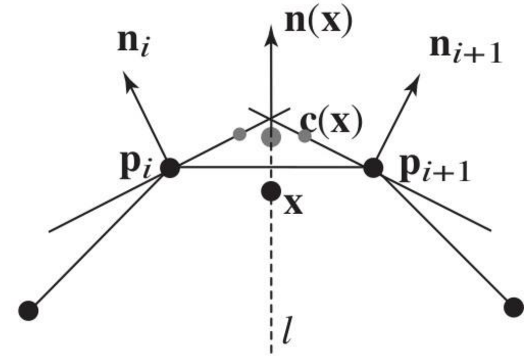




HPSS





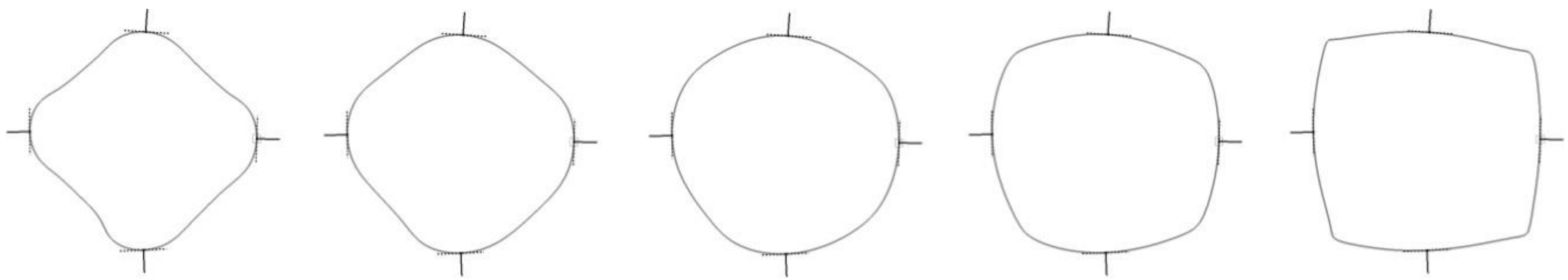


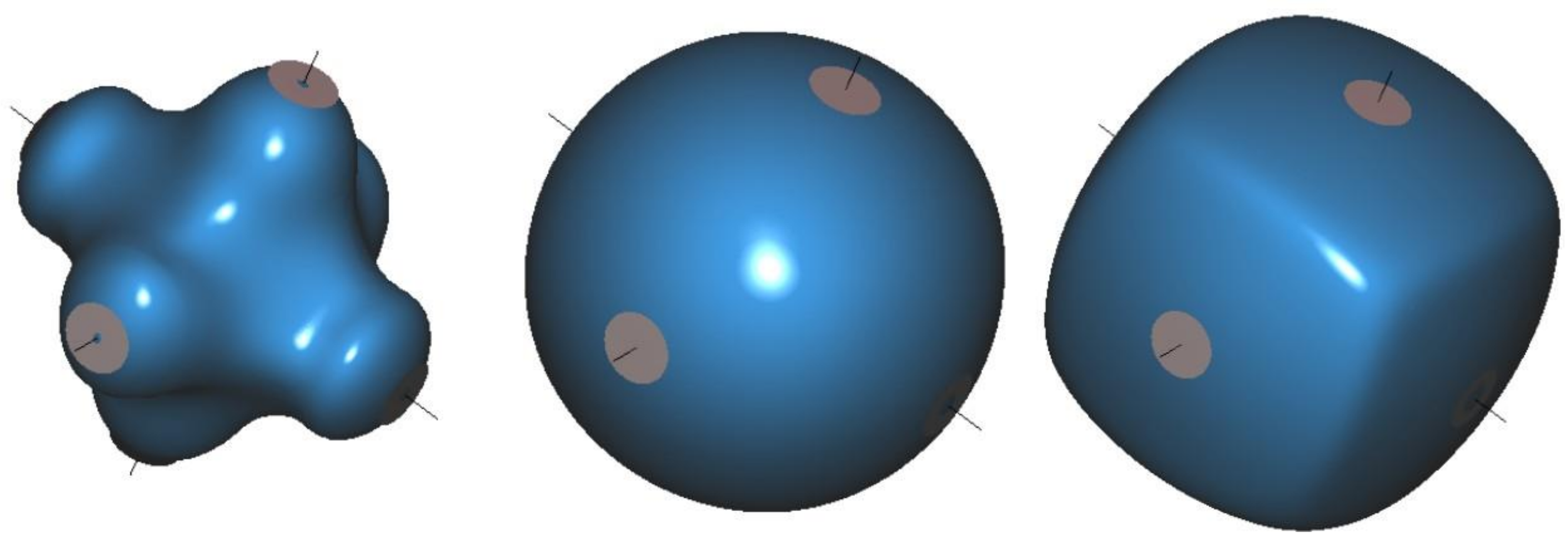
$$f(x) = (x - c(x))^T . n(x)$$

« Hermite Point set surfaces »

Extension : $\widetilde{p}_i(x,s) = s\,\widetilde{p}_i(x) + (1-s)\,p_i$

s=0.8 s=0.9 s=1 s=1.1 s=1.2



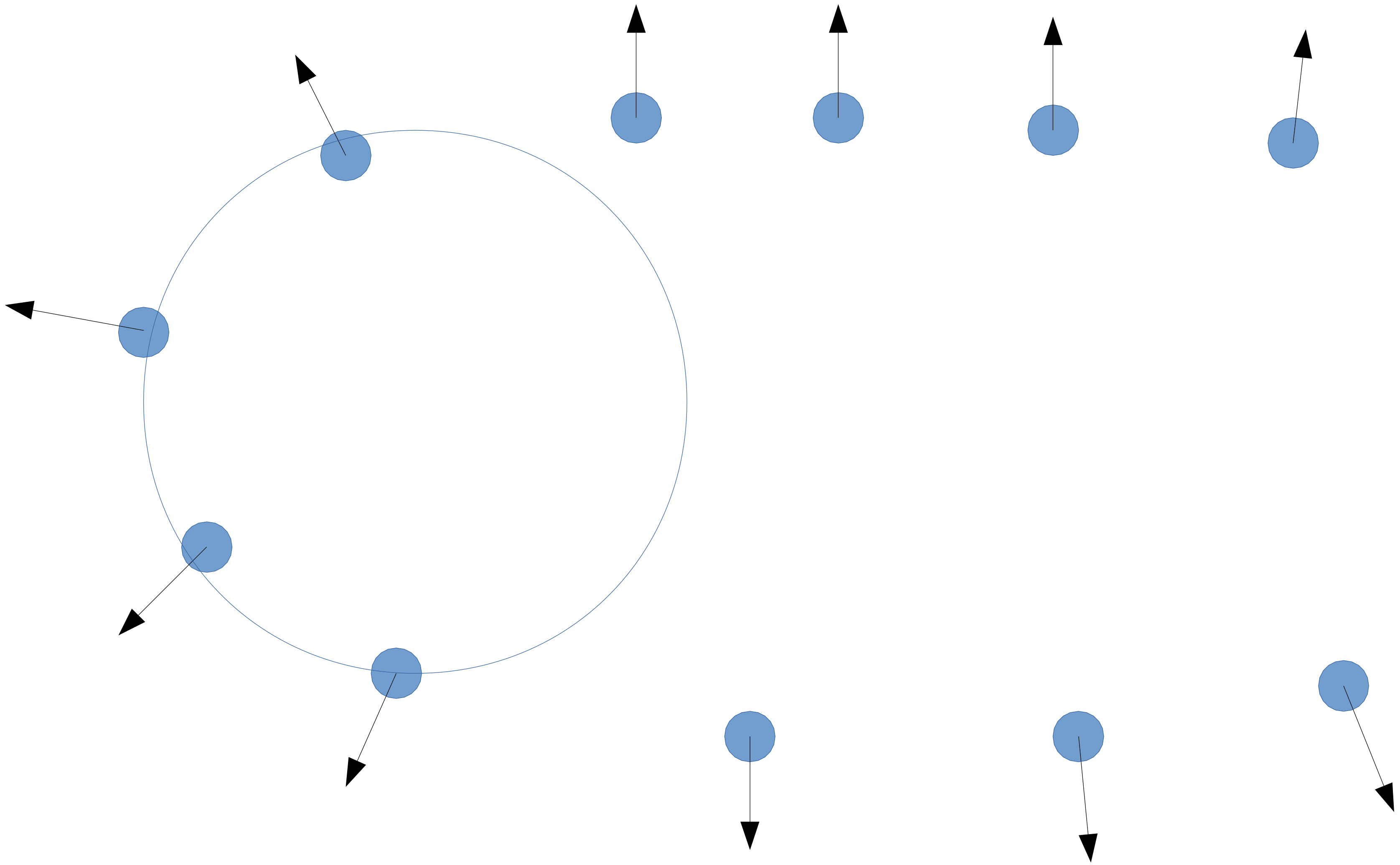


s=0.8 s=1 s=1.1

Questions ?

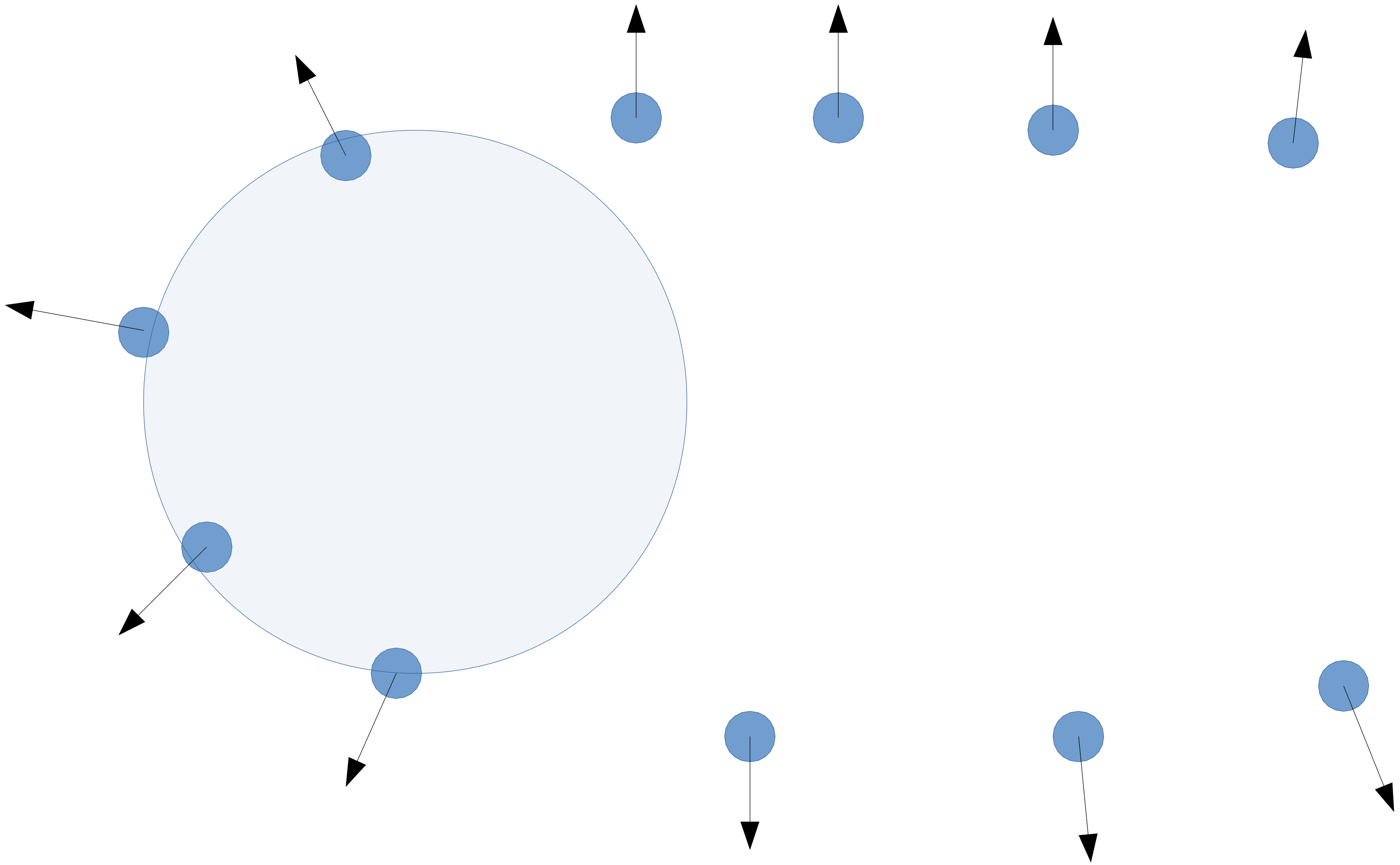
?

Surface de points algébrique



Surface de points algébrique

- Idée principale : projeter sur des sphères plutôt que des plans
- « Algebrique », car une sphère dégénère mal vers un plan (le centre est a l'infini…), mais **son équation algébrique dégénère continûment vers celle du plan**...



Surface de points algébrique

- Idée principale : projeter sur des sphères plutôt que des plans
- « Algebrique », car une sphère dégénère mal vers un plan (le centre est a l'infini…), mais **son équation algébrique dégénère continûment vers celle du plan**...

$$\|X-c\|^2-r^2=0$$

Eq standard d'une sphère

$$(1, X^T, \|X\|^2).(u_0, u_1, u_2, u_3, u_4)^T=0$$

Eq générale d'une sphère algébrique

$$1+\frac{n^T}{d}.X=0$$

Eq standard d'un plan

Surface de points algébrique

$$(1, X^T, \|X\|^2).(u_0, u_1, u_2, u_3, u_4)^T = f(X)$$ Eq générale d'une sphère algébrique

$$f(X)=0$$ Points sur la sphère

$$\nabla f(X)$$ Normale sur la sphère

Surface de points algébrique

$$(1, X^T, \|X\|^2).(u_0, u_1, u_2, u_3, u_4)^T = f(X)$$ Eq générale d'une sphère algébrique

$f(X)=0$ Points sur la sphère

$\nabla f(X)$ Normale sur la sphère

Fitting en deux temps d'une sphère a un ensemble de points : $\{w_i, p_i, n_i\}$

1) Minimiser $\sum_i w_i \|\nabla f(p_i) - n_i\|^2$ (cela définit u_1, u_2, u_3, u_4)

2) Minimiser $\sum_i w_i (f(p_i))^2$ (cela définit u_0)

Stratégie adoptée par :
[Guennebaud et al. 2008] *Dynamic Sampling and Rendering of APSS*

Minimiser l'équation normale

1) Minimiser $\sum_i w_i \|\nabla f(p_i) - n_i\|^2$ (cela définit u_1, u_2, u_3, u_4)

$$f(X) = (1, X^T, \|X\|^2).(u_0, u_1, u_2, u_3, u_4)^T \longrightarrow \nabla f(X) = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} + 2u_4 X = (I_3 | 2X).\begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix}$$

Ou $\sum_i w_i \|\nabla f(p_i) - n_i\|^2 = \sum_i w_i \nabla f(p_i)^T . \nabla f(p_i) - 2\sum_i w_i \nabla f(p_i)^T . n_i + const$

$\longrightarrow$ Minimiser $\sum_i w_i \nabla f(p_i)^T . \nabla f(p_i) - 2\sum_i w_i \nabla f(p_i)^T . n_i$

qui vaut $U^T.(\sum_i w_i (I_3, 2p_i)^T.(I_3, 2p_i)).U - 2(\sum_i w_i (I_3, 2p_i)^T . n_i)^T.U$

Minimiser l'équation de position

1) Minimiser $\sum_i w_i (f(p_i))^2$ (cela définit u_0)

$$f(X) = (1, X^T, \|X\|^2).(u_0, u_1, u_2, u_3, u_4)^T$$

$$f(p_i) = u_0 - (p_i^T, \|p_i\|^2).(u_1, u_2, u_3, u_4)^T$$

→ Simple moyennage barycentrique

Solution exacte

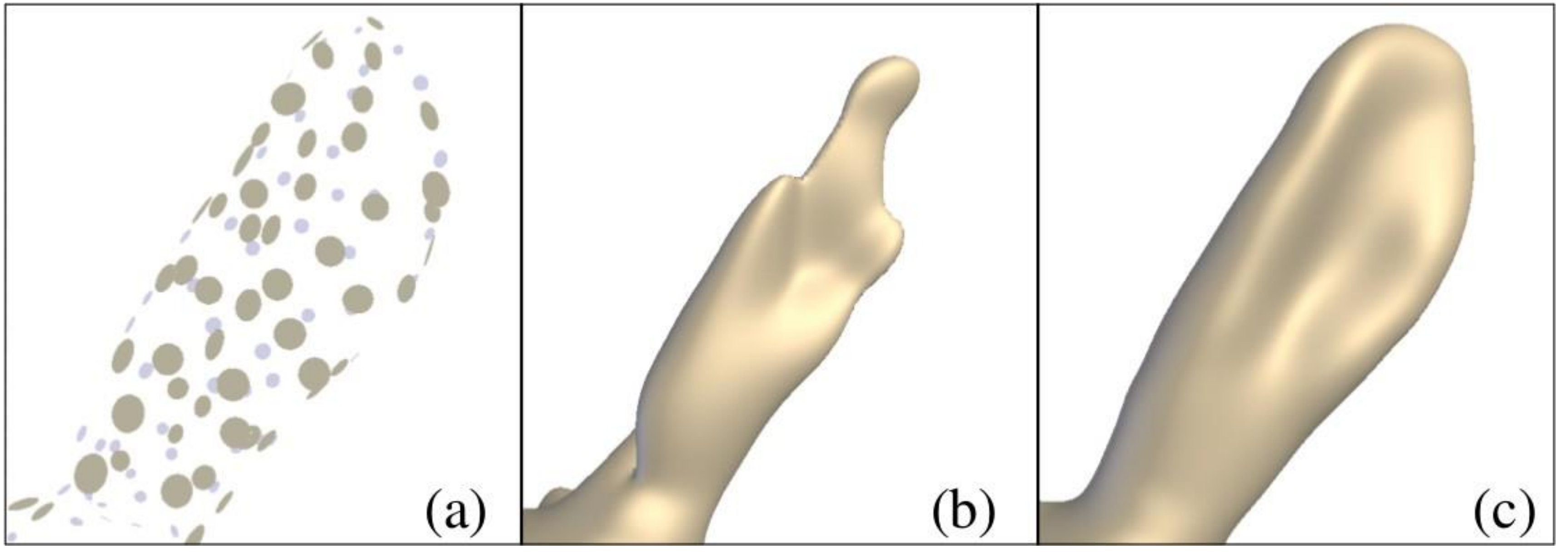
$$\left\{\begin{array}{l} w_j \leftarrow \dfrac{w_j}{\sum_i w_i} \qquad \text{(normalisation des poids)} \\[2em] u_4 = \dfrac{1}{2} \dfrac{\left(\sum_i w_i p_i^T . n_i\right) - \left(\sum_i w_i p_i\right)^T . \left(\sum_j w_j n_j\right)}{\left(\sum_i w_i p_i^T . p_i\right) - \left(\sum_i w_i p_i\right)^T . \left(\sum_j w_j p_j\right)} \\[2em] \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \sum_i w_i (n_i - 2 u_4 p_i) \\[2em] u_0 = -\sum_i w_i ([u_1 u_2 u_3] + u_4 p_i)^T . p_i \end{array}\right.$$

Surface de points algébrique

Note :

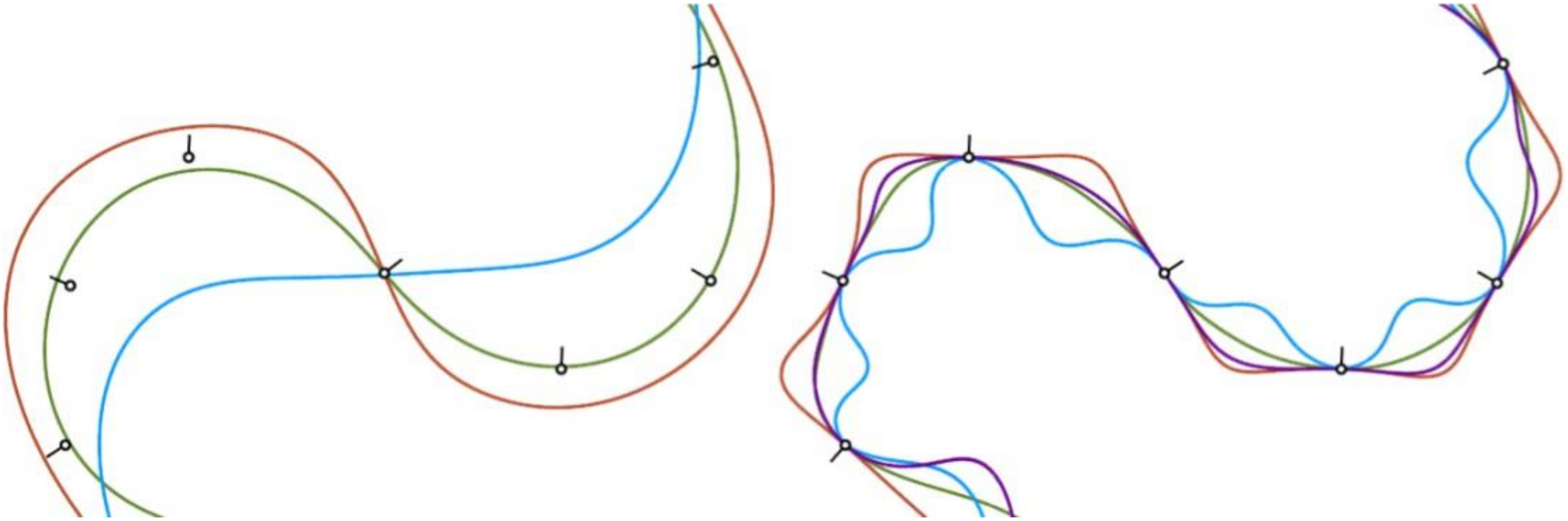
- Si on force $u_4 = 0$, on cherche le meilleur plan défini par les points les plus proches de x, avant de projeter x dessus.
- On retrouve dans ce cas la le modèle SPSS (qui est moins bon que HPSS…)
- → APSS étend le modèle SPSS (d’une autre manière que HPSS)

Surface de points algébrique



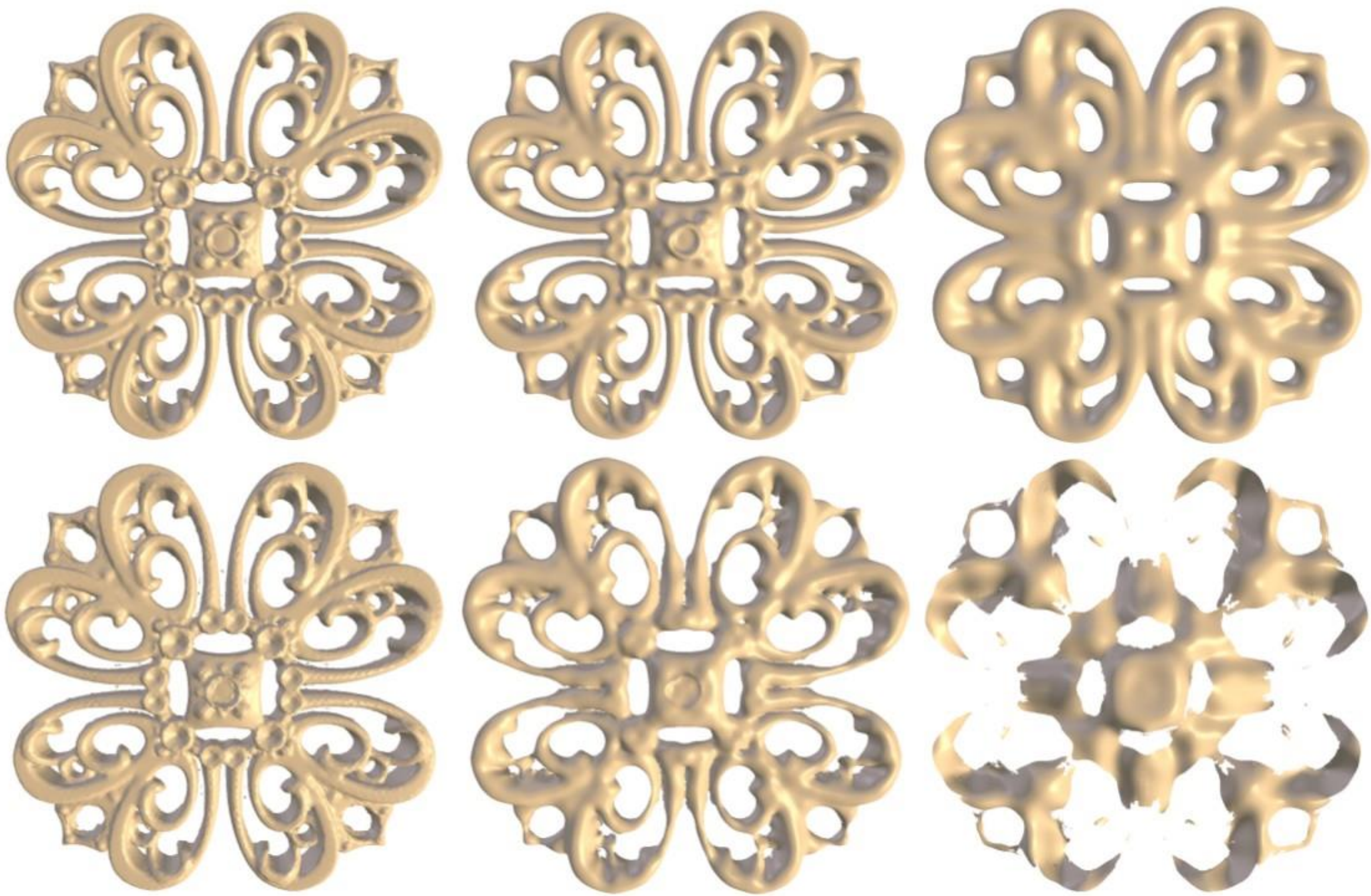
SPSS APSS

Surface de points algébrique



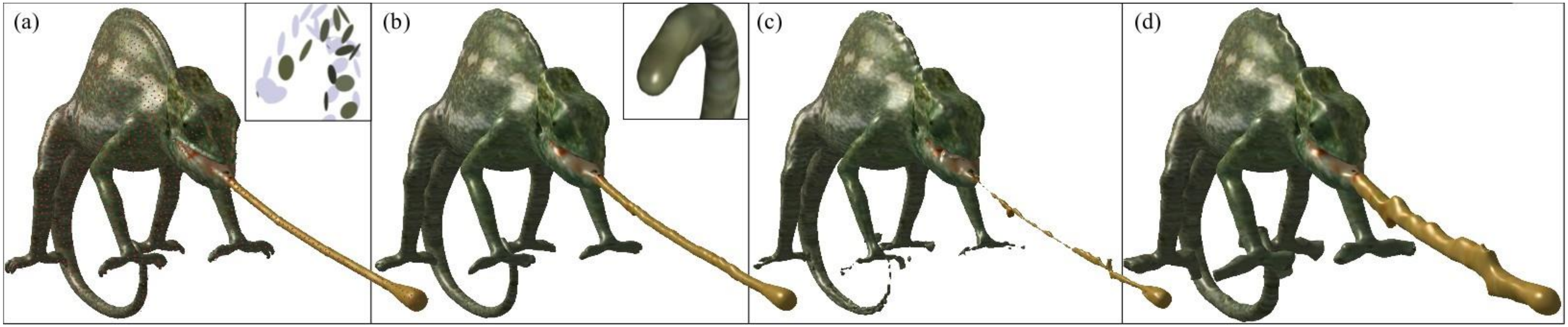
SPSS ***IMLS*** ***HPSS*** ***APSS***

Taille du noyau approximant variant



Comparaison

Downsampling *APSS* *SPSS* *IMLS*



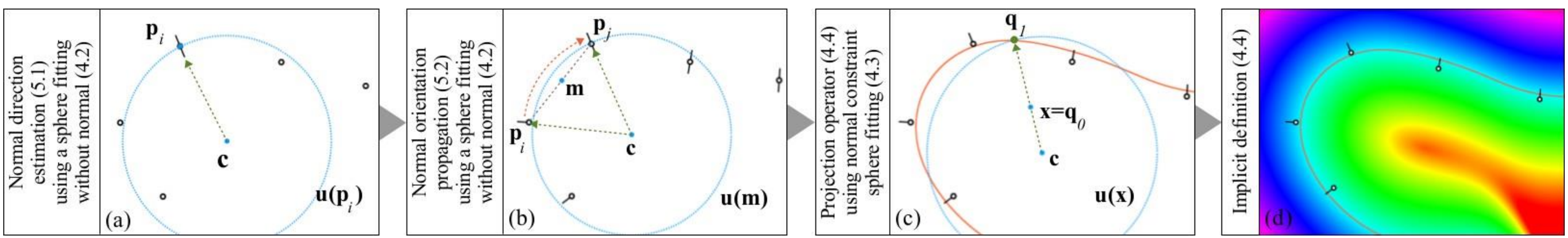
Édition de courbure

$$
\begin{cases}
u_4 = \beta \dfrac{1}{2} \dfrac{\left(\sum_i w_i p_i^T . n_i\right) - \left(\sum_i w_i p_i\right)^T . \left(\sum_j w_j n_j\right) / \left(\sum_k w_k\right)}{\left(\sum_i w_i p_i^T . p_i\right) - \left(\sum_i w_i p_i\right)^T . \left(\sum_j w_j p_j\right) / \left(\sum_k w_k\right)} \\[2ex]
\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \dfrac{\sum_i w_i (n_i - 2 u_4 p_i)}{\sum_i w_i}
\end{cases}
$$

$$
u_0 = \frac{-\sum_i w_i ([\, u_1 u_2 u_3] + p_i)^T . p_i}{\sum_i w_i}
$$

Estimation des normales et fonction implicite

$$f(X) = (1, X^T, \|X\|^2) . U(X)$$



Pour les nuages non-orientes, les auteurs proposent de fitter des spheres aux points sans les normales, de definir la normale ainsi, et de propager l’orientation (distinction des zones convexes et concaves)

Questions ?

?