Probabilistic Robotics Course

Finding Neighbors

Giorgio Grisetti
grisetti@dis.uniroma1.it

Dept of Computer Control and Management Engineering
Sapienza University of Rome

Neighbor Search

Given

- a collection of vectors $\mathcal{P} = \{\mathbf{p}_n\}_{n=1:N} \qquad \mathbf{p}_n \in \Re^k$
- a query vector $\mathbf{p}_q \in \Re^k$
- a distance metric $d(\mathbf{p}_n, \mathbf{p}_q) \in \Re^+$

Find either

- the point in the collection that is “closer” to the query, according to the metric

$$\mathbf{p}_i = \underset{\mathbf{p}_n \in \mathcal{P}}{\operatorname{argmin}}\, d(\mathbf{p}_q, \mathbf{p}_n)$$

- the points in the collection whose distance from the query is smaller than a value ϵ

$$\mathcal{P}' = \{\mathbf{p}_i \in \mathcal{P}, d(\mathbf{p}_q, \mathbf{p}_i) < \epsilon\}$$

Distance Metrics

Examples

- Squared Norm

$$\|\mathbf{p}_i - \mathbf{p}_j\|^2 = (\mathbf{p}_i - \mathbf{p}_j)^T (\mathbf{p}_i - \mathbf{p}_j)$$

- Omega Norm

this should look familiar

$$\|\mathbf{p}_i - \mathbf{p}_j\|_{\mathbf{\Omega}}^2 = (\mathbf{p}_i - \mathbf{p}_j)^T \mathbf{\Omega} (\mathbf{p}_i - \mathbf{p}_j)$$

- Hamming distance (for binary descriptors)

  Integer valued distance between two bit strings having the same dimension. Its value is the number of bit flips required to turn a string into the other

  example:

  hamming(100100,101000)= 2

Trivial Approach

Brute Force:

compute the distance metric between the query point and *each* of the points in the collection and update the minimum

Complexity: `O(N*cost_distance_metric)`

If we need to perform many queries, this results in unacceptable delays

Idea: use auxiliary search structures

Distance Map

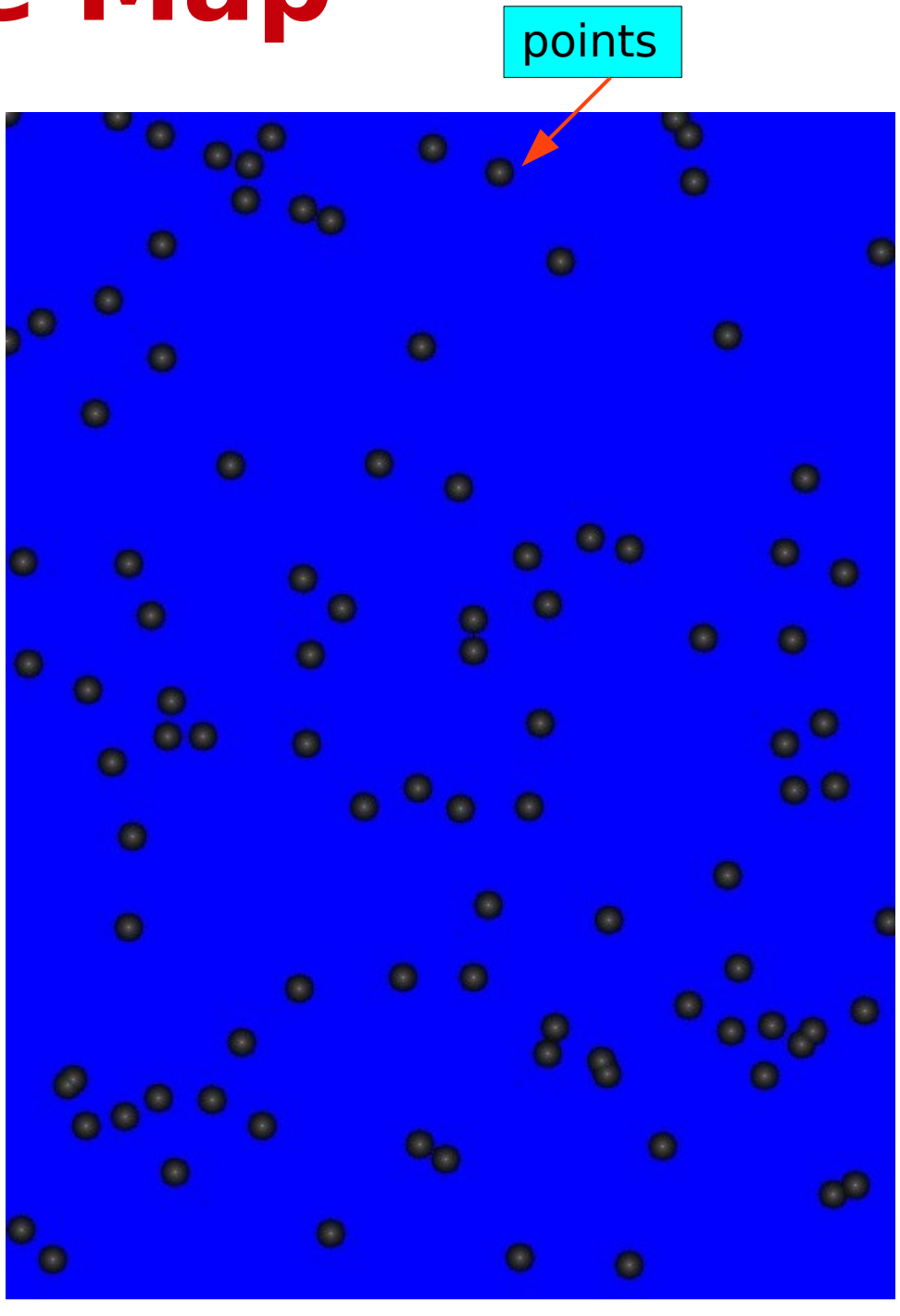
If

- the dimension of the vectors is small (< 3)
- they are spread in a relatively small region of the space

we can pre-compute a grid lookup table

Each cell of the grid contains

- the distance from the closest point
- the identity of the closest point



Distance Map

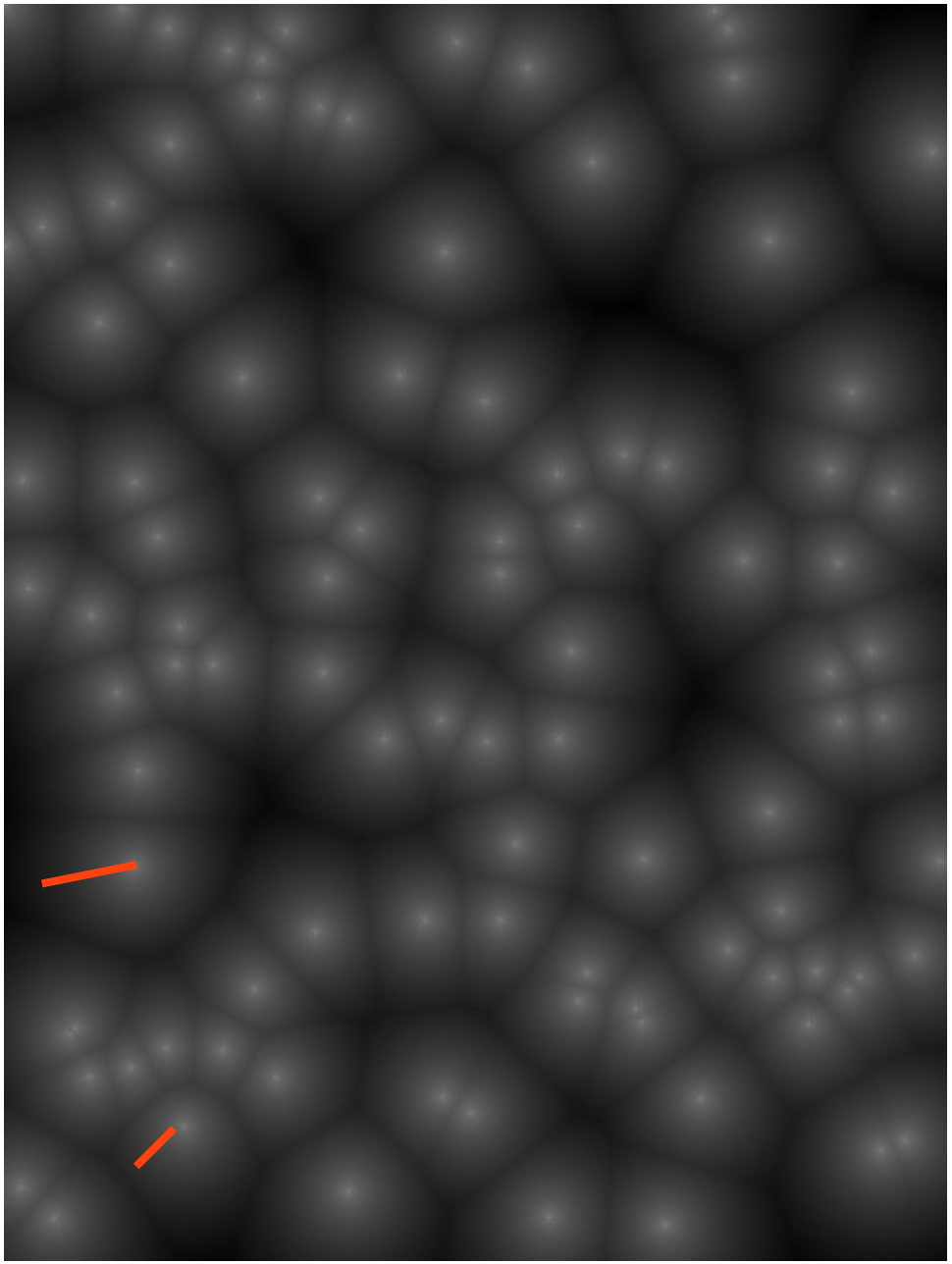
If

- the dimension of the vectors is small (< 3)
- they are spread in a relatively small region of the space

we can pre-compute a grid lookup table

Each cell of the grid contains

- the distance from the closest point (gray value)
- the identity of the closest point (orange line)



Computing the Distance Map

Modified Version of Dijkstra algorithm

Each node **n** stores

- d: distance from “nearest point”
- parent: pointer to the nearest point

A sorted queue **q** contains the nodes to expand

Each time a node $\mathbf{n_{curr}}$ is pulled from the queue, its neighbors $\mathbf{n_i}$ are expanded, and a candidate distance d' is computed as

d'=distance($\mathbf{n_{curr}}$**.parent,ni)**

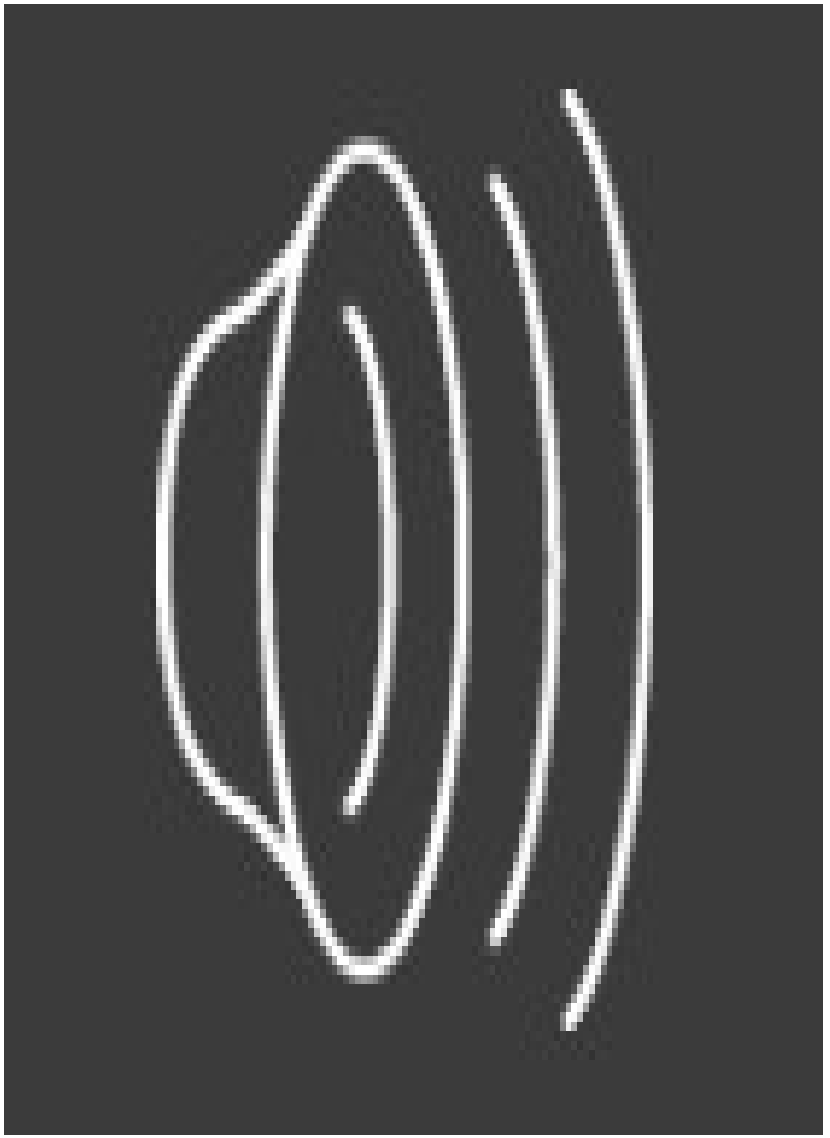
If the current distance is smaller than the previous one, the node is inserted in the queue for further expansion

If d'<ni.d

n_i.d=d';

n_i.parent=n_{curr}.parent;

q.push(n_i);



Distance Map: Complexity

Computing the distance map requires a time

`O(grid_size*log(grid_size))`

One query on the distance map requires

`O(1)`

Good when

- we have many low dimensional points in the collection
- the grid is small
- we can tolerate small association errors

Distance Map: Heuristics

Gating:

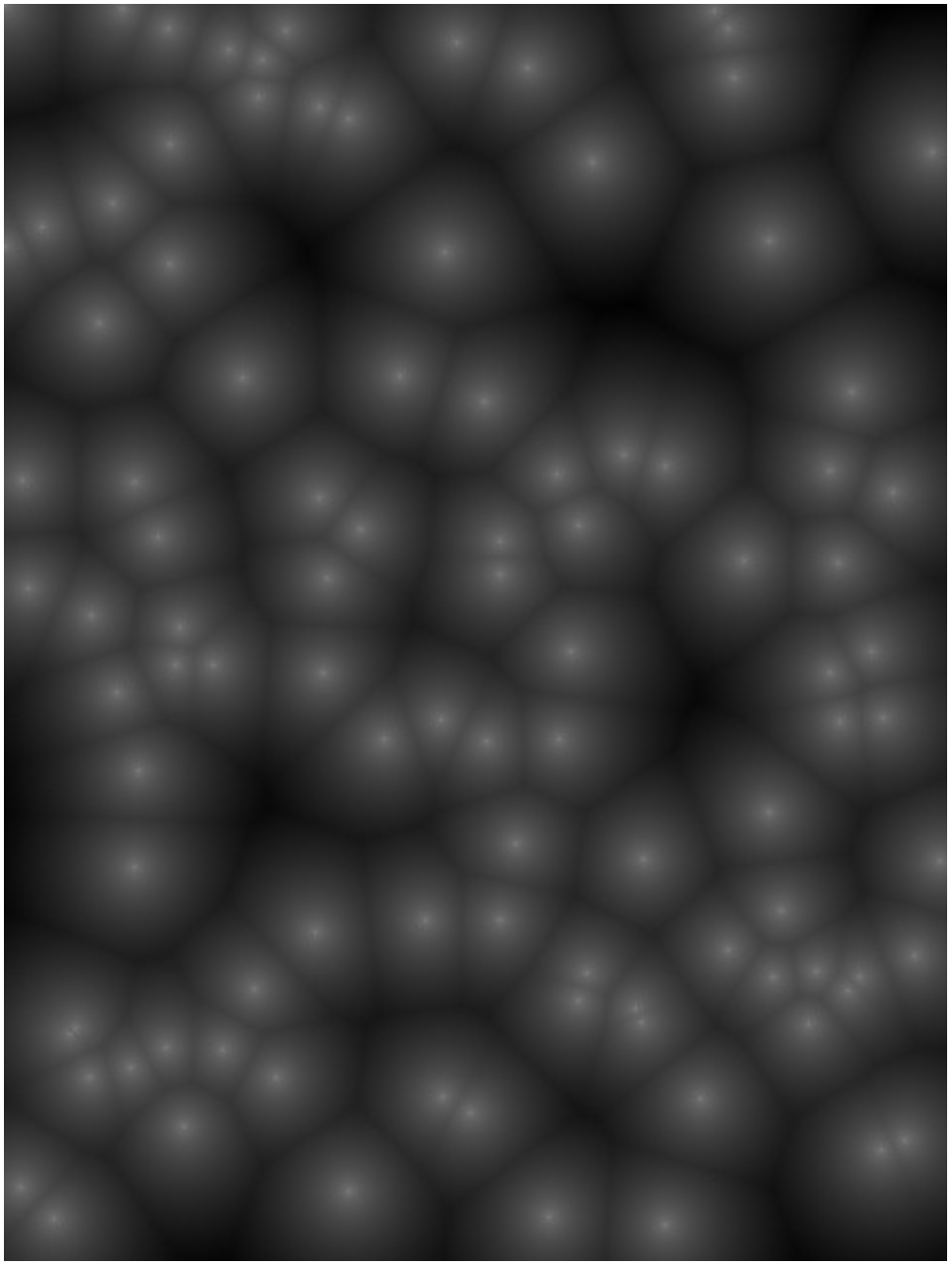
- expand up to a maximum distance

Best friends:

- construct a distance map also for the measurements and cross check

Lonely Best Friends:

- looking up a region of the distance map instead of a single point



KD-Trees

What if the points are K-dimensional, with K large?

- The distance map does not scale well: memory grows exponentially with K, and so the time to construct it

KD-Tree:

- search structure that partitions the query point according to their spatial distribution
- If the tree is balanced, a search takes log(N) with N the number of points

KD-Trees

Constructing KD-trees can be done with a trivial recursion

At each time the set is split in two parts until the number of points in a leaf is smaller than a threshold

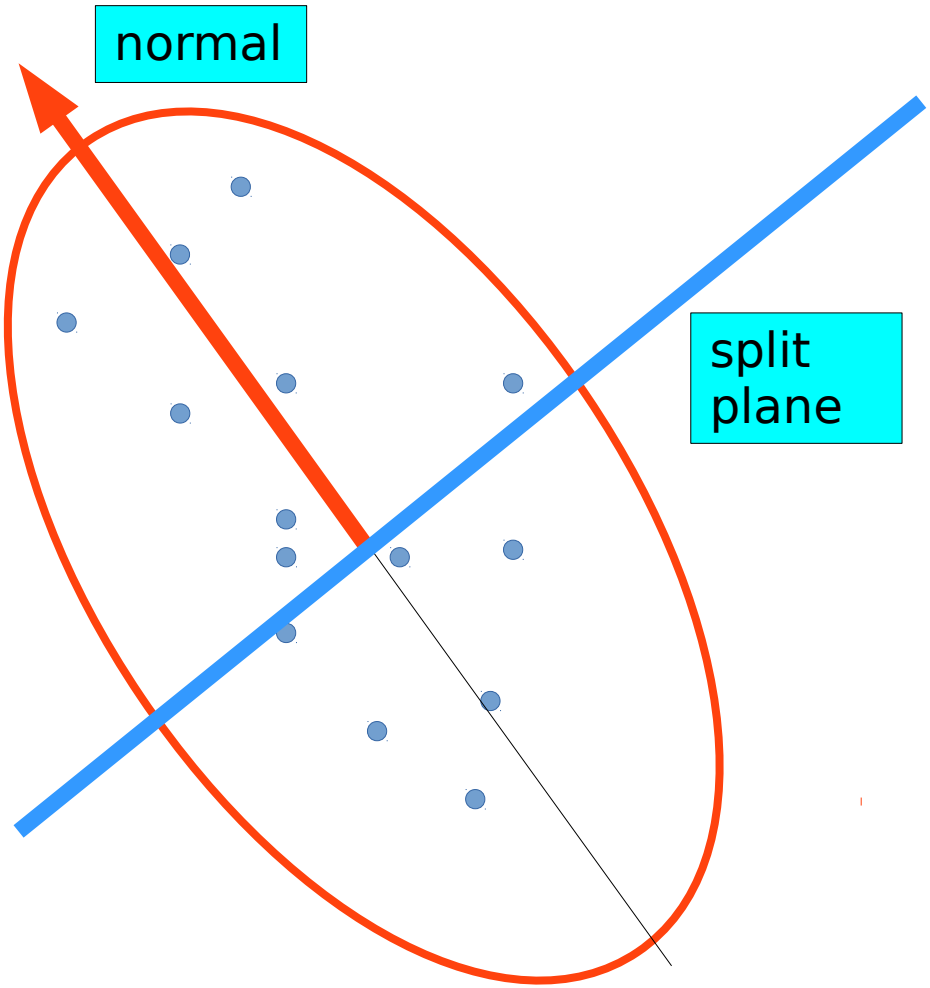
Question:

how to split?

KD-Trees: simple splitting

Consider the n-dimensional points as if they were normally distributed

- Compute the Covariance matrix of the distribution
- Chose a splitting (hyper) plane that passes through the mean and has a normal aligned with the longest axis of the covariance matrix



KD-Trees

A split plane is characterized

- by a “normal” $\mathbf{n}_i$
- a mean μ_i

We can check if a point lies on the one side of the plane by evaluating

$$\mathbf{n}_i^T(\mathbf{p}_q - \mu_i) > 0$$

KD-Trees

Each intermediate node of the tree contains

- The normal of the splitting plane
- The mean
- Pointers to the children nodes

A query requires traversing the tree from the top to the bottom and at each time going left or right

Is a heuristic that might return not the real minimum, depending on how the tree was built

Efficient randomized variants (see ANN c++ library)

Other Tools

Projection to lower dimension

- Reduce the dimension of the query points, by projecting them for instance in 2D
- Use some easy heuristic to perform the association in the lower space

Bag of words

- Used to determine the appearance similarity of multiple points.
- An item of the search collection contains many points.

How to speed up the Localizer?

- What would you use?
- Why?
- How?
- What do you expect?

What about SLAM?

- What would you use?
- Why?
- How?
- What do you expect?