

CS 8803-MDM Lecture 27

Generalized N -Body Problems

Alexander Gray

agray@cc.gatech.edu

Georgia Institute of Technology

Today

1. Basic N -Body Problems
2. Kernel Summation Problems
3. Other N -body Problems

Basic N -body Problems

Basic geometric problems.

Basic N -body Problems

Given a *query* point x_q and a set of *reference* points $\mathcal{R} = \{x_r\}$:

k -nearest-neighbor:

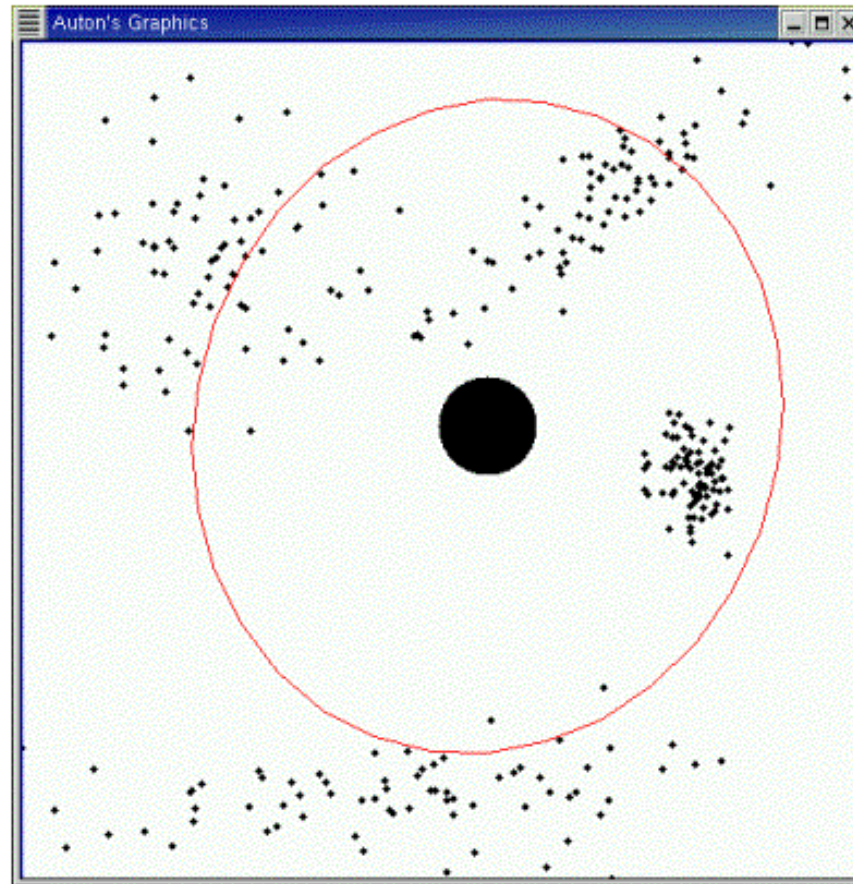
$$\arg \min_r^{(k)} \|x_q - x_r\| \quad (1)$$

Range-count:

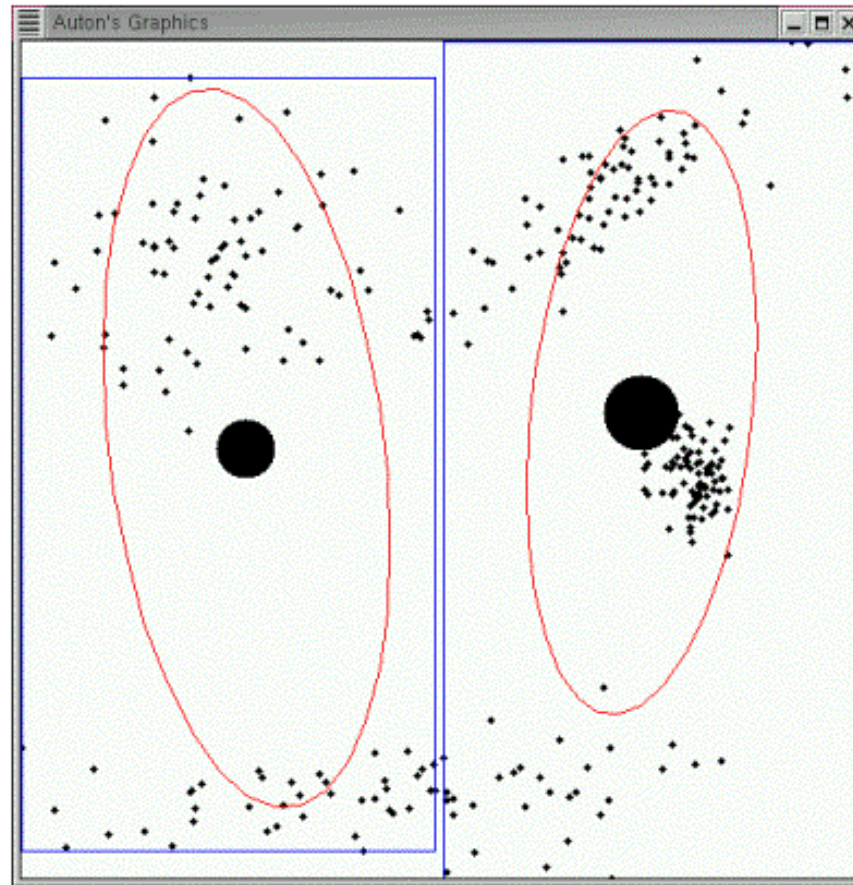
$$\sum_r I(\|x_q - x_r\| < h) \quad (2)$$

Where they come up: nearest-neighbor methods, manifold learning; fractal dimension, spatial statistics, spherical-kernel methods...

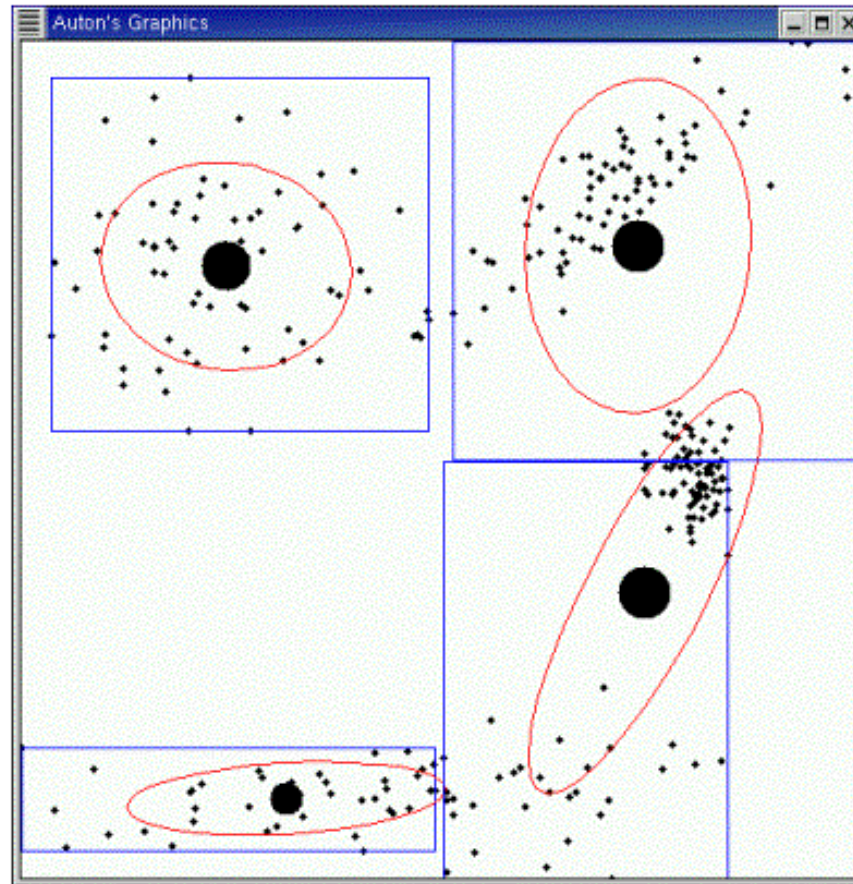
kd-trees



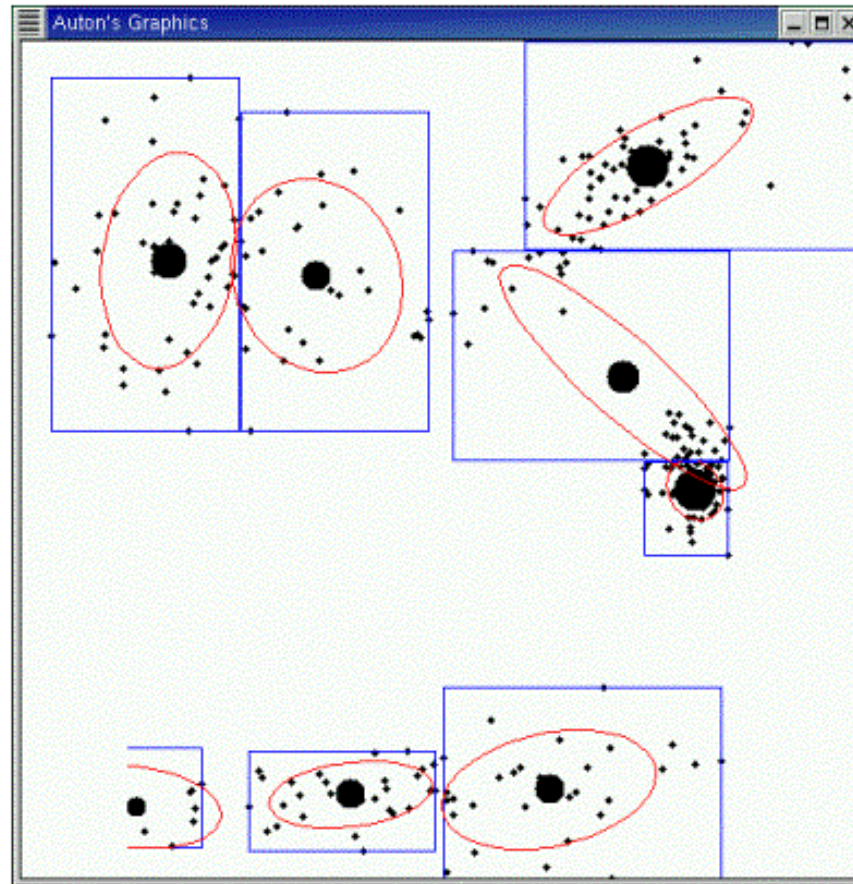
kd-trees



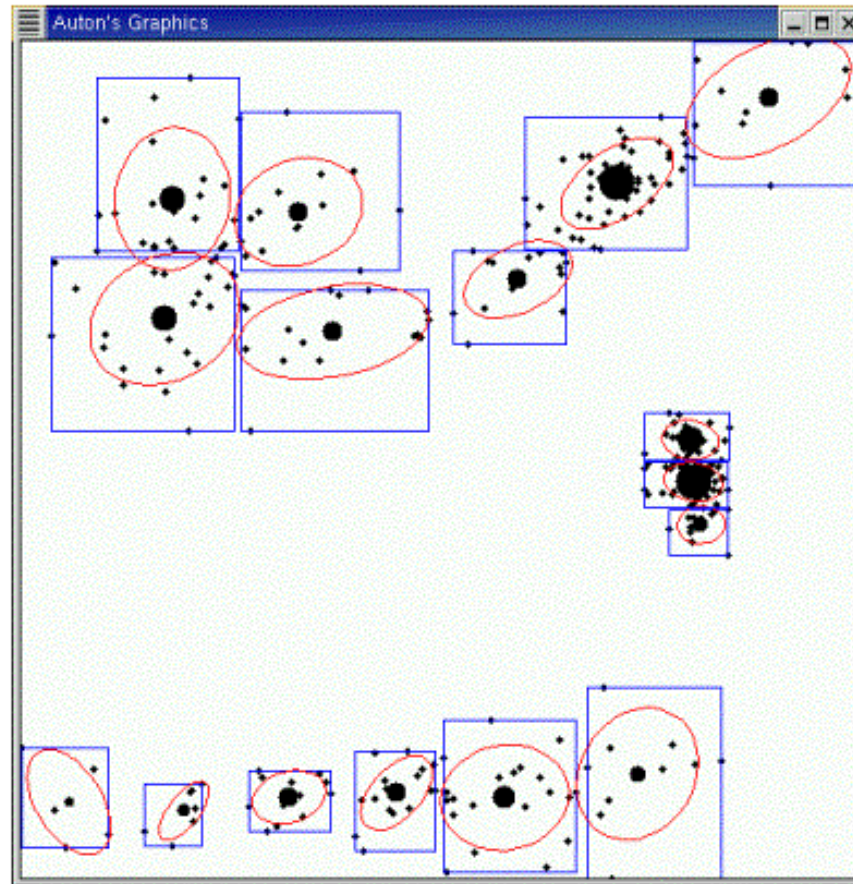
kd-trees



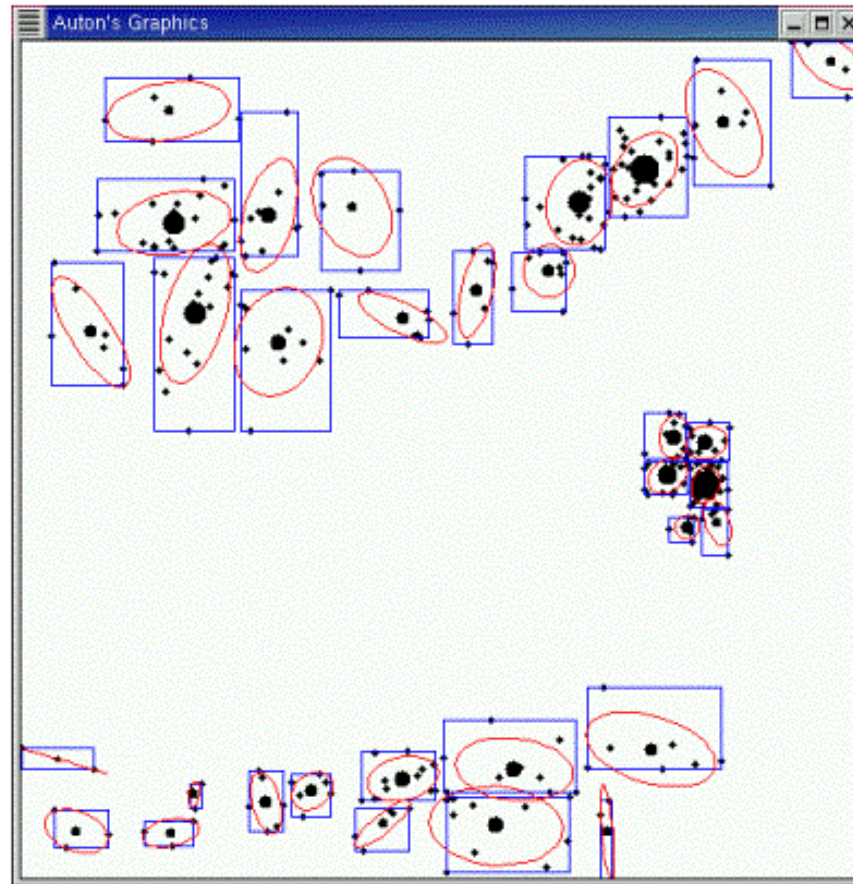
kd-trees



kd-trees

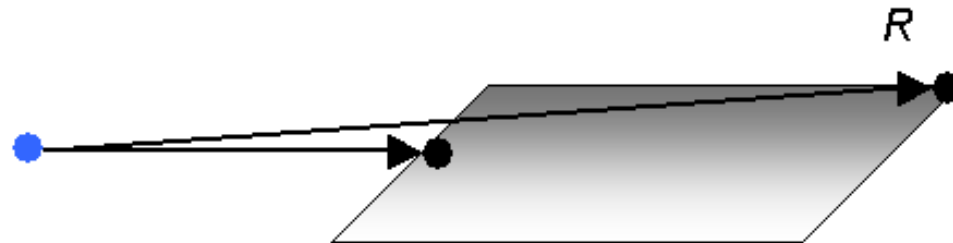


kd-trees

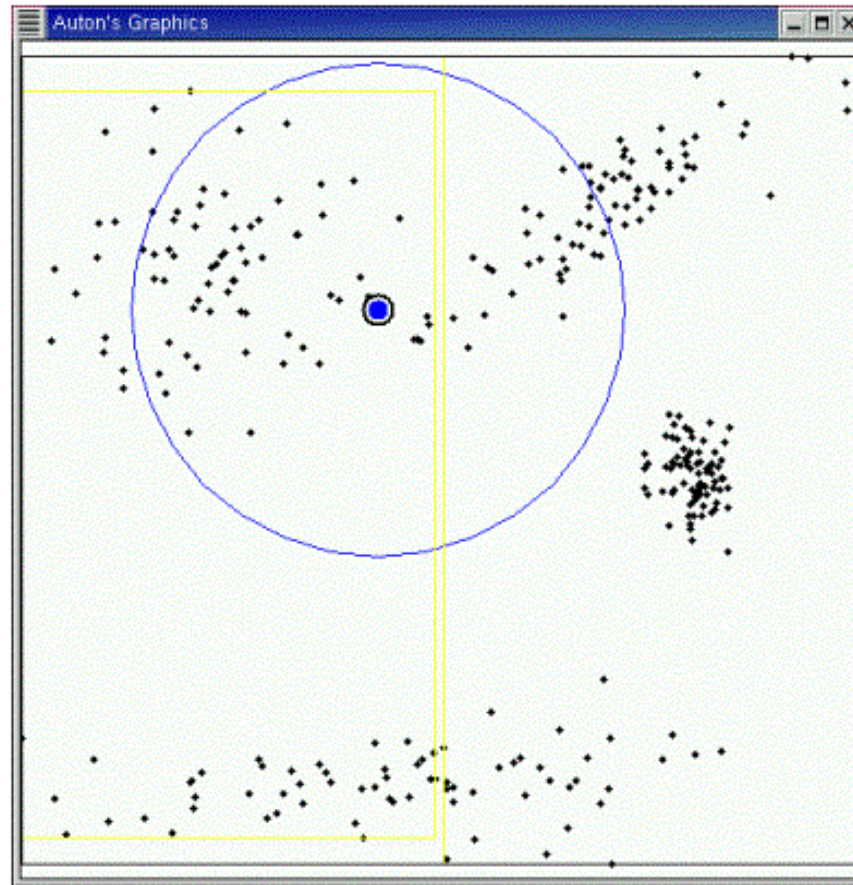


Distance Bounds

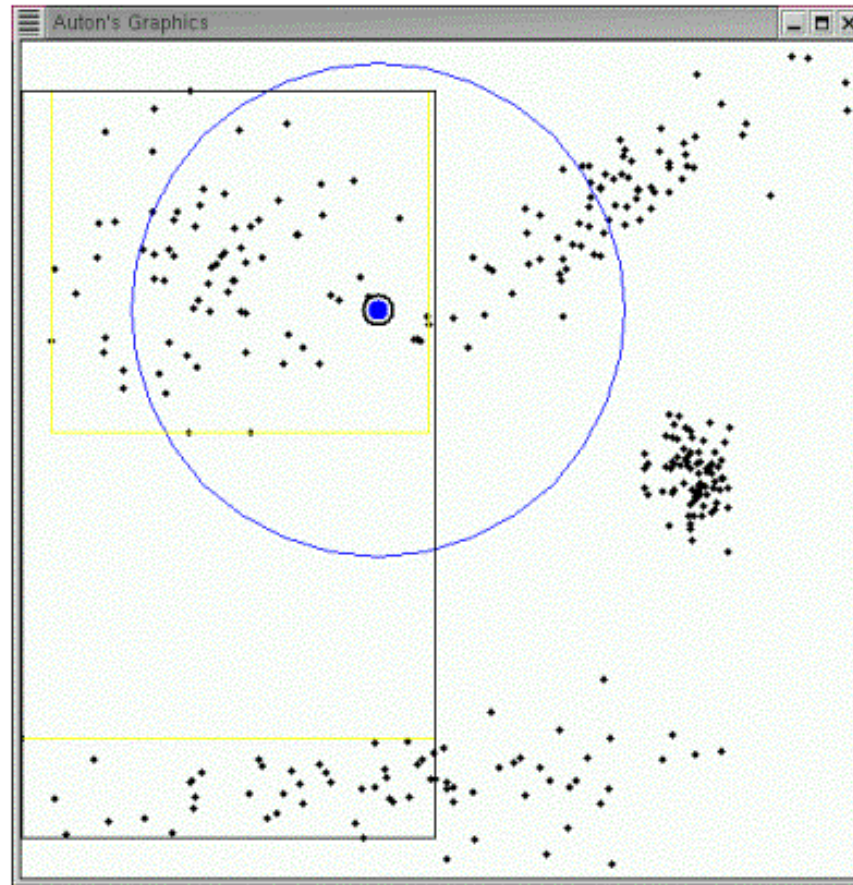
We can compute a lower bound δ_{qR}^L on the distance between x_q and any $x_r \in R$, as well as an upper bound δ_{qR}^U , using the bounding box of the node R , in $O(D)$ time.



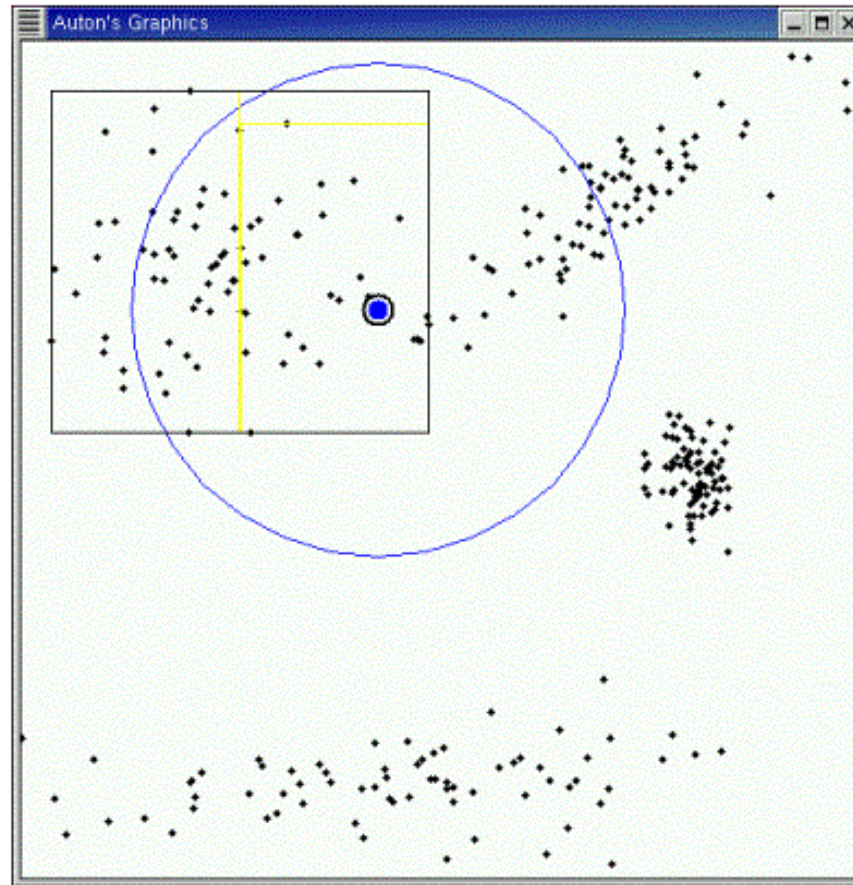
Range-count with kd -trees



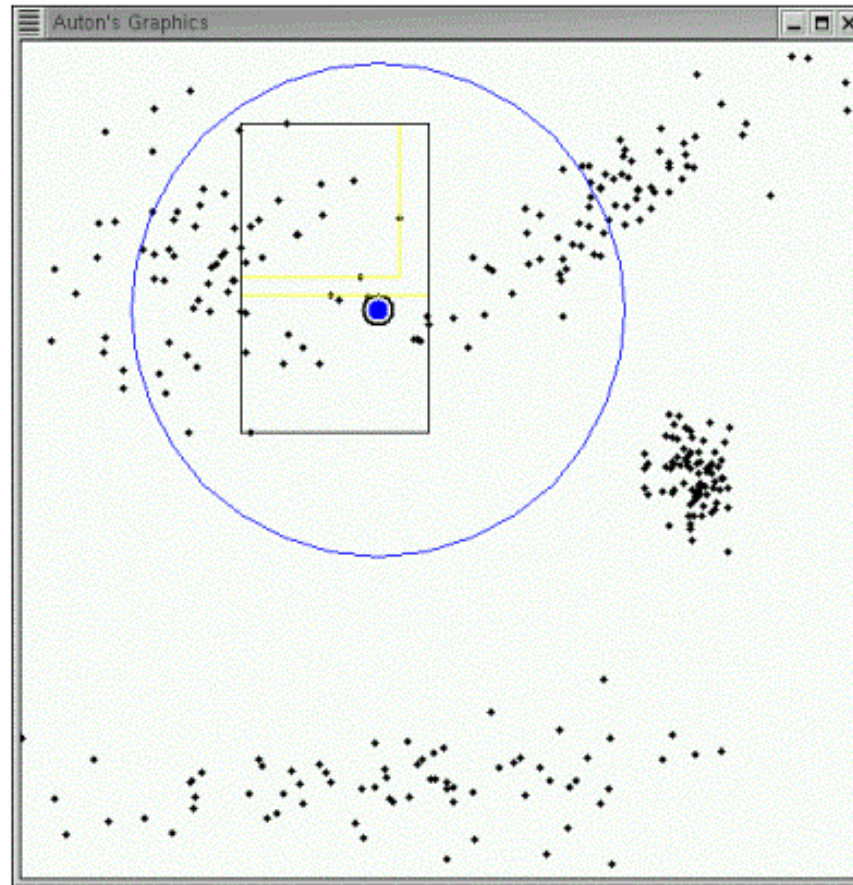
Range-count with kd -trees



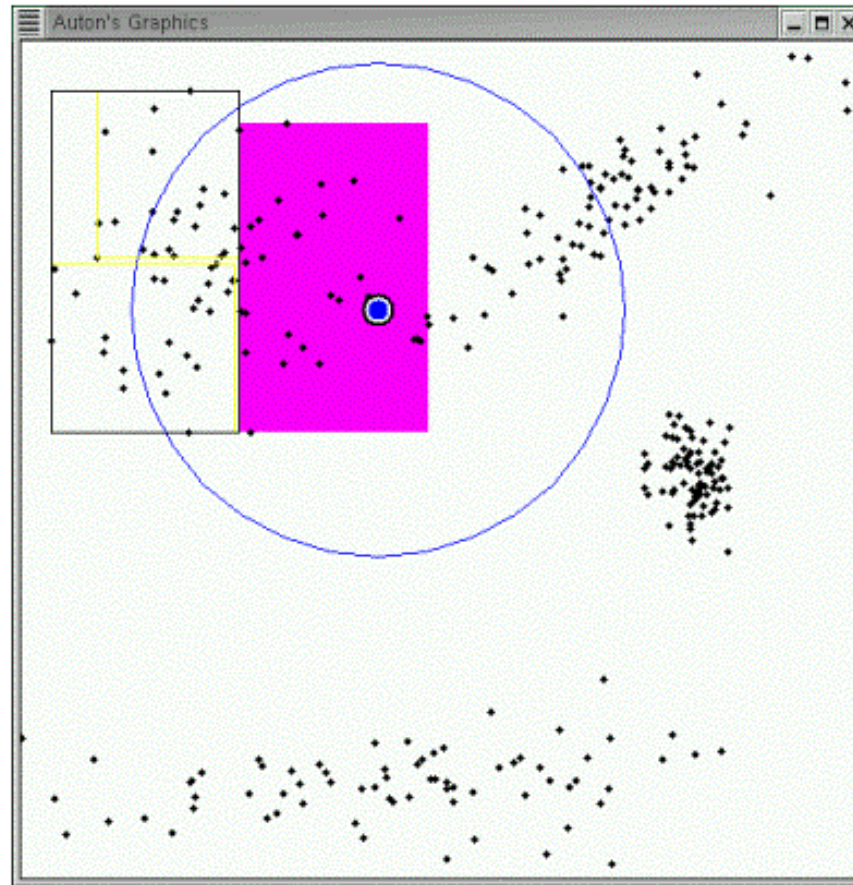
Range-count with kd -trees



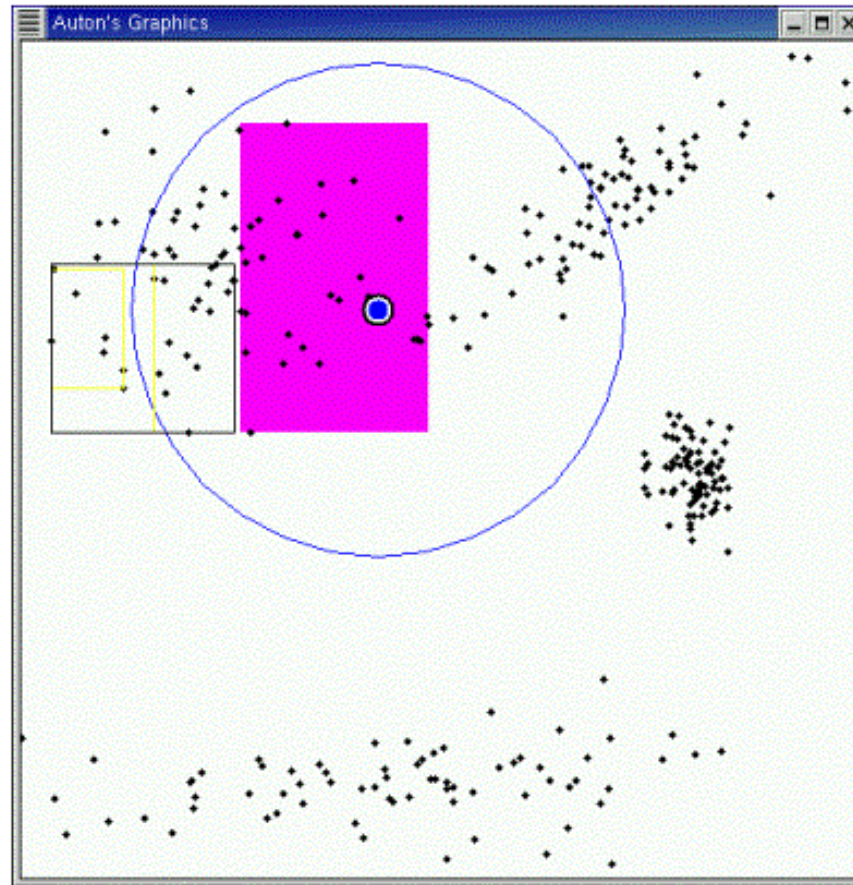
Range-count with kd -trees



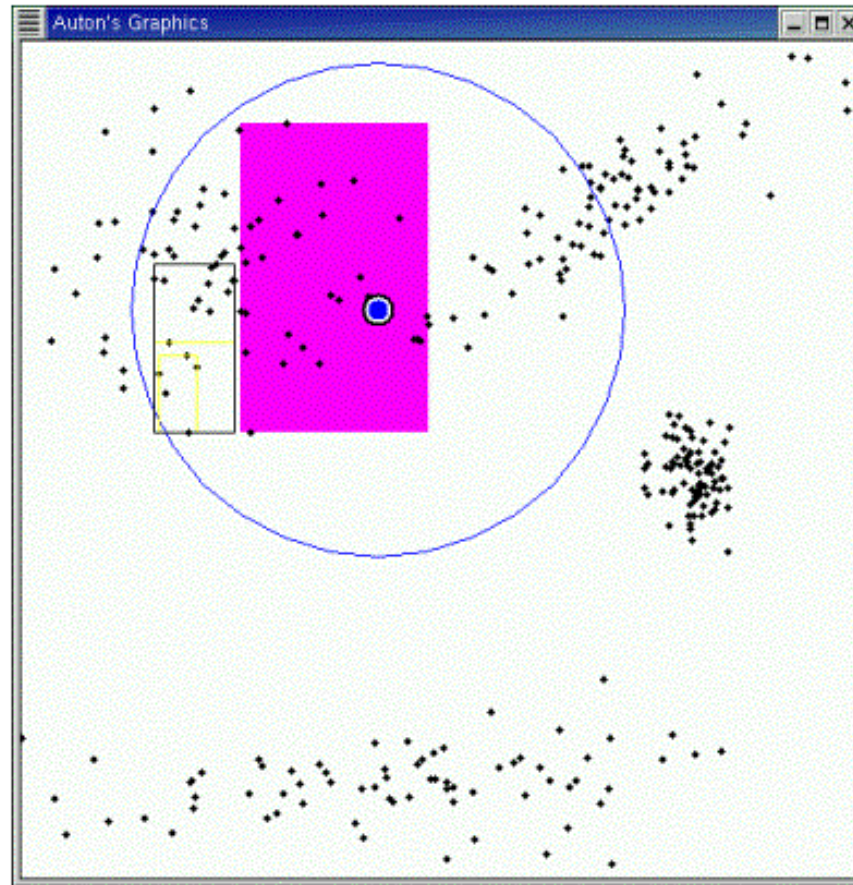
Range-count with kd -trees



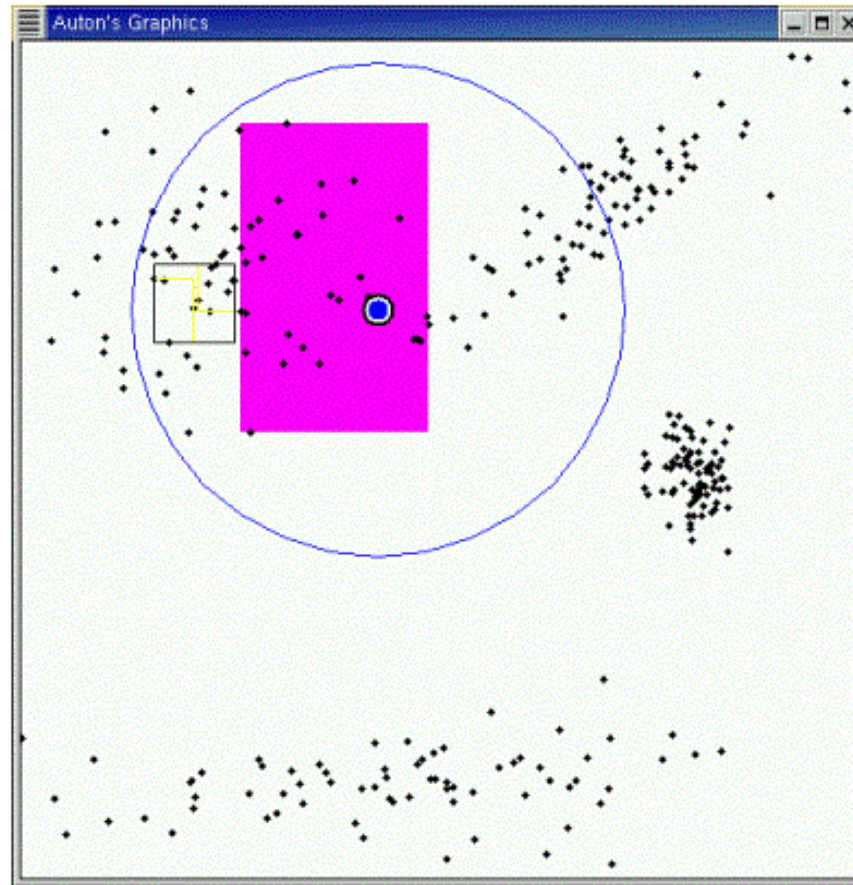
Range-count with kd -trees



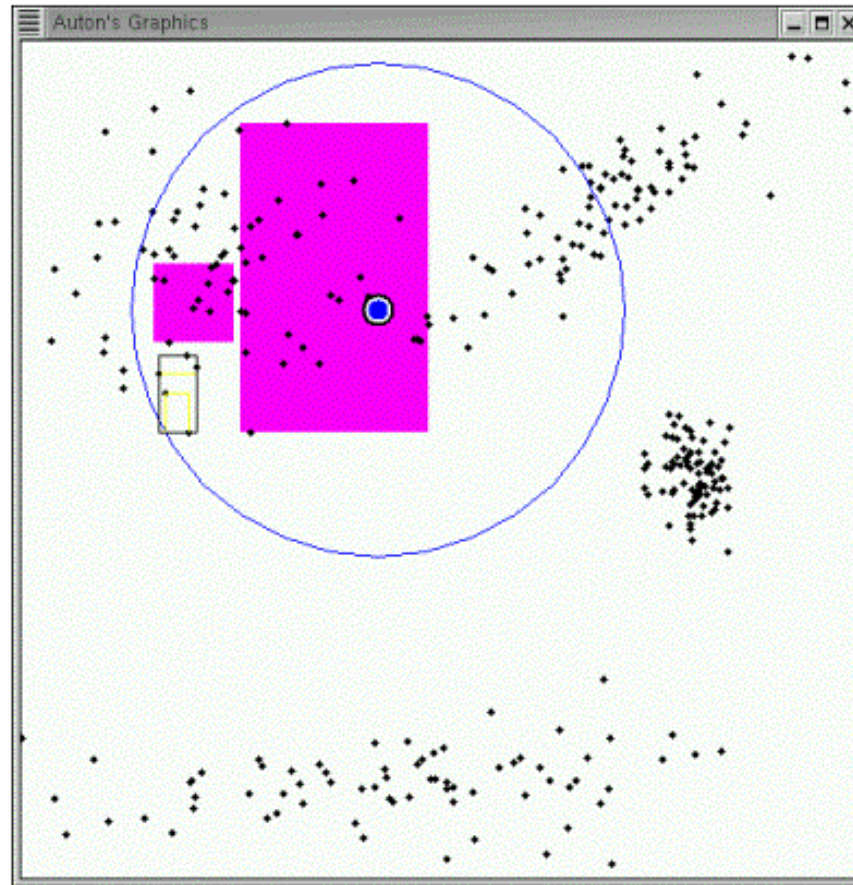
Range-count with kd -trees



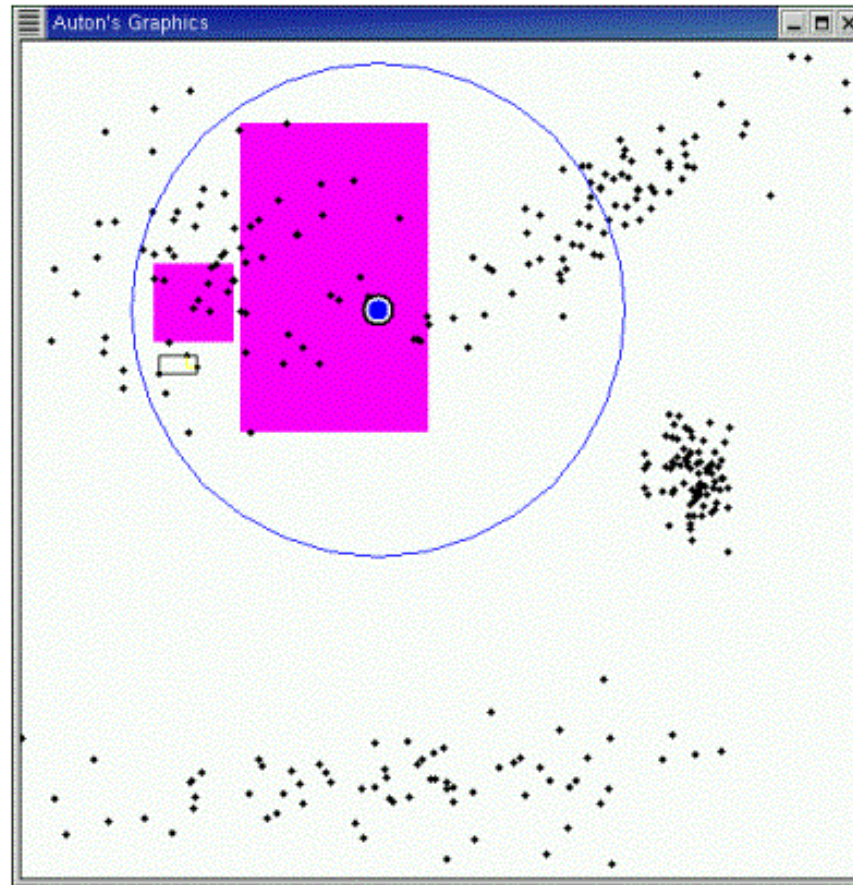
Range-count with kd -trees



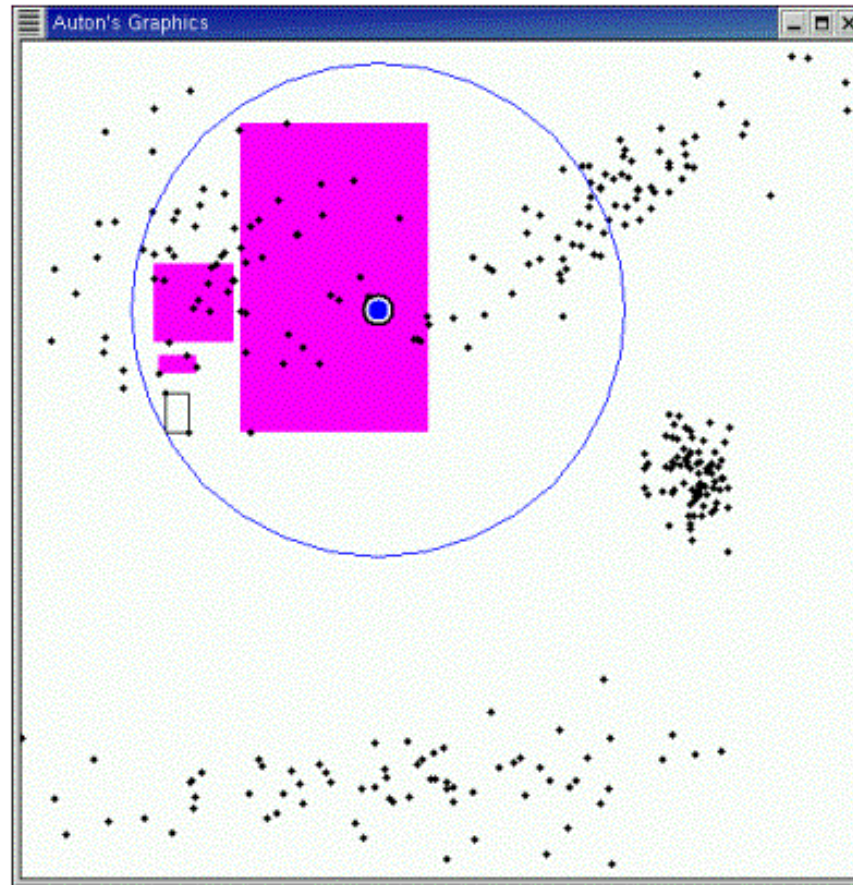
Range-count with kd -trees



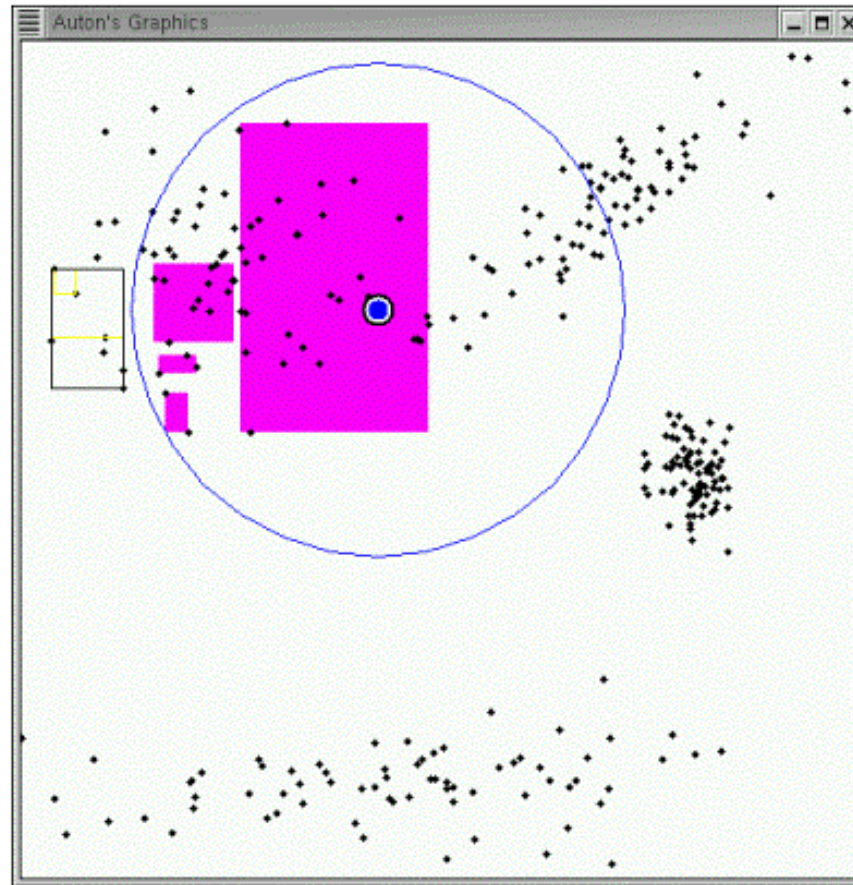
Range-count with kd -trees



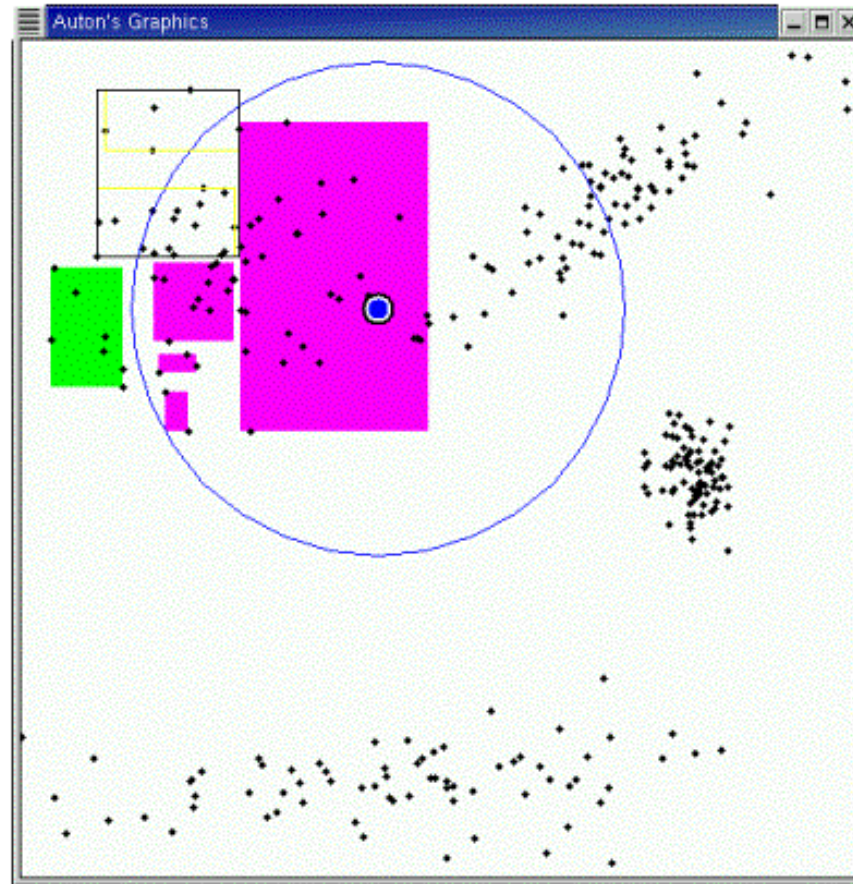
Range-count with kd -trees



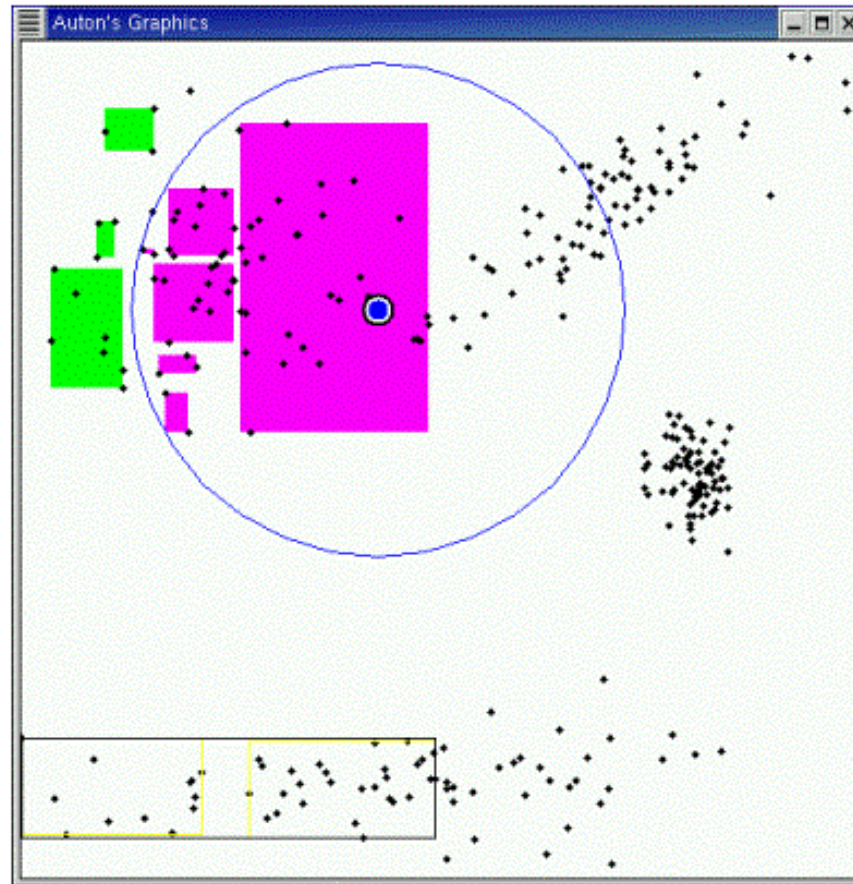
Range-count with kd -trees



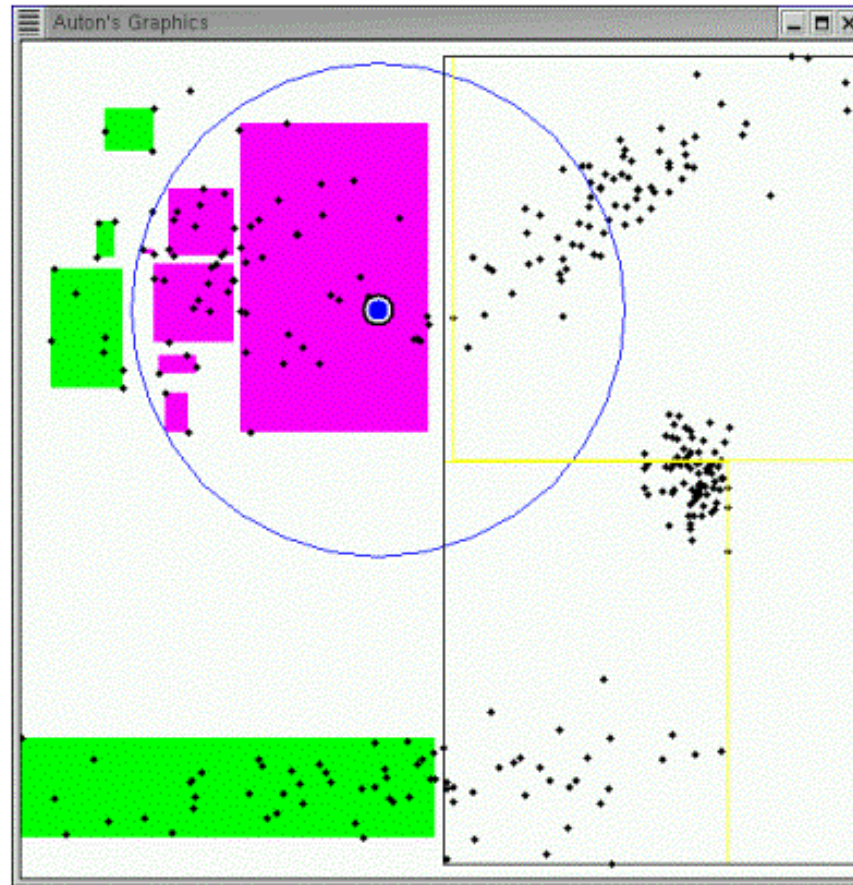
Range-count with kd -trees



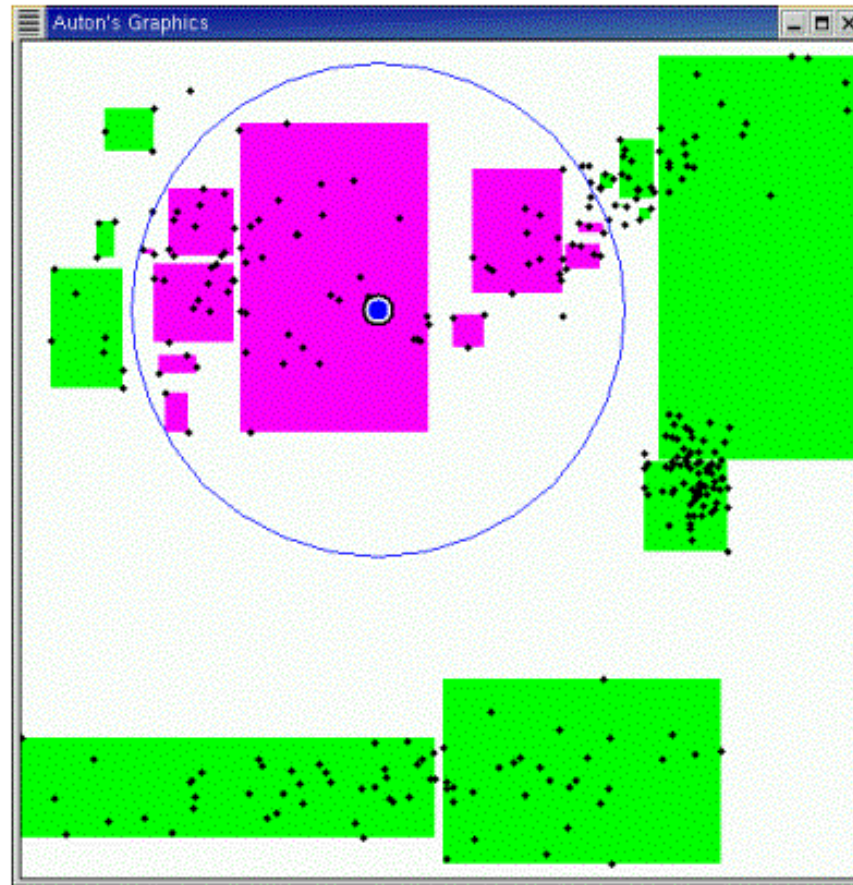
Range-count with kd -trees



Range-count with kd -trees



Range-count with kd -trees



Recursive Algorithm

```
RC( $x_q, R$ )  
  if can-prune( $x_q, R$ ), return.  
  if leaf( $R$ ), RCbase( $x_q, R$ ).  
  else,  
    RC( $x_q, R$ .left).  
    RC( $x_q, R$ .right).
```

The base case examines all points in R exhaustively.

For nearest-neighbor, we maintain the distance of the best candidate nearest-neighbor so far (starting with ∞), and prune based on it.

Runtime

To build a kd -tree: $O(DN \log N)$ time, $O(DN)$ space.

Query-time: Roughly $O(c^D \log N)$, though not rigorous with a kd -tree. Motivating recurrence:

$$T(N) = T(N/2) + O(1) \quad (3)$$

whose solution is $O(\log N)$. Note the curse of dimensionality. In reality it should be $O(c^{D'})$ where D' is some notion of intrinsic dimensionality, though this has not been shown yet.

Problem Variations

k -nearest-neighbor variations: Nearest-neighbor classification, $(1 + \epsilon)$ approximate nearest-neighbor...

Range-count variations: Range-search (return the indices of matching points), rectangular or any polytope (instead of radial region)...

All-type Problems

Given a set of query points $\mathcal{Q} = \{x_q\}$ and a set of reference points $\mathcal{R} = \{x_r\}$:

All- k -nearest-neighbor:

$$\forall x_q \quad \arg \min_r^{(k)} \|x_q - x_r\| \quad (4)$$

All-range-count:

$$\forall x_q \quad \sum_r I(\|x_q - x_r\| < h) \quad (5)$$

If $\mathcal{Q} = \mathcal{R}$ we call this a *monochromatic* problem; otherwise it is *bichromatic*. The basic k -nearest-neighbor and range-count problems are special cases of these problems.

Dual-tree Algorithms

There are now *two* sets over which to perform divide-and-conquer. We now build a tree for each set. Our algorithm will traverse both trees simultaneously (note that we can analogously compute $O(D)$ distance bounds between two nodes):

$\text{AllRC}(Q, R)$

if $\text{can-prune}(Q, R)$, return.

if $\text{leaf}(Q)$ and $\text{leaf}(R)$, $\text{AllRCbase}(Q, R)$.

else,

$\text{AllRC}(Q.\text{left}, R.\text{left})$.

$\text{AllRC}(Q.\text{left}, R.\text{right})$.

$\text{AllRC}(Q.\text{right}, R.\text{left})$.

$\text{AllRC}(Q.\text{right}, R.\text{right})$.

Runtime and Variations

Query-time: Roughly $O(c^D N)$, though not rigorous with a kd -tree. Motivating recurrence:

$$T(N) = 2T(N/2) + O(1) \quad (6)$$

whose solution is $O(N)$.

Problem variations: closest-pair, farthest-pair, Hausdorff distance, n -point correlations, Euclidean minimum spanning tree...

Kernel Summation Problems

Now throw in a continuous function.

Kernel Summation Problems

Given a *query* point x_q and a set of *reference* points $\mathcal{R} = \{x_r\}$:

$$\forall x_q \quad \Phi_q = \sum_r K(\|x_q - x_r\|) \quad (7)$$

where $K()$ is monotonically decreasing with distance and radially symmetric. Examples: Gaussian $K(u) \propto e^{-u^2/2h^2}$, Epanechnikov $K(u) \propto (1 - u^2/h^2)I(u < h)$.

Where they come up: kernel estimators, Gaussian process regression, kernelized methods...

Approximation Error

For infinite-tailed kernels it will be impossible to compute this non-exhaustively without performing some approximation. Ideally we'd like to provide a guarantee that our approximate answer $\tilde{\Phi}$ is close to the true Φ in terms of a relative error tolerance ϵ specified by the user:

$$\forall x_q \quad \frac{|\Phi_q - \tilde{\Phi}_q|}{\Phi_q} \leq \epsilon. \quad (8)$$

We'll maintain various bounds in order to achieve this.

Centroid Approximation and Bounds

A simple approximation of the contribution of a node R to the summation for x_q is

$$\forall x_q \in Q \quad \Phi_{qR} = \sum_{r=1}^{N_R} K(\|x_q - x_r\|) \approx N_R K(\|\mu_Q - \mu_R\|) \quad (9)$$

where μ_Q is the centroid of the query node and μ_R is the centroid of the reference node, and simple bounds such that $\Phi_{qR}^L \leq \Phi_{qR} \leq \Phi_{qR}^U$ are

$$\Phi_{qR}^L = N_R K(\delta_{QR}^U) \quad (10)$$

$$\Phi_{qR}^U = N_R K(\delta_{QR}^L). \quad (11)$$

Approximation Rule

Aside: It is practically equivalent to only compute the lower and upper bounds then take their midpoint at the end (“finite-difference” method).

Stop recursing and use this approximation when:

$$\frac{|\Phi_{qR}^U - \Phi_{qR}^L|}{\Phi_q^L} \leq \left(\frac{N_R}{N} \right) \epsilon. \quad (12)$$

This local rule ensures that the relative error tolerance on the global kernel sum will be met.

Multipole Expansion

How can we go beyond the centroid approximation? One way is the idea of *multipole expansions*. Technically this refers to a different kernel used in physics, $K(u) \propto 1/u$. Let's consider that example.

Multipole Expansion

The multipole expansion is simply a Taylor expansion of the potential at a point x_q due to a point x_r about a third point μ_R , where $\delta_{qr} = \|x_q - x_r\|$:

$$\frac{1}{\delta_{qr}} = \left[\frac{1}{\delta} \right]_{\delta_{qR}} + \sum_d^D \delta_{rRd} \left[\frac{\partial}{\partial x_{rd}} \frac{1}{\delta} \right]_{\delta_{qR}} \quad (13)$$

$$+ \frac{1}{2!} \sum_d^D \delta_{rRd} \sum_{d'}^D \delta_{rRd'} \left[\frac{\partial^2}{\partial x_{rd} \partial x_{rd'}} \frac{1}{\delta} \right]_{\delta_{qR}} + \dots \quad (14)$$

Multipole Expansion: Sum

The total contribution from all the points in node R , each having weight w_r , is then:

$$\begin{aligned}
 \Phi_q &= \sum_r^{N_R} \frac{w_r}{\delta_{qr}} = \sum_r^{N_R} w_r \left\{ \frac{1}{\delta_{qR}} + \sum_d^D \delta_{rRd} \left[\frac{\partial}{\partial x_{rd}} \frac{1}{\delta} \right]_{\delta_{qR}} \right. \\
 &\quad \left. + \frac{1}{2!} \sum_d^D \delta_{rRd} \sum_{d'}^D \delta_{rRd'} \left[\frac{\partial^2}{\partial x_{rd} \partial x_{rd'}} \frac{1}{\delta} \right]_{\delta_{qR}} + \dots \right\} \\
 &= \frac{1}{\delta_{qR}} \sum_r^{N_R} w_r + \sum_d^D \delta_{rRd} \left[\frac{\partial}{\partial x_{rd}} \frac{1}{\delta} \right]_{\delta_{qR}} \sum_r^{N_R} w_r \delta_{rRd} \\
 &\quad + \frac{1}{2!} \sum_d^D \sum_{d'}^D \left[\frac{\partial^2}{\partial x_{rd} \partial x_{rd'}} \frac{1}{\delta} \right]_{\delta_{qR}} \sum_r^{N_R} w_r \delta_{rRd} \delta_{rRd'} + (15)
 \end{aligned}$$

Multipole Expansion: Moments

We can rewrite the last equation by replacing the summations by constants depending only on the points in the node R , *i.e.* not on the query point:

$$\begin{aligned} \Phi_q = & \frac{1}{\delta_{qR}} \alpha_R + \sum_d^D \delta_{rRd} \left[\frac{\partial}{\partial x_{rd}} \frac{1}{\delta} \right]_{\delta_{qR}} \beta_{Rd} \\ & + \frac{1}{2!} \sum_d^D \sum_{d'}^D \left[\frac{\partial^2}{\partial x_{rd} \partial x_{rd'}} \frac{1}{\delta} \right]_{\delta_{qR}} \gamma_{Rdd'} + \dots \quad (16) \end{aligned}$$

α_R (a scalar), β_R (a vector), and γ_R (a matrix) are called the *monopole*, *dipole*, and *quadrupole* moments of the node R , respectively, or *multipole moments* in general.

Multipole Expansion: Gaussian

For the case of the Gaussian kernel, we'll use a slightly different expansion, but the basic idea is the same. We'll use the *Hermite functions* $h_n(u) = e^{-u^2} H_n(u)$, using the *Hermite polynomials* $H_n(u) = (-1)^n e^{u^2} D^n e^{-u^2}$, $u \in \mathbb{R}^1$. After scaling and shifting the argument u appropriately, then taking the product of univariate functions for each dimension, we obtain the multivariate *Hermite expansion*

$$\Phi_{qR} = \sum_{r=1}^{N_R} e^{\frac{-\|x_q - x_r\|^2}{2h^2}} = \sum_{r=1}^{N_R} \sum_{\alpha \geq 0} \frac{1}{\alpha!} \left(\frac{x_r - \mu_R}{\sqrt{2h^2}} \right)^\alpha h_\alpha \left(\frac{x_q - \mu_R}{\sqrt{2h^2}} \right) \quad (17)$$

using the usual multi-index notation for series expansions.

Multipole Expansion: Query-side

That was expanding about μ_R (“reference-side expansion”). We could also perform an expansion (Taylor) about μ_Q (“query-side expansion”):

$$\Phi_{qR} = \sum_{r=1}^{N_R} e^{-\frac{\|x_q - x_r\|^2}{2h^2}} = \sum_{r=1}^{N_R} \sum_{\alpha \geq 0} \frac{1}{\alpha!} h^\alpha \left(\frac{x_r - \mu_Q}{\sqrt{2h^2}} \right) \left(\frac{x_q - \mu_Q}{\sqrt{2h^2}} \right)^\alpha \quad (18)$$

We will actually do both types of expansions. We will:

1. Truncate the reference-side expansion to some number of terms p . We can bound the error of this approximation.
2. Truncate the query-side expansion of this truncated expression to some number of terms p' . We can bound the error of this second approximation.

Multipole Expansion: Algorithm

The overall algorithm requires three kinds of operations, which are described in the original physics literature as *translation operators* on various coefficients, where “far-field” effectively refers to the reference points and “local” effectively refers to the query points:

Far-field to far-field: A bottom-up precomputation of the coefficients of the reference-side expansion of parents from children.

Far-field to local: During the run, the query-side and reference-side approximation of the contribution of a reference node to each point in a query node.

Local to local: A top-down postcomputation which propagates the approximations made at internal nodes down to combine them all at the leaf nodes.

Other N -body Problems

Now throw in a decision, etc.

N -body Decision Problems

Given a *query* point x_q and a set of *reference* points $\mathcal{R} = \{x_r\}$, find the class label(x_q):

k -nearest-neighbor decision:

$$\text{label} \left(\min_r^{(k)} \|x_q - x_r\| \right) \quad (19)$$

Kernel summation decision:

$$\max_c \sum_r K(\|x_q - x_r\|) I(\text{label}(x_r) = c) \quad (20)$$

N -body Decision Algorithms

Where they come up: k -nearest-neighbor classification, kernel discriminant (aka nonparametric Bayes classification), support vector machine...

Basic idea for kernel decision algorithm: Maintain bounds on each sum. Traverse the query and reference tree(s) until we can show that the lower bound for one is definitely larger than the upper bound for the other.

Other N -body Problems

- $(1 + \epsilon)$ approximate nearest-neighbor. We can do this fast, but unfortunately this problem is flawed: it approximates the wrong thing.
- n -point correlation functions. Use n trees.
- Approximate n -point correlation functions. Use Monte Carlo within a tree-based algorithm.
- Mixture of Gaussians. Use a special case of quadratic programming to get bounds for arbitrary-covariance Gaussians.