Alexander G. Gray

BRINGING TRACTABILITY TO GENERALIZED N-BODY PROBLEMS IN STATISTICAL AND SCIENTIFIC COMPUTATION

A dissertation presented to the faculty
of the School of Computer Science of

Carnegie Mellon University

in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

CMU-CS-04-189
April 2003

Committee:
Andrew W. Moore, CS Dept. and Robotics Institute (chair)
Sebastian Thrun, CS Dept. and Robotics Institute
Larry Wasserman, Statistics Dept. and CALD
Robert Nichol, Physics Dept.
Dennis DeCoste, NASA Jet Propulsion Lab

Abstract

Generalized N-body problems.

Taking a bird's-eye view across the fields of computational geometry, statistics and machine learning, computational physics, and a collection of areas which we summarize as computational morphology, we define a class of important problems, **generalized N-body problems**, which share a common structure. Informally these are problems which can be solved by considering in turn each pair (or n-tuple) of points in a metric space. We survey the literature across all of these fields, pointing out the recurring classes of solution attempts and their well-known limitations. We then show that the entire class of problems can be efficiently solved by a common design principle which we propose, which we refer to as *higher-order divide-and-conquer* (HODC) – the extension of the powerful divide-and-conquer principle of algorithm design from the division of single set to the division of multiple sets. This simple idea, stated for the first time here to our knowledge, while quite general, is considered here only in its form for geometric problems, which we call *geometric shattering*.

N-body problems in computational geometry.

To instantiate and demonstrate the most basic geometric shattering algorithm design, we consider the **all-nearest-neighbors** problem. We first introduce adaptive space-partitioning trees (ASPT), of which *kd*-trees and metric- or ball-trees are examples, and review previous observations that they are sensitive to the intrinsic dimension of the data rather than the explicit dimension. We prove that, given the prior construction of an ASPT, the 'dual-tree' algorithm resulting from the shattering principle improves upon the complexity of a standard single-tree approach from $O(N \log N)$ to $O(N)$. This extends related but more limited observations by Vaidya (1989) and Callahan and Kosaraju (1993) in computational geometry and Hjaltason and Samet (1998) in the database literature. This demonstrates in a canonical and well-studied context that the higher-order divide-and-conquer principle is both theoretically insightful and useful (leading to the optimal complexity for this problem).

N-body problems in computational statistics.

To extend shattering to the case of a continuous potential or kernel function between points, we consider the practical problem of **kernel density estimation** (KDE), the principal method of nonparametric estimation in statistics, for which satisfactory computational approaches have been unavailable since formulation of the foundational idea in 1956. Grid-based methods, including FFT elaborations, exhibit poor accuracy and are exponential in the explicit dimension. We develop a function approximation scheme based on a simple recursive *finite-difference* interpolation idea, which is further accelerated by additional techniques involving asynchronous dynamic programming and an extra order of divide-and-conquer. This results in the first practical algorithm for kernel density estimation which is capable of computing densities in seconds for datasets ranging up to millions of points and ranging up to several hundreds of dimensions, with three-significant-digit accuracy or bet-

ter (typically six-digit accuracy), for both kernels with compact support and those with infinite tails. A number of other important problems in statistics and machine learning can be similarly treated.

N-body problems in computational physics.

A central technique of computational fluid dynamics, **smoothed particle hydrodynamics** (SPH), first proposed in 1977 as the Lagrangian (more accurate particle-based) alternative to Eulerian (simple grid-based) approaches to fluid dynamics, represents one of many fundamental N-body problems in physics which are not treatable by the existing well-known linear-time method of Greengard and Rokhlin (1987) based on analytic multipole expansions. We show a number of extensions that can be made to our algorithm for KDE to obtain the first linear-time method for SPH, improving upon the best previous complexity $O(N \log N)$ using the Barnes-Hut method (1986). Our method is importantly derivative-free, eliminating the high barrier to implementation and understanding surrounding the complex multipole methods, and making them practical alternatives even for the original problems where the multipole methods are applicable.

N-body problems in computational morphology.

Finally we treat generalized N-body problems which are defined by n-tuples for $n > 2$ rather than merely pairs. The **n-point correlation functions** form the theoretical foundation for spatial point processes, which are used heavily in statistical physics and particle physics, and are today on the critical path to answering the central questions of cosmology. Despite pursuits for computational tractability in astrophysics beginning in the 1960's, only the simplistic grid- and FFT-based methods had previously existed for this problem in the approximate case, and for the exact case only the naive $O(N^n)$ approach existed; thus even for $n = 3$, the largest true scientific application which had been attempted was for $N \simeq 20{,}000$ points. We demonstrate the natural extension of shattering to obtain an algorithm for arbitrary n, yielding exact solutions in minutes or hours, depending on the ranges chosen, for 75 million points with $n = 3$. For the large ranges which are most difficult for our exact algorithm, we also develop an approximate method which uses shattering to create strata for a stratified Monte Carlo integration method. We develop a new method for adaptive Neyman allocation, which approaches optimal sample allocation and is applicable in any Monte Carlo integration problem. Using the resulting algorithm we demonstrate the computation of the 3-point correlation with 1% maximum error at the 99% confidence level in seconds, regardless of N. Our algorithms are currently in use by astrophysicists in the largest experimental calculations to date, toward answering fundamental questions concerning dark matter and validation of the standard model of cosmology.

Summary.

We have presented substantial advances for two major open computational problems (KDE and n-point), proposed an algorithm for SPH which is superior in complexity though to be tested in future work, and given theoretical perspective on the basic algorithmic insight using the context of the well-studied all-nearest-neighbors problem. Through the examination of these four canonical problems, we effectively introduce a set of simple but powerful techniques for realizing the higher-order divide-and-conquer strategy, comprising a practical toolbox of algorithm designs for the different situations arising in a larger list of thirty or so important problems. We anticipate that the general algorithmic machinery underlying these results will also be useful in statistical and scientific problems beyond these examples.

To my father, Gordon T. Gray (1939-2000).

He showed me, by his brilliant example, how to perceive and think beyond the bounds of others, to pursue value in clear terms, and to yield to no obstacle. He lives in every step of my work.

Acknowledgments

Warm thanks beyond what I can truly express here are due to the following people: Andrew Moore, my advisor, who has been a true inspiration both in his peerless ingenuity and in his uncompromising dedication to value in his work. Nicoleta Serban, my sweetheart, whose belief in me during my darkest hours was stronger than my own, and who has become part of my purpose itself. Rune Jensen, my friend and officemate, without whom the PhD experience would not have been half as enjoyable. All my old friends, who forgave me for disappearing into a cave during all those years of grad school. And Kimberly Gray, my mother, for the lifelong love and support that makes every success possible.

Thanks are also due to all my old colleagues at NASA – my 6 years at JPL gave me the experience and confidence to attack such a wide range of scientific problems simultaneously, which has been a joy, as well as the excellent preparation which allowed this work to be done in 3.6 years.

My graduate study and research was supported in part by the NASA GSRP Fellowship.

Contents

1 Generalized N-Body Problems

Isolating a Ubiquitous Obstacle Across Fields.

Each problem that I solved became a rule which served afterwards to solve other problems. — Rene Descartes (1596 - 1650).

HOW CAN COMPUTER SCIENCE, or more specifically the study of algorithms, change the world? Quite simply, a great number of things people would like to accomplish are limited by the computation required to achieve them. It is these obstacles — *pressing computational problems which really exist in the world*, as opposed to problems which are chosen according to some other measure of interestingness — which drive the viewpoints and approaches taken in this thesis.

What is this thesis about?

This thesis is about three things:

1. First, it identifies a large and real *class of problems*, called **generalized N-body problems** which appear prominently across a number of important disciplines, including statistics, physics, and computer science.

2. The main reason to link these various problems is that they can be efficiently solved by the same kinds of solutions. The second thing this thesis is about is a new *class of solutions* for this new class of problems, called **geometric shattering**.

3. The third concern of this thesis is the design of better *practical* algorithms for specific tasks, as opposed to theoretical algorithm sketches which serve as mathematical existence proofs rather than tools which change the world. Thus, we will be absolutely and happily subject to all of the specific unique constraints for each problem area that its practitioners face, rather than view these constraints as unfortunate limitations which would be removed or relaxed in a perfect paradise. We will choose four specific tasks to build algorithms for, one from each of four different fields, which will represent different forms of our solution strategy corresponding to key aspects which can appear in different problems. These are the **all-nearest-neighbors**, **kernel density estimation**, **smoothed particle hydrodynamics**, and **n-point correlation** problems.

Agenda of this chapter.

In this chapter I'll point out several important problems in various fields whose similarity to the well-known *physical* N-body problem will be apparent. It should also become apparent that these kinds of problems are prevalent and naturally-occurring across several diverse fields, and quite often at the core of those fields in terms of fundamental importance. Along the way I'll sketch the backbone of the organizational structure of this thesis, in previews of each chapter.

§1.1 First questions.

First let's briefly answer some high-level questions about the class of problems we're considering.

So what are generalized N-body problems, roughly?

Very informally, a *generalized N-body problem* is one in which n sets of points in some multi-dimensional space must be compared with each other to obtain the solution. These problems arise when it is generally not possible or clear how to determine the relationships between individual pairs (when $n = 2$ sets, as is usually the case) of points analytically or otherwise *a priori*, without considering each pair individually.

If there are N points in each set, the straightforward solution of explicitly considering each possible pair using a simple double loop costs $O(N^2)$, or $O(N^n)$ in general - and this is in fact most often the best available practical solution for these kinds of problems.

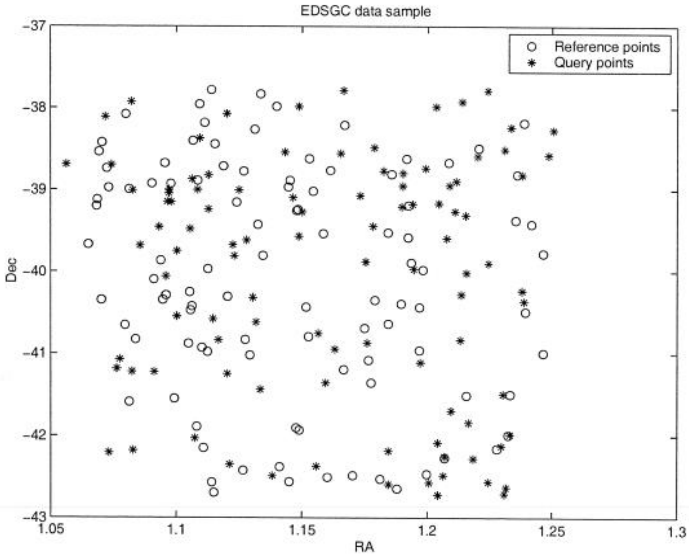

Figure 1.1: POINTS IN SPACE. Shown is a tiny subsample from the EDSGC sky survey. The dimensions RA and Dec represent positional coordinates of sky objects. Shown are two sets of points, in this example, a 'query set' and a 'reference set'.

Typically, we'll have a set of data points, which we'll call the *reference* dataset, $\underline{X}_{\mathcal{R}}$ having size $N_{\mathcal{R}}$. We'll also have a *query* dataset $\underline{X}_{\mathcal{Q}}$ containing $N_{\mathcal{Q}}$ points. Assume all of our points live in some metric space of dimension D, say the Euclidean space $\mathbb{R}^D$ for now. We can think of most of our problems as asking questions about the reference points with respect to one or more of the query points: "For each query point, somehow consider each reference point". A common form of problem, for example, is to ask, given some *kernel function* $\phi()$ defined on the Euclidean distance $\delta_{qr} = \|\underline{x}_q - \underline{x}_r\|$ between a pair of points $\underline{x}_q$ and $\underline{x}_r$, for the sum of the kernel evaluations at each query point:

$$\forall q, \text{Compute } \textstyle\sum_r \phi(\|\underline{x}_q - \underline{x}_r\|) \tag{1.1}$$

But the best way to understand the class is first by the canonical examples we actually have in mind, which we will cover in each of four major fields (next four

Sections). After that, a formal definition can be made (Section 1.6) without being unnecessarily confusing. Note that from now on we'll sometimes just say 'N-body problems' in place of 'generalized N-body problems', and when doing so we'll be sure to say 'the physical N-body problem' when that's what we mean.

Why the name?

I coined the term *generalized N-body problems* in allusion to the physical N-body problem, partly because it is a useful mental prototype for this class due to its concrete physical nature, but mainly because it is the most famous of the problems. Is it more famous than the other problems we'll consider in this chapter because it is the most pressing of the problems in some sense, or because the number of people concerned with it is greatest? Not necessarily. Keep in mind that it was made famous by the fact that a set of beautiful and powerful solutions popped up for it in the mid-80's [BH86, GR87]. Had this not happened, and if physicists still had no idea that there was something better than writing code implementing the simplistic and sluggish $O(N^2)$ solution to this computational problem, it would just be another ugly day-to-day necessity - probably too banal to be given any name at all.

Why wasn't this class pointed out before?

The point here: the existence of non-trivial or interesting *solutions*, or the realization of the possibility of such solutions, is often what makes a problem, or problem class, worth naming and formalizing. Though certain analogies between pairs of the problems I'll mention have been pointed out by a few authors in the past, either no one has observed the larger set of relationships, or no one has decided to delineate the class containing them all. The reason is the fact that no unified *solution* applying to the entire class had previously existed. This thesis demonstrates that a single unified solution methodology for generalized N-body problems indeed exists.

What's the importance of this class?

As it turns out, most of the main problems we'll consider in this thesis were in the same state as the physical N-body problem before the aforementioned solutions came along - namely, though a handful of fairly primitive solutions existed, the best all-around solution was generally the straightforward $O(N^2)$ one. Thus the situation was dramatically improved when the solutions in this thesis were introduced, rather like it was in the mid-80's for the physical N-body problem. As will become apparent, these problems are arguably no less fundamental to their respective fields than the original N-body problem is to physics.

Can't we just apply the previous physics solutions?

As will also be apparent, the answer is unfortunately no. Though our other problems have important similarities to the physics problems, there are also important differences which will require a completely different approach. The less 'physical' our problems become, the less the solutions for the physics problems are relevant. Though we will also do our best to transfer as much as possible from the computational physics techniques (analyzed in Chapter 4) the requirements of many of our problems of interest will lead us down a different path which will culminate with a parallel branch of algorithmic solutions. This branch will yield an alternative solution to the original physical N-body problem itself, with the same complexity but different in several other respects. The next natural question, of how the two approaches might be profitably combined, is also treated in Chapter 4.

So who studies 'points in space'? Let's now follow this common interest, on a tour through various different fields which might seem otherwise unrelated, and in fact by-and-large have little communication with one another, to all of their misfortunes.

§1.2 Basic geometric queries.

Computational geometry is the branch of algorithms (*i.e.* both complexity theory and algorithm design) which treats problems involving the relationships between geometric objects, *e.g.* points, lines, planes, polygons, and onward to include more complex constructions such as Voronoi diagrams [PS85, dBvKOS99]

Among the first and most canonical problems studied in computational geometry are the *proximity problems* - those involving the relative positions of *points* in space. This class is the home for our most elementary N-body problems, which occur in many guises, and is thus our natural starting point.

1.2.1 Proximity problems.

As we'll later see, one of the most ubiquitous N-body problems is as follows: for each of our query points $\underline{x}_q$ we want to know its *nearest neighbor*, *i.e.* the reference point $\underline{x}_r$ whose distance to $\underline{x}_q$ is minimal:

All-nearest-neighbors:

$$\forall q, \text{Compute } NN(\underline{x}_q) = \arg\min_r \|\underline{x}_q - \underline{x}_r\| \tag{1.2}$$

Technically this is the *bichromatic* version of the problem, meaning that $\underline{X}_{\mathcal{Q}} \neq \underline{X}_{\mathcal{R}}$ — usually what is meant is the case where the query and reference datasets are the same. Comparing this to our abstract model problem (Equation 1.1), we see that here the kernel function is the identity function, and that summation has been replaced by minimization.

A common variant is to ask for the κ nearest neighbors, called *all-κ-nearest-neighbors*. Another variant is to ask for the single pair having the smallest distance:

Closest-pair:

$$\text{Compute } \arg\min_q \arg\min_{q \neq q'} \|\underline{x}_q - \underline{x}_{q'}\| \tag{1.3}$$

The related *diameter* of a dataset is the distance of the *farthest pair* of points. The distance between two sets is often defined as:

Set distance:

$$\text{Compute } \min_q \min_r \|\underline{x}_q - \underline{x}_r\| \tag{1.4}$$

which is just a bichromatic version of the closest-pair problem.

A different kind of problem involves *range* queries, in which the kernel function often becomes an indicator or delta function. For example, for each query point, we may wish to find all the reference points within a fixed radius h of it:

All-range-search:

$$\forall q, \text{Compute } h\text{-}set(\underline{x}_q) = \{\underline{x}_r \mid \|\underline{x}_q - \underline{x}_r\| \leq h\} = \bigcup_r \arg_r I(\|\underline{x}_q - \underline{x}_r\| \leq h) \tag{1.5}$$

where $\arg_r I(\delta_{qr}, h)$ means 'return r if the indicator function is satisfied, otherwise $\{\}$'. We sometimes simply want to count them:

All-range-count:

$$\forall q, \text{Compute } h\text{-}count(\underline{x}_q) = \left|\{\underline{x}_r \mid \|\underline{x}_q - \underline{x}_r\| \leq h\}\right| = \sum_r I(\|\underline{x}_q - \underline{x}_r\| \leq h) \tag{1.6}$$

The *total-range-count* problem finds the sum of the counts over the queries.

Database queries.

These operations are in fact heavily studied in another large area of computer science: *database systems*, sometimes under the heading of *spatial indexing* or *information retrieval* [Sam90, SC03]. Even this seemingly most abstract set of N-body problems, then, is of direct real-world interest, in fact spawning mountains of papers on practical algorithmic approaches to these problems, probably more than in computational geometry itself. Note that the same problems are often studied under different names. For example, in the database community, the all-range-search/count problems come under the heading of the *spatial join*. It is important to note, however, that the real-world versions of these computational-geometric problems occurring in databases have an additional twist - the solutions considered useful must also account for other computational issues such as the latencies in the memory/storage hierarchy of modern computers and the impact of the physical layout of data on storage media.

Despite the obvious commonality in problems, these two fields are markedly isolated from one another, the fact that they are both sub-areas of computer science notwithstanding.

1.2.2 Related problems.

At this might we should mention another kind of proximity problem, *Euclidean minimum spanning tree*, or MST construction in a Euclidean space. While having a similar sort of 'closest-point' nature, it cannot be simply characterized as some kind of double-looping procedure. It is slightly more complicated, and falls 'just out' of the class of N-body problems we are considering.

Another whole class of proximity problems are called *dynamic* problems. In these problems the set of points is not static, but changes, and the goal is to compute the solution to the changed problem without redoing unnecessary work. These kinds of problems are also not our concern in this work.

It is worth reemphasizing: not all proximity problems are N-body problems, though many of the core problems are.

1.2.3 Current state-of-the-art and our focus.

Clearly the most straightforward way to solve these problems is to loop over each reference point, for each query point, updating the index, set, or count of interest for each pair. This is what we mean by the naive $O(N^2)$ algorithm (or $O(N_{\mathcal{Q}} N_{\mathcal{R}})$ when $N_{\mathcal{Q}} = N_{\mathcal{R}} = N$). In computational geometry and database systems, however, $O(N \log N)$ approaches have been developed for these problems, and these will in fact be the conceptual starting point for our own algorithmic development. We'll look broadly across both of these fields at the best available solutions.

Preview of Chapter 2.

In Chapter 2 we will take an in-depth look at the central **all-nearest-neighbors** problem in particular. We will show how to extend the standard solutions developed in computational geometry to arrive, perhaps somewhat surprisingly, at an $O(N)$ expected-time algorithm. In doing so we will introduce the main concepts upon which all the algorithms of this thesis rest: certain hierarchical geometric data structures and divide-and-conquer algorithms using them. Chapter 2 will thus be atypical of the other chapters in that we won't be directly concerned with practice (*i.e.* database issues in this case), but rather more abstract notions which will transfer to all our N-body problems.

§1.3 Basic statistical inferences.

In the general endeavor of *multivariate statistics* [Joh98], when all of the measurements in our data are continuous numbers, we can view the situation in at least two different ways. From a data analysis viewpoint, it is useful to think of our data as

consisting of N objects of interest, each having D continuous measurements, also called *features*, or *attributes* - this can be viewed as a table with N rows and D columns. However, it is equivalent to think of these as N points in a D-dimensional space. So once again we are reasoning about points in space.

This 'points-in-space' view is deep in the core of multivariate statistics, particularly in its historical sibling, *pattern recognition* [DH73], the predecessor (as well as a separate contemporary to some degree) of the modern field of *machine learning* [Bis95]. Though in a conceptual sense these latter research areas effectively also study problems of statistical inference, they differ from the older field of *statistics* [CB90] (proper) in focus and approach. One major cultural departure, for example, is their tendency to focus on high-dimensional datasets. Another is almost part of the definition of an offshoot of machine learning, *data mining* [HMS01], which is concerned with the exploratory data analysis of datasets which are typically large, *i.e.* having large N. While sociologically distinct, these various fields form a bit of a tangle, with a certain level of mutual awareness but much much less than might be expected. To keep the technical issues clear I prefer to refer to the *logical* topics in question rather than focus on historical or cultural categories.

In a very brief tour of multivariate statistics, or perhaps more suitably, 'statistics/pattern recognition/machine learning', we'll see that many of its most basic methods are in fact generalized N-body problems.

1.3.1 Classification.

Classification, the prediction of a discrete variable, is one of the basic tasks of multivariate statistics. The discrete variable is usually thought of as an index over K *classes*. Perhaps the most basic (yet surprisingly effective) method is an example we've already seen: the *nearest-neighbor* (or κ-nearest-neighbor) *classifier* is a core tool of statistics and learning. Here the query dataset corresponds to the *test set*, and the reference dataset is the *training set*. The nearest-neighbor rule assigns the class of a test point to be the class of the nearest reference point to it. The κ-nearest-neighbor rule is similarly

κ-nearest-neighbor classifier:

$$\forall q, \text{Compute } Class(\underline{x}_q) = Class(\arg\min_r^{(\kappa)} ||\underline{x}_q - \underline{x}_r||) \tag{1.7}$$

where $\min^{(\kappa)}$ denotes the set of the κ smallest, and the $Class()$ of a set is the majority class within it.

Bayes classifier variants.

This can be seen as a special case of a fundamental tool of statistics and learning, the *Bayes classifier*. For class C_k, $\widehat{P}(C_k|\underline{x}_q) = \frac{\hat{p}(\underline{x}_q|C_k)P(C_k)}{\hat{p}(\underline{x}_q)}$ by Bayes' rule, yielding:

Bayes classifier:

$$\forall q, \forall k, \text{Compute } Class(\underline{x}_q) = \arg\max_k \hat{p}(\underline{x}_q|C_k)P(C_k) \tag{1.8}$$

In the canonical *naive-Bayes* classifier for real-valued inputs, each class is modeled by a single multivariate Gaussian $\mathcal{N}(\hat{\underline{\mu}}_k, \widehat{\underline{\Sigma}}_k)$ where the covariances have the spherical form $\widehat{\underline{\Sigma}}_k = \hat{\sigma}_k^2 I$, in which case

$$\hat{p}(\underline{x}_q|C_k) = \frac{1}{(2\pi\hat{\sigma}_k^2)^{D/2}} e^{-\frac{1}{2}\frac{\|\underline{x}_q - \underline{\mu}_k\|^2}{\hat{\sigma}_k^2}}. \tag{1.9}$$

Classification amounts to selecting the highest-density Gaussian, weighted by the prior. This has the straightforward cost $O(N_Q K)$. The appearance of a weighting factor is a new element which will also appear in many other of our N-body problems.

The kernel function in this problem is the Gaussian, which monotonically decreases with distance. For the case of general covariance matrices $\hat{p}(\underline{x}_q|C_k) = \frac{1}{(2\pi)^{D/2}|\widehat{\underline{\Sigma}}_k|^{1/2}} e^{-\frac{1}{2}(\underline{x}_q - \hat{\underline{\mu}}_k)^T \widehat{\underline{\Sigma}}_k^{-1}(\underline{x}_q - \hat{\underline{\mu}}_k)}$, which slightly obscures the relationship to the general problem form in Equation 1.1, though inconsequentially.

Now in the special case when the covariance matrices are spherical and all $\hat{\sigma}_k$ and priors are equal, the classification rule becomes $Class(\underline{x}_q) = \arg\max_k \hat{p}(\underline{x}_q|C_k)P(C_k) = \arg\max_k \{-\frac{1}{2\hat{\sigma}_k^2}\|\underline{x}_q - \underline{\mu}_k\|^2\} = \arg\min_k \|\underline{x}_q - \underline{\mu}_k\|$, reducing exactly to the nearest-neighbor rule where the reference points are now the class centers:

Equi-spherical naive-Bayes classifier:

$$\forall q, \forall k, \text{Compute } Class(\underline{x}_q) = \arg\min_k \|\underline{x}_q - \underline{\mu}_k\| \tag{1.10}$$

It should also be obvious that replacing the Gaussian with a different probability density function still leaves us with an N-body problem.

1.3.2 Latent-variable models.

A *latent-variable* (or 'hidden-variable') model contains variables which are not directly observable in the data but are responsible for important underlying structure such as multiple modes in the data.

EM for mixtures.

One of the most widely-used and studied methods in statistics and learning, particularly in the last half-decade, is the the canonical example of a latent-variable model, the *mixture model* (more specifically a mixture-of-Gaussians model, typically) for clustering, whose parameters are popularly fitted using the EM algorithm.

In fact, the Bayes classifier model we just saw already implied the mixture-of-Gaussians model, though we didn't need to write out its likelihood. Assignment of test points to clusters is exactly the Bayes classification rule already shown. Furthermore the E-step of EM is exactly inference (in this case, soft assignment of training points to clusters), and this must be performed multiple times with changing parameters (for C_k^t, or class C_k at iteration t, $\underline{\Theta}^t = \{P(C_k^t), \hat{\underline{\mu}}_k^t, \widehat{\underline{\Sigma}}_k^t\}$) until convergence is reached:

Mixture model E-step:

$$\forall q, \forall k, \text{Compute } \frac{\hat{p}(\underline{x}_q|C_k^t)P(C_k^t)}{\hat{p}(\underline{x}_q)} \tag{1.11}$$

Note that though we often show them for clarity, computationally the constant factors are unimportant, in the sense that they can be accounted for before or after the central N-body computation.

The widespread *k-means* algorithm happens to correspond exactly to the special case of this method with identical priors and identical spherical covariances as $\widehat{\underline{\Sigma}}_k \to 0$, making its E-step exactly the now-familiar nearest-neighbor rule:

k-means E-step:

$$\forall q, \forall k, \text{Compute } Class(\underline{x}_q) = \arg\min_k \|\underline{x}_q - \underline{\mu}_k^t\| \tag{1.12}$$

1.3.3 Density estimation.

The mixture-of-Gaussians model can be seen as a method for *density estimation*, or modeling the probability density of the data. Computation of the likelihood of a data point $\hat{p}(\underline{x}_q)$, or $\hat{p}(\underline{x}_q|\underline{\Theta})$, was needed in the E-step for mixtures. The likelihood of the entire dataset $\underline{X}_{\mathcal{Q}}$, normally computed in log form, is one common way of scoring the fit of the model as a density: $\log \widehat{L}(\underline{X}_{\mathcal{Q}}) = \sum_q \log \hat{p}(\underline{x}_q)$:

Mixture model log-likelihood:

$$\forall k, \text{Compute } \log \widehat{L}(\underline{X}_{\mathcal{Q}}) = \sum_q \log \sum_k \hat{p}(\underline{x}_q|C_k)P(C_k) \tag{1.13}$$

For many machine learning practitioners, latent-variable models are probably their main encounter with density estimation.

In fact, all the basic operations of multivariate statistics, including classification and regression (next Section) can be represented in terms of the density of the data. Furthermore, an explicit representation of the density is clearly useful in its own right for visualization, outlier detection, and so on. Though recognized as an abstract unifier of inference tasks in machine learning, the fundamental importance of density estimation is given much more explicit attention in the older (and perhaps wiser) field of statistics as both a theoretical and experimental tool.

1.3.4 Regression.

Regression, the prediction of a continuous variable, is a close sibling of density estimation, because both have the task of estimating a continuous *curve* of some sort.

The similarity is easily seen. Consider a basic (though highly successful empirically) regression model analogous to the Bayes classifier, which might have been more clearly called the 'Bayes regressor', but which has instead historically been called the *radial basis function network*, which weights the target values each of K interpolation points according to the closeness of the query to them under Gaussian densities. Once these values $y(\underline{\mu}_k)$ have been found in the training or 'design' phase, at prediction time we have

Radial basis function network regression:

$$\forall q, \text{Compute } y(\underline{x}_q) = \frac{\sum_k^K y(\underline{\mu}_k)\hat{p}(\underline{x}_q|C_k)P(C_k)}{\sum_k \hat{p}(\underline{x}_q|C_{k'})P(C_{k'})}. \tag{1.14}$$

Locally-weighted regression is a related technique using only the κ-nearest-neighbors for each query point, this requiring an additional computational step which we've already seen.

1.3.5 Flexible models.

Notice that in going from the Bayes classifier to the Bayes regressor, the number of 'classes' K is no longer simply specified by the data - in the regression case it becomes a parameter which can be varied. The more complex or nonlinear the target function is, the more basis functions (radial or otherwise - a great host of different types of basis functions have been explored) are required to adequately model the function, the general philosophy being that the basis functions are themselves simple and are composed to obtain more complex functions. In classification, the natural extension of the Bayes classifier along these lines is to replace the single-Gaussian model of each class with a mixture-of-Gaussians, thereby obtaining the ability to model more complex decision boundaries. Supposing for simplicity that the number of mixture components in each class $L_k = L$, the straightforward computational cost rises to $O(N_{\mathcal{Q}}LK)$:

Mixture Bayes classifer: (1.15)

$$\forall q, \text{Compute } Class(\underline{x}_q) = \arg\max_k \{\sum_{l_k}^{L} \hat{p}(\underline{x}_q|C_{l_k})P(C_{l_k})\}P(C_k)$$

As classification and regression in arbitrarily complex and nonlinear data have been and inevitably continue to be attempted, such flexible methods have become increasingly sophisticated and ever-larger representations have appeared in difficult applications.

1.3.6 Nonparametric methods.

Now let's follow this ascent in representational power to its logical conclusion. Suppose that we use, in some sense, the *maximum possible* number of basis functions by deciding to place a Gaussian on each of the training points $\underline{x}_r$. In this way we might expect that no part of the observed data space is overlooked or under- or over-weighted due to poor or inequitable placement of basis functions. That replaces, for example, the complex nonlinear optimization problem that EM attempts to solve. Perhaps the most difficult and central issue surrounding the function approximation problems we have considered so far is that of *model selection* - or the problem of how to select the right number of basis functions to use for a given dataset. Again, our new strategy replaces whatever complex procedures we might have used for this problem with a simple choice. We are now left free to focus all of our energies on the remaining problem - the scale parameter $\hat{\sigma}_k$ of the basis functions. In fact, making this choice properly becomes even more critical, as it 'contains' the other problems in the sense that all of their consequences have just been tranferred to this problem.

The perspective we have arrived at is that of *nonparametric* estimation, in which the model size (number of parameters) is allowed to be infinite - in this case the number of data, which has no finite upper bound.

Kernel estimators.

The method that we have arrived at is known as *kernel density estimation*, and it is simply a particular mixture model with all class priors (or mixture weights) equal:

Kernel density estimation:

$$\forall q, \text{Compute } \hat{p}(\underline{x}_q) = \sum_r \frac{1}{N_{\mathcal{R}}} \hat{p}\left(\underline{x}_q|\underline{x}_r, \hat{\sigma}_r\right) \quad (1.16)$$

Usually the probability density function $\hat{p}\left(\underline{x}_q|\underline{x}_r, \hat{\sigma}_r\right)$ is written more generally as a kernel function $K\left(\frac{\|\underline{x}_q - \underline{x}_r\|}{\hat{\sigma}_r}\right)$, though in fact it must be a pdf, *i.e.* $\int_{-\infty}^{\infty} K(z)dz = 1$. Also $\hat{\sigma}$, called the *bandwidth* h is standardly taken to be the same for all reference points, though *variable-kernel* density estimation allows them to be different.

In the special case of the basic spherical kernel $K\left(\frac{\|\underline{x}_q - \underline{x}_r\|}{h}\right) = I\left(\frac{\|\underline{x}_q - \underline{x}_r\|}{h} < 1\right)$, or $I\left(\|\underline{x}_q - \underline{x}_r\| < h\right)$, we have exactly the all-range-count problem.

The importance of this nonparametric end of the spectrum is that the kernel density estimator now has the property of *consistency* as an estimator of the true underlying density $p()$ of any arbitrary dataset - a property which no particular parametric choice of model can have. Its fatal drawback is that its straightforward cost is $O(N_{\mathcal{Q}} N_{\mathcal{R}})$.

The theory of nonparametric estimation is filled with a class of quantities called *U-statistics*, which are based on pairwise interactions, and thus exactly N-body problems.

The classification version of this method is immediate - simply model each class in the Bayes classifier with the kernel density estimator for the data having that class:

Kernel density Bayes classifier:

$$\forall q, \text{Compute } Class(\underline{x}_q) = \arg\max_k \{\sum_{r_k} \frac{1}{R_k} \hat{p}(\underline{x}_q | \underline{x}_{r_k}, \hat{\sigma}_{r_k})\} P(C_k) \tag{1.17}$$

The regression version, called the *Nadaraya-Watson estimator*, or simply 'kernel regression' is also readily apparent by extension of what we've already seen:

Nadaraya-Watson regression:

$$\forall q, \text{Compute } y(\underline{x}_q) = \frac{\sum_r y(\underline{x}_r) \hat{p}(\underline{x}_q | \underline{x}_r, \hat{\sigma}_r)}{\sum_r \hat{p}(\underline{x}_q | \underline{x}_r, \hat{\sigma}_r)} \tag{1.18}$$

1.3.7 Related problems.

The Euclidean minimum spanning tree appears in statistics in the form of *single-linkage clustering*.

Dynamic problems appear widely in the statistical setting in the form of *incremental* or 'online' learning. Another example of a dynamic proximity problem occurs in the construction of *dendrograms*, or 'hierarchical clustering' methods [Epp98].

1.3.8 Current state-of-the-art and our focus.

While we obviously haven't covered the entire space of methods of multivariate statistics, it is also clear that a surprisingly large fraction of them are computationally characterizable in a similar way. In fact, arguably the methods forming the primary backbone of the statistical pattern recognition style of approaches all fall into category of N-body problems, once we expand our notion of N-body problems to include those requiring evaluations of continuous functions of distance, such as radial basis functions and other probability density functions.

A look at the main prior work on these computational problems is instructive. Having entered the world of continuous functions, we will see the beginning of a pattern - attempts to use explicit grid-based methods, the most sophisticated version of which uses the Fast Fourier Transform (though in a rather awkward way).

Only a very few researchers seem to have noticed the geometric structure of such statistical problems and sought to exploit it, with the notable exception of a few who discussed the connection in a very broad sense [Sha75, Omo87, ML98] as well as some notable uses of computational geometry for specific problems [Pri94, ZRL96]. Straightforward instances of the nearest-neighbor problem are the ones to have been treated geometrically, and in those cases the approach has followed the basic known approach described in Chapter 2. The less obviously geometric problems involving continuous kernel functions have been largely untouched except for a few problem instances, in which powerful $O(N \log N)$ approaches using space-partioning trees have been introduced with great success. Overall however, the general disconnection of these statistical problems from the areas which study algorithmic techniques is painfully clear.

Preview of Chapter 3.

It is hopefully clear, from the progressive development of this Section, that there is a certain correspondence between statistical representational power and computational expense. We should not be surprised that the most powerful and empirically successful methods have the highest cost. In Chapter 3 we will focus on **kernel density estimation**, which represents a certain computational extreme. Its particularly acute computational expense has historically been a major impediment to many potential applications. We'll expend particular effort to treat all of its pragmatic issues

with vigor. The resulting method dramatically changes the tractability status of this problem.

Because it can be seen as an extreme case of many of the other computational problems in this Section, its solution can be used as a prototype for them. Moreover it represents our general approach to continuous kernel functions. This will transfer readily to the next Section.

§1.4 Simulation of basic systems.

It is not enough to say that the correspondence between geometry and physics is natural. As stated by Galileo, "The book of nature is written in the characters of geometry." And in Einstein's words, "Geometry is the earliest form of physics." Of course many types of physical objects, across all known scales, are conveniently modeled as points in space. The approximation of real objects with non-zero extent as zero-dimensional point masses tends to be most sensible on the tiniest and largest of scales, where inter-body distances can be assumed to be vastly large relative to the effective diameter of the individual bodies. It is also necessary that the masses are sufficiently large that classical mechanics are sufficient to describe the system of interest. (We will briefly touch upon the type of system where quantum mechanical effects are non-negligible later.) Even after these caveats have been accounted for, a gigantic range of systems fall naturally under this sort of consideration: for example those involving (most) atoms, molecules, planets, stars, or galaxies.

Simulation problems.

Computational physics refers generally to any kind of study in physics for which intensive computation is necessary or integral - it thus includes all manner of numerical methods (which generally treat problems which cannot be approached analytically) which might arise in physical calculations [Ves94, Gio97]. Forward simulation of physical equations often serves this purpose, *i.e.* providing an alternative where analytical calculation is infeasible. For example, consider Kepler's problem of computing the trajectories of celestial bodies. Applying calculus to the equations of motion of two bodies under gravitation yielded an analytical expressions for their trajectories. However, for three or more bodies no expression is known [Ves94]. Hence the transformation of the 'N-body problem' from the mathematical realm to the computational.

However, the role of computation in physical studies is elevated to a new level when it effectively becomes the experimental ground itself. The ability to simulate a physical system has changed the face of physics by adding a distinct new investigative strategy to the toolbox of the scientific method: observation and measurement of phenomena in artificial representations of systems rather than actual physical systems. The broad activity of physical simulation is the classical source of N-body problems.

There are two major types of simulation problems: *dynamical* and and *equilibrium* simulations. The first is concerned with the time evolution of a system, for which we compute the potentials and thus forces acting on each particle at each time point. The second is concerned with the equilibrium configuration of a set of particles, generally averages or minima of quantities over all possible configurations. For example the *thermodynamic average* of a quantity $f(\underline{X}_{\mathcal{Q}})$ of a 'configuration' or dataset $\underline{X}_{\mathcal{Q}}$ is $\langle f \rangle = \int_{\mathcal{Q}} f(\underline{X}_{\mathcal{Q}}) p(\underline{X}_{\mathcal{Q}}) d\underline{X}_{\mathcal{Q}}$. In the second case the Monte Carlo method is generally used to generate configurations.

In principle either (or both) can be the goal, given a kind of object and the interaction kernel $\phi()$ between the objects (in this Section, the kernel function is often directly interpretable as a potential energy, though not always). The basic template for the calculation in each of the systems we'll survey is the same: for each

point $\underline{x}_q$ compute $\sum_{q' \neq q} \phi(\underline{x}_q, \underline{x}_{q'})$. In a dynamical simulation we also compute the force vector $\sum_{q' \neq q} \{-\nabla \phi(\underline{x}_q, \underline{x}_{q'})\}$, which we'll henceforth imply without stating. Generally we are in 3 dimensions or less in physical N-body problems.

1.4.1 Coulombic interactions.

Many kinds of physical interactions have the basic form $\phi(\underline{x}_q, \underline{x}_{q'}) = \frac{w(\underline{x}_q) w(\underline{x}_{q'})}{\|\underline{x}_q - \underline{x}_{q'}\|^a}$, where $w(\underline{x}_q)$ is the mass or charge of the q^{th} particle. We'll call these *generalized Coulombic* interactions. Typically the interest in this situation is in dynamical simulation. Notice that now a weighting factor attached to the query point enters the kernel function, though that turns out to be inconsequential.

When $a = 1$ we have the Coulomb interaction of electrostatics, occurring in molecular and atomic simulations:

Coulomb interaction:

$$\forall q, \text{Compute } \sum_{q' \neq q} \frac{\alpha_q \alpha_{q'}}{\|\underline{x}_q - \underline{x}_{q'}\|} \tag{1.19}$$

For $a = 6$ we have the London dispersion. The $a = 2$ case corresponds notably to the *gravitational* interaction (or the shielded Coulombic interaction), occuring perhaps most prototypically in celestial mechanics simulations:

Gravitational interaction:

$$\forall q, \text{Compute } \sum_{q' \neq q} \frac{\alpha_q \alpha_{q'}}{\|\underline{x}_q - \underline{x}_{q'}\|^2} \tag{1.20}$$

This particular kernel function, the generalized Coulombic interaction, is particular well-studied, and the corresponding algorithms for this case are quite good, though they also have certain disadvantages. We'll in fact visit those methods in detail.

1.4.2 Non-Coulombic interactions and quantities.

In statistical mechanics simulations a number of non-Coulombic interactions appear. Non-Coulombic kernel functions also appear heavily in contexts outside of just potential energies and forces.

Model systems.

Perhaps the simplest model system of the so-called *molecular dynamics* method is the standard *hard-spheres*, or 'hard-disks' model. Rather than a potential energy, the goal is to compute, for each particle, the smallest collision time (and the corresponding partner, which is implied):

Hard-spheres collision time:

$$\forall q, \text{Compute } \min_{q' \neq q} \frac{-b - \sqrt{b^2 - v^2(\|\underline{x}_q - \underline{x}_{q'}\|^2 - h^2)}}{v^2} \tag{1.21}$$

where h is the sphere diameter, $b = (\underline{x}_{q'} - \underline{x}_q) \cdot (\underline{v}_{q'} - \underline{v}_q)$, and $v = \|\underline{v}_{q'} - \underline{v}_q\|$.

A move up in fidelity brings us to the standard model for simple liquids, which is characterized by the *Lennard-Jones* interaction:

Lennard-Jones interaction:

$$\forall q, \text{Compute } \sum_{q' \neq q} \left\{ \left(\frac{\sigma}{\|\underline{x}_q - \underline{x}_{q'}\|} \right)^{12} - \left(\frac{\sigma}{\|\underline{x}_q - \underline{x}_{q'}\|} \right)^{6} \right\} \tag{1.22}$$

where σ is a substance-specific parameter. In polar fluids a Coulombic term is also added to the kernel.

Ensemble averages.

We have already seen some examples of a fundamental type of all-pairs computation which is not simply the pairwise potential or force between bodies that the term 'N-body problems' connotes in physics.

Many common thermodynamic averages are N-body problems, such as the two most elementary observables, the *internal energy* of a piece of matter:

Internal energy:

$$\text{Compute } U = NkT + \frac{1}{2}\langle \sum_q \sum_{q \neq q'} \phi(\|\underline{x}_q - \underline{x}_{q'}\|) \rangle \tag{1.23}$$

and the *pressure*:

Pressure:

$$\text{Compute } p = \frac{NkT}{V} - \frac{1}{6V}\langle \sum_q \sum_{q \neq q'} \delta_{qq'} \frac{d\phi}{d\delta}|_{\delta_{qq'}} \rangle \tag{1.24}$$

where $\delta_{qq'} = \|\underline{x}_q - \underline{x}_{q'}\|$.

A simulation's quality must be evaluated, via diagostics. These include the mean *density*, which will appear in various forms later. A fundamental statistic to measure is the *2-point correlation function* or 'pair correlation function' or 'radial distribution function':

2-point correlation function:

$$\forall h, \text{Compute } \sum_q \sum_{q' \neq q} \zeta(h) = \frac{1}{N_{\mathcal{Q}}(N_{\mathcal{Q}}-1)} I(\|\underline{x}_q - \underline{x}_{q'}\|_t < h) \tag{1.25}$$

This will reappear in a much more general form in the next Section. One interesting twist introduced by this problem is the fact that we must compute this for a range of different h's. We'll see why this is significant later.

Quantum mechanics.

Despite our caveats about being sure we can model systems as point objects, these sorts of simulation methods can be applied usefully to a large extent in the quantum mechanical realm.

In the *diffusion Monte Carlo* method, we consider $N_{\mathcal{Q}}$ ensembles made up of $N_{\mathcal{R}}$ particles each, and wish to compute the *average density* at time t:

Average density:

$$\text{Compute } \langle \hat{p}_h \rangle = \sum_q \sum_r \frac{1}{N_{\mathcal{Q}}} \frac{1}{N_{\mathcal{R}}} I(\|\underline{x}_q - \underline{x}_r\|_t < h) \tag{1.26}$$

This has also appeared already in the guise of the 2-point correlation, in a single-dataset form.

In the *path integral Monte Carlo* approach, we must compute the P-element *ring chain average*, yet another kind of mean potential:

Ring chain average: (1.27)

$$\text{Compute } \langle \Phi \rangle = \frac{1}{P} \sum_p \frac{1}{N_{\mathcal{Q}}} \frac{1}{N_{\mathcal{Q}}-1} \sum_q \sum_{q \neq q'} \phi(\|\underline{x}_q^p - \underline{x}_{q'}^p\|)$$

For the *wave packet dynamics* method, we must compute averages where particles are smeared out by a Gaussian or other convolution, approximating quantum wave packets. The wave function is then represented as

Wave function: (1.28)

$$\forall q, \text{Compute } \Psi(\underline{x}_q) = \prod_{q \neq q'} \phi(\|\underline{x}_q - \underline{x}_{q'}\|) \tag{1.29}$$

Fluid dynamics.

In fluid dynamics methods which use aggregation approximations, various contrived kernel functions appear, for example in Chorin's vortex blob method for viscous flow.

A more modern example is the widely-used *smoothed particle hydrodynamics* method for which contains two N-body problems; one is to compute the density at each point:

Smoothed particle hydrodynamics (density):

$$\forall q, \text{Compute } \hat{p}(\underline{x}_r) = \sum_r \frac{1}{N_{\mathcal{R}}} m_r \phi\left(\frac{\|\underline{x}_q - \underline{x}_r\|}{h}\right) \tag{1.30}$$

where $w(\underline{x}_r) = m_r$ is the mass of the r^{th} particle and h is the chosen bandwidth. Note that this is no different from kernel density estimation, with the small exception of a weighting factor. SPH also seems to be missing the important notion that h should be estimated according to rigorous statistical criteria. The second problem problem is no different from kernel regression:

Smoothed particle hydrodynamics (function):

$$\forall q, \text{Compute } f(\underline{x}_q) = \sum_r \frac{1}{N_{\mathcal{R}}} m_r \frac{f(\underline{x}_r)}{\hat{p}(\underline{x}_r)} \phi\left(\frac{\|\underline{x}_q - \underline{x}_r\|}{h}\right) \tag{1.31}$$

where the function is some quantity of interest such as the pressure.

For various physical reasons, though certain customized smoothing kernels $\phi()$ are preferred in SPH, such as spline kernels. Another thing to note is that the preferred SPH method selects bandwidths h_r dependent on the reference point, as in variable-kernel estimation.

1.4.3 Current state-of-the-art and our focus.

It should be clear at this point that a large number of all-pairs problems arise in computational physics quite naturally, far beyond the prototypical 'N-body problem' of Coulombic/gravitational interactions. In particular, statistical mechanics is rife with these problems. It is possible that physical N-body problems, of various sorts, are the single largest consumer of supercomputer time and expense.

The history of work in physics on these computational problems is actually exceptional in its sophistication, though of somewhat narrow scope. Grid-based methods and the fast Fourier transform appear quite prominently, just as we saw in statistics. However, two methods emerged which demonstrated dramatic advancement over these traditional approaches. The *Barnes-Hut* method is a direct analog of the successful tree-based methods for statistical problems mentioned in 1.3.8, as we will see, having $O(N \log N)$ complexity. The Greengard-Rokhlin *fast multipole method* (FMM) improves that complexity to $O(N)$, in cases where the kernel function is appropriately mathematically amenable. The generalized Coulombic interactions represent the class for which this has been worked out (with the exception of the Gaussian kernel in very low dimensions via the variant which was named the *fast Gauss transform*).

While inferior in complexity, Barnes-Hut coexists with the FMM today for three main reasons. Implementation of the FMM is far more difficult to understand and perform, and becomes a specialty in itself, limiting its widespread use. Furthermore, it applies to only a few cases of kernel function - though they are important ones, the FMM is not relevant to a great many important N-body problems. Finally, even in terms of performance, the FMM does not clearly dominate Barnes-Hut in many practical regimes, despite its asymptotic advantage - the reason being the massive constants in its runtime caused by the multipole expansions at its heart.

Preview of Chapter 4.

We will fill the gap left by this situation. In Chapter 4 we will focus on one of the more recent N-body simulation methods to be developed, the **smoothed particle hydrodynamics** method, which is of great current interest and actually responsible for much of the work being done on supercomputers today. It is also a prominent example of a non-Coulombic problem in physics, thus out of the reach of the linear-time FMM. Fortuitously we'll see that almost all of what we develop for the kernel estimators in our statistics focus will transfer here seamlessly. The result is in some sense the best of both of the Barnes-Hut and FMM worlds - an $O(N)$ method which is extremely simple to implement and understand, and is applicable to virtually any kernel function.

The multipole expansion is not to be forgotten however, as it is a powerful tool in our arsenal when it can be applied. Despite the level of mathematical sophistication in the computational physics approaches to N-body problems, the lack of contact with fundamental computer science ideas is evident. We will suggest ways to improve the existing FMM algorithm, using our new computational-geometric perspectives.

§1.5 Basic morphological questions.

What is computational morphology?

There are a number of areas in which the goal is to somehow formalize and operationalize notions of *shapes* or complex and distinct structures in data. I'll call this type of activity *computational morphology*, to use a term that has been previously coined to capture the variety of technical approaches that have been developed for this pursuit. This includes, at a minimum, large parts of spatial statistics [Rip81], statistical physics [Kad00, MS00], stochastic geometry [MSK96], fractal geometry [Man82], dynamical systems, the part of machine learning concerned with nonlinear dimensionality reduction, certain self-contained theories of shape arising in different fields such as statistics and astrophysics, and the very new field of computational topology.

Note that computational geometry, pattern recognition, and indeed a few of the problems we have already encountered can arguably be considered exercises in characterizing 'morphology' or spatial patterns - one perhaps notable example being clustering. Nonetheless the distinction appears to be increasingly necessary, as very recent communication between various diverse fields (see for example [MS00]) has begun to crystallize previously disparate approaches into a distinct body of methodologies and aims. The sudden emergence of computational topology as an self-standing area of study as well as the very new embedding methods from machine learning add to this mass of new approaches and goals. Regardless, the kind of N-body problems we'll need to consider in this Section are different from previous ones we've covered in a key way.

Astrophysics.

To make things concrete as soon as possible, a chief thing we have in mind is a problem faced by astrophysicists when looking at the points in the sky after long efforts by instrument builders, expert observers, data cleaners, and database builders: what is the *large-scale structure* of the universe? *Astrophysics*, a natural and prolific source for many of the N-body problems we've already described. Aside from obvious fits like the gravitation simulations and magnetohydrodynamic simulations, it should also come as no surprise that kernel density estimation, kernel regression, and spatial database queries are all of central interest in astrophysics. Because computational morphology is on the critical path to the questions it is trying to answer in a very direct way, astrophysics, in particular *cosmology* [**?**], is the natural home for us in this Section.

Again, the jump to modeling the objects of interests as points in a certain space should not be done blindly - in astrophysics we only do so subject to caveats about the fair sample hypothesis, the assumption that we are only effectively observing objects which are not too distant in time, and non-negligible differences between sky objects of different types [Pee80]. Those caveats aside, we can proceed to make inferences regarding more global properties of sets of points than the simple geometric relationships we have seen so far. To do so with rigor, the theoretical foundation of *spatial statistics* will permeate everything that we do, and indeed it permeates (and to an extent unites) many of the fields we have mentioned which are concerned with computational morphology.

Note that in astrophysics the outcomes of these morphological analyses can and do have dramatic consequences - consider Arp's quasar alignments claim [**?**], which if properly supported by spatial statistics threatens to upset the entire big bang model [BF96]. The importance of rigorous and efficient spatial statistics tools is in fact unquestionably at the heart of cosmology, to the point of defining the edge of its reach, as we'll see in the corresponding chapter of this thesis.

1.5.1 Embedding and dimensionality.

The new methods of *embedding* which have appeared in machine learning have the ambitious goal of constructing the presumed lower-dimensional nonlinear manifold upon which the data lie. In the *locally linear embedding* method, the first and most costly operation is an all-k-nearest-neighbors computation.

Estimation of the *intrinsic dimension* (or 'fractal dimension', 'Hausdorff dimension', 'correlation dimension', or 'Procaccia-Grassberger dimension') turns out to be identical to the 2-point correlation that we have already seen, modulo a log transformation:

Intrinsic dimension:

$$\forall h, \text{Compute } \sum_q \sum_{q' \neq q} \frac{1}{N_{\mathcal{Q}}(N_{\mathcal{Q}}-1)} I(\|\underline{x}_q - \underline{x}_{q'}\|_t < h) \tag{1.32}$$

A metric called the *Hausdorff distance* comes up in the study of fractals and interated function systems.

The *Hausdorff distance* from $\mathcal{Q}$ to $\mathcal{R}$ is asymmetrically defined as:

Haussdorf distance:

$$\text{Compute } H(\mathcal{Q},\mathcal{R}) = \max_q \min_r \|\underline{x}_q - \underline{x}_r\| \tag{1.33}$$

and the Hausdorff distance *between* $\mathcal{Q}$ and $\mathcal{R}$ is $\max(H(\mathcal{Q},\mathcal{R}), H(\mathcal{R},\mathcal{Q}))$.

1.5.2 Spatial statistics.

Spatial statistics is the area of computational morphology draws upon whenever we must make inferences regarding shapes or patterns formed by data points. It traditionally has arisen in studies of the distribution of sky objects, trees in forests, and atoms in gases.

The most basic model of spatial statistics is the *Poisson point process*. The natural starting point for elementary statistics of Poisson processes is the distribution of nearest-neighbor distances. The cdf can be investigated directly, or test statistics can be constructed as in the *Skellam-Moore* test:

Skellam-Moore statistic:

$$\forall q, \text{Compute } \frac{N(B(\sum_q \min_r \|\underline{x}_q - \underline{x}_r\|^2))}{N(B(h))} \tag{1.34}$$

where $B(\delta)$ means the ball of radius δ and $N(B(\delta))$ means the number of points falling within $B(\delta)$. This can be computed with an all-nearest-neighbor(-distance) operation followed by an all-range-count with query-dependent radius. A large number of variants on this kind of statistic have been proposed and used.

The n-point correlation.

The most widely-used and foundational theory of spatial point processes surrounds the study of $N(B)$, for all bounded Borel sets B. This can in fact be linked to the nearest-neighbor statistics above via the 2-point correlation, which we have already seen in 1.4.2 (though we won't detail this connection.).

The 2-point correlation is just the beginning, as it turns out, of an entire hierarchy called the *n-point correlation functions*, or 'spatial correlation functions', corresponding to moments of $N(B)$.

In 1.25 we denoted the 2-point correlation by $\zeta(h)$, where h represented the upper threshold on the distance δ_{qr} between points. In the more general context of n-point correlations, however, we'll need to add indices, denoting the n-point correlation for $n = 2$ by $\zeta_2(h_{qr})$.

The 3-point correlation function requires the computation of a quantity called the *reduced 3-point correlation function*, a function of 3 lengths, corresponding to the sides of a triangle:

Reduced 3-point correlation function:

$$\forall h_1, h_2, h_3, \text{Compute} \zeta(h_1, h_2, h_3) =$$
$$\sum_q \sum_{q'>q} \sum_{q''>q'>q} \frac{1}{N_{\mathcal{Q}}(N_{\mathcal{Q}}-1)(N_{\mathcal{Q}}-2)}$$
$$I(\|\underline{x}_q - \underline{x}_{q'}\| < h_1, \|\underline{x}_{q'} - \underline{x}_{q''}\| < h_2, \|\underline{x}_{q''} - \underline{x}_q\| < h_3) \tag{1.35}$$

where the indicator function is 1 when all three conditions are met. We may also write the indicator function as a product of indicators $I(\|\underline{x}_q - \underline{x}_{q'}\|_t < h_1, \|\underline{x}_{q'} - \underline{x}_{q''}\|_t < h_2, \|\underline{x}_{q''} - \underline{x}_q\|_t < h_3) = I(\|\underline{x}_q - \underline{x}_{q'}\|_t < h_1) I(\|\underline{x}_{q'} - \underline{x}_{q''}\|_t < h_2) I(\|\underline{x}_{q''} - \underline{x}_q\|_t < h_3)$. For general n the normalizing factor is $\begin{pmatrix} N_{\mathcal{Q}} \\ n \end{pmatrix}$.

More generally we can also specify lower thresholds on the lengths, for example for the 3-point we would add l_1, l_2, and l_3:

Reduced 3-point correlation function (fully parametrized):

$$\forall l_1, l_2, l_3, h_1, h_2, h_3, \text{Compute } \zeta(l_1, l_2, l_3, h_1, h_2, h_3) = \tag{1.36}$$
$$\sum_q \sum_{q'>q} \sum_{q''>q'>q} \frac{1}{N_{\mathcal{Q}}(N_{\mathcal{Q}}-1)(N_{\mathcal{Q}}-2)}$$
$$I(l_1 \leq \|\underline{x}_q - \underline{x}_{q'}\| < h_1, l_2 \leq \|\underline{x}'_q - \underline{x}_{q''}\| < h_2, l_3 \leq \|\underline{x}''_q - \underline{x}_q\| < h_3).$$

Also more generally, the multi-chromatic version of the n-point correlation, sometimes called the *n-point cross-correlation*, in the 3-point example it would be:

Reduced 3-point cross-correlation function (fully parametrized):

$$\forall l_{qr}, l_{rs}, l_{sq}, h_{qr}, h_{rs}, h_{sq}, \text{Compute } \zeta(l_{qr}, l_{rs}, l_{sq}, h_{qr}, h_{rs}, h_{sq}) = \tag{1.37}$$

$$\sum_q \sum_r \sum_s \frac{1}{N_{\mathcal{Q}} N_{\mathcal{R}} N_{\mathcal{S}}}$$
$$I(l_{qr} \leq \|\underline{x}_q - \underline{x}_r\| < h_{qr}, l_{rs} \leq \|\underline{x}_q - \underline{x}_r\| < h_{rs}, l_{sq} \leq \|\underline{x}_s - \underline{x}_q\| < h_{sq}).$$

In general the parameters can be specified by two lower (or upper) triangular matrices $\underline{L}_n$ and uH_n containing $n(n-1)/2$ elements each. The general n-point cross-correlation is then:

Reduced n-point cross-correlation function (fully parametrized):
$$\forall \underline{L}_n, \underline{H}_n, \text{Compute } \Xi_n(\underline{L}_n, \underline{H}_n) =$$
$$\sum_{q_1} \cdots \sum_{q_n} \frac{1}{N_{\mathcal{Q}_1} \ldots N_{\mathcal{Q}_n}} \prod_{\nu\upsilon} I(l_{\nu\upsilon} \leq \|\underline{x}_{q_{1\nu}} - \underline{x}_{q_{n\upsilon}}\| < h_{\nu\upsilon}) \qquad (1.38)$$

Another kind of simple generalization leads to the *marked n-point correlation*, in which a function $f(\underline{x}_q)$, called a 'mark', is associated with each point $\underline{x}_q$, which is often an indicator corresponding to different discrete types of points but can also take other forms. A special case of the marked n-point correlation is the *weighted n-point correlation*, in which the marks are continuous weights w_q associated with each point $\underline{x}_q$. The *projected n-point correlation* accounts for the viewing plane using a coordinate transformation of the n points before matching.

Gibbs processes.

We have already seen an instance of the *Gibbs process*, in the hard-sphere model of statistical mechanics. It represents a basic departure from the Poisson process, in which the points interact. This branch point leads to its own large set of N-body problems, as one might easily imagine. For example we only mention here the entire class of *pair-potential processes*, a special case of a more general class of Gibbs processes having the form:

Notably, the need to process n-tuples is evident in this setting as well.

Combinatorial proximity problems.

We now return full-circle to the basic proximity problems of computational geometry. It is apparent that the basic n-point correlation can be thought of as a certain generalization of the total-range-count problem from pairs to n-tuples, which we might could call the *total-n-tuples-range-count* problem:

Total-n-tuples-range-count:
$$\text{Compute } h\text{−}count = \textstyle\sum_{q_1} \cdots \sum_{q_n} \prod_{\nu\upsilon} I(\|\underline{x}_{q_{1\nu}} - \underline{x}_{q_{n\upsilon}}\| < h) \qquad (1.39)$$

The entire set of generalizations of the proximity problems does not need enumeration, but one variation that might provoke ideas for applications is one we might call *largest-span set*:

Largest-span set:
$$\text{Compute } \arg\max_{q_1} \ldots \max_{q_n} \textstyle\sum_{\nu\upsilon} \|\underline{x}_{q_{1\nu}} - \underline{x}_{q_{n\upsilon}}\| \qquad (1.40)$$

or 'find the n-tuple of points whose span, or sum of pairwise distances, is maximal over all n-tuples'.

Constellation pattern-matching.

The use of shape templates occurs in computer vision when patterns of interest are naturally defined as n-tuples of points having a relationship $f(\underline{x}_1 \ldots \underline{x}_n)$ which can be defined in terms of their pairwise distances. For example, face-like patterns naturally correspond to a kind of variation of the marked n-point correlation where mouth-like objects, nose-like objects, and eye-like objects (corresonding to different marks) match constraints based on relative distances. More generally, such *constellation*-like patterns can be found by:

Constellation search: (1.41)

$$\forall q_1 \ldots \forall q_n \text{Compute } \{(\underline{x}_{q_1} \ldots \underline{x}_{q_n}) \mid I(\underline{x}_{q_1} \ldots \underline{x}_{q_n}) = 1\} = \bigcup \arg I_\Theta(\Delta_n)$$

where $I()$ encapsulates the matching criteria, based on the $n \times n$ matrix of pairwise distances Δ_n and parameters Θ incorporating quantities like relative length or angle factors.

1.5.3 Related problems.

Other intriguing theories of shape and morphology exist, some of which are just beginning to emerge as possibly viable practical tools. These include the theory of random sets, Kendalls's stochastic-geometric theory of shape, the Minkowski functionals, and computational topology. N-body problems are of course present in many of these approaches, but because their utility has not been fully established yet, we will not be discussing these N-body problems in this thesis.

As already mentioned, minimum spanning trees are useful tools in computational morphology, and have been applied to problems such as filament-finding.

Dendrograms have also found use in morphological investigations in astrophysics.

1.5.4 Current state-of-the-art and our focus.

Poisson point processes and the theory of n-point correlations constitute the best theoretical tools available for the statisical analyses of morphology required in astrophysics and many other natural sciences. As a very practical matter, the larger the value of n we can compute, the better. The 2-point correlation, sitting at $O(N^2)$ in straightforward cost, already poses a significant obstacle, particularly in the new age of massive sky surveys (*i.e.* large N). Consider also that, as is the case with many N-body problems, for a given dataset, it must be computed with many different parameter settings and often under many randomly generated datasets. The grid-based makes its customary appearance, with the same disappointing results. The FFT solution, with all its inadequacies, has been the main crutch in astrophysics for making progress under these conditions. The daunting $O(N^n)$ curve provides little reason for optimism concerning higher moments.

Despite the apparent hopelessness of the gap between the best available theory and its computational realizability, astrophysicists have long realized that the mere ability to compute the 3-point correlation for datasets beyond a toy-ish scale would provide a dramatic advance in insight concerning the cosmic structure [Pee80, SDS, TJ03]. Unfortunately the lack of an algorithmic approach has placed a cap even on the 3-point correlation, limiting this line of thinking mostly to unfulfilled rumblings.

Preview of Chapter 5.

In Chapter 5 we will develop algorithms for the **n-point correlation**, in its full generality, in particular for general n. Recognizing that the n-point correlation represents the generalization of the more common pairwise N-body problem from 2-tuples to n-tuples leads to a very natural extension of our shattering approach to n trees. The result is an elegant exact algorithm easily yielding billion-fold speedups over the standard computation for 3-point and 4-point cases.

Nonetheless, with the large number of experimental calculations that must be performed for the study of a single dataset as described earlier, and the fact that higher values of n are still out of reach, we explore a second new attack on the hardest n-point correlation computations. This extends the idea of geometric shattering with the Monte Carlo method, providing a unique blend of exact determistic computation and statistical averaging. We demonstrate a form of geometric stratified sampling providing confidence bounds on the error tight enough to satisfy scientific requirements, while yielding orders of magnitude in additional speedup. Together, these algorithms have changed the practical status of the n-point correlation as a computational obstacle.

§1.6 The generalized perspective.

A *generalized N-body problem* Γ is a 6-tuple $\{n, \chi, \Omega, \phi, w, \Theta\}$, expressing an instantiation of the following variables:

1. *Order.* n is the tuple *order*, *e.g.* $n = 2$ for pairs.

2. *Chromatic number.* χ is the *chromatic number* or 'chromaticity' if preferred, meaning the maximum number of datasets in the problem which can differ. For example the all-nearest-neighbors problem where the query and reference sets can be different is called the *bichromatic* version. Clearly $1 \leq \chi \leq n$.

3. *Operator set.* Ω indicates the set $\{\Omega_1 \ldots \Omega_n\}$, indicating the *operators* acting on each of the n datasets. Each operator indexes over the data in its corresponding dataset. Each operator is assumed to have the property of *decomposability over subsets*, *e.g.* if $A_1 \subset A, A_2 \subset A$, $\sum A = \sum A_1 + \sum A_2$. We could have also adopted the commutative semigroup formalism originally used by Fredman to unify nearest-neighbor and range-searching/counting [Fre81, AE97], but the notion is simple enough that this seems unnecessarily heavy-handed.

4. *Kernel function.* $\phi()$ is the *kernel function* defining the interaction between the n-tuples, usually a function of the distance(s) between them.

5. *Weight function.* w indicates a *weight function* which acts on single variable; in some conceivable cases there may be a different weight function for each dataset, making w a set, but we'll stick to the simpler case.

6. *Parameters.* Θ is a set containing arbitrary parameters for the kernel function and possibly also the operators.

The inputs of a generalized problem are D-dimensional datasets $\underline{X}_1 \ldots \underline{X}_n$, where up to χ of them differ. The output Y is a set of real values, ranging in possibility from a single count to a set of D-dimensional points. The size and form of Y depends on the operators Ω.

Definition 1 (Generalized N-body problem) *A generalized N-body problem is the 6-tuple $\{n, \chi, \Omega, \phi, w, \Theta\}$ (order, chromatic number, operator set, kernel function, weight function, parameters) whose solution is*

$$Y = \Omega_{i_1} \ldots \Omega_{i_n} \phi(\underline{x}_{i_1}, .., \underline{x}i_n) \tag{1.42}$$

where $\underline{x}_{i_1} \in \underline{X}_1, \ldots, \underline{x}_{i_n} \in \underline{X}_n$.

Problem	n,χ	Cardinalities	Operators	Kernel function	Dim.	Other parameters
BASIC GEOMETRIC QUERIES						
All-nearest-neighbors	2, 1	N	$\forall, \arg\min$	$\delta_{qq'}$	D	
Bichromatic all-κ-nearest-neighbors	2, 2	$N_\mathcal{Q}, N_\mathcal{R}$	$\forall, \arg\min^\kappa$	δ_{qr}	D	κ
Closest-pair	2, 1	N	$\arg\min, \arg\min$	$\delta_{qq'}$	D	
Diameter	2, 1	N	$\arg\max, \arg\max$	$\delta_{qq'}$	D	
Set distance	2, 2	$N_\mathcal{Q}, N_\mathcal{R}$	$\arg\min, \arg\min$	δ_{qr}	D	
All-range-search	2, 2	$N_\mathcal{Q}, N_\mathcal{R}$	$\forall, \bigcup \arg$	$I_h(\delta_{qr})$	D	h
All-range-count	2, 2	$N_\mathcal{Q}, N_\mathcal{R}$	$\forall, \sum$	$I_h(\delta_{qr})$	D	h
BASIC STATISTICAL INFERENCES						
κ-nearest-neighbor classifier	2, 2	$N_\mathcal{Q}, N_\mathcal{R}$	$\forall, \arg_k \arg\min^\kappa$	δ_{qr}	D	κ, $\mathcal{Q}_{[T]}$
	K+1, K+1	$N_\mathcal{Q}$, $N_{\mathcal{R}_1}, .., N_{\mathcal{R}_K}$	$\forall, \arg_k$ $\arg\min^\kappa, .., \arg\min^\kappa$	δ_{qr_k}	D	κ, $\mathcal{Q}_{[T]}$
Naive-Bayes classifier	2, 2	$N_\mathcal{Q}, K$	$\forall, \arg\max$	$e^{-\frac{1}{2}\delta_{qk}^2\ \sigma_k^2} P(C_k)$	D	$\{\sigma_k, P(C_k)\}$, $\mathcal{Q}_{[T]}$
Equi-spherical naive-Bayes classifier	2, 2	$N_\mathcal{Q}, K$	$\forall, \arg\min$	δ_{qk}	D	$\mathcal{Q}_{[T]}$
Mixture model E-step	2, 2	$N_\mathcal{Q}, K$	$\forall, \forall$	$e^{-\frac{1}{2}\delta_{qk}^2\ \sigma_k^2} P(C_k)$	D	$\{\sigma_k, P(C_k)\}$, $\mathcal{Q}_{[T]}$, $\mathcal{R}_{[I]}$
K-means E-step	2, 2	$N_\mathcal{Q}, K$	$\forall, \arg\min$	δ_{qk}	D	$\mathcal{Q}_{[T]}$, $\mathcal{R}_{[I]}$
Mixture model log-likelihood	2, 2	$N_\mathcal{Q}, K$	$\sum, \log \sum$	$e^{-\frac{1}{2}\delta_{qk}^2\ \sigma_k^2} P(C_k)$	D	$\{\sigma_k, P(C_k)\}$, $\mathcal{Q}_{[T]}$, $\mathcal{R}_{[I]}$
Radial basis function network	2, 2	$N_\mathcal{Q}, K$	$\forall, \sum$	$e^{-\frac{1}{2}\delta_{qk}^2\ \sigma_k^2} P(C_k)$	D	$\{\sigma_k, P(C_k)\}$, $\mathcal{Q}_{[T]}$
Mixture Bayes classifier	2, 2	$N_\mathcal{Q}, LK$	$\forall, \arg\max_k \sum_{l_k}$	$e^{-\frac{1}{2}\delta_{ql_k}^2\ \sigma_k^2} P(C_k)$	D	$\{\sigma_k, P(C_k)\}$, $\mathcal{Q}_{[T]}$
	K+1, K+1	$N_\mathcal{Q}, L, .., L$	$\forall, \arg\max_k \sum, .., \sum$	$e^{-\frac{1}{2}\delta_{ql_k}^2\ \sigma_k^2} P(C_k)$	D	$\{\sigma_k, P(C_k)\}$, $\mathcal{Q}_{[T]}$
Kernel density estimation	2, 2	$N_\mathcal{Q}, N_\mathcal{R}$	$\forall, \sum$	$\phi(\frac{\delta_{qr}^2}{h^2})$	D	$h_{[B]}$, $\mathcal{Q}_{[T]}$
Nadaraya-Watson regression	2, 2	$N_\mathcal{Q}, N_\mathcal{R}$	$\forall, \sum$	$y_r \phi(\frac{\delta_{qr}^2}{h^2})$	D	$h_{[B]}$, $\mathcal{Q}_{[T]}$
Kernel density Bayes classifier	2, 2	$N_\mathcal{Q}, N_\mathcal{R}$	$\forall, \arg\max_k \sum_{r_k}$	$\phi(\frac{\delta_{qr_k}^2}{h_k^2}) P(C_k)$	D	$\{h_{k[B]}, P(C_k)\}$, $\mathcal{Q}_{[T]}$
	K+1,K+1	$N_\mathcal{Q}$, $N_{\mathcal{R}_1}, .., N_{\mathcal{R}_K}$	$\forall, \arg\max_k \sum, .., \sum$	$\phi(\frac{\delta_{qr_k}^2}{h_k^2}) P(C_k)$	D	$\{h_{k[B]}, P(C_k)\}$, $\mathcal{Q}_{[T]}$
SIMULATION OF BASIC SYSTEMS						
Generalized Coulombic interaction	2, 1	N	$\forall, \sum$	$\alpha_q \alpha_{q'} \frac{1}{\delta_{qq'}^a}$ *	1-3	$\mathcal{Q}_{[S]}$, $\mathcal{Q}_{[I]}$
Hard-spheres collision time	2, 1	N	$\forall, \min$	$\sqrt{b^2 - v^2 \delta_{qq'}^2 - h^2}$	1-3	$\mathcal{Q}_{[S]}$
Lennard-Jones interaction	2, 1	N	$\forall, \sum$	$(\frac{\sigma}{\delta_{qq'}})^{12} - (\frac{\sigma}{\delta_{qq'}})^{6}$ *	1-3	$\mathcal{Q}_{[S]}$, $\mathcal{Q}_{[I]}$
Thermodynamic average	2, 1	N	$\sum, \sum$	$\phi(\delta_{qr})$	1-3	$\mathcal{Q}_{[S]}$, $\mathcal{Q}_{[I]}$
Average density	2, 2	N	$\sum, \sum$	$I_h(\delta_{qr})$	1-3	$h_{[B]}$, $\mathcal{Q}_{[S]}$, $\mathcal{Q}_{[I]}$
Wave function	2, 1	N	$\forall, \prod$	$\phi(\delta_{qq'})$	1-3	$\mathcal{Q}_{[S]}$, $\mathcal{Q}_{[I]}$
Smoothed particle hydrodynamics						
(density)	2, 1	N	$\forall, \sum$	$\phi(\frac{\delta_{qq'}^2}{h_{q'}^2})$	1-3	$h_{[B]}$, $\mathcal{Q}_{[T]}$, $\mathcal{Q}_{[I]}$
(function)	2, 1	N	$\forall, \sum$	$y_r \phi(\frac{\delta_{qq'}^2}{h_{q'}^2})$	1-3	$h_{[B]}$, $\mathcal{Q}_{[T]}$, $\mathcal{Q}_{[I]}$
BASIC MORPHOLOGICAL QUESTIONS						
Intrinsic (fractal) dimension	2, 1	N	$\sum, \sum$	$I_h(\delta_{qq'})$	D	$h_{[B]}$
Hausdorff distance	2, 2	$N_\mathcal{Q}, N_\mathcal{R}$	$\max, \min$	δ_{qr}	D	
Skellam-Moore statistic						
(neighbor distances)	2, 1	N	$\forall, \min$	$\delta_{qq'}$	D	
(neighbor counts)	2, 1	N	$\forall, \sum$	$I_{h_{q'}}(\delta_{qq'})$	D	$h_{q'}$
(reference counts)	2, 1	N	$\forall, \sum$	$I_h(\delta_{qq'})$	D	h
2-point (cross-)correlation	2, 2	$N_\mathcal{Q}, N_\mathcal{R}$	$\sum, \sum$	$I_h(\delta_{qr})$	D	$h_{[B]}$, $\mathcal{R}_{[S]}$
n-point (cross-)correlation	n, 2	$N_\mathcal{Q}, N_\mathcal{R}$	$\sum, .., \sum$	$\prod_{\nu\nu} I_{lh_{\nu\nu}}(\delta_{q_\nu q_\nu})$	D	$\underline{L}_{[B]}$, $\underline{H}_{[B]}$, $\mathcal{R}_{[S]}$
Largest-span set	n, 1	N	$\max, .., \max$	$\sum_{\nu\nu} \delta_{q_\nu q_\nu}$	D	
Constellation search	n, 1	N	$\forall, .., \forall \bigcup \arg$	$I_\Theta(\Delta_n)$	D	Θ

Table 1.1: GENERALIZED N-BODY PROBLEMS. χ is the chromatic number. Quantities in brackets [·] indicate the multiplicity of the associated variables. * indicates that there is a second kernel function corresponding to a force.

In other words its solution can be found by straightforward application of its operators to its argument datasets. Since each operator indexes over each element of its corresponding dataset, the cost of computing a generalized N-body problem by following its definition is $O(N^n)$ if each dataset is of size N.

Left implicit in this problem description is the underlying space, assumed to be a *metric space* for the purposes of this thesis, though we'll shortly mention other possibilities. We'll assume the Euclidean metric throughout, though the generalization of everything we'll do to any Minkowski norm seems unopposed.

A problem is also usefully annotated with some 'typical' properties. In Table 1.1 we list the associations of the *cardinalities* or sizes of the datasets, *e.g.* the fact that in a mixture model one of the datasets corresponds to the K states, and the typical dimensionality, listed simply as D when it can be arbitrary. It also sometimes the case that a problem typically needs to be solved for multiple settings of certain variables or parameters, in which the *multiplicity*, or number of such settings, is shown in square brackets with the parameter. For example $\mathcal{Q}_{[T]}$ indicates that the query dataset often differs over T different *tasks*, $\mathcal{Q}_{[I]}$ that the data changes over I iterations, $\mathcal{R}_{[S]}$ that the reference set is replaced according to S different Monte Carlo samples, and $h_{[B]}$ that the bandwidth ranges over a set of B scales.

Table 1.1 reveals certain themes which characterize different sub-classes of generalized N-body problems:

- *Composite problems and simultaneous solutions.* Some problems seem to be characterizable in terms of multiple steps, which are each N-body problems – for example the Skellam-Moore statistic and smoothed particle hydrodynamics. While such 'composite' problems could be solved by separate algorithms for each of the steps, a more efficient single-pass or *simultaneous* solution might be possible.

- *Properties of kernel function.* There is intuitively some kind of qualitative difference between kernel functions which have more of a discrete character such as indicator functions, and general continuous kernel functions such as polynomials and Gaussians. We will see that certain other mathematical properties of the kernel function will make a difference in the choice of solution, such as the analyticity or differentiability properties of the function.

- *Dynamic problems.* In general, we'll see that *multiplicity* in a problem can be a source of opportunity. When the different parameter settings correspond to problems with similar structure, some of the computational effort can be shared. The *dynamic* case, in which the N-body problem changes from one iteration to the next — but not very much — is one in which we need not redo all our computational steps.

- *Enumeration problems.* It is important to note that the $\forall$ operator expands the size of the output. For example, finding the nearest-neighbor for *every* query point means returning a vector of $N_{\mathcal{Q}}$ indices, while finding the closest pair means returning a single pair of indices. These distinct types of problems typically do not actually differ in worst-case complexity, unfortunately — though it is intuitive that less *careful* work needs to be done in the latter case, and so it typically can be performed in less time. Another important aspect of an enumeration is that the space needed to store the answer may begin to be a problem in some cases, particularly when there are multiple $\forall$ operators.

- *Optimization and decision problems.* The $\min$ and $\max$ operators imply a minimization/maximization, or *optimization* problem. Optimization over discrete choices can also be called a *decision* problem. A typical form in our case is finding the class for which a certain sum is highest. While an optimization problem can be reduced to an enumration problem, it is intuitive that this is not necessarily going to be the most efficient approach. An observation along these lines is to be found in Table 1.1 in the alternative formulations of the κ-nearest-neighbor classifier, mixture Bayes classifier, and kernel density classifier. Here we have noted that it might be profitable to view each class as forming its own dataset, and the resulting computation as a joint optimization problem over all of these K datasets in concert with the query dataset. The possibility that this might yield speedup over the usual train-test two-dataset perspective will not be explored in this thesis but we regard it as an avenue to be investigated. This example illustrates the general fact that the appearance of an optimization operator in a problem opens to the door to a larger set of algorithmic possibilities.

1.6.1 Computer science perspective.

The computer science toolbox, ideally, applies to all problems computational, and could and should form a powerful central resource from which any field facing a computational obstacle can draw. Statisticians and natural scientists *must* solve their most pressing computational problems in one manner or another, and if necessary will resort to home-brewing tools so that they can make progress. However, technically trained people in almost any field know that if they come across a sorting problem, there are non-obvious solutions lying in computer science, which are likely to be better than what one might cook up oneself as a side occupation.

In our tour of problems and fields we saw this has not been the case for the generalized N-body problems, for which computer science has had little to say, and home-brewed computational solutions have filled the void. The root of this is an artifact of cultural divides in research; in order for computer science to be relevant to problems of human interest, someone must define them as *computer science problems*. Normally, however, it is up to the computer scientist to do this rather than the statistician or the biologist, because the business of framing things usefully as computer science problems requires computer science expertise. Sorting and other classical discrete problems were defined by computer scientists, but computer science must step out of itself to be relevant in the wider world of computational problems.

We have taken this step, showing that many problems are generalizations or siblings of some problems which have already been defined as computer science problems, the proximity problems. It required entering other fields to a certain depth, but reaped rewards even without any further work - the best existing solutions can now be matched to problems in fields that had been unaware of them, based on an abstract categorization. Now we're in a position to take the next step and develop a new class of computer science tool for this new class of computer science problem.

1.6.2 Overall state of affairs.

Our tour through the various problems and their home fields revealed some distinct recurring patterns. A small number of different types of solutions appear in different guises for different problems, but are usually easily recognizable.

Here we summarize the main properties generally required by practical applications, as well as a qualitative evaluation of how well existing solution classes meet these desiderata. In general, what characterizes a 'good' solution in statistical and scientific computation?

- **Large N.** For many reasons, ever-larger numbers of points are demanded in applications. Generally the ability to use a much larger N makes a significant *qualitative* difference in what can be achieved. Of course an approximate method must perform its computation with accuracy to be valuable.

- **Arbitrary D.** In many problems, the need to incorporate ever-larger numbers of different kinds of measurements means that arbitrary dimensionalities D must be handled with efficiency. This is not necessarily the case, as in many physical problems which are invariably in 3 or fewer dimensions.

- **Controllable error.** The ability to arbitrarily control the error, *i.e.* specify the accuracy-performance tradeoff of an approximate algorithm, is often possible only through tweak parameters whose effect on the final error of the approximation is not known directly, thus requiring time-consuming experimentation by the user.

- **Known actual error.** Although most approximate algorithms admit some form of at least indirect error control, bounds on the *actual* error resulting from any particular run of the algorithm are generally unknown. This reduces use of the algorithm to an act of faith, which is not particularly desirable in supposedly-careful statistical or scientific analyses.

- **Not obtuse.** Experience shows that the primary factor in selecting which algorithms are most often used in real-world applications is their simplicity. An algorithm intended to make an impact in practical problems cannot be frustrating to understand, diagnose, or implement from scratch. Scientists, in particular, do not trust their analyses to opaque algorithms.

- **General.** Algorithms which are highly specialized to only certain kinds of problems, *i.e.* not robust to moderate changes in the problem context or definition, tend to be much useful in practice because few statisticians and scientists have the time or patience to become familiar with a large number of special-case methods.

Another important constraint, which might be assumed to hold for any given algorithm but is in fact often violated, is that the algorithm should not solve a *different* problem than the one actually posed. Of course each problem also has its own problem-specific constraints and desiderata.

	Grids	FFT	Barnes-Hut	FMM	new?
Large N?	–	+	++	+++	?
Arbitrary D?	–	–	–	–	?
Control error?	+	+	+	++	?
Known error?	–	–	–	+++	?
Not obtuse?	+++	–	+++	–	?
General?	+++	–	+++	–	?

Table 1.2: EXISTING SOLUTION CLASSES FOR GENERALIZED N-BODY PROBLEMS. The number of '+' symbols indicates the extent to which the property in question is satisfied. '–' indicates failure to satisfy the property.

We will consider these solutions in greater problem-specific detail in the appropriate individual chapters, but it is possible here to get a feeling for the overall state of affairs for generalized N-body problems, summarized in Table 1.2. Each of the existing solution classes has its strengths, but none of them is widely satisfactory, posing the challenge of proposing a new class of solutions which has a more desirable blend of practical properties.

§1.7 Related observations.

The main contribution of this chapter was a set of high-level bird's-eye-view observations concerning some global connections. Subsets of these observations have been made in by at least a few authors we are aware of, in each of the different areas.

From computational geometry.

Computational geometry might be expected to naturally find opportunities in other disciplines in which to formalize geometric problems and develop solutions for them. This has certainly not been the case by-and-large, however. As might be expected, the generality of a geometric perspective implies that there a vast number of geometric problems, occurring in virtually every scientific and engineering endeavor. Computational geometry *per se* has focused on a fairly narrow subset of these problems, while hordes of them find study in their respective applied fields — computer graphics, computer vision, manufacturing, computer-aided design, robotics, and geographic information systems, to name a few. Not unrightfully so, computational geometry represented the necessary development of the study of geometric problems from a fundamental computer science perspective, and thus might have been more correctly called 'geometric complexity theory'.

This precise issue, with a focus on the disparity between the field and its general disconnection from real-world applicability, was the topic of a recent task force report written by 20 leading researchers in computational geometry [Cha99]. Their candid critique of their own field was laudable and full of insights regarding potential connections between computational geometry and problems in other disciplines. One of these which is relevant here is the potential lying in astrophysics — though they only noted the obvious problems of spatial database querying and N-body simulations for celestial mechanics. One major connection along the latter vein was made by the work on the well-separated pair decomposition [Cal95], which we'll visit in more detail in Chapter 4. The authors of that work also noted the connection between the N-body problem and the all-nearest-neighbors problem.

Certain statistics problems were considered quite early in the development of computational geometry by Bentley and Shamos in [Sha75, BS76], though this worthy-seeming line did not seem to be pursued to the point of affecting practice.

Eppstein has collected a good online bibliography listing instances where computational geometry has touched statistical problems [Epp99]. Most of these problems concern very basic quantities such as centroids, medians, and contours, rather than the statistical *inferences* we have considered here,with the exception of a number of papers which have considered clustering, of both the K-means and hierarchical variety. The nearest-neighbor problem is quite widely understood to be a pattern recognition problem, and is fact universally cited as a motivation for studying the problem. It is difficult to find mention of the all-nearest-neighbors problem in this context, however.

The statistical relevance of the Euclidean minimum spanning tree, Voronoi tesselation, convex hull, and Delaunay triangulation problems, all fundamental in computational geometry has been noted by many authors, but these are not N-body problems. Computational geometry has made significant connections with computational physics in the area of mesh generation for finite-element methods — while sharing some abstract similarities to our approaches, these are also not N-body problems.

From computational statistics.

Some computational-geometric approaches have originated from within statistics and machine learning. Two of the broadest research programs, in terms of pointing out the entirety of the possible scope of the applicability of computational geometry to the kinds of statistical inference problems considered here are those of Omohundro [Omo87, Omo90, Omo91] and Moore [DM95, PM99, ML98, Moo99]. It is in fact these approaches which we now generalize and extend to the current context of N-body problems. Uses of tree structures have appeared in the pattern recognition literature for specific individual problems, though generally the focus has been on variations of the nearest-neighbor problem.

From computational physics.

The discovery of geometric methods in computational physics [BH86, GR87], though regarded as a breakthrough for physical N-body problems, still has essentially not progressed beyond the most simplistic approach to geometric divide-and-conquer. The few exceptions to this are treated in Chapter 4.

A handful of attempts have been made to connect the well-known fast multipole method (though not Barnes-Hut) to the problem of nonparametric density estimation. Greengard himself [GS91] noted the connection, in the Gaussian kernel context. Two recent papers have also rediscovered this [LHB+99, AED01]. However, they also discovered the reasons that these methods have limited applicability to this problem. This will be discussed in Chapters 3 and 4.

From computational morphology.

There seems to have been no major computational-geometric approaches of note arising from this set of disparate areas, several of which are relatively new.

§1.8 Summary of this chapter.

We can now summarize the main points of this chapter:

- **Inter-field and intra-field connections.** By unifying various disparate problems in a common definitional framework, we implicitly made a network of connections, as noted earlier. This opens the door to transfer of separately-developed insights and solutions across fields and problems within the same field.

- **These problems are hard and important.** We saw that generalized N-body problems of various sorts lie at the cores of several fields, and quite often their

inherent difficulty stands as a barrier to progress in major branches of statistical and scientific investigation. Why do these problems tend to be central? One explanation is that their common nature in the sense of direct comparison or interaction between arbitrary points in space is intuitively somehow basic and raw. In some sense a generalized N-body is what one is left with once the artifice of assumptions or approximations such as periodic structure, uniformity, Gaussianity, or mean-field validity are stripped away.

- **Types of N-body problems.** The class of problems we have called generalized N-body problems can be clustered according to some key properties. We shall see how this grouping naturally corresponds to the sub-problems we will solve in the subsequent chapters.

Publications.

The publications related to the specific problems (and their solutions) of each chapter will be covered in the relevant chapter.

The description and unification of our statistical problems as N-body problems was for the most part given in:

- Gray, A. and Moore, A. *N-Body Problems in Statistical Learning*, NIPS 2001 (selected for oral presentation).

It seems that at least three other publications could do much to fill the holes of awareness of the global picture we have painted:

- An article for the general computer science community (*e.g.* JACM) describing these connections from the point of view of opportunities for computer science contributions.
- An article for the statistical physics community describing the many N-body problems there, which do not seem to have been treated by even the quite-effective and existing Barnes-Hut method. This would of course also introduce for the problems the dual-tree methods we'll develop in this thesis.
- An article for the spatial statistics community, or perhaps a somehow audience which somehow also at least encompasses fractal geometry, detailing the geometric treatment of these problems as developed in this thesis.

What's next?

The rest of this thesis fleshes out the trace we've made through the four different fields. This field-by-field organization happily coincides with a logical problem-type-by-problem-type organization treating major branches of the N-body problem taxonomy. We'll start with solutions for the elementary proximity problems in computational geometry. Then we'll show how to extend the same solution method in the more general case of kernel function; the main vehicle for this will be statistics. We'll then traverse a natural bridge to physics problems and show that the same solution can be used there; we'll also at that point take a deep look at the state-of-the-art methods from physics, contrasting and combining them with our new approach. Finally we'll use the problems in computational morphology to demonstate the significant case where pairs are generalized to n-tuples, and show that there is a natural generalization of our solution approach for that case; we'll also at that point augment our geometric approach with a probabilistic method.

2 N-Body Problems in Computational Geometry

Geometric Shattering I: Divide-and-Conquer Tools.

Reduce big troubles to small ones, and small ones to nothing. — Chinese proverb.

THE TOOLBOX OF COMPUTER SCIENCE, while immense, is at the same time understandable in terms of a smaller number of templates, or algorithm *design principles*. There are hammers, motors, and lenses, which are really templates for many ingenious variations tuned for specific tasks like contact lenses and space telescopes. In this chapter we will look at new forms of *divide-and-conquer*, one of the most fundamental algorithmic design principles in computer science.

Agenda of this chapter.

In this chapter we'll review divide-and-conquer in various forms of interest to us, specifically one form employing geometric tree data structures — our main tool from computational geometry. We'll see how to *extend* this tool (and in fact divide-and-conquer more generally) with the method of *geometric shattering*. Our main vehicle for demonstrating and explaining this will be the **all-nearest-neighbors** problem, though we'll also show that this easily extends to the other major N-body problems in computational geometry.

§2.1 Proximity problems.

The *proximity problems*, or *closest-point* problems, were usefully collected under a single heading early in the history of computational geometry by Shamos and Hoey [SH75]. The main justification for this was in fact that a common solution method exists for the problems in this class (and in fact the only real definition of the class is in these terms).

Interestingly though, that common solution was *not* divide-and-conquer — in fact classical divide-and-conquer approaches that were natural for some problems could not be generalized to other problems in the class [PS85]. It was discovered that they could all be solved by reduction to the same problem - that of constructing a *Voronoi diagram*, leading to a clean and tight algorithmic theory for proximity problems... But only in the plane — for $D > 2$ the Voronoi diagram approach breaks down, as its size is exponential in the dimension.

Some recent treatments, *e.g.* [AE97], encapsulate this class of problems under the class of 'range searching', which generalizes to the class of 'intersection searching', but otherwise use a similar semigroup framework for unifying the operations of range-searching and range-counting, nearest-neighbor-searching, along with many other kinds of problems which will not be our focus.

Unfortunately the type of complexity exhibited by the Voronoi approach is typical of *almost all* of the work in computational geometry on proximity problems aside from a certain class of approach. And this brings us directly to what we'll consider the practical constraints on solutions to this problem.

2.1.1 Practical considerations and constraints.

In order to make our solutions to these basic proximity problems useful and transferable to the problems existing in the statistical and scientific settings we are interested in solving, we need to delineate a number of considerations which will constrain our approach.

Arbitrary dimension D.

Algorithms which scale exponentially in D are said to suffer from the *curse of dimensionality*. This behavior tends to quickly render problems beyond even one or two dimensions intractable, and indeed this has been the weakness of all virtually all the standard solutions to N-body problems. This is the fundamental point of departure which leads to more sophisticated perspectives from computational geometry.

One very significant constraint we will place on our algorithmic approach to all-nearest-neighbors and its close relatives is that it work as well as possible in *arbitrary dimension*, *i.e.* not just 2, but possibly 10, or 1,000 or more in some cases. Thus the Voronoi approach is ruled out. A return to divide-and-conquer, though in a different form, will turn out to provide an approach that succeeds in uniting the proximity problems, and also provides efficiency in much higher dimensionalities.

Reusable structure S.

A second important constraint on an algorithmic approach to proximity problems is whether reusable data structures are allowed. For example in the basic nearest neighbor problem (coming up), the standard setup for the problem allows the construction of an indexing structure S, a one-time preprocessing cost, which can be utilized at query time.

In all of the practical problems of interest in this thesis, this is not an issue provided the cost of building the data structure is still reasonable for very large datasets in arbitrary dimension. The reason our complexity requirement on this construction phase is somewhat lax is that it need be performed only once for the entire lifetime of the dataset, while in practice N-body computations of interest in statistics and science are repeated multiple times on the same data under different variations.

Note that this is not always sensible in the abstract; for example in the all-nearest-neighbors problem the queries do not change, and thus there is no reuse possible in the normal sense. However even for a problem like all-nearest-neighbors, the ability to essentially pre-factor part of the computation out has benefit for us when there are other N-body problems present in the same context, which can all share the precomputation. Our unified approach algorithmic actually makes this the case. It also provides efficiency when different query sets are expected to be reused with different reference sets, as is generally the case in most of N-body problems in this thesis in practice.

All that having been said, the whole issue is moot if the preprocessing cost is actually negligible. For the *kd*-trees and ball-trees that we use, this is arguably the case for all but the largest problem sizes. Nonetheless, we do not want to rule out the possibility of reusable data structures which might be slightly more costly to build but provide a higher payoff in efficiency.

Ease of implementation.

Very generally, any algorithm sees more use when it is relatively *easy to implement*. Part of the cost of any project, in many terms, is the implementation of the tech-

niques themselves. This fact of life has been proven in practice time and again, and needs no further explanation.

Tweak parameters.

It is also well-known that algorithms containing many *tweak parameters*, or values whose settings significantly affect the performance or behavior of the algorithm, are frustrating and time-consuming to use. Such parameters become an issue when there is no clear theoretical guidance or rigorous methodology for determining their optimal value. These arise heavily in virtually all of the standard solutions to N-body problems.

Storage media and systems.

In the database community, the assumption is made that the datasets are much too large to fit in main memory. This leads to a very different set of constraints on problems such as nearest-neighbor. A significant one is that the indexing structure must be built in an on-line fashion or dynamic fashion rather than in batch. Another significant constraint is that the index itself is stored in secondary memory. This implies that very little information be stored in the nodes so that many accesses of disk blocks can obtain as many nodes as possible.

The database-centric issues regarding the physical storage and manipulation of data in secondary memory are beyond the scope of this thesis. We'll be focused on what can or cannot be done at the rawer algorithmic level, assuming our data can be stored in the most favorable way (all in RAM) and touching only briefly on the secondary-storage case. We will mainly be focused on trying to extract whatever tools and perspectives we can leverage from computational geometry and database systems to help us attack the broad panoply of N-body problems we're considering.

2.1.2 Nearest-neighbor search.

The basic problem of *nearest neighbor* searching is the prototype for all the proximity problems we'll consider. It has received the most attention by far, lying at the center of a vast sea of literature in several fields (which we'll navigate later). Arguably it is the canonical problem in computational geometry, if not for any other reason than the It is simply the one-query version of all-nearest-neighbors:

Nearest neighbor:

Compute $NN(\underline{x}_q) = \arg\min_r ||\underline{x}_q - \underline{x}_r||$ (2.1)

It is of course only an N-body problem in a degenerate sense, but it is worth our investigation because our solution to all-nearest-neighbors will end up being a very natural generalization of the standard solution to nearest neighbor.

It is also much easier to think about. In fact, the seeming simplicity of the problem is probably a big part of the reason so many attempts at it have been made, aside from its well-observed appearance in so many contexts. Another reason is the fact that it still has no entirely satisfactory solution. Our approach will be based on the standard solution used in practice, which though old and simple, still has not been convincingly bested by any comers. We'll discuss the reasons for this later.

The problems of *range searching* and *range counting* are the analogous one-query versions of all-range-search and all-range-count. After we go into depth on the nearest neighbor problems we will return to these problems, and we'll see that we can treat them with essentially the same approach.

§2.2 Divide-and-conquer.

Divide-and-conquer is a very general algorithmic strategy, rather than a particular set of algorithm instances. Though informal use (or misuse) of the term is common and quite variable outside of computer science, here we mean something that is actually somewhat specific. The usual form of a divide-and-conquer algorithm is something like this:

```
DivConq(R)
  DivConqWork(R).
  if base-case(R), DivConqBase(R), return.
  else,
    {R.right, R.left} = DivConqSplit(R).
    DivConq(R.right).
    DivConq(R.left).
    DivConqMerge(R.left, R.right).
```

Figure 2.1: ABSTRACT DIVIDE-AND-CONQUER ALGORITHM. Note that 'left' and 'right' don't necessarily have any ordered meaning.

This represents the 'divide-conquer-merge' type of algorithm. We can define such an algorithm by the 4-tuple $A = \{$ Split(), Merge(), Work(), Base() $\}$, whose components are the sub-algorithms fleshing out the divide-and-conquer template. Other variants exist, but we shall not discuss them here.

This strategy is successful whenever there is significant *decoupling* between the parts of the problem, such that the cost of solving the two halves of a problem plus the cost of the division itself (*i.e.* the splitting required to obtain the two halves plus the merging required to obtain the overall solution using the two solutions for the halves) is less than the cost of solving the entire problem in a direct fashion.

2.2.1 Search techniques.

A special class of divide-and-conquer intersects with the notion of *search*, which has some roots in the field of AI as well as the central study of algorithms and data structures in computer science.

Pre-factoring strategy.

The perspective of search usually assumes that the 'split' part of the method is implicit, presuming the existing of a search tree which is either given naturally by the problem or is a data structure which can be constructed offline, *i.e.* as a preprocessing step. This will be our approach in this thesis. Using a pre-built tree as a substrate for divide-and-conquer is tantamount to *pre-factoring part of the computational cost out*, so that it can be reused for any further computations with the same dataset. It will in fact turn out for many of our problems that this pre-factored part of the overall cost dominates the rest. The flexibility we lose is the ability to make splitting decisions on-the-fly, for the specific case of query set and reference set, or possibly other parameters of the problem, that we are dealing with. While potentially important, this strategy of performing part of the divide-and-conquer computation *a priori*, *i.e.* before observing the problem instance, nonetheless proves to be highly robust and successful in practice.

Prioritized search.

Note that we can generalize the basic divide-and-conquer schema slightly by using a *priority function* to choose between the two branches, rather than always taking the left or the right first, *i.e.* *depth-first* traversal, which was implicit in our abstract divide-and-conquer template. All the simple tree-traversal policies, including breadth-first traversal and in-order traversal, can be generalized using the idea of a

priority queue, a data structure which allows efficient selection of the currently-best search branch according to an arbitrary priority function.

We will make the distinction between *global* priority search, in which a priority queue is necessary (along with its associated costs) and *local* priority search, in which the better of the two search branches is determined by a priority function at every branch point only locally.

All of our algorithms can be written either in a priority queue form or a 'pure recursive' form. The basic tradeoff here is one of slightly increased efficiency (the global priority queue effectively allows more knowledge to be considered when choosing the next node(s) to expand) versus space efficiency (the priority queue can grow quite large depending on the specific recursion strategy).

§2.3 Adaptive space-partitioning trees.

Space-partitioning trees form a large class of geometric tree data structure. It is large mainly because many variants (mostly insignificant) of the following main types of structures have been proposed. Our approach actually works with any of the forms of space-partitioning trees that have been proposed for nearest neighbor problems. We emphasize that we mean *adaptive space-partitioning trees* (ASPT's), since there also exist fixed varieties, often referred to by the same names, which are not as powerful.

2.3.1 kd-trees.

kd-trees [Ben75, FBF77] are simple yet effective space-partitioning trees, where each node in the tree defines a distinct region in the data space using a bounding hyperrectangle (which is maximally tight in each coordinate) and each split is made along a single coordinate, the one having largest variance [FBF77]. The tree is often built all the way down to some small predefined number of points ρ at the leaves.

Adaptivity.

Such a hierarchy of nested hyperrectangles can be seen as a generalization of the simple Cartesian grid. But besides generalizing the flat grid to higher levels of resolution, the variance-minimizing property of *kd*-trees effectively places hyperrectangles in a manner which is sensitive to the shape of the density, in contrast to the data-blind nature of simple grids, which simply divide up space into fixed-size hypercubes.

This adaptation to the shape of the data is the key property which separates these kinds of structures from many others which have been and continue to be proposed, in two ways. The first is that they can be effective beyond low dimensions, and the second is that theoretical analyses of algorithms based on such structures are derailed by the fact that the statistics of the input becomes critical, requiring analytical tools beyond that of the usual discrete arguments.

Cached sufficient statistics.

We use an extension called *multi-resolution* kd-*trees*, or mrkd-*trees* [DM95] which also contain local sufficient statistics of the data such as the mean and covariance within each node.

The choice of which statistics or quantities to store is determined by the problem.

Distance bounds.

Importantly, bounds on the distance of a point $\underline{x}_q$ to any point $\underline{x}_r$ in region R can be computed in $O(D)$ time:

$$\min_r \|\underline{x}_q - \underline{x}_r\|^2 \geq \sum_{d=1}^{D} \Big(\max\{(\underline{l}_R^d - \underline{x}_q^d)^2, 0\} + \max\{(\underline{x}_q^d - \underline{u}_R^d)^2, 0\} \Big) \quad (2.2)$$

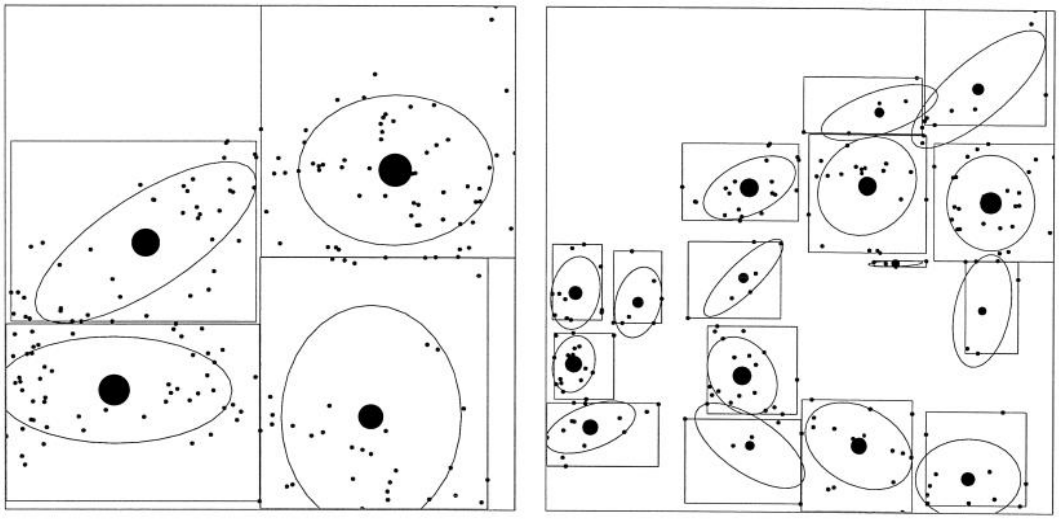

Figure 2.2: LEVELS OF AN *MRKD*-TREE. The second and fourth levels are shown, respectively. The dots are the individual data points. The sizes and positions of the black disks show the node counts and centroids. The ellipses and rectangles show the covariances and bounding boxes.

$$\max_r \|\underline{x}_q - \underline{x}_r\|^2 \quad \leq \quad \sum_{d=1}^{D} \max\left\{(\underline{u}_R^d - \underline{x}_q^d)^2, (\underline{x}_q^d - \underline{l}_R^d)^2\right\}$$

where $\underline{l}_R$ and $\underline{u}_R$ are the lower and upper D-dimensional corner points defining R. This effectively yields a hierarchy of cheap bounds, which increase in tightness as we descend to the leaves.

The analogous bounds can be written for the closest and farthest distances between any two points of a query *node* and a reference node.

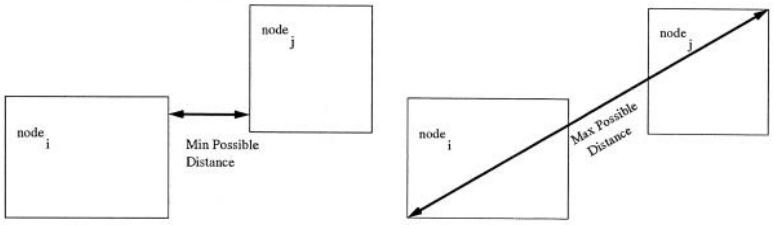

Figure 2.3: DISTANCE BOUNDS. The lower and upper bound on pairwise distances between the points contained in each of two *kd*-tree nodes.

Complexity.

The space cost of a *kd*-tree is $O(DN)$ (there are $2N$ nodes in a full balanced tree). The time to build to a *kd*-tree is $O(DN \log N)$ (at each node we must pay a cost proportional to the size of the node's data subset). We'll visit the *run-time* complexity of typical algorithms which use *kd*-trees shortly.

2.3.2 Ball-trees.

We can in fact improve the data-adaptivity of *kd*-trees by removing their axis-parallel restriction. If we instead partition by selecting centroid *points* to define the left and right sets of the partitioning (according to the Voronoi diagram induced by the two points), we can escape this representation problem as well, effectively replacing

axis-parallel hyperplane separators with arbitrarily-aligned hyperplanes. The result is called a *ball-tree* [Omo91] or 'metric tree' [Uhl91, CNBYM01]. As with *kd*-trees we augment the structure with problem-dependent cached sufficient statistics.

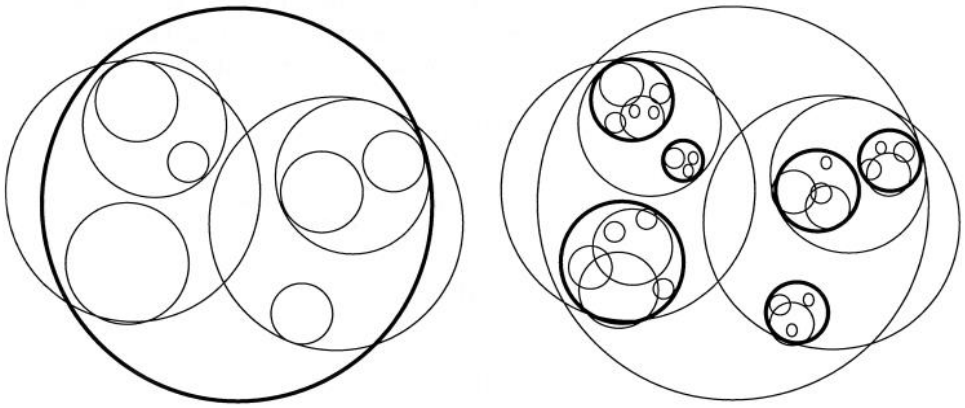

Figure 2.4: LEVELS OF A BALL-TREE. The bold circles indicate the root (left figure) and the 'middle' layer of nodes occurring in the anchors construction algorithm (right figure).

Distance bounds.

We again have $O(D)$ bounds due to the triangle inequality:

$$\begin{aligned} \min_r \|\underline{x}_q - \underline{x}_r\|^2 &\geq \max\left\{(\|\underline{x}_q - \underline{c}_R\| - s_R)^2, 0\right\} \\ \max_r \|\underline{x}_q - \underline{x}_r\|^2 &\leq (\|\underline{x}_q - \underline{c}_R\| + s_R)^2 \end{aligned} \tag{2.3}$$

where $\underline{c}_R$ is the centroid of R and s_R is its radius, the distance of the farthest point in R to $\underline{c}_R$.

Again, analogous bounds can be written for the closest and farthest distances between two nodes.

Complexity.

Ball-trees can be built efficiently and with high quality using the *anchors hierarchy* algorithm [Moo00]. Ball-trees built using that method have been demonstrated to be effective in up to thousands of dimensions in some cases. The build cost was constrained to be $O(DN^{3/2})$. The space cost is also the same as a *kd*-tree, $O(DN)$.

2.3.3 Basic divide-and-conquer algorithms.

Nearest neighbor algorithm.

The form that the divide-and-conquer principle takes for the nearest-neighbor problem is quite simple. First, the fact that we have already built a tree before runtime means that the 'split' part of the algorithm has already been taken care of in preprocessing. The 'merge' part of the algorithm is trivial is $O(1)$ since it amounts to updating a single scalar quantity. In fact this specific form of divide-and-conquer (where merging is really just updating a bound) is sometimes called 'branch-and-bound' in older literatures and in the operations research community.

The basic idea is to compare the query point to nodes in the tree, starting at the root. At all times we maintain the distance δ_q^{NN} of the query to its nearest neighbor found so far. At the start of the search this is set to ∞. If the lower bound on the distance from the query $\underline{x}_q$ to any point in the node R, $\delta_{qR} = \|\underline{x}_q - R\|^{lo}$, is greater than δ_q^{NN}, we know that no point in R could be $\underline{x}_q$'s nearest neighbor, so need not

recurse on the children of R. Otherwise we recurse, taking the more promising of its children first according to a local priority function — in this case we use the lower bound on the distance to the node. Since leaves can contain up to ρ points, there is a special base case for the leaves, in which we exhaustively compute the distance of each to the query point, updating the nearest neighbor guess and its distance as needed. (Note that no special base case is needed if $\rho = 1$.)

NN(q,R)
  $\delta_{qR}^{lo} = \|\underline{x}_q - R\|^{lo}$.
  `if` $\delta_{qR}^{lo} \geq \delta_q^{NN}$, `return`.
  `if` leaf(R), **NNBase**(q,R).
  `else`,
    NN$(q$,closer-of$(q,\{R.\text{left},R.\text{right}\})$.
    NN$(q$,farther-of$(q,\{R.\text{left},R.\text{right}\})$.

NNBase(q,R)
  `foreach` $\underline{x}_r \in R$,
    $\delta_{qr} = \|\underline{x}_q - \underline{x}_r\|$.
    `if` $\delta_{qr} < \delta_q^{NN}$,
      $\delta_q^{NN} = \delta_{qr}$, $NN_q = r$.

RS(q,R)
  $\delta_{qR}^{lo} = \|\underline{x}_q - R\|^{lo}$, $\delta_{qR}^{hi} = \|\underline{x}_q - R\|^{hi}$.
  `if` $\delta_{qR}^{lo} > h$, `return`.
  `if` $\delta_{qR}^{hi} \leq h$,
    add-to-set$(h{-}Set_q, R)$, `return`.
  `if` leaf(R), **RSBase**(q,R).
  `else`,
    RS$(q$,closer-of$(q,\{R.\text{left},R.\text{right}\})$.
    RS$(q$,farther-of$(q,\{R.\text{left},R.\text{right}\})$.

RSBase(q,R)
  `foreach` $\underline{x}_r \in R$,
    $\delta_{qr} = \|\underline{x}_q - \underline{x}_r\|$.
    `if` $\delta_{qr} \leq h$,
      add-to-set$(h{-}Set_q, r)$.

Figure 2.5: BASIC NEAREST-NEIGHBOR AND RANGE SEARCH ALGORITHMS. In the pseudocode a += b means a = a + b, and a = b denotes assignment while a == b denotes equality checking. Note that these algorithms can be written in a slightly different way such that distance computations are never repeated, but this presentation is more conceptually transparent for our purposes. closer-of(A,B) returns the closer of its two node arguments A and B; farther-of() is similar. add-to-set(A,B) implements $A = A \cup B$ and could also have been written $A \cup = B$.

Range search/count algorithm.

The algorithm for range search is completely analogous. The one interesting notable difference is that the inverse of the node pruning operation that was used in the nearest-neighbor algorithm is also used — if the *upper* bound $\delta_{qR}^{hi} \leq h$, we know that all the points in R belong to $h{-}Set_q$. With the appropriate data structures we can simply note that all of R should be added to the set, or alternatively we can explicitly add each point at the current moment in the search.

The algorithm for range count simply updates an integer count c_q instead of maintaining the set of indices $h{-}Set_q$.

Single-tree algorithms.

What about all-nearest-neighbor and all-range-search/count? These can clearly be solved using the previous solutions, as shown. The reason we refer to these as *single-tree* algorithms will become clear shortly.

AllNN$(\mathcal{Q},R)$
  `foreach` $q \in \mathcal{Q}$, $NN_q =$ **NN**(q,R).

AllRS$(\mathcal{Q},R)$
  `foreach` $q \in \mathcal{Q}$, $h{-}Set_q =$ **RS**(q,R).

Figure 2.6: SINGLE-TREE ALGORITHMS FOR ALL-NEAREST-NEIGHBOR AND ALL-RANGE-SEARCH.

§2.4 Higher-order divide-and-conquer.

We can generalize the standard notion of divide-and-conquer on a set to divide-and-conquer on *multiple sets*, which we call *higher-order* divide-and-conquer (HODC). For example, in Figure 2.7 a basic template for the $n = 2$ case is shown.

```
HODC(Q,R)
  HODCWork(Q,R).
  if base-case(Q,R), HODCBase(Q,R), return.
  else,
    {Q.right,Q.left} = HODCSplit(Q).
    {R.right,R.left} = HODCSplit(R).
    HODC(Q.left,R.left).
    HODC(Q.left,R.right).
    HODC(Q.right,R.left).
    HODC(Q.right,R.right).
    HODCMerge(Q.left,Q.right).
    HODCMerge(R.left,R.right).
```

Figure 2.7: ABSTRACT HIGHER-ORDER DIVIDE-AND-CONQUER ALGORITHM.

Note that Q and R can be the same set. The analogous schemas hold for any number of additional sets S, T, U, ... We could define a divide-and-conquer algorithm of this basic form by the tuple $\{ \forall\nu \text{ Split}_\nu(), \text{Merge}_\nu(), \text{Work}_\nu(), \text{Base}_\nu() \}$.

Note the unusual four-way recursive step. This ensures that each child of the first node is compared with each child of the second node. Only a simple depth-first search is shown rather than, say, local priority search.

This implicitly assumes that the trees are full and have the same topology. More generally the complete set of recursion cases must account for the cases in which one node is a leaf while the other is not. This is shown in Figure 2.8, which is also specialized to the pre-built-tree case that will be the situation in all of this thesis. No merge step is shown because it will be implicit in bounds updates as we have seen earlier.

```
HODC(Q,R)
  HODCWork(Q,R).
  if leaf(Q) and leaf(R), HODCBase(Q,R), return.
  if leaf(Q) and !leaf(R),
    HODC(Q,R.left).
    HODC(Q,R.right). return.
  if !leaf(Q) and leaf(R),
    HODC(Q.left,R).
    HODC(Q.right,R). return.
  if !leaf(Q) and !leaf(R),
    HODC(Q.left,R.left).
    HODC(Q.left,R.right).
    HODC(Q.right,R.left).
    HODC(Q.right,R.right). return.
```

Figure 2.8: ABSTRACT HODC ALGORITHM, GENERAL CASES. $!A$, where A is a boolean expression, means 'not A', or A evaluates to false.

Because this is slightly cumbersome to write, we will use a slightly compressed notation which assumes that for a leaf node A, A.left $= A$ and A.right $= A$, and the exact same recursive call is not made twice. This allows us to write the same algorithm equivalently as shown in Figure 2.9.

```
HODC(Q,R)
  HODCWork(Q,R).
  if leaf(Q) and leaf(R), HODCBase(Q,R), return.
  else,
    HODC(Q.left,R.left).
    HODC(Q.left,R.right).
    HODC(Q.right,R.left).
    HODC(Q.right,R.right).
```

Figure 2.9: ABSTRACT HODC ALGORITHM, COMPRESSED NOTATION.

Shattering.

Shattering is really just a shorter synonym for 'higher-order divide-and-conquer'. *Geometric shattering* is really just a shorter synonym for 'higher-order divide-and-conquer on space-partitioning trees'. The intuition is that if we are really dealing with *multiple* sets, it is wasteful not to consider breaking up *all* of the sets. It works whenever additional dividing will reveal more pieces that are easily conquered. The idea of breaking *everything* into little pieces (of various sizes) then putting them back together with minimum effort is the intuition behind the name 'shattering'.

2.4.1 Search techniques.

Recursion pattern.

The move to higher-order divide-and-conquer opens up a new type of choice, the recursion pattern. By this we mean that the basic four-way recursion shown in 2.7 can also be replaced by a pattern in which, say, only one of Q and R is split, to form two recursive calls rather than four. If say, the larger (greater number of data) of them is always chosen, the selection of Q versus R will roughly alternate.

All of our algorithms can be equivalently written with alternative recursion patterns (in fact we'll use the alternative just described in Chapter 5. This choice does not seem to affect performance, though we did not fully pursue this possibility experimentally.

All-nearest-neighbors algorithm.

Understanding the shattering solution to the all-nearest-neighbors problem should be trivial now. The idea is to now build a second tree for the query set (it is the same tree if the query set is equal to the reference set), and generalize the algorithm from one which performs point-node comparisons to one which performs node-node comparisons.

The last few lines of the base case for the all-nearest-neighbors algorithm maintain the bound δ_Q^{NN} for node Q, ensuring that it is the largest of all the nearest-neighbor bounds δ_q^{NN} for the data in Q. The bound is updated for non-leaves in the line following the recursive calls. The looser of the childrens' bounds replaces the node's current bound if it improves it.

AllNN(Q,R)
$\delta_{QR} = \|Q - R\|^{lo}$.
`if` $\delta_{QR} \geq \delta_Q^{NN}$, `return`.
`if` leaf(Q) and leaf(R), **AllNNBase**(Q,R).
`else`,
AllNN$(Q$.left,closer-of$(Q$.left,$\{R$.left,R.right$\})$.
AllNN$(Q$.left,farther-of$(Q$.left,$\{R$.left,R.right$\})$.
AllNN$(Q$.right,closer-of$(Q$.right,$\{R$.left,R.right$\})$.
AllNN$(Q$.right,farther-of$(Q$.right,$\{R$.left,R.right$\})$.
$\delta_Q^{NN} = \min(\delta_Q^{NN}, \max(\delta_{Q.\text{left}}^{NN}, \delta_{Q.\text{right}}^{NN}))$.

AllNNBase(Q,R)
`foreach` $\underline{x}_q \in Q$,
`foreach` $\underline{x}_r \in R$,
$\delta_{qr} = \|\underline{x}_q - \underline{x}_r\|$.
`if` $\delta_{qr} < \delta_q^{NN}$,
$\delta_q^{NN} = \delta_{qr}$, $NN_q = r$.
`if` $\delta_{qr} < \delta_Q^{NN}$, $\delta_Q^{NN} = \delta_{qr}$.

AllRS(Q,R)
$\delta_{QR}^{lo} = \|Q - R\|^{lo}$, $\delta_{QR}^{hi} = \|Q - R\|^{hi}$.
`if` $\delta_{QR}^{lo} > h$, `return`.
`if` $\delta_{QR}^{hi} \leq h$,
add-to-set$(h\text{-}Set_Q, R)$, `return`.
`if` leaf(Q) and leaf(R), **AllRSBase**(Q,R).
`else`,
AllRS$(Q$.left,closer-of$(Q$.left,$\{R$.left,R.right$\})$.
AllRS$(Q$.left,farther-of$(Q$.left,$\{R$.left,R.right$\})$.
AllRS$(Q$.right,closer-of$(Q$.right,$\{R$.left,R.right$\})$.
AllRS$(Q$.right,farther-of$(Q$.right,$\{R$.left,R.right$\})$.

AllRSBase(Q,R)
`foreach` $\underline{x}_q \in Q$,
`foreach` $\underline{x}_r \in R$,
$\delta_{qr} = \|\underline{x}_q - \underline{x}_r\|$.
`if` $\delta_{qr} \leq h$,
add-to-set$(h\text{-}Set_q, r)$.

Figure 2.10: ALL-NEAREST-NEIGHBORS AND ALL-RANGE-SEARCH/COUNT ALGORITHMS. A leaf's left or right child is defined to be itself. In the actual code repeated recursion cases are prevented.

All-range-search/count algorithm.

The analogous algorithm for the range-searching/counting problem is trivially seen. The corresponding algorithm for the closest-pair problem is also a trivial modification of the all-nearest-neighbors algorithm.

§2.5 Complexity.

The expected time complexity of the basic space-partitioning tree approach for finding the nearest neighbor in D dimensions was shown by [FBF77] to be asymptotically $O(\log N)$ for the case of *kd*-trees, though the argument is easily extended to ball trees. Note that this holds for any input distribution, assuming only that the query and reference data come from the same distribution.

We now show how the analysis of [FBF77] can be extended to the single-tree and dual-tree algorithms for the all-nearest-neighbors problem.

Theorem 1 (All-nearest-neighbors runtime) *The asymptotic expected runtime is $O(N \log N)$ for the single-tree algorithm and $O(N)$ for the dual-tree algorithm.*

Proof: We'll start with the single-tree case, which is easier to understand. Because of the minimum-distance priority function used in the search, the first descent down the tree simply locates the leaf node containing the query point $\underline{x}_q$ itself (or the closest reference point in its vicinity) in expected $O(\log N)$ time, following from the recurrence of binary search:

$$T(N) = T(N/2) + O(1); \;\; T(1) = O(1) \tag{2.4}$$

To see where this comes from, consider the more explicit recurrence

$$T(N) = T(N/2) + \alpha_N T(N/2) + O(1); \;\; T(1) = O(1) \tag{2.5}$$

which represents the fact that after descending down the correct branch, *i.e.* the one containing the true nearest neighbor, there remains a probability which may depend on N, call it α_N, that the heuristic used to select the descent branch (the lower bound on the distance, obtained from the boundary of each node) selected the wrong branch. It is after all only a lower bound, with some unknown slack with respect to the true minimum distance. In *expectation*, α_N is zero, until we reach the depth in the tree at which we must perform a minimal amount of backtracking to ensure that we have the true nearest neighbor. This is why the analysis only holds in the expected sense.

The amount of backtracking we must do corresponds to the expected number of nodes whose boundaries will intersect the nearest-neighbor ball $NNball(\underline{x}_q) = \{\underline{x} \mid ||\underline{x} - \underline{x}_q|| < \delta_q^{NN}\}$. Let $f(D)$ denote the factor relating the volume of $NNball(\underline{x}_q)$ to the volume of the smallest region containing $NNball(\underline{x}_q)$ which is representable by the tree's node shape. This is 1 in the case of ball-trees, larger than 1 for *kd*-trees. If the number of points in leaves is allowed to be ρ, from [FBF77], the expected number of points C examined is bounded by $(1 + f(D)^{1/D})^D \rho$, which is independent of N and the probability distribution of the data.

This is equivalent to writing $T(C) = O(1)$ for the second part of the recurrence. The solution is thus $O(N \log N)$, for N repeated calls costing $O(\log N)$ each.

The same argument holds for the dual-tree case, replacing the previous recurrence with the one appropriate for dual-tree search. For each query child, there are two possible branching choices, leading to the recurrence

$$\begin{aligned} T(N) &= T(N/4) + \alpha_N T(N/4) + T(N/4) + \alpha_N T(N/4) + O(1); \\ T(1) &= O(1) \end{aligned} \tag{2.6}$$

but by the same expectation argument this is equivalent to

$$T(N) = 2T(N/4) + O(1); \; T(1) = O(1) \tag{2.7}$$

whose solution is $O(N)$. ■

Note that this implies $O(1)$ complexity per query point in the dual-tree case. Both the single-tree and dual-tree algorithms have $O(N)$ memory cost, due to the trees themselves as well as the cost of storing all N answers during execution.

2.5.1 Dependence on dimensionality.

Work on the difficult problem of characterizing the runtime of these data-adaptive structures has very recently re-awakened [Boh00, CPZ98]. One thrust of these efforts, though it has not been formalized satisfactorily, is the growing agreement that these structures are not truly exponential in the explicit dimension D, but rather in some 'intrinsic dimension' D'. This is in fact borne out by experiment and the discrepancy with the worst-case has long been observed [Spr91]. Faloutsos and co-workers [FK94] characterized the expected number of page accesses for range queries in R-trees in terms of the box-counting fractal dimension and the correlation dimension of Procaccia and Grassberger [GP83], and nearest-neighbor queries have been similarly characterized [PM97]. Recently in computational geometry similar lines of reasoning have been hinted at in some works, *e.g.* [EPY00], and notably made explicit in [MM01].

Even so, the ability of adaptive space-partitioning trees to adapt to a lower-dimensional manifold within the data is not perfect. The tradeoff between such a structure's statistical modeling power and its cost of construction ('learning') is a significant factor affecting this. The entire perspective of ASPT's as *statistical models* of the data in themselves has not been pursued and is something we regard as a fruitful area for future research. The upshot is that while they remain the best available option for exact computation in high-dimensional spaces, their performance is often still suboptimally sensitive to the dimensionality in practice.

§2.6 Performance.

For any space-partitioning scheme, the dual-tree algorithm for the all-nearest-neighbor problem will be more efficient than the standard single-tree algorithm. We know this is true asymptotically. Though we are not particularly interested in all-nearest-neighbor as a practical application, here we show a quick empirical demonstration of the complexity superiority of dual-tree versus single-tree.

The dataset is a 2-dimensional set of point-spread function values for stars and galaxies. The experiment was performed on a 500 MHz Pentium workstation. All runtimes reported in this thesis are given in seconds.

N	Tree Build Time	Single Tree Time	Dual Tree Time
100k	4	11	1
200k	8	26	3
500k	24	78	7
1M	50	182	15

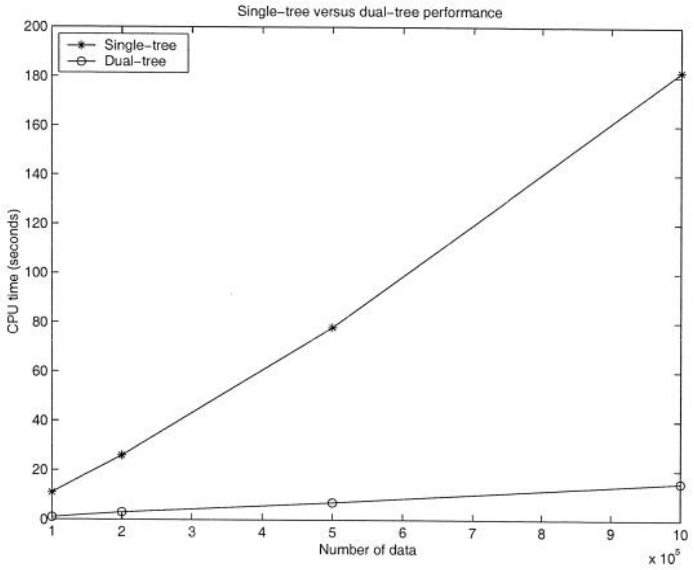

Figure 2.11: SINGLE-TREE VERSUS DUAL-TREE PERFORMANCE.

§2.7 Related problems and approaches.

It is of interest to compare these new algorithms at a theoretical level to other existing schemes, mainly to see if there is something else we can leverage for our range of N-body problems.

2.7.1 Nearest-neighbor problems.

Although a stunning multitude of approaches to the nearest-neighbor problem have been proposed, it is safe to say that none of these has particularly distinguished itself from the crowd by convincingly demonstrating empirical superiority over the approach based on space-partitioning trees which we have covered. Wide-spanning surveys describing a large number of these attempts can be found in [GG98, CNBYM01].

Despite the impressive diversity of approaches to nearest-neighbor which have been explored, the practical state of the art can be safely said to still revolve around forms of *kd*-trees and ball-trees. Consider for example the most recent paper of Clarkson, one of the most well-known nearest-neighbor researchers from the theory community, whose goal is to finally identify a method which works robustly in arbitrary dimensions for realistic data in practice [Cla02] – the proposed method is a standard variant of ball-tree.

Other space-partitioning trees.

Most of the proposals are based on variants of *kd*-trees, including *R-trees*, their secondary-storage analogs, to one extent or another. We scoured the literature in the hopes of finding interesting variants which could provide efficiency benefits,

but found very few which convincingly offered more than the two basic types of space-partitioning trees we've discussed.

It should be noted that many of the proposed approaches originated in the database community, and thus assume some different constraints on the nearest-neighbor problem, as noted earlier. This entire class of approaches is outside the scope of our concern.

One of the best hopes in our opinion lies in methods for capturing the distributional or statistical structure of the data as well as possible while still maintaining efficiency in searching and building, as well as space efficiency. The method we have used [Moo00] represents the best we are aware of, however further research in this vein may provide payoff. Examples of other methods for ball-tree construction occur in [Omo89].

A proposal by Sproull [Spr91] has received surprisingly little attention (perhaps due to the unassuming title of the paper). It introduces the notion of using the principal axis (or first principal component) of the data to create non-axis-parallel splits. This approach seems not to have been explored further. The relationship of this statistical approach to selecting arbitrary decision hyperplanes versus the statistical approach implicit in ball-tree construction represents an interesting open question.

Grid-based methods and hashing.

A simplistic attempt at a hashing scheme is represented by the *VA-file* [WB97]. However since no locality-preserving property or mechanism was shown, the method boils down to $O(N)$ linear scan for each nearest-neighbor query, placing emphasis on reducing the constant term via low-level programming techniques.

A different grid-based approach [**?**] which constructs a grid structure then hashes new points to the appropriate hypercube in the grid purports expected *constant-time* complexity in arbitrary dimension for the nearest-neighbor problem (which of course implies $O(N)$ complexity for several N-body proximity problems, including closest-pair). However, this complexity assumes a class of 'bounded' distributions which may or may not include realistic data distributions (even the normal distribution is not included in this class, for example). It also assumes the constant-time floor function, an issue of some debate [FH91] (see also 2.7.1 below), though we do not take any particular stance on this issue. The most impractical aspect of this approach is the $O(N^D)$ size of the grid structure.

At the end of the paper the authors discuss the possiblities of extending this approach to more realistic uneven distributions by using adaptive grids. Note that this unrealized idea would lead in the direction of space-partitioning trees. In fact the distinction between tree search for finding leaf nodes and hashing for finding grid buckets lies only in whether the finding operation is constant-time or not. The fuzziness of the floor function's status in this regard thus blurs the distinction between 'hashing' and a simple non-adaptive space-partitioning tree. Indeed, similar discussions in [WB97] consider ways to prioritize the order in which grid cells are searched, leading dangerously close to the idea of hierarchy.

The I-grid index [AY00] is a grid-based method purporting to *reverse* the curse of dimensionality, but in order to achieve this a new distance measure is proposed in place of the Euclidean metric. Therefore it does not treat the same problem.

Hashing methods taking a randomized approach are discussed below.

Graph separation.

An interesting approach based on the graph-theoretic perspective of distances as edge weights [DL76, LT77] represents perhaps the most serious assault on the problem by the computer science theory community in this classical direction. The 'planar

separator' achieves worst-case $O(\log N)$ run time and $O(N)$ space, but only in 2 dimensions as implied. Unfortunately the preprocessing and storage requirements have been noted by several authors to be prohibitive for practical use, including the creators of the method themselves.

Reductions from other problems.

Some N-body problems in computational geometry are reducible from the another problem which is sometimes more general, so that a fast solution to that problem yields a fast solution to the proximity problem of interest.

The approach of unifying many proximity problems as reductions from the Voronoi diagram [SH75] we discussed earlier has $O(\lceil D/2 \rceil ! N^{\lceil D/2 \rceil})$ complexity.

The diameter problem is an example of an N-body problem which is quickly solvable given the convex hull. [BS78] gives a linear-expected-time deterministic divide-and-conquer approach to finding convex hulls in 3 dimensions or less. [PH77] is another example of the convex hull approach.

Formulating nearest-neighbor as a special case of the 'ray-shooting' problem [AE97] yields a solution which, if allowed $O(M)$ space, requires $O(N/M^{1/\lceil D/2 \rceil})$ time, again exponential in D.

Randomized algorithms.

Randomized algorithms represent an algorithmic strategy using randomness in a way which is distinct from the *Monte Carlo method* for integration, which we will use in Chapter 5. This type of algorithm is arguably the chief focus of the computer science theory community, and is a typical approach to achieving so-called 'approximation algorithms'. The question of whether randomness in algorithms represents a kind of algorithmic power which cannot be achieved by deterministic algorithms is an open one, though at the moment there are few instances of problems having randomized solutions without known deterministic analogs.

Unfortunately, the typical theoretical scenario considered is one in which not only is the answer approximated, but the error bound is not strict — instead it is a *confidence bound, i.e.* the bound holds with a certain probability. While this can be tolerated in real applications, some constructions of randomized algorithms hide undesirable properties. An example is the the inability to retry a computation because the randomness does not occur not at runtime. This occurs in algorithms which use randomness at preprocessing time rather than runtime. In these types of algorithms the user cannot simple run the algorithm several times for the same input to mitigate the effect of random error.

A very influential paper by Rabin [Rab76] which helped to introduce the randomized algorithm strategy using two examples, one of which presents a randomized solution to the closest-pair problem with linear-expected-time complexity in arbitrary dimension. However, this complexity advance was shown by [FH91] to be partially due to the assumption of a more powerful model of computation than previously assumed, by assuming the floor function as a constant-time operation. Regardless, it is not obvious how to generalize the method to other proximity problems. [Wei78] gave a different randomized algorithm for the closest-pair problem.

Clarkson gives a randomized algorithm in [Cla88] which answers a nearest neighbor query in time $O(2^D \log N)$, improved to $O(D^3 \log N)$ by [Mei93], unfortunately using a structure of size $O(N^{\lceil D/2 \rceil + \epsilon})$.

Other papers by Clarkson discuss randomized approaches in computational geometry [Cla87, CS89] in a general context.

2.7.2 Approximate nearest neighbors.

The idea of replacing the nearest-neighbor problem with that of the $(1+\epsilon)$-*approximate nearest-neighbor* problem deserves special attention because so much implication has

surrounded it as a long-awaited solution to the curse of dimensionality.

One of the only widely-known facts about high-dimensional distances is that as the dimension grows, the variance of distances shrinks [Ham50]. In other words, the size of the distances approach the same value. This fact is commonly misinterpreted to mean that distances are meaningless in high dimensions. This is fallacious since the problem is that of finding the *nearest* neighbor, *i.e.* the information is in the relative *ranks* of the distances, not their numerical values.

This fact has also been exploited to grandly imply that the curse of dimensionality can be solved by allowing approximation in the nearest-neighbor problem — surely there is no harm in accepting neighbors whose distance is a factor of $1+\epsilon$ times that of the true nearest neighbor. In fact, this is also fallacious. In a high enough dimensionality, *all* the distances might be within $1+\epsilon$ of each other. This does not change the fact that their ranks contain information. Thus, while it is true that the hardness of the nearest-neighbor problem, which is in the ranks, can be removed by allowing an 'approximation' which is insensitive to ranks, this approach simply skirts the actual problem.

A meaningful formulation of an approximation notion for the nearest-neighbor problem must be in terms of their ranks. So far this has not appeared in the literature. Thus, we do not consider the notion problem to have yet been formulated properly.

Randomized and deterministic algorithms.

The most recent approximate nearest-neighbor approach is 'locality-sensitive hashing' [GIM99], a hashing scheme based on the hope of operationalizing the oft-quoted and re-proved Johnson-Lindenstrauss Lemma, which says roughly that points which are nearby in a high-dimensional space are nearby in a projection of the points onto a random subset of the dimensions, with high probability. A query point is hashed to a bucket in each of a large number M of hash tables, each corresponding to a different random subset of the dimensions. The hope is that points hashing to the same buckets are proximal with high probability. Thus at runtime the algorithm consists of computing the distance of the query to all of the points in each of the M hash buckets, and returning the closest one among them. This querying time is $O(DN^{1/(1+\epsilon)})$, while the time to build the preprocessing structure is $O(N^{1/(1+\epsilon)+1}+DN)$ For realistic values of ϵ having some hope of preserving distance ranks in arbitrary dimension, these complexities are not encouraging. Unfortunately the empirical evaluation of the method performed by the authors, though laudable as a rare undertaking by theorists, is unconvincing. The method was empirically compared to the SR-tree, an R-tree variant from the database community which performs online tree structuring — which has been shown to degrade performance considerably in comparison to batch construction of trees. The SR-tree also performs exact querying, and was not modified to perform $1+\epsilon$ approximate querying. The method requires a large number of tweak parameters (α,l,and k) which must be chosen without theoretical guidance — the influence of ϵ is through these parameters rather than being specifiable directly by the user. This also means that ϵ is not a parameter which can be selected at query time. Because all the randomness occurs at preprocessing time rather than at query time, a query for which the approximation bound has failed cannot be retried — that query simply can never be answered correctly by the algorithm. Finally, because the approach is inextricably based on comparison of bit-strings, the metric must be the L_1 norm, and the Euclidean distance cannot be used.

It should also be noted that other hashing approaches exist which are in the same vein as locality-sensitive hashing, *i.e.* attempt to use locality-preserving embeddings

(or subspaces), often many together. However they are deterministic and potentially more practical, *e.g.* [SZM99].

Other randomized approaches to finding approximate nearest neighbors include [Cla88, Kle97].

Note that approximate nearest-neighbor finding represents a trivial change to the space-partitioning tree approach, though this was apparently not realized until [AM93, AMN+98]. The authors of the randomized approaches do not compare to this visible-enough approach for some reason, which has the advantage of yielding deterministic hard bounds, 100% of the time. Its preprocessing structure is also independent of the approximaton setting ϵ.

2.7.3 Range-searching/counting problems.

The vast majority of the work on range-searching is on other variants of the problem we are concerned with, which is *spherical* range-searching. Virtually all of the large body of work on range-searching treats orthogonal range queries, triangular range queries, simplex range queries, and half-space range queries. We must also be wary of certain other distractions such as 'fixed-radius range queries' which are spherical but assume that the radius of the query is known at the time the preprocessing structure is built, *e.g.* [CE85]. The main reason focus has shifted to these other problems is that are easier than the spherical range search problem [Yao82].

Approximate range searching.

The idea of $1+\epsilon$ approximation in range searching is more sensible than in the nearest-neighbor setting, since the specification of a numerical range implies that its meaning is known to the user — its sensibility can be decoupled from the distribution of distances, unlike in the nearest-neighbor case.

The randomized approximation methods do not seem to carry over to range-searching problems. The fairly obvious approximation trick that can be used to implement an approximate nearest-neighbor algorithm using standard space-partitioning trees [AM93] can also be used in the range-searching context [AM95].

2.7.4 Divide-and-conquer.

Other divide-and-conquer strategies.

A notable part of the literature is [Ben80], in which Bentley gives a *general strategy* for divide-and-conquer and shows examples of how to apply this algorithmic design principle for several different proximity problems in arbitrary dimension D, rather similarly to our aim in this thesis. The paper even considers some fundamental statistical problems which have geometric formulations. Unfortunately the approach applies divide-and-conquer *in the dimensions*, and unavoidably results in algorithms which are exponential in the explicit dimension D.

The generalization of the space-partitioning tree approach to finding nearest neighbors as priority search was apparently not published until [AMN+94].

Higher-order divide-and-conquer.

The higher-order divide-and-conquer principle of algorithm design does not appear to have been previously formulated in a general form. However, in its geometric form it has appeared for some special cases of N-body problems (mainly all-nearest-neighbor) as we discuss next.

2.7.5 Dual-tree algorithms.

In the history of the all-nearest-neighbors problem, we see a handful of approaches by authors who have also observed the 'node-node' idea, which in our general bichromatic perspective is a 'dual-tree' notion. Outside of all-nearest-neighbor, algorithms

of the dual-tree style do not appear to have been proposed, with the arguable exception of the interesting paper by Appel [App81, App85] describing an approach that was even earlier than the much better-known [BH86] and [GR87], to be discussed in Chapter 4.

The fact that notions similar to the dual-tree idea have been independently noted by different authors in different fields testifies to its naturalness. These authors unfortunately did not realize the scope of the underlying algorithmic concept as we present it here (higher-order divide-and-conquer). They also each have certain drawbacks over the fairly elegant approach we showed.

Vaidya's algorithm.

The first is the algorithm of Vaidya [Vai89], which upon dissection can be interpreted as the version of the dual-tree algorithm we showed which performs splitting on the fly rather than as a preprocessing phase. The structure implicitly created is essentially a form of *kd*-tree, except that every dimension is split to form 2^D recursive calls. This unfortunate fact makes gives the algorithm's runtime a factor of $O(D^D)$, despite the fact that it is otherwise worst-case $O(N \log N)$ due to the implicit tree construction. Predecessors also containing node-node notions include [Cla83] and [GBT84].

Well-separated pair decomposition.

The second is the *well-separated pair decomposition* (WSPD) [Cal95], which we'll also discuss in Chapter 4. The algorithm for computing the WSPD data structure can be regarded as a form of dual-tree algorithm. The viewpoint is different though, in that the WSPD-based algorithm for solving all-nearest-neighbors then operates upon the intermediate WSPD data structure, a list of all the pairs of points satisfying the 'well-separated' condition. The runtime analysis is also different – they showed that computing the WSPD is worst-case $O(N \log N)$ and that the size of the structure is $O(N)$. Obtaining the nearest-neighbor pairs is done by examining every node-pair in the WSPD so that the runtime is $O(N)$ given construction of the WSPD. In our case the dual-tree algorithm is in the final stage, not the preprocessing stage.

It is unfortunate that even for the problems for which the idea was considered (all-nearest-neighbors and the physical N-body problem), the approach of creating algorithms through the WSPD 'data structure' is rather awkward, particularly in all-nearest-neighbors, where the idea of well-separatedness is superfluous. The need to build a structure cluttered by a large number of irrelevant node-pairs is unclear when the dual-tree procedure can simply be used to find the nearest neighbors directly.

Though we agree with the idea that an $O(N \log N)$ preprocessing can be usefully extracted from the computation for re-use, leaving an $O(N)$ computation at runtime, the WSPD does not appear to be natural for any problems other than the physical N-body problem. Even in that case, it can be eliminated by just performing the relevant computations directly as we'll see in Chapter 4. Thus the WSPD approach will suffer an additional constant factor of inefficiency versus our direct method.

Spatial join methods.

The third is the method of [HS98], which was proposed for a problem they coined the 'distance join', which returns all pairs of points in order of interpair distance, and for a problem they coined the 'distance semi-join', which appears to be exactly the all-nearest-neighbor problem. It has its roots in several earlier works in the spatial join literature employing node-node ideas including [BKS93, HJR97] – one special case of the spatial join is the all-range-search problem [Rot91] though the focus is often on intersection of polygons. The distance semi-join algorithm of [HS98] can actually be regarded as the priority-queue version of the dual-tree algorithm we showed, if the database-centric aspects of the work are ignored. The paper focuses

on the problems which arise with priority queues. Our pure-recursive alternative can be regarded as a fix eliminating the need for priority queues when they are problematic to use. They also did not provide a complexity analysis. Otherwise we regard the basic insight as the same.

§2.8 Summary of this chapter.

Let's review the main points of this chapter:

- **Basic perspective of proximity problems.** We introduced the viewpoint of computational geometry and its proximity problems. These provide a conceptual foundation for the way we'll discuss our problems (in terms of terminology such as 'bichromatic' and 'all-') and approach them (in terms of reusable data structures, in particular adaptive space-partitioning trees, which lead naturally to thinking about divide-and-conquer).
- **Higher-order divide-and-conquer.** We introduced a natural extension to the standard computer science toolbox and showed how this general algorithm design technique can be applied using four simple proximity problems as examples.
- **All-nearest-neighbor algorithm.** These examples in fact all represent novel algorithms with some advantages over existing algorithms in the literature. In particular we focused on the canonical all-nearest-neighbor problem. Mainly this was an example to show the utility of the higher-order divide-and-conquer principle.
- **Review of alternative approaches.** We also surveyed as much of the related literature as we could find in order to scour it for potential tools we could leverage.

Publications.

The following should be put forth to any community in which space-partitioning trees are used heavily in practice:

- An article demonstrating how and why the shattering principle yields efficiency over the standard use of space-partitioning trees, when going to the 'all-' version of a proximity problem such as nearest-neighbor.

What's next?

So far our kernel function has been as simple as possible - either the identity function of the distance, or a delta function implementing a simple threshold. This has allowed us to focus on just the geometric issues. In the next Chapter we approach the whole issue of arbitrary (or nearly so) kernel functions.

3 N-Body Problems in Computational Statistics

Function Approximation I: Finite-Difference Methods.

We live in succession, in division, in parts, in particles. — Ralph Waldo Emerson.

TWO WORLDS COLLIDE in many of the N-body problems we saw in Chapter 1; that is to say, these problems have both *continuous* and *discrete* properties. (This two-sidedness is in fact a theme apparent in the historical approaches to this problem as well as the organization of this thesis.) Computational statistics serves well as our first reason to consider non-trivial continuous kernel functions, since it abounds with a large variety of different kinds of kernels. This will take us beyond the purely discrete computations of computational geometry into that of continuous function approximation.

Agenda of this chapter.

We'll invoke perhaps the simplest principle for approximating a continuous function, that of *finite-differencing*. This general strategy abounds in the continous world of applied mathematics and numerical methods, and amounts to another instance of *dividing a problem into many smaller pieces*. Our goal will be a simple approximation theory which allows the rigorous guarantee of a user-specified error tolerance, with seamless operation in our hierarchical geometric context. All the while in this chapter, we will be sensitive to the nitty-gritty practicalities of our example task, **kernel density estimation**. The upshot of this is that we will actually provide in the end a *real* solution to this problem that people can use.

§3.1 Nonparametric function estimation.

Function estimation.

We refer to the estimation of continuous statistical functionals, of which density estimation and regression are the main examples we have in mind, as *function estimation*, sometimes also called 'curve estimation'.

The task of estimating a probability density from data is a fundamental one, upon which subsequent *inference*, or 'learning', or 'decision-making' procedures are often explicitly based. Statistically speaking, *all* inferences can be defined as estimating functions of the density, though in practice many inference methods for different tasks bypass direct density estimation.

Regression is such a close cousin to density estimation in terms of both the theoretical issues and practical methods that it will suffice for us to focus on density estimation mainly, while obtaining solutions for both problems in the end.

3.1.1 Nonparametric estimation.

Parametric methods are useful when the underlying distribution is known in advance or is simple enough to well-modeled by a standard distribution. Models which are sometimes called *semi-parametric* (such as mixtures of a fixed number of simpler distributions) are more flexible and more forgiving of the user's lack of the true model, but usually require significant computation in order to fit the resulting nonlinear models (such as the EM iterative re-estimation method). *Nonparametric* methods assume the least structure of the three, and take the strongest stance of letting the data speak for themselves [Sil86]. They are useful in the setting of arbitrary-shape distributions coming from complex real-world data sources.

Versus parametric.

Neither parametric nor nonparametric estimators are to be preferred in all situations. For example, when the sample size is small, there are sometimes statistical reasons to prefer parametric methods - though in the increasingly common data mining setting we are generally dealing with large if not massive datasets. If compression is desired, nonparametric methods are entirely inappropriate. On the other hand, often strong parametric assumptions are inappropriate, perhaps nowhere more so than in exploratory data analysis. In practical terms, incorrect assumptions generally lead to incorrect inferences.

When do we know the underlying distribution? *Almost never*, except in the most artificial situations. The higher accuracy of nonparametric methods in general, and the resulting improvement in inferences, has been widely observed both theoretically and in practice. However, they apparently often come at the heaviest computational cost of the three types of models. This has, to date, been the fundamental limitation of nonparametric methods for function estimation. It prevents practitioners from applying them to the increasingly large datasets that appear in modern real-world problems, and even for small problems, their use as a repeatedly-called basic subroutine is limited.

Estimation with minimal assumptions.

Nonparametric methods make minimal or no distribution assumptions and can be shown to achieve asymptotic estimation optimality for *any* input distribution under them. For example using KDE (detailed below), with no assumptions at all on the true underlying distribution, given only that the scale $h_N \to 0$ and $Nh_N \to \infty$, and that the kernel $K()$ is a non-negative Borel function whose integral is 1 (easily satisified by all commonly-used kernels), then with probability 1,

$$\int |\hat{p}(\underline{x}) - p(\underline{x})| d\underline{x} \to 0 \quad as \quad N \to \infty \tag{3.1}$$

i.e. as more data are observed, the estimate converges to the true density [DG85].

This *consistency* property is clearly one that no particular fixed parametric (or 'semi'-parametric) form can achieve. [1]

1. Though semi-parametric approaches provide a certain middle-ground between parametric and nonparametric approaches, utilizing a number of components/hidden units/basis functions which is smaller than the training set, but are generally characterized by nonlinear optimization procedures which either find only locally-optimal solutions dependent on starting conditions (*e.g.* EM, gradient descent) or are acutely expensive (*e.g.* quadratic programming). Semi-parametric methods such as mixtures of Gaussians or sigmoidal neural networks can be shown to to approximate any density, *but* only if the number of components/hidden units is allowed to grow to infinity with the data: at this point they are by definition nonparametric methods. When viewed this way, as universal approximators with a very large number of components, such models suffer from a similar computational problem to the one we address here, and as we saw in Chapter 1 they are often also N-body problems.

For this reason nonparametric estimators are the focus of a considerable body of advanced statistical theory [Rao83, DL01].

3.1.2 Kernel density estimation.

The task is to estimate the density $\hat{p}(\underline{x}_q)$ for each point $\underline{x}_q$ in a query (test) dataset $\underline{X}_{\mathcal{Q}}$ (having size $N_{\mathcal{Q}}$), from which we can also compute the overall log-likelihood of the dataset $\widehat{L}_{\mathcal{Q}} = \sum_{q=1}^{N_{\mathcal{Q}}} \log \hat{p}(\underline{x}_q)$. *Kernel density estimation* (KDE) is the most widely analyzed and used nonparametric density estimation method. (There exist many elaborations upon the basic model presented here, but most of them do not pose any particular problems for our computational approach.) The 'model' is the reference dataset $\underline{X}_{\mathcal{R}}$ (having size $N_{\mathcal{R}}$) itself, in addition to a local kernel function $K()$ centered upon each training datum, and its scale parameter h (the 'bandwidth'). The density estimate at the q^{th} test point $\underline{x}_q$ is

$$\hat{p}(\underline{x}_q) = \frac{1}{N_{\mathcal{R}}} \sum_{r=1}^{N_{\mathcal{R}}} \frac{1}{V_{Dh}} K\left(\frac{\|\underline{x}_q - \underline{x}_r\|}{h}\right) \tag{3.2}$$

where D is the dimensionality of the data and $V_{Dh} = \int_{-\infty}^{\infty} K_h(z)dz$, a normalizing constant depending on D and h. We'll write $K_h(\|\underline{x}_q - \underline{x}_r\|)$ when it is clearer to do so.

3.1.3 Practical considerations and constraints.

We'll now consider the main practicalities, or real-usage constraints, that must be used to judge the effectiveness of any solution to the KDE problem.

The discussion regarding the need for handling arbitrary D and the ability to reuse structures S from 2.1.1 apply with equal force here, in addition to the following added sources of difficulty.

Optimal bandwidth h^*.

As mentioned, the central issue of estimating a density optimally with KDE is selecting the optimal bandwidth h^*. Across statistical learning, *model selection* in current practice often amounts to evaluating a set of learned models (representing a finite set of parameter settings chosen from the set of all possible parameters) under a score function and a dataset (where the score may correspond to, for example, a Bayesian posterior, structural risk minimization, maximum entropy, maximum likelihood, least-squares, and so on).

Cross-validation. In common practice, the primary data-driven alternative to asymptotic analytical choices for bandwidth selection is *cross-validation* [Bow85, JMS96], a particular method of scoring. The two most widely-used methods both end up being a form of *leave-one-out* cross-validation. *Likelihood cross-validation* [HHv74] is derived by minimization of the Kullback-Liebler information $\int p(\underline{x}) \log \frac{p(\underline{x})}{\hat{p}(\underline{x})} d\underline{x}$, yielding the score

$$CV(h) = \frac{1}{N_{\mathcal{R}}} \sum_{r=1}^{N_{\mathcal{R}}} \log \hat{p}_{-r}(\underline{x}_r) \tag{3.3}$$

where the $-r$ subscript denotes an estimate using all $N_{\mathcal{R}}$ points except the r^{th}. *Least-squares cross-validation* [Rud82] minimizes the integrated squared error criterion $\int [\hat{p}(\underline{x}) - p(\underline{x})]^2 d\underline{x}$, yielding the computationally similar score

$$M_1(h) = \frac{1}{N_{\mathcal{R}}} \sum_{r=1}^{N_{\mathcal{R}}} \hat{p}_{-r}(\underline{x}_r) \tag{3.4}$$

where the density estimate uses the derived kernel $K^*() = K() * K() - 2K()$, * denoting convolution.

Either scoring procedure requires N density estimates, each based on $N-1$ points; thus cross-validation scoring is itself an N-body problem.

Bandwidth search. Further, this is done for each of the B bandwidths under consideration, making the cost of estimation even more acutely felt. Usually this set of bandwidths to evaluate is simply chosen to be equally-spaced bandwidths within some predetermined range, though one might imagine a more adaptive procedure for determining which bandwidths to evaluate. Thus in addition to scoring, there is either an implicit or explicit *bandwidth search* procedure which selects the bandwidths to be evaluated.

Multiple bandwidths h.

So *scoring models for selection* is one big reason we'll need to compute multiple densities, each for a different value of h. In addition, other common tasks demand the computation of models for the same data but B different bandwidths.

Scoring models for combination. An alternative to strict model selection is *model combination*, in which the estimates of multiple learned models (again corresponding to some finite set of chosen parameter settings) are combined to form a final estimate, weighted by their score. Examples include Bayesian model combination and stacking. This methodology has been the focus of considerable attention in the learning literature in recent years, mainly for the task of classification – however, the same principle applies to density estimation, as noted by [SW99].

Exploratory visualization. It is often useful in exploratory data analysis to visualize the curve representing the score as a function of the bandwidth. Figure 3.1 shows an example of the kind of curve we would like to be able to generate quickly. Shown are the cross-validated likelihood scores for 1000 bandwidths ranging from 0.0001 to 0.1, along with the optimal bandwidth h^* (about .00774), for the astrophysics dataset described later in the empirical results.

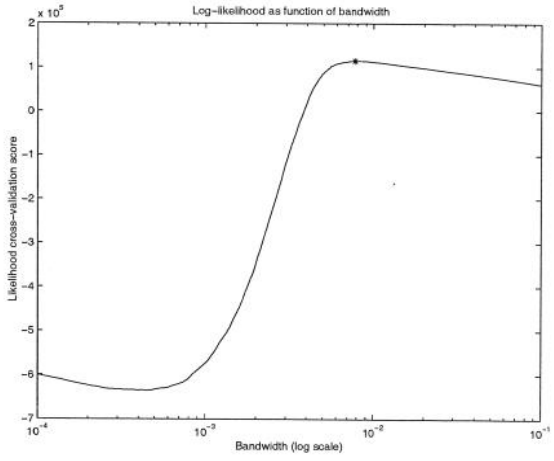

Figure 3.1: EXAMPLE OF VISUAL BANDWIDTH ANALYSIS.

Kernel function choice $K()$.

The choice of $K()$ has also been the subject of much investigation, and thus varies widely.

Common multivariate choices for $K()$ include the spherical, Gaussian, and Epanechnikov kernels. The spherical kernel ($K_h(||\underline{x}_q - \underline{x}_r||) = 1$ if $||\underline{x}_q - \underline{x}_r|| < h$, otherwise

0, with normalizing constant V_{Dh}^s, the volume of the sphere of radius h in D dimensions) is simplest but can introduce sharp discontinuities for small datasets. The Epanechnikov kernel ($K_h(\|\underline{x}_q - \underline{x}_r\|) = 1 - \|\underline{x}_q - \underline{x}_r\|^2$ if $\|\underline{x}_q - \underline{x}_r\| < h$, otherwise 0, with normalizing constant $\frac{D+2}{2V_{Dh}^s}$) has the property of asymptotically minimal MISE among all possible kernels [HL56], and so is the default used in our studies.

Note that this is in constrast to common belief among naive users of KDE, particularly non-statisticians, that the Gaussian kernel is sufficient for all purposes — in fact, many do not even realize that there exists a world of kernel functions beyond the Gaussian. However, aside from theoretical understanding that other kernels provide faster convergence, many data analysis tasks require alternative properties such as finite tails.

For this reason our method is specifically designed to work well under very weak assumptions on the kernel function - namely, that $K()$ is *monotonic*. (In this presentation we'll also assume that it is also positive and *decreasing* away from zero, though these can actually be dropped in principle since our core approach does not rely on these to hold.) This admits practically any kernel function considered reasonable. The effect of the bandwidth size and shape of the kernel function on the efficiency of our method are examined later in the paper.

§3.2 Monopole approximation.

We'll first develop a simple single-tree algorithm for KDE, *i.e.* where a tree partitions the reference dataset $\underline{X}_{\mathcal{R}}$ and we compute the density at a single point $\underline{x}_q$.

3.2.1 Exclusion and inclusion.

We'll begin with some intuition for how such space-partitioning data structures can be used for efficient summation of continuous kernel values. At each node R encountered during the traversal, using the boundary of the data $\underline{X}_R$ (having size N_R) in the node, we obtain bounds on the distance of $\underline{x}_q$ to any point $\underline{x}_r \in \underline{X}_R$ as we did in the nearest-neighbor case.

The evaluation of $K_h()$ on these values yields bounds on the mass contribution of $\underline{X}_R$ to $\hat{p}(\underline{x}_q)$. Suppose the kernel has finite extent, such as the Epanechnikov kernel. If the maximum density contribution of R is zero, *i.e.* the lower bound on the distance was greater than h, R can be *pruned* from the search (*i.e.* we do not need to recurse on its children). We call this *exclusion*;

Exclusion rule:

$$\text{If } \delta_q^{\min} > h, \quad \Phi(\underline{x}_q) \mathrel{+}= 0; \text{ return} \tag{3.5}$$

The opposite, *inclusion*, is also possible, for example in the case of the spherical kernel, in which any distance less than h yields a constant kernel value:

Inclusion rule:

$$\text{If } \delta_q^{\min} \leq h, \quad \Phi(\underline{x}_q) \mathrel{+}= N_R; \text{ return} \tag{3.6}$$

Note that in the case of a finite-extent kernel, exclusion/inclusion results in no error, yet can eliminate large chunks of data from consideration in a single sweep.

3.2.2 Monopole approximation rules.

In order to generalize beyond finite-extent kernels, we'll generalize our notion of (exact) pruning to one of approximation.

If the difference between these bounds is smaller than some predetermined small τ, we can 'prune' the node by approximating its mass contribution by its centroid $\underline{\mu}_R$:

Monopole approximation with simple rule:

$$\text{If } K(\delta_q^{\min}) - K(\delta_q^{\max}) < \tau, \quad \Phi(\underline{x}_q) \mathrel{+}= N_R K(\underline{\mu}_R) \tag{3.7}$$

Exclusion and inclusion are actually both special cases of this more general pruning rule. For this reason we will sometimes use the terms 'pruning' interchangeably with 'approximation'.

This is exactly the Barnes-Hut algorithm, which we'll consider in more depth in the next chapter.

A variant which has good properties in practice uses a different rule for deciding when to apply the monopole approximation:

Monopole approximation with ratio rule:

$$\text{If } \frac{K(\delta_q^{\min}) - K(\delta_q^{\max})}{\Phi(\underline{x}_q)^{lo}} < \tau, \quad \Phi(\underline{x}_q) \mathrel{+}= N_R K(\underline{\mu}_R) \tag{3.8}$$

So far this is the algorithm of [DM95], adapted slightly for kernel density estimation instead of locally weighted kernel regression.

§3.3 Finite-difference approximation.

3.3.1 Quadrature, interpolation, and finite differences.

We can regard our problem as a kind of *quadrature*, or numerical integration problem, where the problem is to find $\int_{\delta_{QR}^{\max}}^{\delta_{QR}^{\min}} K_h(\delta) d\delta$, though we actually want the value at a finite number of evaluation points given by our data, $\sum_{qr} K_h(\delta_{qr} d\delta_{qr})$. Within quadrature is an *interpolation* problem, namely that of approximation function values lying between the quadrature points.

Taking the simplest form of *Newton-Cotes* formula, the two-point form, gives the familiar *trapezoidal rule*, as shown in Figure 3.2.

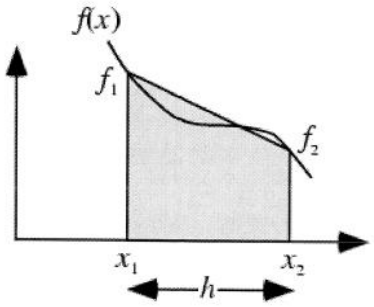

Figure 3.2: TRAPEZOIDAL RULE FOR NUMERICAL INTEGRATION.

Here the function values lying between the endpoints, the quadrature points in this case, are approximated by a linear function of the abscissa (polynomial in general). Recalling Taylor's theorem,

$$\begin{aligned} f(x) &= \sum_{p=0}^{\infty} \frac{f^{(p)}(a)}{p!}(x-a)^p \\ &= f(a) + f'(a)(x-a) + \dots \end{aligned} \tag{3.9}$$

we see that this particular choice of interpolation corresponds to the Newton-Stirling formula (using central differences) or *Gregory-Newton* formula (using forward differences, shown), finite analogs of Taylor's theorem:

$$f(x) = f(x_i) + \frac{1}{2}\frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}(x - x_i) + \dots \tag{3.10}$$

This is called the *finite-difference* approximation and is used in many forms throughout applied mathematics. The basic idea of approximating continuous functions with a finite set of pieces lies at the heart of all methods for numerical integration and the important special case of solving differential equations.

3.3.2 Adaptive quadrature rule.

Let $\delta_{QR}^{\min}$ and $\delta_{QR}^{\max}$ be our bounds on the distance between respective points in Q and R, yielding lower and upper bounds on R's mass contribution to Q of $K_h(\delta_{QR}^{\max})$ and $K_h(\delta_{QR}^{\min})$, respectively. An obvious special case of the Gregory-Newton formula is

$$\begin{aligned} K_h(\delta_{QR}) &= K_h(\delta_{QR}^{\min}) + \frac{1}{2}(\delta_{QR} - \delta_{QR}^{\min})\frac{\{K_h(\delta_{QR}^{\max}) - K_h(\delta_{QR}^{\min})\}}{\delta_{QR}^{\max} - \delta_{QR}^{\min}} \\ &\quad + O((\delta_{QR} - \delta_{QR}^{\min})^2) \end{aligned} \tag{3.11}$$

Since we are using $\delta_{QR}^{\min}$ and $\delta_{QR}^{\max}$ as our two quadrature points, and the placement of node boundaries is determined by the distribution of the data, we are in the realm of *adaptive quadrature* rather than the familiar fixed-width quadrature.

Intuitively, the closer $K_h(\delta_{QR}^{\max})$ and $K_h(\delta_{QR}^{\min})$ are to each other, the better we can approximate R's contribution by $N_R\overline{K_h}$ where $\overline{K_h} = \frac{1}{2}\{K_h(\delta_{QR}^{\max}) + K_h(\delta_{QR}^{\min})\}$.

Recall that our ultimate aim in this is not really to estimate the integral's value via some mean and standard deviation, say. (Though we will discuss this option later.) Instead, we are interested in hard error guarantees in the current context, and thus we need *bounds* on the integral (actually, sum). In principle we could integrate the interpolation polynomial over the interval, but we don't know the placement of the evaluation points. However with the assumption $K_h()$ is *monotonic*, we can obtain simple bounds. The error of this linear approximation with respect to any point $\underline{x}_q \in Q$ is

$$\begin{aligned} e_{QR} &= \sum_r \left|K_h(\|\underline{x}_q - \underline{x}_r\|) - \overline{K_h}\right| \le \frac{N_R}{\delta_{QR}^{\max} - \delta_{QR}^{\min}} \int_{\delta_{QR}^{\max}}^{\delta_{QR}^{\min}} \left|K_h(\delta) - \overline{K_h}\right| d\delta \\ &= \frac{N_R}{2}\left\{K_h(\delta_{QR}^{\min}) - K_h(\delta_{QR}^{\max})\right\}. \end{aligned} \tag{3.12}$$

To ensure that every $\Phi(\underline{x}_q)$'s error e_q meets the user-specified ϵ tolerance, or $\frac{e_q}{\Phi(\underline{x}_q)} \le \epsilon$, we can enforce that $\frac{e_{QR}}{\Phi(\underline{x}_q)} \le \frac{N_R}{N}\epsilon$ by using the running lower bound $\Phi_Q^{\min}$ for $\Phi(\underline{x}_q)$, yielding

$$\frac{\frac{N_R}{2}\left\{K_h(\delta_{QR}^{\min}) - K_h(\delta_{QR}^{\max})\right\}}{\Phi_Q^{\min}} \leq \frac{N_R}{N}\epsilon \tag{3.13}$$

or more simply,

$$K_h(\delta_{QR}^{\min}) - K_h(\delta_{QR}^{\max}) \leq \frac{2\epsilon}{N}\Phi_Q^{\min} \tag{3.14}$$

as a *local* pruning criterion which ensures the *global* error tolerance ϵ. It can be seen that exclusion and inclusion are also special cases of this generalized rule.

Error tolerance specification.

This design now allows the user to simply specify an error tolerance directly rather than fiddle with indirect parameters (such as τ or a number of grid points M) with no known relationship to the error.

Ideally the algorithm would meet the prescribed error exactly, *i.e.* it would do no more work than necessary to achieve that error. If the size of the difference between the real error and the prescribed error is unpredictable, then effectively the parameter ϵ is once again an indirect parameter with no known relationship to the error, other than being an upper bound. The situation is much improved by knowledge of the latter fact, but still not optimal. In actuality our algorithm falls short of this ideal, since the algorithm will overshoot the minimal amount of work. On the other hand, the actual error is typically less than one order of magnitude (or one significant digit) smaller than the error requested. This behavior is much better than that of, say, the FMM. This issue will be discussed further in Chapter 4.

3.3.3 Guaranteed anytime bounds.

Now we seek to design an algorithm which provides *hard* (as opposed to say, confidence bounds holding with some probability less than 1) bounds on the error at all times during its execution.

We maintain at all times bounds $\Phi_q^{\min}$ and $\Phi_q^{\max}$ on the unnormalized sum $N_{\mathcal{R}}\hat{p}(\underline{x}_q)$. We work with unnormalized quantities to avoid pointless (though admittedly minor) divisions as well as to keep floating point numbers within a favorable ranges. We'll sometimes refer to *mass* when talking about unnormalized density.

We begin with maximally pessimistic bounds and tighten them as we recurse and observe training points at increasingly finer granularity. We start by agnostically setting the lower bound to assume that no training points contribute any mass, and the upper bound to assume that all training points contribute maximum mass. At the end of the computation, the estimate $\Phi(\underline{x}_q)$ is based on the midpoint between $\Phi_q^{\min}$ and $\Phi_q^{\max}$.

Such an algorithm has the property of *anytime* operation, *i.e.* in principle if it is stopped at any time during its execution, it will output a valid answer, whose quality increases with running time.

Bound tightening.

Assuming the kernel function has maximum value 1 (without loss of generality as other values can be accounted for), the maximally pessimistic upper bound $\Phi_q^{\min}$ for every $\Phi(\underline{x}_q)$ is set to $N_{\mathcal{R}} \cdot 1 = N_{\mathcal{R}}$. Assuming the kernel function has minimum value 0 (again without loss of generality), the maximally pessimistic lower bound $\Phi_q^{\max}$ is set to $N_{\mathcal{R}} \cdot 0 = 0$ for every query point.

Each time a pruning operation is performed, in which a reference node R's contribution to the query node Q is (implicitly) approximated by $N_R\overline{K_h}$, we gain the

knowledge that $N_R K_h(\delta_{QR}^{\max})$ is the minimum possible contribution of node R to Q and $N_R K_h(\delta_{QR}^{\min})$ is the maximum possible contribution. The proper update to the bounds is then:

$$\begin{aligned} \forall q \in Q, \Phi_q^{\min} &+= N_R K_h(\delta_{QR}^{\max}) \\ \forall q \in Q, \Phi_q^{\max} &+= N_R[K_h(\delta_{QR}^{\min}) - 1] \end{aligned} \tag{3.15}$$

The last update accounts for the *a priori* mass contribution of 1 assumed for each of the N_R reference points, which can now be undone given the observation of their true maximum contribution. This is the explanation for the resulting subtractions of N_R which appear through the algorithm pseudocode, which may be confusing without knowing this.

3.3.4 Dual-tree finite-difference algorithm.

We now apply the shattering principle to obtain a *dual*-tree algorithm for KDE, in which a second tree is also built to partition the query set $\underline{X}_Q$. This should seem very natural after the last chapter. Again, if $\underline{X}_Q = \underline{X}_R$ there is only one tree.

The density bounds $\Phi_q^{\min}$ and $\Phi_q^{\max}$ become $\Phi_Q^{\min}$ and $\Phi_Q^{\max}$, holding for all the query points in Q.

Delayed summation.

One technique which can only be applied in the dual-tree case is that of *delayed summation*. Suppose that a query node meets a reference node which can be pruned, *i.e.* the mass contained in the reference node should be added to all elements of the query node. Rather than explicitly iterating over every $q \in Q$ as implied earlier, we can now simply add the mass contributions to $\Phi_Q^{\min}$ and $\Phi_Q^{\max}$, which is $O(1)$ rather than $O(N_Q)$.

These mass contributions all wind up in the final answers via a single post-processing pass over the query tree, which simply adds any mass stored in these variables to the query points in the relevant nodes. This can be arranged to be done in $O(N_Q)$ time.

KDE(Q,R,h)
$dl = N_R K_h(\delta_{QR}^{\max})$, $du = N_R K_h(\delta_{QR}^{\min}) - N_R$.
`if` $K_h(\delta_{QR}^{\min}) - K_h(\delta_{QR}^{\max}) \leq \frac{2\epsilon}{N}\Phi_Q^{\min}$,
`foreach` $\underline{x}_q \in Q$, $\Phi_q^{\min}$ += dl, $\Phi_q^{\max}$ += du.
`return`.
`else`,
`if` leaf(Q) `and` leaf(R), **KDEBase**(Q,R,h), `return`.
KDE(Q.left,closer-of(Q.left,$\{R$.left,R.right$\}$,h)).
KDE(Q.left,farther-of(Q.left,$\{R$.left,R.right$\}$,h)).
KDE(Q.right,closer-of(Q.right,$\{R$.left,R.right$\}$,h)).
KDE(Q.right,farther-of(Q.right,$\{R$.left,R.right$\}$,h)).

KDEBase(Q,R,h)
`foreach` $\underline{x}_q \in Q$,
`foreach` $\underline{x}_r \in R$,
$c = K_h(\|\underline{x}_q - \underline{x}_r\|)$, $\Phi_q^{\min}$ += c, $\Phi_q^{\max}$ += c.
$\Phi_q^{\max}$ -= N_R.
$\Phi_Q^{\min} = \min_{q \in Q} \Phi_q^{\min}$, $\Phi_Q^{\max} = \max_{q \in Q} \Phi_q^{\max} - N_R$.

Figure 3.3: DUAL-TREE ALGORITHM, BASIC FORM. In the pseudocode a += b means a = a + b. A leaf's left or right child is defined to be itself. In the actual code repeated recursion cases are prevented.

§3.4 Optimization of upper and lower bounds.

We can enhance performance by maximizing the tightness of the bounds, which allows approximation pruning as early in the search as possible. These techniques (not shown in the algorithm for simplicity of presentation) are not strictly necessary in order to realize the primary gain in efficiency yielded by the dual-tree structure of the algorithm, but offer more opportunities for acceleration, which are not available to a single-tree algorithm.

3.4.1 Up-down mass propagation.

Cross-scale information maximization.

Our delayed summation technique actually causes a certain problem. Each local bounds update due to a prune can be regarded as a new piece of information which is known only locally. For example, when we store a reference node R's contribution in the bounds for Q, none of Q's children know about it, though logically all the bounds in the entire sub-tree of Q should reflect this information, as they merely represent the same data elements at different scales. Information in the tree as a whole can be maximized by upward and downward propagation.

Downward propagation recursively passes dl and du to the entire subtree below Q. This can be done by a simple pre-order traversal. It is performed whenever new mass is obtained by the node.

Upward propagation can be done simply by taking the min/max of the children's bounds. This technique can be seen as an instance of tree-based dynamic programming. In the pure recursive form of the algorithm it is performed after returning from the recursive calls on a node's children. In the priority queue version of the algorithm it is performed upon entry of a node.

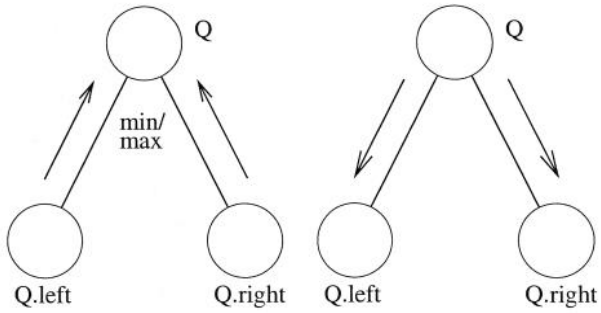

Figure 3.4: UP-PROPAGATION AND DOWN-PROPAGATION.

3.4.2 Deferred asynchronous propagation.

To avoid the cost of full downward propagation down to the leaves upon every prune, we can employ a frugal asynchronous dynamic programming method: dl and du are passed only to Q's immediate children, in temporary holding slots for 'owed' mass. Whenever any node is considered during the search, it first checks these slots for any mass owed to it, integrates that mass into its bounds, and passes the mass to its immediate children's 'owed' slots.

Propagation with this technique now has cost $O(1)$ rather than scaling with the number of nodes in the tree.

§3.5 Multiple density models.

We now generalize the algorithm to include a range of bandwidths indexed by b^{lo} and b^{hi} during consideration of each node-pair, which is recursively narrowed as search progresses toward the leaves. This corresponds to a second application of the principle of higher-order divide-and-conquer.

All the bounds, such as $\Phi_q^{\min}$ and $\Phi_q^{\max}$, generalize from the scalar quantities of the single-bandwidth algorithm to *vector* quantities, containing bounds for each bandwidth.

3.5.1 Sharing and nesting.

Sharing distance computations.

In the base case, we reuse each distance computation δ_{QR} for each of the $b^{hi}-b^{lo}+1$ bandwidths that remain upon reaching the base case, just as the naive exhaustive algorithm can do if modified for the multi-bandwidth problem.

MultiKDEBase(Q, R, b^{lo}, b^{hi})
`foreach` $\underline{x}_q \in Q$,
  `foreach` $\underline{x}_r \in R$,
    `foreach` $b \in [b^{lo}, b^{hi}]$,
      $c = K_{h_b}(\|\underline{x}_q - \underline{x}_r\|)$, $\Phi_{qb}^{\min} \mathrel{+}= c$, $\Phi_{qb}^{\max} \mathrel{+}= c$.
  `foreach` $b \in [b^{lo}, b^{hi}]$,
    $\Phi_{qb}^{\max} \mathrel{-}= N_R$.
`foreach` $b \in [b^{lo}, b^{hi}]$,
  $\Phi_{Qb}^{\min} = \min_{q \in Q} \Phi_{qb}^{\min}$, $\Phi_{Qb}^{\max} = \max_{q \in Q} \Phi_{qb}^{\max} - N_R$.

Figure 3.5: MULTI-BANDWIDTH DUAL-TREE BASE CASE.

Bandwidth indexing.

In order to be able to efficiently locate the bucket (*i.e.* inter-bandwidth range) that a given distance falls into (we'll see why we want this in a moment), the buckets can be indexed efficiently using a binary search tree if B is large. We denote the operation of finding the index of the smallest bin b^* in the set $\mathcal{B}$ containing the distance δ by $\lfloor \delta \rfloor_{\mathcal{B}}$.

For moderate B a reasonable alternative is to compute the kernel evaluations for each bandwidth from highest to lowest when checking for exclusion.

Nested exclusion/inclusion.

Nested exclusion and inclusion. In the finite-extent kernel case, we can quickly perform an exclusion once we locate the smallest bandwidth still containing δ_{QR}^{lo}. This is due to the *nesting* property that if a bandwidth h is excludable, so are all bandwidths $h' < h$. This nesting property also applies to inclusion in the spherical kernel case, in reverse.

$\delta_{QR}^{lo} = \|Q - R\|^{lo}$, $\delta_{QR}^{hi} = \|Q - R\|^{hi}$.
$b_{new}^{lo} = \lfloor \delta_{QR}^{lo} \rfloor_{\mathcal{B}}$.
foreach $b \in [b_{new}^{lo}, b^{hi}]$,
  $\Phi_{Qb}^{\max}$ -= N_R.
$b_{new}^{hi} = \lceil \delta_{QR}^{hi} \rceil_{\mathcal{B}}$.
foreach $b \in [b^{lo}, b_{new}^{hi}]$,
  $\Phi_{Qb}^{\min}$ += N_R.

Figure 3.6: NESTED EXCLUSION AND INCLUSION.

Infinite-extent case. If neither of these cases holds, the normal approximation criterion can simply be tested for each bandwidth.

Propagation in ranges.

Updating of bounds, including the necessary upward and downward propagation procedures, is also now generalized to ranges. For example an exclusion can now result in bounds updates for a sequence of multiple bandwidths as we saw.

3.5.2 Recursive range-narrowing.

After bounds updating and propagation for the appropriate sub-ranges, we narrow the range of bandwidths which still need to be considered and recurse. Note that with this procedure there is no loss of information nor pruning opportunity, with respect to the single-bandwidth algorithm.

This can be seen as another application of the higher-order divide-and-conquer principle.

MultiKDE(Q,R,b^{lo},b^{hi})
$\delta_{QR}^{lo} = \|Q-R\|^{lo}$, $\delta_{QR}^{hi} = \|Q-R\|^{hi}$.
$b_{new}^{lo} = \lfloor \delta_{QR}^{lo} \rfloor_{\mathcal{B}}$.
`foreach` $b \in [b_{new}^{lo}, b^{hi}]$,
$\Phi_{Qb}^{\max}$ -= N_R.
$b_{new}^{hi} = \lceil \delta_{QR}^{hi} \rceil_{\mathcal{B}}$.
`foreach` $b \in [b^{lo}, b_{new}^{hi}]$,
$\Phi_{Qb}^{\min}$ += N_R.
`if` $b_{new}^{hi} - b_{new}^{lo} == 0$, `return`.
`else`,
`if` leaf(Q) and leaf(R), **MultiKDEBase**$(Q,R,b_{new}^{lo},b_{new}^{hi})$, `return`.
MultiKDE(Q.left,closer-of(Q.left,{R.left,R.right} $b_{new}^{lo},b_{new}^{hi}$)).
MultiKDE(Q.left,farther-of(Q.left,{R.left,R.right} $b_{new}^{lo},b_{new}^{hi}$)).
MultiKDE(Q.right,closer-of(Q.right,{R.left,R.right} $b_{new}^{lo},b_{new}^{hi}$)).
MultiKDE(Q.right,farther-of(Q.right,{R.left,R.right} $b_{new}^{lo},b_{new}^{hi}$)).

Figure 3.7: DUAL-TREE ALGORITHM, MULTI-BANDWIDTH. For illustration the spherical kernel case is shown.

§3.6 Find-bandwidth procedure.

We can utilize the multi-bandwidth algorithm for several purposes, as previously noted. We now show an effective method for searching for the optimum bandwidth h^* which utilizes the multi-bandwidth procedure and well as single-bandwidth probes of the cross-validation score.

We assume that the cross-validation score curve has a single optimum. While not necessarily valid in general, this appears to hold fairly robustly in practice. For simplicity let us consider only the least-squares score, a quantity to be minimized. The strategy is to start from the high end, with a theoretical 'over-smoothing' bandwidth [Sco92], and move down in orders of magnitude until the descent in score stops. We refer to the first bandwidth at which the score increases as the 'critical bandwidth' h^{crit}. s_h refers to the score for bandwidth h and s^* refers to the best score. Because discretization and other effects can in practice cause the procedure to sometimes prefer zero bandwidth, a minimum bandwidth is chosen *a priori* as five orders of magnitude below the oversmoothing bandwidth.

FindBW(Q.root,R.root)
$h^{crit} = \infty$, $h^{max} = h_{OS}$, $h^{min} = h^{max} \times 10^{-5}$.
while $h \leq h^{min}$,
$s_h =$ **KDE**(Q.root,R.root,h).
if $s_h > s^*$,
if !slow, $h^{crit} = h$, slow $= 1$, $s^* = \infty$.
else break.
else $s^* = s$, $h^* = h$.
if !slow,
$\mathcal{H} = [h^{crit} \times 1.1, h^{crit} \times 90]$.
$s_{\mathcal{H}} =$ **MultiKDE**(Q.root,R.root,$\mathcal{H}$).
$h^* =$ argmin($s_{\mathcal{H}}$).
else $h = h/10$.
if $h^{crit} = \infty$, return ∞.

Figure 3.8: FIND-BANDWIDTH FUNCTION. Note that **KDE**() is actually implemented as a special case of **MultiKDE**().

If ∞ is returned, then no critical bandwidth was found. In this case the user is informed and can exercise the fallback option of using a theoretical plug-in estimate for the bandwidth.

§3.7 Performance.

Empirical study.

We measure seconds of actual runtime on a modern Pentium-Pro Linux desktop workstation with 2Gb of RAM. Asterisks denote times estimated from smaller problem sizes using the known algorithm complexity. All density estimates were performed at the optimal bandwidth h^* as found by likelihood cross-validation [HHv74], chosen over the set $\{0.25, 0.5, 0.75, 1\} \times 10^i$ for $-5 \leq i \leq 1$. The greatest computation requirement almost always occurred at the optimal bandwidth, as evidenced by the third table (in fact we conjecture that this may be a provable property of the algorithm), and so represents a worst case in terms of bandwidth. In all experiments a leave-one-out computation is measured, so the training set and test set have the same size N. The approximation parameter is set in all cases so that the maximum possible error in the overall log-likelihood was no more than 10^{-3}, *i.e.* one tenth of one percent away from the true value, and in most cases was no more than 10^{-6}. The kernel function used is the Epanechnikov kernel, which has optimal efficiency among all kernel functions. In all of the experiments only ball-trees are used.

Datasets.

Most experiments are on a segment of the Sloan Digital Sky Survey—a data collection of current scientific interest, and the active subject of ongoing nonparametric density estimation studies. It contains spatial coordinates in the first two dimensions - the dataset containing these attributes is called RA-Dec. The Sloan data includes an additional 20 color attributes from various instruments, which we test in a separate dataset called Colors. We also test a 5-dimensional biological screening dataset called BIO5.

3.7.1 Scaling with dataset size.

From our theoretical analysis we showed that single-tree algorithm scales as $O(N \log N)$ and the dual-tree algorithm scales as $O(N)$. These conclusions are supported by our empirical observations.

Data = SDSS, $D = 2$					
N	h^*	Tree Build Time	Naive Time	Single Tree Time	Dual Tree Time
12.5K	.0025	.3	7	.45	.12
25K	.0025	.6	31	1.4	.31
50K	.001	1	123	2.1	.46
100K	.00075	3	494	5	1.0
200K	.0005	6	1976*	10	2
400K	.0005	15	7904*	27	5
800K	.00025	33	31616*	49	10
1.6M	.00025	70	126465*	127	23

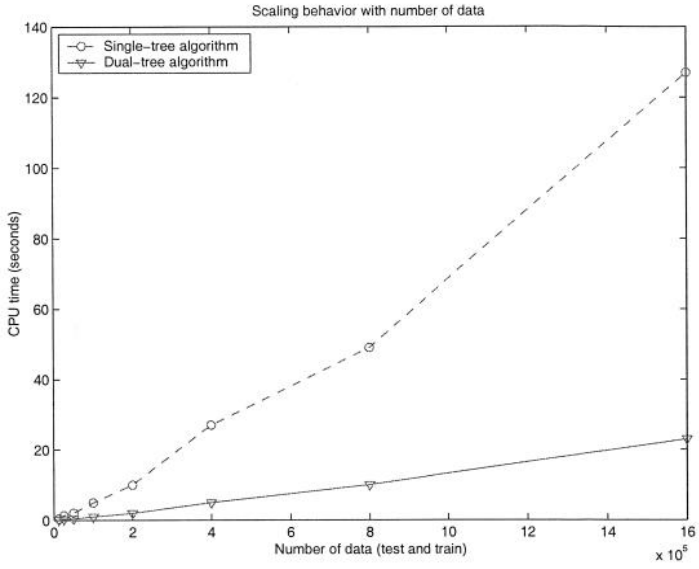

Figure 3.9: SCALING WITH DATASET SIZE.

3.7.2 Scaling with approximation.

Studying the effect of approximation for the Gaussian kernel (the finite-extent kernels yield near-exact estimates almost irrespective of the approximation level), we see a dramatic drop in runtime as the ϵ tolerance is increased. Note that the maximum possible error bounds provided by the algorithm are typically about two orders of magnitude more conservative than the actual error in log-likelihood.

Data = SDSS, $N = 1.6M$ Kernel = Gaussian		
ϵ	Max. Poss. Error	Dual Tree Time
.1	0.00084595	51
1	0.0149808	41
10	0.0753523	32
100	0.155863	21

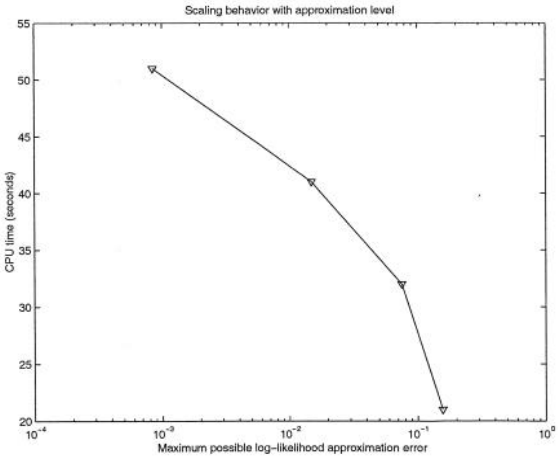

Figure 3.10: SCALING WITH APPROXIMATION.

3.7.3 Scaling with dimensionality.

As we add more of the dimensions of the SDSS dataset, the space becomes more complex and datasets are harder for the trees to localize. Note, however, that the growth is polynomial rather than exponential in D.

Data = SDSS, $N = 100,000$			
D	h^*	Naive Time	Dual Tree Time
2	.00075	494	1
3	.0075	543	6
4	.025	579	18
8	.05	945	41
16	.025	1424	43
32	.1	3326	57

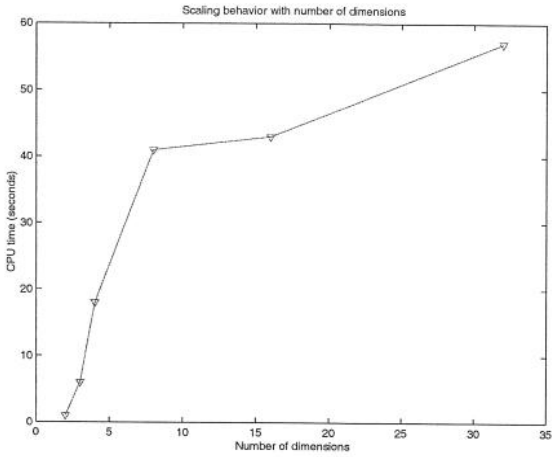

Figure 3.11: SCALING WITH DIMENSIONALITY.

3.7.4 Effect of bandwidth.

As noted earlier, estimation at bandwidths larger or smaller than the optimal bandwidth is typically much less expensive. The reason that small bandwidths are cheap is that more of the space can be pruned by exclusion, or its generalized notion for continuous functions. Likewise, the reason that large bandwidhths are cheap is that more of the space can be pruned by inclusion or its continuous form.

SDSS, $D = 2$ $N = 1.6M$	
h	Dual Tree
$.001h^*$	9
$.01h^*$	9
$.1h^*$	10
h^*	23
$10h^*$	18
$100h^*$	2
$1000h^*$	1

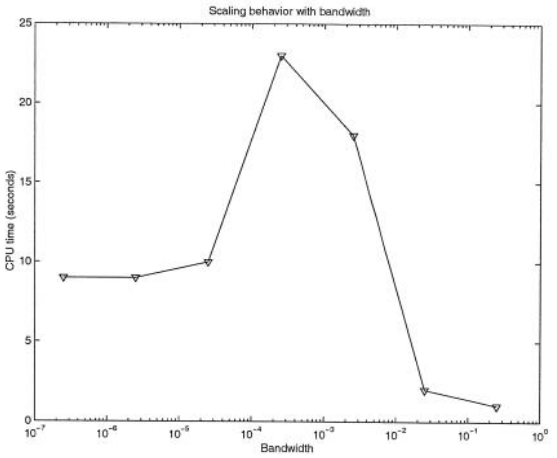

Figure 3.12: SCALING WITH BANDWIDTH.

3.7.5 Effect of kernel function. The infinite-extent Gaussian kernel yields approximately double the cost of the other two, finite-extent kernels.

Data = SDSS, $D = 2$				
N	Tree Build Time	Dual Tree Spher. Time	Dual Tree Epan. Time	Dual Tree Gauss. Time
12.5K	.3	.11	.12	.32
25K	.6	.31	.31	.70
50K	1	.45	.46	1.1
100K	3	1.0	1.0	2
200K	6	2	2	5
400K	15	5	5	11
800K	33	10	10	22
1600K	70	23	23	51

Figure 3.13: EFFECT OF KERNEL FUNCTION.

3.7.6 Other kinds of data. To exhibit the algorithm's behavior under varying settings, we explore a sampling of datasets generated by various different kinds of processes.

Other Datasets				
Dataset	N	D	h^*	Dual Tree Time
BIO5	103,016	5	10^{-2}	10
CovType	136,081	38	10^{-4}	8
MNIST	10,000	784	10^{-5}	24
PSF2d	3,056,092	2	10^{-3}	9

Figure 3.14: OTHER KINDS OF DATA.

3.7.7 Scaling with number of bandwidths, near optimum. We first examine the case in which the bandwidths fall on a scale ranging over one order of magnitude roughly centered around the optimum bandwidth h^*.

For all of the multi-bandwidth results, we measured the runtimes on a different Alpha-processor-based desktop workstation, which is not as fast as the most recent Pentium-Pro-based workstations but has 14Gb of RAM.

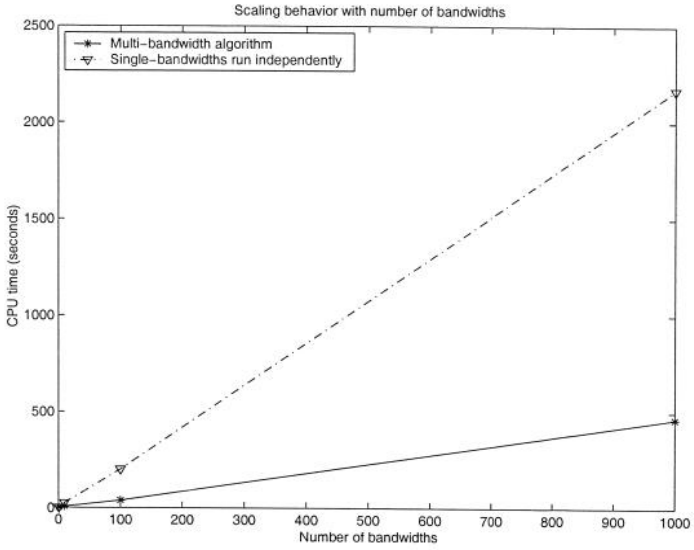

Figure 3.15: SCALING WITH NUMBER OF BANDWIDTHS: SIMULTANEOUS VS. SEPARATE COMPUTATIONS.

RA-Dec, $N = 100$K, $h \in [0.001, 0.01]$				
B	Multi Time	Indep Single Time	Naive Time	Speedup Over Naive
1	1.4	1.4	1204	889
10	7	25	3631	518
100	39	201	26859	671
1000	465	2170	374098	805

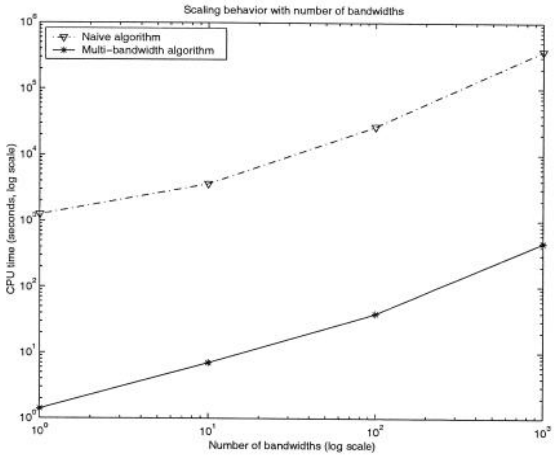

Figure 3.16: SCALING WITH NUMBER OF BANDWIDTHS: COMPARISON TO NAIVE METHOD NEAR OPTIMUM.

Though the theoretical complexity of the scaling of the multi-bandwidth algorithm is $O(B)$, the log-log plot of the growth in actual CPU time shows a superlinearity. Though relatively mild in the range of B we are typically interested in (100 models is probably a reasonable number for most purposes), it is curious that the naive exhaustive method displays the same superlinearity.

This appears to be a side-effect of a limitation of the multi-bandwidth algorithm – it has a large memory footprint necessitated by the fact that it must store B entire densities (actually bounds on them), each of size $O(N)$. For the largest number of bandwidths, the large RAM of our test workstation was taxed near its limit, limiting the size of dataset that can be processed. Further, far below the point of swapping, the surprising effect of hardware cache-locality issues becomes significantly evident. Note that the naive multi-bandwidth method also shares this

limitation. One advantage of independent single-bandwidth computations is that this memory consumption can be avoided if necessary.

3.7.8 Scaling with number of bandwidths, far from optimum.

We next create difficulty for the algorithm by making it evaluate densities over a much broader range, covering 3 orders of magnitude. In this case we expect less sharing to be possible between the simultaneous computations.

RA-Dec, $N = 100$K, $h \in [0.0001, 0.1]$				
B	Tree Build Time	Multi Time	Naive Time	Speedup Over Naive
1	5	1.4	1204	889
10	5	110	3631	33
100	7	545	26859	49
1000	10	6240	374098	60

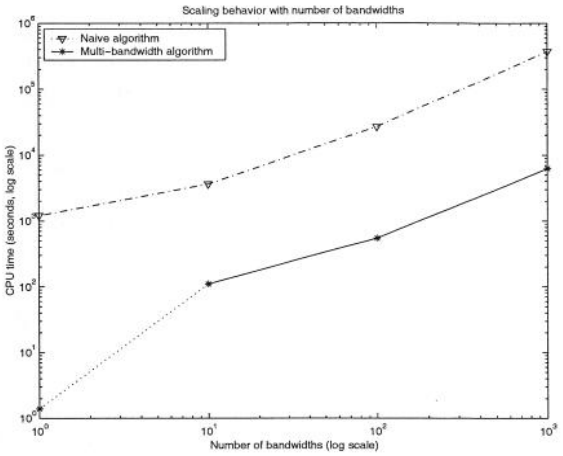

Figure 3.17: SCALING WITH NUMBER OF BANDWIDTHS: COMPARISON TO NAIVE METHOD FAR FROM OPTIMUM.

We indeed observe a large degradation in the performance of the multi-bandwidth algorithm in this case. Over an order of magnitude of computational advantage is lost (note that the first data point in the plot of **??** is misleading).

3.7.9 Other kinds of data.

To exhibit the algorithm's behavior under varying settings, we explore a sampling of datasets generated by various different kinds of processes.

$N = 100$K, near h^*				
B	RA-Dec (2-d)	BIO5 (5-d)	Colors (20-d)	Naive Time
1	1.4	14	20	1204
10	7	90	105	3631
100	39	449	413	26859
1000	465	5341	6675	374098

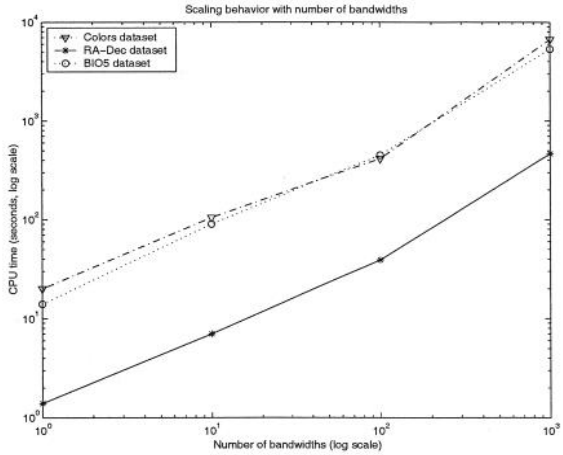

Figure 3.18: SCALING WITH NUMBER OF BANDWIDTHS: OTHER DATASETS.

§3.8 Related problems and approaches.

3.8.1 Kernel density estimation.

Though previous computational solutions have been proposed, their many inadequacies have still left the exhaustive method as a necessity for this task in general.

Grid-based approaches.

The idea of gridding (or 'binning') is to approximate the data by chopping each dimension into a fixed number of intervals M, then assigning the original data to neighboring grid points to obtain grid counts, representing the amount of data in its neighborhood. The kernel function is evaluated at grid points rather than actual points. The main problem with gridding is that the number of grid points required is M^D, exponential in the number of dimensions. The cost of estimating a density given N reference and query points is $O((M^D)^2)$, where presumably M is somehow set proportionally to N if accuracy is to be maintained (*e.g.* consider the fact that $h^* \propto N^{1/(D+4)}$ [Epa69, Sil86], *i.e.* the optimal bandwidth is dependent on the number of data), though principled guidance for choosing M is generally absent. (One would expect much of this computation to be essentially wasted, since in higher dimensions most of the grid cells are likely to be empty, unless only non-empty cells were kept in a linked list, say – something that was not originally proposed). Query points falling in the regions in-between are linearly interpolated, yielding another source of error. Tight bounds on the overall error are not provided by such a method, and indeed the error of such methods has been a point of concern in the literature.

The current workaround in general use is Scott's binning procedure [Sco85], which is the default used in the widespread S system.

FFT approach.

An elaboration uses the fast Fourier transform [Sil82, Sil86] on gridded data, performing discrete convolutions to combine the grid counts and kernel weights. It has been noted that the need for zero-packing causes significant loss of computational advantage for the FFT (which was designed for a regularly-spaced univariate time-varying signal, explaining its awkwardness in this context). Because a grid still underlies the method, it still suffers from similarly explosive scaling and error limitations. Its cost is $O(M^D \log(M^D))$.

For these obvious reasons, these grid-based methods originated in the univariate setting and are hardly considered for D higher than 2 or perhaps 3. In a thorough study by Wand [Wan94] of multivariate extensions of binning including the FFT extension, in tests of dimensionality up to 3 and data size up to 10,000 points, it was concluded empirically that this method can give speedups of at most 5 over the naive quadratic method, and in many cases incurs about the same computational cost as the quadratic method.

Forcing a calculation which is based on arbitrarily-placed points into a periodic representation has several severe effects, including significant artifacts occurring at the boundaries of the data and loss of accuracy caused by any unevenness in the distribution, as the underlying grid cannot be made adaptive.

Despite its numerious difficulties in this context, the FFT is the most-often-quoted solution to the KDE computational problem. The fact that it recurs as a prominent solution attempt in every one of our chapters is clearly a testament to its household familiarity rather than its appropriateness for N-body problems.

Data reduction approaches.

Viewing the N-body problem as the problem of 'having too much data', several proposals have been put forth for systematically deleting elements of the dataset. The approach by Girolami [GH03], which formulates KDE data elimination as an

SVM-like optimization represents a recent approach of this kind. The problem with these approaches is that a satisfactory criterion for performing such a modification of the data has not proposed, *e.g.* one which somehow guarantees preservation of the accuracy of the original estimator.

Multipole methods.

At least three attempts have been made to apply the multipole methods of computational physics to kernel density estimation [GS91, AED01, LHB+99]. The first suggestion of this kind was made by Greengard himself, with Strain, in 1989. The idea of doing this seems compelling, in light of the well-known success of these methods for physical problems which at least superficially appear similar. Unfortunately the multipole methods do not represent a general solution to the kernel density estimation problem for three key reasons — among the many choices for the KDE kernel, they are effectively only possible for the Gaussian kernel; they are exponential in the explicit dimension; and in their current form, multipole methods do not handle non-uniform distributions efficiently. The next chapter will discuss in much greater depth the properties of multipole methods for different kinds of N-body problems and how they might be usefully extended.

§3.9 Chapter summary.

It's now time to review what we've done in this chapter:

- **Function approximation with a finite-difference approach.** We developed an approach to function approximation which is designed to mesh naturally with the hierarchical geometric machinery of the last Chapter. This treats a key branch of N-body problems, namely those with continuous kernel functions. We in fact we did this without the need for derivatives. A full appreciation of the significance of this fact won't be possible until the next Chapter. We also showed that this provides a cleaner solution than the more simplistic monopole approach — which will again take on a new light in the next Chapter.
- **Kernel estimation problems and real solutions.** We reviewed the key theoretical aspects making kernel estimation problems foundational and the key practical constraints under which a solution attempt must operate efficiently. We then proceeded to develop a solution meeting these criteria for the first time, via a number of specialized techniques.
- **Multiple scales and optimal scale.** We will see that several of these techniques may also find application in other N-body problems, most notably the mechanisms for treating multiplicity in bandwidths and for locating the optimum bandwidth.
- **Up-down propagation.** We developed an additional methodology based on dynamic programming which streamlines the basic shattering approach.

Publications.

The KDE methodology shown here was developed over this series of papers:

- Gray, A. and Moore, A. *N-Body Problems in Statistical Learning*, NIPS 2001 (selected for oral presentation).
- Gray, A. and Moore, A. *Nonparametric Density Estimation: Toward Computational Tractability*, SIAM Data Mining 2003 (Best Algorithms Paper Award).
- Gray, A. and Moore, A. *Very Fast Kernel Density Estimation via Computational Geometry*, Joint Statistical Meeting 2003 (Statistical Computing Student Paper Prize).

- Gray, A. and Moore, A. *Rapid Evaluation of Multiple Density Models*, AI and Statistics 2003 (selected for oral presentation).

However, several elements still have not been published, and should all appear in:

- A journal version summarizing all the methodologies developed for this problem, for the statistics audience.

What's next?

The work we've done in this chapter turns out to have a second easy payoff, besides the trivial application to Nadaraya-Watson regression. By fortune it turns out that kernel density estimation and regression translate almost exactly to a core problem in physics, giving us an excuse to visit that world including the interesting approaches developed there.

4 N-Body Problems in Computational Physics

Function Approximation II: Multipole Methods.

When the doors of perception are cleansed, man will see things as they truly are, infinite. — William Blake (1757 - 1827).

WE SAW THE CONTINUUM reduced to the discrete in a certain sense in the last chapter. In this chapter we'll look at another major way in which the infinite is usefully replaced by the finite as a general strategy. One of the pillars of applied mathematics, and the focus of what is sometimes called 'approximation theory', is the idea of infinite series expansions. The most well-known and widely applicable of these arises from Taylor's Theorem. We shall explore the nature of its applicability to N-body problems, in the form of the famous 'multipole methods'. In particular, we will focus on the large subspace of N-body problems in computational physics which are *not* amenable to the multipole methods.

Agenda of this chapter.

The main problem we'll concentrate on is a fundamental tool arising in many sciences, the Lagrangian approach to computational fluid dynamics, called **smoothed particle hydrodynamics** (SPH). While the iron of the last chapter is hot, we will explain how extensions of the methods we used to solve the KDE problem can be used to create an equally powerful solution for SPH with several advantages over existing approaches. SPH will also serve as a vehicle for getting a feel for the sorts of constraints which are typical in N-body simulation methods for physics in general. Then we'll visit the approaches developed for physical N-body problems, discussing their scope and their relationship to the methodologies presented in this thesis.

§4.1 Computational fluid dynamics.

4.1.1 Dynamical simulation.

The problem we are considering in this chapter, simulation of fluid motion, lies at the more elaborate end of the spectrum of dynamical simulation. All simulations of physical dynamics simply operationalize Newton's law for a finite set of objects:

$$\forall q, \quad m_q \frac{\partial^2 \underline{x}_q}{\partial t^2} = -\nabla_q \Phi \tag{4.1}$$

where the force is obtained from the gradient of the potential function Φ, given some masses m_q for each of Q particles and their starting configuration of positions and velocities.

The general physical N-body simulation problem then, is to:

1. Compute the potential Φ at each point $\underline{x}_q$.
2. Compute the force $\nabla\Phi$ at each point $\underline{x}_q$.
3. Compute new positions $\underline{x}'_q$ for each point.

These three steps are done for each of a large number of finite timesteps t. The number and size of the timesteps depends on the system and phenomena being studied.

4.1.2 Navier-Stokes equations.

So far we have described the 'standard' N-body simulation problem with a simple force field (just gravity, say). This is the scenario considered by the Barnes-Hut [BH86] and Greengard-Rokhlin [GR87] papers, forexample. In this chapter we'll consider one of the most complex scenarios we could find, arising in computational fluid dynamics (CFD). The idea was to treat a sample task having as many complicating factors as possible — the CFD scenario we consider has all of the elements of a basic N-body simulation, plus a number of other ones.

The motion of a fluid in $\mathbb{R}^D$ is described by the Navier-Stokes equations [Fef00],

$$\frac{\partial}{\partial t}\underline{v}_d + \sum_{d'}^{D} \underline{v}_{d'} \frac{\partial \underline{v}_d}{\partial \underline{x}_{d'}} = V \sum_{d'}^{D} \underline{v}_{d'} \frac{\partial^2 \underline{v}_d}{\partial \underline{x}_{d'}^2} - \frac{\partial p}{\partial \underline{x}_d} + f_d(\underline{x}, t) \tag{4.2}$$

$$\sum_{d}^{D} \frac{\partial \underline{v}_d}{\partial \underline{x}_d} = 0, \tag{4.3}$$

with initial conditions $\underline{v}(\underline{x}, 0) = \underline{v}^0(\underline{x})$, where $\underline{x}$ denotes position, $\underline{v}$ denotes velocity, t denotes time, $f()$ is a given, externally-applied force such as gravity and V is the constant of viscosity.[1] The Euler equations correspond to the case where $\underline{v} = 0$. The unknown velocity and pressure are to be solved for.

They simply specify the basic constraints on fluid motion — the first equation is just Newton's law $f = ma$ for a fluid subject to an external force and to forces due to pressure and friction. The second equation just states that the fluid is incompressible, in this case. (Note that all of our discussion actually applies to both compressible and incompressible fluids.)

While essential in a huge number of applied physics and engineering problems from aircraft flight to combustion engine design, the Navier-Stokes equations are not well understood analytically. Even basic facts concerning the existence of smooth solutions are lacking. Understanding of analytical solutions of the Navier-Stokes equations remains in fact a major open problem of mathematics [Fef00]. However, we are always free to *run* the equations, *i.e.* simulate fluid systems according to the known equations, and observe the values we desire, such as the velocity and pressure. This is similar in end-result to the situation in the N-body problem we mentioned in 1.4.

4.1.3 The Lagrangian approach.

The *smoothed particle hydrodynamics* method, introduced in 1977 by [Luc77] and [GM77] is a *Lagrangian* simulation method, as opposed to an *Eulerian* method. This means simply that some representative particles are tracked through space, rather

1. It is sometimes written more succinctly using the divergence $div\underline{v} = \sum_d^D \frac{\partial}{\partial \underline{x}_d}$ and the Laplacian $\Delta\underline{v} = \sum_d^D \frac{\partial^2}{\partial \underline{x}_d^2}$.

than a grid being used, where grid cells become the new theoretical focus. This is the source of its advantages over competing methods and the reasons for its long-standing and growing popularity for fluid dynamics problems of all sorts.

Versus Eulerian.

Besides the computational advantages, which we'll describe as usual in the related work near the end of the chapter, the Lagrangian approach constitutes several *representational* advantages: it imposes no constraints on the geometry of the system or in how far it may evolve from its initial conditions, and simulation resolution naturally self-adapts as needed via the particle density. SPH is particularly suited to compressible fluids (*i.e.* gases and plasmas), where non-uniformity of the distribution is especially debilitating for grid-based methods. This general robustness makes SPH much more widely applicable than other methods.

Further, much of the complication incurred by grid-based representations is eliminated as a result of the fact that the method operationalizes the mathematical form of the governing equations in the most direct and intuitive way possible. Closing the intuition gap between the mathematics and the computational method is much of the reason SPH has found relatively widespread application.

4.1.4 Smoothed particle hydrodynamics.

As we've already seen, part of the SPH method is exactly a kernel density estimate for the density at each point:

$$\hat{p}(\underline{x}_r) = \sum_r \frac{1}{N_{\mathcal{R}}} m_r \phi(\frac{||\underline{x}_q - \underline{x}_r||}{h}) \tag{4.4}$$

with the inclusion of a weighting by each particle's mass m_r.

Finding values of quantities of interest such as the pressure or velocity is tantamount to kernel regression:

$$f(\underline{x}_q) = \sum_r \frac{1}{N_{\mathcal{R}}} m_r \frac{f(\underline{x}_r)}{\hat{p}(\underline{x}_r)} \phi(\frac{||\underline{x}_q - \underline{x}_r||}{h}) \tag{4.5}$$

As pointed out by Monaghan [Mon92], the kernel estimates inherit the differentiability properties of the kernel $\phi()$, so that as long as the kernel is differentiable, finding quantities such as the pressure gradient is easy and natural:

$$\nabla f(\underline{x}_q) = \sum_r \frac{1}{N_{\mathcal{R}}} m_r \frac{f(\underline{x}_r)}{\hat{p}(\underline{x}_r)} \nabla\phi(\frac{||\underline{x}_q - \underline{x}_r||}{h}) \tag{4.6}$$

though in practice a rearranged form of this equation is used.

Recall that these calculations are performed at each time step t.

Certain considerations entering the kernel choice arise from the equation of motion used in SPH, which arises from symmetrizing the pressure gradient term and adding an *artificial viscosity* term $\varphi()$:

$$\frac{d\underline{v}_q}{dt} = -\sum_r m_r \left(\frac{p_q}{\hat{p}_q^2} + \frac{p_r}{\hat{p}_r^2} + \varphi_{qr} \right) \nabla_r K_{qr} \tag{4.7}$$

where it is assumed that $K_{qr} = K_{rq}$. The artificial viscosity term is used to allow for entropy production by shocks.

4.1.5 Practical considerations and constraints.

The set of practical issues we must now consider differs and goes beyond that of kernel density estimation in several ways.

Low dimensionality D.

One important constraint that we can now relax is that of arbitary dimensionality. Except for the possibility of higher-dimensional spaces occurring in several proposed grand unification models of physics, we can assume the dimension is at most 3.

Modest accuracy requirements.

SPH applications typically do not require extremely high accuracy tolerances, generally focusing more on qualitative characteristics — hence the appropriateness of Barnes-Hut algorithms.

Optimal bandwidth h^*.

The meaning and effect of the bandwidth h in SPH is absolutely no different than it is for kernel estimation. The idea is that quantities which actually are defined on a much larger true number of particles approaching infinity are estimated using a smaller finite representative sample of particles, by using an estimate of the local density based on the finite sample. The extent to which the finite sample represents the larger number of particles well depends on the local scale h chosen, and some h^* gives the best possible representation.

Despite borrowing its basic concept from kernel estimation, the notion of the optimal bandwidth h^* has eluded the field of SPH, which seems to have forgotten or ignored the statistical origin of the ideas, or perhaps the fact that it has a mathematical basis (nonparametric estimation theory). Thus, having at least recognized the criticality of the issue, investigations by physicists on this question, *e.g.* [Ras99], continue without the benefit of decades of work in statistics on it. We have already incorporated the best existing statistical prescriptions for treating this question in our methodology in the last chapter.

Variable bandwidths h_r.

The idea of allowing a different bandwidth for each reference point corresponds to *variable-kernel* or 'adaptive-kernel' density estimation. Though proposed as early as [TS92], this has been practically pursued only in a relatively fledgling fashion in statistics. This is perhaps because the fixed-bandwidth scenario is deemed to pose significant difficulty alone. Nonetheless, it is intuitively clear that such a estimator is more powerful than a fixed-bandwidth estimator in general, and particularly when the data exhibits long tails, outliers, or regions which otherwise differ greatly in density, and empirically it has given good results in the literature. In the practice of SPH, it is the norm.

The idea of variable bandwidths is in fact taken slightly further in SPH, as we'll see in the kernel functions typically used.

Time-varying bandwidths $h_r(t)$.

Some formulations of SPH even use time-varying bandwidths, accounting for the fact that the distribution of points changes.

Kernel function choice $K()$.

Kernel symmetry. The kernel itself is typically defined to reflect both the bandwidth of the reference point and that of the query point [HK89]:

$$K_{qr} = \frac{1}{2}\left\{K\left(\frac{||\underline{x}_q - \underline{x}_r||}{h_q}\right) + K\left(\frac{||\underline{x}_q - \underline{x}_r||}{h_r}\right)\right\}, \tag{4.8}$$

which guarantees that $K_{qr} = K_{rq}$ even when the bandwidths of $\underline{x}_q$ and $\underline{x}_r$ differ.

Base kernel choice. The most common choice for the base kernel function is a form of spline kernel such as $\phi(\frac{\|\underline{x}_q - \underline{x}_r\|}{h}) = 1 - \frac{3}{2}q^2 + \frac{3}{4}q^3$ for $0 \leq \frac{\|\underline{x}_q - \underline{x}_r\|}{h} \leq 1$ and $\frac{1}{4}(2-q)^3$ for $1 \leq \frac{\|\underline{x}_q - \underline{x}_r\|}{h} \leq 2$, otherwise 0. The arguments supporting these choices include smoothness and statistical efficiency. In addition, kernels which are not smooth or do not have compact support have been shown to cause a number of problems in the SPH setting, including ringing, density overestimates, interpenetration of particles, and inability to model shocks accurately [MPT93], as well as computational efficiency. Recent versions of SPH have even proposed anisotropic kernels.

Artificial viscosity. The artificial viscosity function $\varphi()$ is generally also a function of distance, and can take a multitude of complex forms depending on the problem. It can be essentially thought of as another kernel function, so that the 'overall' kernel function is actually the product of $\varphi()$ and $\phi()$.

The most important thing to note about the kernel functions used in SPH is that they are not generally standard pdf's such as the Gaussian, and in fact are a matter of considerable art, as many aspects of the problem at hand are reflected in the crafting of the kernel function. This holds in particular for the artificial viscosity function. Another item of note is that the kernels used in SPH almost always have compact support. These facts eliminate the multipole methods from consideration for SPH, but represent no issue for the much more flexible finite-difference approach.

Gravity component.

Note that the equations of fluid motion contain a component due to gravity. This is handled in SPH by finding the gravitational component of the force with a separate algorithm, usually even implemented in a separate piece of software, specialized for gravity.

Dynamic update of S.

In a dynamical simulation the points move at every time step. In other words, in principle a new N-body problem must be solved at every time step. Fortunately, they generally move only slightly between time steps, so that the N-body problem at time t is very close to that at time $t-1$.

One clear upshot of this new dynamic aspect is that the tree structure S must be kept consistent with the points somehow.

Time-stepping.

In the most basic form of N-body simulation, the increment Δt between time steps is constant, and is the same for all points.

Adaptive time-stepping. The issue of selecting an efficient time-stepping strategy to minimize wasted computational effort while maintaining high accuracy can be viewed in the general setting of numerically integrating a time-varying equation. Thus well-known approaches such as Runge-Kutta, the predictor-corrector method, and so on, enter this potentially significant subproblem of N-body simulation. In fact, it has been argued that the issue of efficient time-stepping strategies holds the potential for the last orders-of-magnitude to be gained in the N-body simulation problem [LQR97]. In SPH a simple second-order explicit *leap-frog* method is almost always used [MPT93, FO02].

Interaction with N-body problem. To what extent does the time-stepping aspect of the simulation problem interact with the strict N-body problem that must be solved at each time step? In other words, can accounting for the dynamical aspect of the problem result in only needing to solve the N-body problem for a subset of the data? The idea of varying the time resolution for different query points, though intuitively very promising for reducing unnecessary computation, does not seem to

have entered the mainstream mindset concerning N-body problems, though it has been pursued seriously in the context of some specific physical N-body problems, for example in simulations of the heat equation [?] and in molecular dynamics [GHWS91, CHH85]. Instead, in standard practice the time-stepping problem is decoupled from the N-body problem. While we actually feel that substantial gains are to be made by coupling the problems, this will not be our focus and we will focus only on the N-body problem as standardly posed.

Our method works with any standard time-stepping scheme and can likely be modified for any coupled formulation of the N-body problem with no more difficulty than any other N-body approach for SPH.

Boundary conditions.

The principal weakness of SPH is actually a recurring theme of N-body approaches in general — near the boundary neighborhood relations are artificially distorted by the lack of points beyond the boundary. [MPT93] surveys a number of approaches for dealing with this, none of which affect the computational approach in any significant way.

§4.2 Extensions.

Now we show how an algorithm for SPH can be derived from the algorithm we developed for KDE in the last chapter.

4.2.1 Function estimation extensions.

Here we treat the aspects of SPH which are actually also variations of statistical kernel estimation methods.

Variable-bandwidth estimation.

Variable or data-dependent bandwidths represent a very minor change in the KDE algorithm, though at the cost of some efficiency. The minimum mass contribution of a reference node is computed using the smallest bandwidth in the reference node, and the maximum mass contribution is computed using the largest bandwidth.

This ensures that the bounds are valid, though large differences in the minimum and maximum bandwidths within a node will cause its pruning bounds to be loose. However in practice, there will be fairly strong spatial consistency in the bandwidths, *i.e.* nearby points will tend to have similar bandwidths, given a reasonable scheme for selecting the values for the data-dependent bandwidths.

Standard practice in SPH uses the heuristic of keeping the number of neighbors constant, *i.e.* each reference point's bandwidth is chosen to be the distance of its κ^{th} nearest neighbor. This corresponds roughly to schemes proposed in the statistics literature [TS92], though a cross-validation computation of some sort to choose the best value of κ should be performed before the simulation begins. We have already shown the dual-tree method for this sub-problem in Chapter 2. The total additional overhead of performing this during this simulation is considered below in 4.2.1.

Currently SPH practitioners choose global smoothing parameters arbitrarily [MPT93, FO02].

Simultaneous computation: regression.

Ideally, the regression step(s) of SPH could be performed simultaneously with the density estimation step, *i.e.* in the same node-node comparison, both quantities could be computed. Unfortunately, the value $f(\underline{x}_q)$ at time t depends on knowing the values $\hat{p}(\underline{x}_r)$ at time t for every point in a reference node. Thus we are forced, for a given time step t, to fully compute the density values for each point, then subsequently compute the regression values for each point. If the values $f(\underline{x}_q)$ at time t were redefined to be based on the values $\hat{p}(\underline{x}_r)$ at time $t-1$, we could eliminate this doubling of the computational effort. The corresponding loss of fidelity

should be relatively minor. To be clear, this represents a possible way to redefine the standard SPH updates, which is not necessary but allows a possible factor of 2 savings.

Time-varying bandwidths.

The inclusion of time-varying bandwidths is computationally insignificant *per se*, except for the issue of possibly re-estimating the optimal bandwidths, which again has more to do with the standard choices made in SPH than inherent computation. If the nearest-neighbor approach is followed, the bandwidths change as the neighborhood of each point changes over time. This means computing the all-κ-nearest-neighbor distance at each time step. Unfortunately this cannot be performed simultaneously with the regression or density estimations, since it is needed to determine the bandwidths upon which they are based. Again, simultaneity is possible if we choose to allow the bandwidths of time t to be based on the neighbor distances of time $t-1$. Since the neighbor distances should be changing very slightly between time steps, another option is to compute them only every τ timesteps.

4.2.2 Physical extensions.

Simultaneous computation: gravity.

At each time step t, we must compute the force on each point due to gravity (which has a contribution from each of the other points), in addition to the other SPH quantities. Because these values are not interdependent within the same time step, as was the case with the density and regression values, in this case we are able to compute gravity simultaneously with the other SPH quantities. Thus the need for a separate 'gravity solver' is removed. Such solvers are particularly wasteful if they employ expensive methods designed for very high accuracy (*i.e.* multipole methods). The fact that the accuracy required of the gravity component of likely to be on the same order as that required of the other quantities also suggests simultaneous computation.

For that matter, if the gravitational force is not treated with its own separate accuracy criterion, but lumped together with the total force, no disjunction need appear at all.

Force vectors.

In the KDE problem, only a scalar potential Φ is computed at each point. In a dynamical simulation the output also includes a force *vector* $\nabla\Phi$ for each point. This poses no additional problem beyond the fact that D values must be stored for each query point instead of one. (In fact $2D$ values are needed since we maintain upper and lower bounds.)

4.2.3 Dynamical extensions.

Dynamic tree updates.

In methods which use non-adaptive fixed-spacing structures, such as the original Barnes-Hut and Greengard-Rokhlin methods, maintainence of the values stored in the tree was a minor issue since each point can be hashed (whether or not it is called this) to its appropriate cell. Tree maintenance can thus be done in $O(N)$ time.

For our adaptive tree structures, the situation is more troublesome. Complete reconstruction of the tree at each timestep would cost us $O(NlogN)$, and as we saw in the empirical results for KDE, the cost of tree-building actually dominates that of the N-body computation itself.

One thing that comes to mind is the idea of dynamic data structures in the vein of AVL trees or splay trees [ST85], for which *kd*-tree variants exist, for example

[M. 91]. There has been a great deal of study of dynamic data structures and algorithms in computational geometry in general [CT92]. However, these designs are appropriate for the situation in which most of the data stays the same, and a relative few are added or deleted, or 'moved' by deletion followed by addition. This is not our situation, since all of the data move at each time step, in principle.

Minimal propagation method.

Fortunately, the movements of points between time steps is expected to be very slight, if reasonable simulation quality is to be maintained. Thus a very simple and cheap procedure can be employed to avoid reconstruction of the tree at every timestep, which we refer to as a *minimal propagation method* for tree updates.

In our finite-difference approach, only the boundaries of the nodes are important — no other quantities such as multipole moments, even centroids, are critical to the algorithm. It must be ensured, however, that points lie within the boundaries indicated by the nodes containing them, or else our approximation bounds will be invalid. Assuming a standard mechanism for traversing the leaves of the tree linearly, as in a *threaded* search tree (R-trees and B-trees are examples), we can simply check each point to see if it has moved outside its leaf node's boundary. If so, the boundary is appropriately updated. For those leaves which have changed, the boundary change is propagated upward to its parent. If the parent's boundary needs to be updated, the process is continued recursively, until the root is reached. In the low dimensionalities of this problem, only a small number of data points will trigger boundary changes, and further propagation becomes less and less likely as the change ascends to the root. Over a large number of timesteps, this process will tend to introduce overlap between the boundaries of nodes. Note that this causes no issue regarding correctness, only causing a decrease in efficiency with increasing amounts of overlap. Ball-trees in fact contains large amounts of overlap while remaining very robust in terms of efficiency. If desired, the tree can be reconstructed from scratch every τ time steps to correct for this effect.

We can now contrast the properties of our overall method to the state-of-the-art methods in SPH. However, to do so with maximum insight, it will be necessary to review in depth the non-trivial computational literature for physical N-body simulation problems.

§4.3 Multipole Methods

The term *multipole methods* is used in two different ways. The first sense refers to a very specific kind of function representation based on Taylor's Theorem. The second sense refers more generally to this function representation coupled with a certain data structure strategy and a certain approximation strategy (what we mean by these will be clear in a moment) to form a specific overall method for N-body problems. Examples are the original Barnes-Hut method and the original Greengard-Rokhlin method, which imply certain choices other than the use of the multipole representation *per se*.

Much confusion surrounds the multipole methods even within its own literature, despite the fact that they are so well-known. Part of the reason is due to the presentational choices made by Greengard and Rokhlin in their seminal paper, which presented the method in terms of spherical harmonics and complex analysis, rather than real numbers and Cartesian coordinates. They also chose a somewhat obtuse theorem-proof format for explaining the essential operations. The presentation which follows is my reconstruction of the multipole methods from scratch, in real numbers and Cartesian coordinates, and from the more informed computer science perspective of divide-and-conquer and data structures, where the discrete aspects of the methods are concerned.

4.3.1 Reference-side expansion.

The multipole method, from the point of view we can take after having developed the chapters so far, is little more than a double application of Taylor's Theorem to the kernel function, along with some extra thinking about how to use this kind of approximation within a hierarchical context.

Multipole expansion.

The terminology behind the *multipole expansion* [PP62, Bot73] is due to one of the main sources of the N-body problem, classical electromagnetism, in which the points are charged masses and the kernel function is the Coulomb potential. This classical setting is in fact the one considered by Greengard and Rokhlin. The multipole expansion is simply a Taylor expansion of the potential at a point $\underline{x}_q$ due to a point $\underline{x}_r$ about a third point $\underline{x}_R$. This corresponds to the point-node scenario shown in the figure, where $\underline{x}_R$ can be thought of as the center or some other reasonable representative of a node R. In terms of the distance $\delta_{qr} = \|\underline{x}_q - \underline{x}_r\|$, Taylor expansion of the potential function yields

$$\frac{1}{\delta_{qr}} = \left[\frac{1}{\delta}\right]_{\delta_{qR}} + \sum_{d}^{D} \delta_{rRd} \left[\frac{\partial}{\partial \underline{x}_{r\,d}} \frac{1}{\delta}\right]_{\delta_{qR}} + \frac{1}{2!} \sum_{d}^{D} \delta_{rRd} \sum_{d'}^{D} \delta_{rRd'} \left[\frac{\partial^2}{\partial \underline{x}_{r\,d} \partial \underline{x}_{r\,d'}} \frac{1}{\delta}\right]_{\delta_{qR}} + \ldots \tag{4.9}$$

The total Coulombic potential at $\underline{x}_q$ due to all the points in node R each having charge w_r is then

$$\begin{aligned}
\Phi(\underline{x}_q) &= \sum_{r}^{N_R} \frac{w_r}{\delta_{qr}} \\
&= \sum_{r}^{N_R} w_r \left\{ \frac{1}{\delta_{qR}} + \sum_{d}^{D} \delta_{rRd} \left[\frac{\partial}{\partial \underline{x}_{r\,d}} \frac{1}{\delta}\right]_{\delta_{qR}} \right. \\
&\qquad \left. + \frac{1}{2!} \sum_{d}^{D} \delta_{rRd} \sum_{d'}^{D} \delta_{rRd'} \left[\frac{\partial^2}{\partial \underline{x}_{r\,d} \partial \underline{x}_{r\,d'}} \frac{1}{\delta}\right]_{\delta_{qR}} + \ldots \right\} \\
&= \frac{1}{\delta_{qR}} \sum_{r}^{N_R} w_r + \sum_{d}^{D} \delta_{rRd} \left[\frac{\partial}{\partial \underline{x}_{r\,d}} \frac{1}{\delta}\right]_{\delta_{qR}} \sum_{r}^{N_R} w_r \delta_{rRd} \\
&\qquad + \frac{1}{2!} \sum_{d}^{D} \sum_{d'}^{D} \left[\frac{\partial^2}{\partial \underline{x}_{r\,d} \partial \underline{x}_{r\,d'}} \frac{1}{\delta}\right]_{\delta_{qR}} \sum_{r}^{N_R} w_r \delta_{rRd} \delta_{rRd'} + \ldots
\end{aligned} \tag{4.10}$$

Multipole 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:

$$\begin{aligned}
\Phi(\underline{x}_q) &= \frac{1}{\delta_{qR}} \alpha_R + \sum_{d}^{D} \delta_{rRd} \left[\frac{\partial}{\partial \underline{x}_{r\,d}} \frac{1}{\delta}\right]_{\delta_{qR}} \underline{\beta}_{Rd} \\
&\qquad + \frac{1}{2!} \sum_{d}^{D} \sum_{d'}^{D} \left[\frac{\partial^2}{\partial \underline{x}_{r\,d} \partial \underline{x}_{r\,d'}} \frac{1}{\delta}\right]_{\delta_{qR}} \underline{\gamma}_{Rdd'} + \ldots
\end{aligned} \tag{4.11}$$

α_R, $\underline{\beta}_R$, and $\underline{\gamma}_R$ are called the *monopole*, *dipole*, and *quadrupole* moments of the node R, respectively.[2] $\underline{\beta}_R$ is a vector, $\underline{\gamma}_R$ is a matrix, and so on into tensors.

Importantly, the multipole moments represent one way to *generalize the idea of the centroid approximation* we saw in the last chapter, answering a question we raised in the last chapter. We should keep in mind that there might or might not also be other ways as well, though no others have been proposed.

Derivatives.

An easily-overlooked but important point arises when actually taking the required derivatives. The 'direction' implicit in the distances are not interchangeable, *i.e.* it is not the case that $\delta_{qr} = \delta_{rq}$ for this purpose. As an example,

$$\begin{aligned}
\frac{\partial}{\partial \underline{x}_{r\,d}} \frac{1}{\delta} &= \frac{\partial}{\partial \underline{x}_{r\,d}} \frac{1}{[(\underline{x}_{q\,1} - \underline{x}_{r\,1})^2 + \ldots (\underline{x}_{q\,D} - \underline{x}_{r\,D})^2]^{1/2}} \\
&= \frac{\partial}{\partial \underline{x}_{r\,d}} [(\underline{x}_{q\,1} - \underline{x}_{r\,1})^2 + \ldots (\underline{x}_{q\,D} - \underline{x}_{r\,D})^2]^{-1/2} \\
&= -\frac{1}{2}[(\underline{x}_{q\,1} - \underline{x}_{r\,1})^2 + \ldots (\underline{x}_{q\,D} - \underline{x}_{r\,D})^2]^{-3/2} 2(\underline{x}_{q\,d} - \underline{x}_{r\,d})(-1) \\
&= \frac{1}{2}[\delta^2] 2\delta_{qr\,d} \\
&= \delta^{-3}\delta_{qr\,d} \\
&= \frac{\delta_{qr\,d}}{\delta^3} \qquad (4.12)
\end{aligned}$$

so that $\left[\frac{\partial}{\partial \underline{x}_{r\,d}} \frac{1}{\delta}\right]_{\delta_{qR}} = \frac{\delta_{qr\,d}}{\delta^3_{qR}}$.

Approximation.

The total potential at $\underline{x}_q$ is then approximated by truncating the series at some number of terms P, here $P = 2$:

$$\Phi(\underline{x}_q) \simeq \frac{1}{\delta_{qR}}\alpha_R + \frac{1}{\delta^3_{qR}} \sum_d^D \delta_{qR_d} \underline{\beta}_{Rd} + \frac{1}{2\delta^5_{qR}} \sum_d^D \sum_{d'}^D (3\delta_{qR_d}\delta_{qR_{d'}} - I_{d=d'}\delta^2_{qR})\underline{\gamma}_{R\,dd'} + \ldots \qquad (4.13)$$

The moments are functions only of the points in R and thus can be precomputed and stored in each node of the tree. The rest depends on the query point and thus must be computed at runtime. The complexity of approximation at runtime is $O(P^D)$. The *storage* cost due to the moments is also $O(P^D)$.

4.3.2 Query-side expansion.

So far we have a point-node approximation scheme. Now let's draw a more complete picture, in which $\underline{x}_q$ lives in a node Q, having center (or other representative) $\underline{x}_Q$, to obtain a node-node scheme.

The idea is to now approximate the potential at each $\underline{x}_q$ in terms of the representative $\underline{x}_Q$. This implies a Taylor approximation of the previous Taylor approximation. It should now be clear why we have called these approximations the *reference side* and *query-side* approximations, respectively.

2. We have written this in the form used by [Kut95] rather than that used by [DKG92], which we find clearer.

$$
\begin{aligned}
\Phi(\underline{x}_q) &= [\Phi(\underline{x})]_{\underline{x}_Q} + \sum_d^D \delta_{qQ\,d} \left[\frac{\partial}{\partial \underline{x}_{q\,d}} \Phi(\underline{x})\right]_{\underline{x}_Q} \\
&\quad + \frac{1}{2!} \sum_d^D \delta_{qQ\,d} \sum_{d'}^D \delta_{qQ\,d'} \left[\frac{\partial^2}{\partial \underline{x}_{q\,d} \partial \underline{x}_{q\,d'}} \Phi(\underline{x})\right]_{\underline{x}_Q} + \ldots \\
&= \left[\frac{1}{\delta_{\cdot R}} \alpha_R + \frac{1}{\delta_{\cdot R}^3} \sum_d^D \delta_{\cdot Rd} \underline{\beta}_{Rd} \right. \\
&\quad \left. + \frac{1}{2\delta_{\cdot R}^5} \sum_d^D \sum_{d'}^D (3\delta_{\cdot Rd}\delta_{\cdot Rd'} - I_{d=d'}\delta_{\cdot R}^2)\underline{\gamma}_{Rdd'} + \ldots \right]_{\underline{x}_Q} \\
&\quad + \sum_d^D \delta_{qQ\,d} \left[\frac{\partial}{\partial \underline{x}_{q\,d}} \left(\frac{1}{\delta_{\cdot R}} \alpha_R + \frac{1}{\delta_{\cdot R}^3} \sum_d^D \delta_{\cdot Rd} \underline{\beta}_{Rd} \right.\right. \\
&\quad \left.\left. + \frac{1}{2\delta_{\cdot R}^5} \sum_d^D \sum_{d'}^D (3\delta_{\cdot Rd}\delta_{\cdot Rd'} - I_{d=d'}\delta_{\cdot R}^2)\underline{\gamma}_{Rdd'} + \ldots \right)\right]_{\underline{x}_Q} \\
&\quad + \frac{1}{2!} \sum_d^D \delta_{qQ\,d} \sum_{d'}^D \delta_{qQ\,d'} \left[\frac{\partial^2}{\partial \underline{x}_{q\,d} \partial \underline{x}_{q\,d'}} \left(\frac{1}{\delta_{\cdot R}} \alpha_R + \frac{1}{\delta_{\cdot R}^3} \sum_d^D \delta_{\cdot Rd} \underline{\beta}_{Rd} \right.\right. \\
&\quad \left.\left. + \frac{1}{2\delta_{\cdot R}^5} \sum_d^D \sum_{d'}^D (3\delta_{\cdot Rd}\delta_{\cdot Rd'} - I_{d=d'}\delta_{\cdot R}^2)\underline{\gamma}_{Rdd'} + \ldots \right)\right]_{\underline{x}_Q} \qquad (4.14)
\end{aligned}
$$

By rearranging the terms, we can also view this as Taylor-expanding each of the terms of the reference-side expansion about $\underline{x}_Q$:

$$
\begin{aligned}
\Phi(\underline{x}_q) &= \left[\frac{1}{\delta_{\cdot R}} \alpha_R\right]_{\underline{x}_Q} + \sum_d^D \delta_{qQ\,d} \left[\frac{\partial}{\partial \underline{x}_{q\,d}} \left(\frac{1}{\delta_{\cdot R}} \alpha_R\right)\right]_{\underline{x}_Q} \\
&\quad + \frac{1}{2!} \sum_d^D \delta_{qQ\,d} \sum_{d'}^D \delta_{qQ\,d'} \left[\frac{\partial^2}{\partial \underline{x}_{q\,d} \partial \underline{x}_{q\,d'}} \left(\frac{1}{\delta_{\cdot R}} \alpha_R\right)\right]_{\underline{x}_Q} \\
&\quad + \left[\frac{1}{\delta_{\cdot R}^3} \sum_d^D \delta_{\cdot Rd} \underline{\beta}_{Rd}\right]_{\underline{x}_Q} + \sum_d^D \delta_{qQ\,d} \left[\frac{\partial}{\partial \underline{x}_{q\,d}} \left(\frac{1}{\delta_{\cdot R}^3} \sum_d^D \delta_{\cdot Rd} \underline{\beta}_{Rd}\right)\right]_{\underline{x}_Q} \\
&\quad + \frac{1}{2!} \sum_d^D \delta_{qQ\,d} \sum_{d'}^D \delta_{qQ\,d'} \left[\frac{\partial^2}{\partial \underline{x}_{q\,d} \partial \underline{x}_{q\,d'}} \left(\frac{1}{\delta_{\cdot R}^3} \sum_d^D \delta_{\cdot Rd} \underline{\beta}_{Rd}\right)\right]_{\underline{x}_Q} \\
&\quad + \left[\frac{1}{2\delta_{\cdot R}^5} \sum_d^D \sum_{d'}^D (3\delta_{\cdot Rd}\delta_{\cdot Rd'} - I_{d=d'}\delta_{\cdot R}^2)\underline{\gamma}_{Rdd'}\right]_{\underline{x}_Q} \\
&\quad + \sum_d^D \delta_{qQ\,d} \left[\frac{\partial}{\partial \underline{x}_{q\,d}} \left(\frac{1}{2\delta_{\cdot R}^5} \sum_d^D \sum_{d'}^D (3\delta_{\cdot Rd}\delta_{\cdot Rd'} - I_{d=d'}\delta_{\cdot R}^2)\underline{\gamma}_{Rdd'}\right)\right]_{\underline{x}_Q} \\
&\quad + \frac{1}{2!} \sum_d^D \delta_{qQ\,d} \sum_{d'}^D \delta_{qQ\,d'}
\end{aligned}
$$

$$\left[\frac{\partial^2}{\partial \underline{x}_{q\,d}\partial \underline{x}_{q\,d'}}\left(\frac{1}{2\delta^5_{\cdot R}}\sum_d^D\sum_{d'}^D(3\delta_{\cdot Rd}\delta_{\cdot Rd'}-I_{d=d'}\delta^2_{\cdot R})\underline{\gamma}_{Rdd'}\right)\right]_{\underline{x}_Q}+\ldots \tag{4.15}$$

Rather than explain it this way, the Greengard-Rokhlin paper presents the query-side expansion as one of a number of 'translation operators' for 'converting expansions into other expansions' in the form of three theorems.

Conceptual importance.

It should be understood that *the idea of query-side expansion is the main conceptual breakthrough of the Greengard-Rokhlin paper* [GR87]. It is not the node-node idea *per se*, as Appel [App81, App85] (a computer scientist) had introduced that idea long before, though in a confusing fashion that effectively left it by the wayside. It is not the idea of multipole expansion, as earlier grid-based methods had made use of that idea. It is not the fact that the tree structure used in the Greengard-Rokhlin method is of a more primitive fixed-spacing type. And it is not the idea of 'working with the field rather than the interactions', one of the more groping attempts by some authors to somehow place the Greengard-Rokhlin method with respect to the other more intuitive N-body approaches such as Barnes-Hut.

4.3.3 Gaussian case.

There are other things to be considered to arrive at a complete method for solving an N-body problem, but first let's visit the case of the Gaussian kernel. The Gaussian is of some interest to us because of its wide use, particularly in statistics. While not always the best choice for an application, it is often at least a good thing to try and tends to be relatively robust for many uses. For example, an algorithm for kernel density estimation which only works with the Gaussian kernel would be annoying and even unworkable for many applications, but for many users with generic applications for which the density estimation component need not be particularly refined, restriction to the Gaussian case could be tolerated. This is similarly the case for many other statistical inference methods.

The Gaussian is essentially the only other major type of function beyond the generalized Coulombic potential, $\frac{1}{\delta^\alpha}$, for which multipole methods exist (we will discuss the full scope of multipole methods in more depth soon). Unfortunately, the Gaussian case of the 'fast multipole method' was published under the name 'fast Gauss transform' [GS91], and the link between the two only quickly mentioned. The approach is explained as an application of Hermite polynomials, which in addition to adding unnecessary obfuscation and little additional insight, steers attention away from the fact that the fast Gauss transform is in fact simply the fast multipole method where the kernel function is the Gaussian.

Reference-side Taylor expansion of the Gaussian yields

$$\begin{aligned}
e^{-\delta_{qr}/2h^2} &= \left[e^{-\delta^2/2h^2}\right]_{\delta_{qR}}+\sum_d^D\delta_{rRd}\left[\frac{\partial}{\partial \underline{x}_{r\,d}}e^{-\delta^2/2h^2}\right]_{\delta_{qR}} \\
&\quad +\frac{1}{2!}\sum_d^D\delta_{rRd}\sum_{d'}^D\delta_{rRd'}\left[\frac{\partial^2}{\partial \underline{x}_{r\,d}\partial \underline{x}_{r\,d'}}e^{-\delta^2/2h^2}\right]_{\delta_{qR}}+\ldots \\
&= \left[e^{-\delta/2h^2}\right]_{\delta_{qR}}+\sum_d^D\delta_{rRd}\left[\frac{\delta_{qr\,d}}{h^2}e^{-\delta^2/2h^2}\right]_{\delta_{qR}}
\end{aligned}$$

$$+\frac{1}{2!}\sum_{d}^{D}\delta_{rRd}\sum_{d'}^{D}\delta_{rRd'}\left[\left(\frac{\delta_{qr\,d}\delta_{qr\,d'}}{h^4}-\frac{I_{d=d'}}{h^2}\right)e^{-\delta/2h^2}\right]_{\delta_{qR}} \quad (4.16)$$

Note the appearance of the Gaussian function again in every term. Continuing onward as before, we can obtain an expression for the total potential $\Phi(\underline{x}_q) = \sum_r^{N_R} w_r e^{-\delta_{qr}^2/2h^2}$, from which we'll be able to extract pre-computable multipole moments.

Hermite polynomial perspective.

An alternative form of this expression is the one used by Greengard and Strain for the total potential, expanding about $\underline{x}_R$:

$$\Phi(\underline{x}) = \sum_{\rho} C_\rho e^{-\delta/2h^2} H_\rho\left(\frac{\|\underline{x}-\underline{x}_R\|}{\sqrt{2}h}\right) \quad (4.17)$$

where H_ρ are the Hermite polynomials [AS72] where the coefficients C_ρ are given by

$$C_\rho = \frac{1}{\rho!}\sum_{r}^{N_R} w_r\left(\frac{\|\underline{x}_r-\underline{x}_R\|}{\sqrt{2}h}\right)^\rho \quad (4.18)$$

and confusingly, ρ represents a *multi-index*, a notational compression of the sums indexed over the dimensions arising from the Taylor expansion. While perhaps useful in approximation theory, its use here is somewhat obfuscating when trying to understand the mapping to implementation.

The main point of this discussion is to make it clear that when we discuss properties of the 'fast multipole method', the 'fast gauss transform' has those same properties, because they are the same method.[3]

4.3.4 Computing the moments.

Again using the language of 'translating expansions', and referring to the multipole moments interchangably with expansions, the Greengard-Rokhlin paper describes how the moments can be 'shifted' from leaf centers to the center of their parent, and so on recursively. From our point of view this is simply another kind of composable 'cached sufficient statistic', which is built bottom-up during tree construction.

For example for the monopole moments at the leaves, $\alpha_R = \sum_r w_r$, and for non-leaves

$$\begin{aligned}\alpha_R &= \sum_k\sum_{r\in R_k} w_r \\ &= \sum_k \alpha_{R_k}.\end{aligned} \quad (4.19)$$

For the dipole moments at the leaves, $\underline{\beta}_{Rd} = \sum_r w_r\delta_{rRd}$, and for non-leaves

$$\underline{\beta}_{Rd} = \sum_k\sum_{r\in R_k} w_r(\delta_{rR_k\,d}+\delta_{RR_k\,d})$$

3. Except that in [GS91], the tree structure was omitted, and only the function approximation was described. The other slight practical difference is that no 'well-separated' constraint is needed for the approximation bounds to hold, for the particular bounds developed for the Gaussian case.

$$= \sum_k \left\{ \sum_{r \in R_k} w_r \delta_{r R_k d} + \sum_{r \in R_k} w_r \delta_{R R_k d} \right\}$$
$$= \sum_k \left\{ \underline{\beta}_{R_k d} + \alpha_{R_k} \delta_{R R_k d} \right\} \quad (4.20)$$

§4.4 Variations and new possiblities.

Now we'll take a look at the main variations that have been proposed for how to use the basic idea of multipole approximation. This will allow us to propose improvements to current uses of the multipole idea.

The chief benefit of multipole methods is the gain in accuracy. In particular, the first handful of additional moments beyond the first yield a high gain in accuracy. However, they face a number of practical limitations which limit their scope of applicability and motivate improvements.

4.4.1 Practical limitations of multipole methods: applicability.

We are now in the world of analytic approaches, which in contrast to the finite-difference approach of the last chapter, attempt to take advantage of analytic information of the kernel function (*i.e.* derivatives). While there are advantages to be gained (namely higher accuracy in low dimensions as we'll see), this is what makes analytic approaches fairly narrow in scope — essentially each kernel function has a different set of properties, so that an analytic approach for a given kernel function generally does not transfer to a different kernel function. The fast multipole method was developed for the Coulombic potential, and though the idea of Taylor approximation would seem general enough, in practice the machinery has not been developed for any other class of functions except for the Gaussian, for reasons we'll discuss.

Useful differentiability.

Use of the multipole method with order P requires that the kernel function have P useful derivatives. A large number of common kernel functions are low-order polynomial in nature, thus having few non-zero derivatives. Multiple methods cannot achieve arbitrary accuracy unless P is allowed to be arbitrarily large. Furthermore a large number of kernels which are desirable or even strictly necessary for certain types of problems have compact support or are discontinuous. The Coulombic potential and the Gaussian both happen to be analytic functions which have an infinite number of non-zero derivatives. In fact they seem to be rather rare in this regard.

New mathematics for each kernel.

There are no tight general bounds on the error of a Taylor approximation. The mean value theorem can always be used, but generally does not result in a useful bound. Thus new bounds must be derived for each new kernel function to be used. For example, the central contribution of the paper introducing the specialization of the fast multipole method to the Gaussian case [GS91] is the bound on the error. This necessity for new mathematics severely limits the application of multipole methods to new problems.

Opaqueness of theory and implementation.

We have done our best in the above explication to demystify and deconvolute the ideas behind the multipole methods. However, existing published explanations which offer intuition, clarity, and details at all levels, from theory to implementation, seem nowhere to be found. Confusions about multipole methods in the literature are easy to find, as we've already touched upon. The result seems to be that a fairly small number of groups have specialized in the complex art of multipole methods, rather than the multipole methods representing a well-understood standard tool that

can be pulled off the shelf easily for new problems, as for example the hundreds of computational methods each compactly summarized in Numerical Recipes. There is even a certain thread of research on 'multipole methods without multipoles' [And92], at least partially motivated by a desire for greater simplicity and clarity.

The relative opaqueness surrounding the multipole methods, despite the inherent simplicity of the idea, is our guess as to why this technology has had limited impact beyond the specific problems it was originally demonstrated for. To date, these methods do not seem to have been applied above 3 dimensions or outside of physics, with the exception of [AED01, LHB+99], though with limited success.

The simplest of the multipole methods, Barnes-Hut, is much better in this regard than the Greengard-Rokhlin method. This is because the entire multipole expansion part of the story need never enter. It was originally described in only the monopole form, which is just the center of mass. At this level it is quite simple and intuitive, but not $O(N)$, begging the question of whether a linear-time method might exist which is also easy to understand and adapt to new problems.

4.4.2 Practical limitations of multipole methods: efficiency.

Curse of dimensionality.

The basic nature of the multipole expansion in arbitrary dimensions unavoidably enforces an explicit curse of dimensionality. Since P tends to be fairly large, the curse is further exacerbated, limiting the utility of the multipole idea to low-dimensional problems. Fortunately physical N-body problems fit the bill nicely, hence their demonstrated utility in this area.

Cost of approximation.

Even in low dimension, the $O(P^D)$ constant is significant. Its size in fact defeats the complexity advantage of the Greengard-Rokhlin method over Barnes-Hut until N becomes quite large [BN97]. This fact has been another obstacle to the use of the Greengard-Rokhlin approach for all but the largest problems, even in the limited regime for which it is best-suited.

Uneven distributions.

The Greengard-Rokhlin method was analyzed for the uniform distribution of data, and it was empirically demonstrated on the uniform distribution. For uneven distributions the performance of the method degrades significantly. The Barnes-Hut method handles non-uniformity more robustly, but still less so than what is clearly possible. Two main sources of non-adaptivity accounting for these properties will be discussed shortly.

Tweak parameters.

Both the Barnes-Hut and Greengard-Rokhlin methods effectively have tweak parameters. The net effect of this in practice is that optimal performance might not be achieved if the optimal setting tweak parameters are not found somehow by the user. The user may choose to spend a lot of time finding efficient settings, but then this has to be accounted for a loss of *human* efficiency, which is ultimately most important.

4.4.3 Data structure strategies.

Fixed structures.

The Barnes-Hut and Greengard-Rokhlin approaches, the two primary approaches used for N-body problems, both used fixed hierarchical grid structures, with fixed-spacing cells, such as oct-trees and quad-trees. The main inspiration for this seems to be that it makes the methods easier to analyze and think about; Appel's approach

was actually very much an adaptive-tree approach, but was perhaps too much to swallow at the time, leading to the counter-reaction of Barnes-Hut, a much more naive algorithm.

There is also an advantage in the cost of tree building. The fixed structures are trivial to build, and are thus rebuilt at each time step. Note that by 'tree-building', we include the process of processing the 'cached sufficient statistics' in nodes, such as the multipole moments.

Adaptive structures, and the WSPD.

However, it is clear to the intuition that adaptive trees will perform better when the distribution is non-uniform. This has also been verified experimentally, as many researchers have made this observation. Computer scientists, in particular, seem to suggest this as their first reaction upon exposure to the current methods for the N-body problem.

The most well-known of these, at least in computer science, is represented by the *well-separated pair decomposition* method of Callahan and Kosaraju [Cal95]. This was a proposal to allow the use of adaptive *kd*-trees, under the name of 'fair-split trees', rather than oct-trees or quad-trees. The idea was to compute a list of all the pairs meeting the 'well-separated' criterion. The list of all such pairs, called the 'well-separated pair decomposition' (WSPD), is then passed to an N-body solver which would use it to decide which nodes to approximate by multipole expansion, the rest being handled directly. The cost of computing the WSPD was shown to be worst-case $O(N \log N)$, using $O(N)$ storage. Interestingly, the WSPD algorithm is exactly an instance of the recursive dual-tree algorithms advocated in this thesis.

Unfortunately, the authors seem to have never implemented the algorithm in an N-body solver, nor has any other author apparently been compelled to use it in its intended setting. The critical question is thus left unanswered by the WSPD, which is how the tree will be efficiently updated or constructed from time step to time step. This question *must* be answered by any attempt to apply adaptive trees to the dynamic simulation setting. Despite this, computer scientists, perhaps either convinced of or simply wishful for the relevance of computer science to a famous problem in the physical sciences, continue to publish proclamations that the WSPD has solved this problem.

Given the perspective we have built up so far, on-the-fly computation of well-separatedness is trivial. We just need to check the well-separatedness criterion at the beginning of each node-node comparison. (If it weren't so trivial, we could give it a fancy appelation like 'simultaneous computation of well-separatedness'.) Thus there is no need to compute the full 'well-separated pair decomposition' as a separate data structure requiring a whole separate computation.

To date, adaptive trees have not yet entered N-body solvers except possibly for limited cases, *e.g.* [NT94]. Part of the reason for this is perhaps that the cost of updating the tree does not seem to have been seriously treated. For example, with the introduction of a simple scheme for tree-updates such as the one we gave, preferably in addition to skipping the construction of an unnecessary separate data structure, the WSPD authors would have actually given a solution to this problem.

4.4.4 Adaptive multipole extension.

Is it possible to extend our algorithms to use the multipole approximation instead of the finite-difference approximation (for problems for which it is appropriate)? The answer is yes, but we must consider some different cases.

Single-tree adaptive multipole.

For the single-tree case, the extension is straightforward. The bounds on the reference-side multipole approximation error simply replace the simpler finite-difference

bounds.

Dual-tree adaptive multipole I.

The situation is slightly more difficult in the dual-tree case. To maintain our full error-control behavior, we need bounds on the minimum and maximum values that the contribution from the reference node could be for any point in a query node, computable *without* running over all the points in the query ndoe. The query-side multipole approximation error bounds merely give us the width of the bounds, not the center. We cannot find the absolute maximum potential values without knowing the centers. A fallback position is to simply use a large value of P which absolutely bounds the error given by any reference node and knowledge of the maximum number of reference nodes that any query node could enter to obtain a loose bound on the overall error. This is exactly what the Greengard-Rokhlin method effectively does, thus obtaining a weak form of error control.

Dual-tree adaptive multipole II.

A possible way out is to optimize over the query region to find the points yielding minimum and maximum contribution from the reference node. This sets up a nonlinear optimization over the query-side expansion which must be solved at each node-node comparison. While perhaps sounding expensive, this optimization would be contrained by the query node region, which in the hyperrectangle, might be possible relatively efficiently, as in the similar case with [Moo99]. This suggestion must be considered a mere possibility rather than a prescription as it may not be feasible or efficient at this speculative stage.

4.4.5 Approximation strategies.

Direct accuracy specification.

The most desirable state of affairs for any algorithm which makes approximations is that the user is able to specify an error tolerance (which resumably affects the algorithm's cost in some inverse proportion) and the algorithm performs as little work as possible while achieving that minimal accuracy requirement. This simply says that there should ideally be an explicit way for the user to specify a desired point along some known performance-error tradeoff curve. Very few algorithms provide this capability in reality. The connection between error and performance is usually through some auxiliary parameter, often multiple parameters, of the algorithm which are related to those quantities in an unknown, often nonlinear manner.

The Barnes-Hut algorithm is an example of this. Whether a node is pruned or recursed upon is determined by a user-chosen threshold τ on the 'opening angle'. This parameter has no direct known connection with either the actual maximum error or the actual runtime of the algorithm, and thus the user is left with the task of experimentation, specialized to his/her data and other variables, for values of τ achieving the performance-error tradeoff point desired.

The Greengard-Rokhlin algorithm does better in this regard, but is still a method for which direct accuracy specification is not possible. Though a relationship between the error and the runtime is known, in practice the tradeoff point is chosen by choosing the multipole expansion order P by experimentation. This is because the bounds are several orders of magnitude too loose in practice, *i.e.* the algorithm performs much more work (and achieves much higher accuracy) than the minimum needed to achieve the accuracy specifed by the bounds.

By contrast, the finite-difference approach we developed allows direct error control.

Data-adaptive accuracy.

By 'data-adaptive accuracy control' we mean a mechanism for choosing the amount of work to do (*i.e.* degree of approximation to make) based on the data being dealt with at the moment a point-node or node-node comparison is being made.

Barnes-Hut attempts to do this with its opening-angle heuristic. However as we have said, this heuristic is disconnected (at least in a known fashion) to the error made at that step. It also has no connection to the actual shape of the kernel function. The multipole order P is also the same for all nodes. A very recent paper [?] presents an approach allowing the selection of P adaptively at each point-node comparison, and shows that this results in speedups up to 50% over the normal method.

The Greengard-Rokhlin method is semi-blind in this respect. The same value of P is used no matter what the nodes are. However, this is treated to some extent by allowing different node sizes to be compared, larger ones at further fixed distances. This results in a limited amount of adaptivity. To counter-balance this, P is often set to a fairly high value. Published values are typically 6 or 8, and range up to 20.

Our finite-difference method is fully adaptive in this regard - the pruning decision is based on bounds on the actual error incurred at each node-node comparison.

4.4.6 Other analytic expansions.

There are of course a number of other asymptotic expansions which have found use in various parts of applied mathematics, and some of these have also been used in place of Taylor expansion, embedded in the same basic algorithms, to form alternative types of multipole-like methods. Examples include [And92, BN98]. In the cases where the same kernel function is treated, a convincing performance advantage has not been shown so far — though it is conceivable that some series might converge faster than others for the same function. However, it will certainly be the case that different types of expansions will be more or less useful or applicable to different types of kernel functions. Nonetheless, virtually all of the properties of the overall method will remain the same regardless of the kind of expansion used.

4.4.7 Relationships.

Now that we've identified the primary variables, or axes along which to place the variations on multipole methods which exist, we can consruct a matrix of relationships depicting the relationships among them.

				Multipole	Derivative-free
Node-node	Adaptive	Direct error-control	Fully error-adaptive and anytime	*Dual-ASPT multipole with min. prop. and query bounds opt.*	*Dual-ASPT finite-diff. with min. prop.*
		Weak error-control	Weakly error-adaptive	*Dual-ASPT multipole with min. prop.*	
		No error control	Somewhat error-adaptive	*Appel multipole [App85]*	*Appel monopole [App85]*
	Fixed	Weak error-control	Weakly error-adaptive	*Greengard-Rokhlin multipole [GR87]*	
Point-node	Adaptive	Direct error-control	Fully error-adaptive and anytime	*Single-ASPT multipole with min. prop.*	*Single-ASPT finite-diff. with min. prop.*
	Fixed	Weak error-control	Somewhat error-adaptive	*Barnes-Hut multipole [BH86]*	
		No error control			*Barnes-Hut monopole [BH86]*

Figure 4.1: RELATIONSHIPS BETWEEN N-BODY SOLVERS.

The single- and dual-ASPT (adaptive space-partitioning tree) approaches are the ones now made possible by the techniques we've designed. The 'multipole' designation refers to the version of a method in which multipole expansion is performed rather than another method of function approximation such as monopole approximation or finite-differencing.

§4.5 Related problems and approaches.

Let's review the computational alternatives for the SPH problem.

4.5.1 Grid methods

Grid methods are the competitor to the Lagrangian method as represented by SPH. The comparison between the two was discussed in **??**. Grid-based methods for SPH include the particle-mesh and particle-in-cell methods [HE88], the pressure method, and the marker-and-cell method [Ves94]. It must be noted the choice of a grid-based or Eulerian method is more than simply a computational one; the theoretical properties and overall applicability of the method for different fluid dynamics problems changes, so that it is also a change of *model* in some sense.

4.5.2 Ewald Sums

Ewald sums [Ewa21] represent the FFT approach to the problem, and were proposed in the context of electrostatics problems. It has all the disadvantages of FFT methods which we have already discussed.

4.5.3 The Greengard-Rokhlin Algorithm

We have already discussed the Greengard-Rokhlin algorithm, aka the 'fast multipole method'. It is not applicable in the case of SPH due to the non-arbitrarily-differentiable nature of the kernels used in SPH. Furthermore in SPH the kernel function is subject to much change, particularly the artificial viscosity part of the kernel, so that even if multipole approximation to a high order were possible, the sheer mathematical overhead of multipole methods would make them very painful to use in SPH. The Greengard-Rokhlin algorithm shines best in a low-dimensional, high-accuracy setting. High accuracy is generally not required in the SPH setting.

4.5.4 The Barnes-Hut Algorithm

The state-of-the-art for SPH is the Barnes-Hut algorithm, as exemplified by the TreeSPH code [HK89]. Our method significantly improves upon Barnes-Hut by providing error control and adaptivity. Adaptivity should aid efficiency significantly in non-uniform distributions. Furthermore the dual-tree nature of our method reduces the complexity from $O(N \log N)$ to $O(N)$. Like Barnes-Hut it is also easy to implement.

§4.6 Chapter summary

Let's take a look at what we've done:

- **Function approximation with analytic methods.** We reviewed in-depth the well-known approach to the function approximation sub-problem based on Taylor expansion that was introduced for Coulombic N-body problems. This yields two main insights — the main difficulties which prevent the application of this approach to many problems outside the low-dimensional physics context, and suggestions for how to augment currently used multipole-based algorithms using the geometric insights gained from our shattering perspective.

- **Computational fluid dynamics problems and real solutions.** We reviewed the key theoretical aspects making the smoothed particle hydrodynamics approach fundamental and the key practical constraints under which a solution attempt must operate efficiently. We then developed a solution meeting these criteria for the first time, via a set of modifications of the machinery we developed in the last chapter for the new aspects of this problem. As a side effect we have in fact provided the first linear-time method for SPH. As a bonus, the mechanisms we have designed within our approach for efficient bandwidth selection can be transferred to SPH, which currently lacks any rigorous methodology for bandwidth selection.

Publications.

This work should be made available in the form of at least the following publications:

- An article presenting the dual-tree finite-difference method using minimal propagation for the SPH problem.
- Perhaps concomitant with the previous article should be one introducing the much-needed theoretical and practical machinery of optimal nonparametric estimation to the SPH world.

What's next?

This concludes our examination of pair-wise N-body problems, including the classical ones from physics. Our in-depth discussion of analytic approximation methods could be seen as asking what we can do in a more limited case than usual (the case where the dimension is fixed to be very low). In the next chapter we'll consider a much harder case, and one which has not been previously considered, that of n-tuples instead of just pairs.

5 N-Body Problems in Computational Morphology

Geometric Shattering II: n-tuples and Monte Carlo.

Each success only buys an admission ticket to a more difficult problem.
— Henry Kissinger.

SOME PERSPECTIVE is always to be gained by reaching higher and beyond. Though the pair-wise N-body problems we have dealt with thus far are difficult enough to satisfactorily grind computers to a halt in their own right, reaching higher in tuple order makes us really appreciate how lucky we've been in the previous chapters. It is difficult to even conceive of an approach to a problem which seems to demand consideration of all $O(N^n)$ n-tuples of points.

Agenda of this chapter.

In this chapter we'll how the shattering concept generalizes naturally to any tuple order n. We'll also examine Monte Carlo methods, which on pause one might expect to appear in a story about N-body problems. Monte Carlo approaches represent in some sense the final resort — when the problem to be solved is important enough to persist, even enough to make less-than-certain answers acceptable. It also so happens that the central Markov chain Monte Carlo method, or the Metropolis method, was originally developed for N-body problems in statistical physics which we have already seen. Our example task will be the *n-point correlation functions*, whose computational solution turns out to be of pivotal importance in modern cosmology.

§5.1 Point processes.

Point processes are stochastic processes whose realizations consist of point events in space (or the special case of time). Statistical inferences about patterns of points thus concern this fundamental part of statistics.

Poisson processes.

The *Poisson process* is the most basic and important point process model. It is the model for a completely 'random' distribution of points in space, without any interaction. It can be thought of as the spatial analog to independent observations. It is used theoretically as a building block for constructing other processes, and practically as the baseline distribution for testing for or characterizing various properties of observed datasets.

It is standardly defined by either or both of these properies [Rip81, MS00]:

- The number of points $N(A)$ falling within any set A has a Poisson distribution with mean $\lambda V(A)$, where λ is a constant called the *intensity* and $V(A)$ is the volume of A.

- The number of points falling in disjoint sets A_1 and A_2, *i.e.* $N(A_1)$ and $N(A_2)$, are independent.

We have actually defined the standard Poisson process which is *homogeneous* (or 'stationary'), *i.e.* λ is invariant to translation, and *isotropic*, *i.e.* λ is invariant to rotation.

Moments of counts.

If λ, the mean count density, corresponds to the first moment of the counts from a Poisson distribution, *i.e.* $E[N(A)]$, what about the higher moments? The *2-point correlation* is a statistic related to the second moment of Poisson counts, $E[N(A_1)N(A_2)]$. It can be defined in a number of slightly different ways, but we'll use the most standard definition used in astrophysics [Pee80], which is in terms of the joint probability of finding a point in both of the volume elements dV_q and dV_r at separation δ_{qr}:

$$dP = \lambda^2 dV_q dV_r [1 + \xi(\delta_{qr})] \tag{5.1}$$

where $\xi(\delta_{qr})$ is the *2-point correlation function* of the pairwise separation δ_{qr}. Further discussion of the derivation of moments of Poisson counts can be found in [Rip81].

$\xi()$ then, represents an amount of deviation from the Poisson. Since $\xi()$ is a function of the distance δ_{qr}, we sometimes refer to the 2-point correlation *function*.

5.1.1 The n-point correlation function.

The n-point correlation corresponds to the n^{th} moment of Poisson counts. For example the joint probability of finding points in each of the three volume elements dV_q, dV_r and dV_s is given by

$$dP = \lambda^3 dV_q dV_r dV_s [1 + \xi(\delta_{qr}) + \xi(\delta_{rs}) + \xi(\delta_{sq}) + \zeta(\delta_{qr}, \delta_{rs}, \delta_{sq})] \tag{5.2}$$

where δ_{qr}, δ_{rs}, and δ_{sq} are the sides of the triangle defined by the three points $\underline{x}_q$, $\underline{x}_r$, and $\underline{x}_s$. $\zeta()$ is called the *reduced* 3-point correlation function. In general we refer to this quantity in place of the full correlation function since it is what we need to concern ourselves with computationally.

The expressions for the higher correlation functions become increasingly unwieldy, as does theoretical and practical understanding of how to use them. Discussion up to the 4-point correlation can be found in [Pee80].

The full distribution of counts is given by an infinite sum of all the higher-order correlation functions, *i.e.* $n = 1$ to ∞. Unfortunately, White [Whi79] shows that on practically all scales except the tiniest, the distribution depends significantly upon the higher-order functions and cannot be well-approximated using only the lower-order terms. Importantly, the higher-order terms are also necessary for characterizing the variance, or error, of the lower-order terms.

A foundational theory for spatial statistics.

The hierarchy of n-point correlation functions forms the foundation of the theory of point processes. Various formalizations of spatial statistics can be unified within this framework [DVJ72, Rip76].

The n-point correlation functions can be used to derive and understand basic inferences about points in space. Examples include characterizing the distribution of clusters of points [Whi79], the distribution of nearest-neighbor distances and the distribution of counts in cells [Pee80], the probability of a void of a certain size between points [BF96], the structure of filaments [Fry86], and many others.

Importance in cosmology.

The n-point correlation function take center-stage in astrophysics for several reasons. One is that the types of spatial statistics inferences for which we gave examples occur in the daily business of working with astronomical observations.

However, the main reason lies at the core of cosmology itself. The n-point functions are used to compare different cosmological models, by measuring the similarity between simulated data generated by a given model and observed data. Typically the parameters which lie inside a model are estimated using such a statistical methodology. Further, comparison between different *types* of models can be done in this fashion.

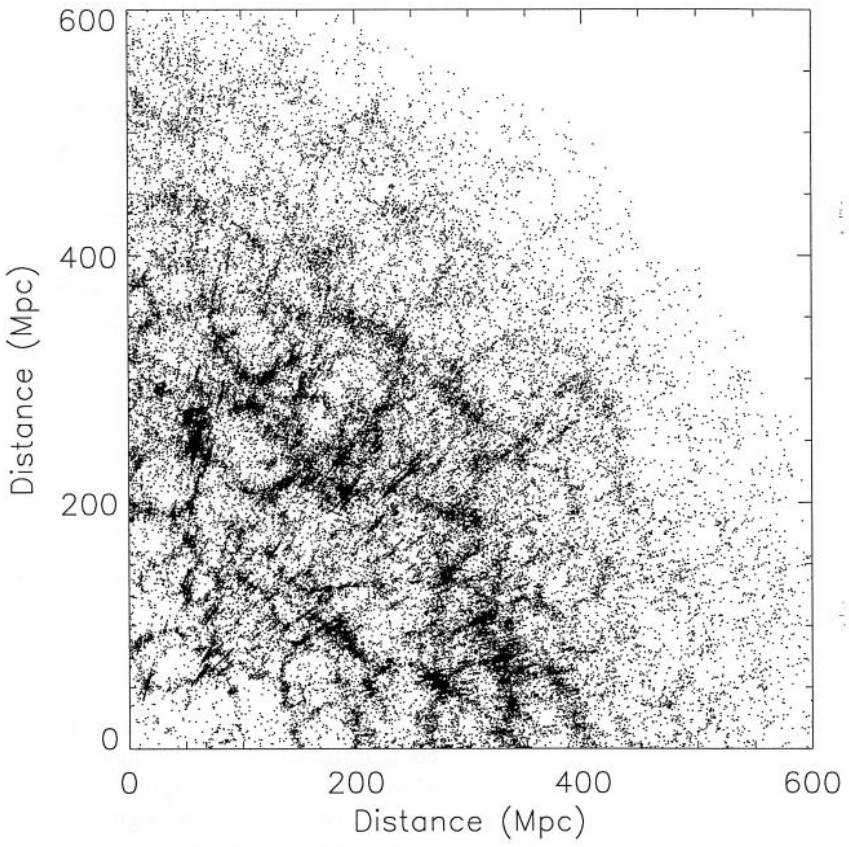

Figure 5.1: VIRGO DARK MATTER SIMULATION DATA.

On large scales, count probabilities reveal important clues about the origin of primordial density fluctuations. The simple standard models of inflationary cosmology predict Gaussian probability distributions while topological defect models (e.g. cosmic strings, texture) predict non-Gaussian fluctuations [Pee80, Fry84, Ber92]. The question of which type of model is correct is still very much open and is one of the fundamental questions of cosmology [SDS].

Figure 5.1 shows an instance of simulation data currently in use for exactly this purpose, from the Virgo project. We will describe it in more detail in the experimental section.

The need for higher-order statistics.

Though lower-order correlations are in agreement with the standard model, a number of issues cause the answer to remain inconclusive, including edge effects and odd geometries in existing samples, which are compounded by their small sample sizes, and the aforementioned fact that the lower-order correlations alone are insufficient to capture the distribution.

The higher-order n-point functions, along with observational data of high quality and in massive quantities from the emerging Sloan Digital Sky Survey and others to come, represent the central inferential path which can decide the question. See for example [TJ03].

Figure 5.2 shows a clear example of the inadequacy of the lower-order correlations alone.

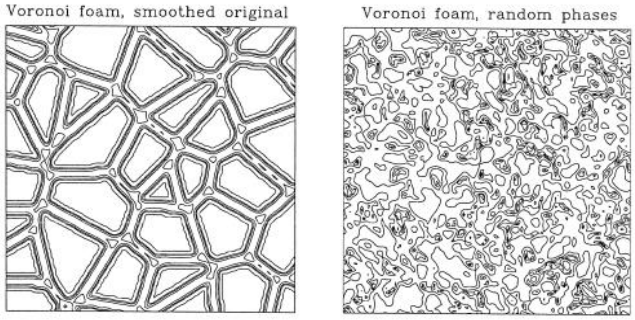

Figure 5.2: TWO DISTRIBUTIONS WITH THE SAME CORRELATION FUNCTION. The left-hand figure is a two-dimensional Voronoi foam, generated by the median surfaces between Poisson "seeds" at a mean separation of 100 h-1 Mpc . In this simple toy model, galaxies reside only on the walls of the foam, smoothed to give the walls a finite thickness. The structure has a well defined second-order statistic, but it also has correlated phases. This picture has been Fourier transformed, all the phases randomized, then transformed back again. The result is another two-dimensional density plot, shown on the right hand side, with the same second-order properties, but with a Gaussian density distribution function. — *Figure and caption both from [SDS], produced by Alex Szalay.*

Computational status.

Now that the issue of observational data is beginning to disappear, the obvious computational problem stands alone as the clear bottleneck. The practical severity of the problem is also significantly compounded by certain issues which we'll discuss shortly. As it stands, no significant computational progress has been made on the n-point correlation functions in three decades. As a benchmark, the largest value of N in recent memory for which the 3-point correlation was usefully computed was 20,000 or so [Nic03].

5.1.2 Practical considerations and constraints.

There are a number of issues we must consider if we wish to provide a working solution for the most demanding customer of the n-point correlation functions, the astrophysics community.

As in the last chapter, the dimensionality D is something which is not an issue in the astrophysical context we are mainly focused on, as we are working in 2 or 3 dimensions strictly.

General n.

In principle astrophysicists and spatial statisticians would like to compute correlation functions for n as high as possible. However, the inability to realize such functions

computationally has significantly impeded motivation for even the theoretical development of the correlation functions. The recent appearance of major advances in the theory of the 3-point correlation [TJ03] follows the recent appearance of large-scale datasets making higher-order analyses possible. To date, no major text or article has seen a need to go beyond the 4-point function. Undoubtedly the removal of computational obstacles will have a similar effect on theoretical development.

Simple and compound matchers.

We call the set of thresholds which determine whether an n-tuple is to be counted or not the *matcher*. A *simple* matcher has only one parameter, such as a single pair separation threshold for the 2-point function, or a single edge length for an equilateral-triangle 3-point function. A *compound* matcher allows a different parameter for each of the $n(n-1)/2$ possible lengths. This flexibility is important for studying asymmetric structures in data such as filaments.

More generally correlation functions can be defined with both lower and upper thresholds on each distance. In the general case then, a matcher is parametrized by two triangular matrices $\underline{L}_n$ and uH_n as we wrote in Chapter 1.

Multiple matchers.

Because the ultimate end of the computation is generally the visualization of the n-point correlation *function* curve, we generally need to compute counts for multiple matchers on the same data.

Edge effects.

The fact that estimation of quantities such as the 2-point is corrupted near the edges of the dataset is a central trouble of spatial statistics, for which many corrections have been proposed both in statistics and in astrophysics. See for example [Rip88]. In astrophysics the currently preferred solution is that of the Landy-Szalay estimator [LS93]:

$$\hat{\xi} = \frac{\mathrm{DD} - 2\mathrm{DR} + \mathrm{RR}}{\mathrm{RR}} \tag{5.3}$$

which is generalized to higher orders by [SS98]. Here D and R stand for 'data', the dataset in question, and 'random', a synthetic dataset consisting of points drawn from a Poisson distribution within the region of interest, and DD, DR, and RR stand for the corresponding 2-point (cross-)correlation counts. While overcoming many of the subtleties and difficulties of more analytical corrections, this approach is most effective when R is very large. Furthermore the estimate should be averaged over many instances of R.

Weighted data.

Another solution to the edge-effect problem is to down-weight data near the edges. This necessitates computing the weighted generalization of the n-point functions. Weighting is also useful in general to penalize data points which are corrupted in some way yet not so much as to warrant complete removal - this strategy abounds when data are expensive to obtain (as is very much the case in astronomy), or when the sample size is small for other reasons.

§5.2 n-Tree shattering.

5.2.1 Geometric shattering with n trees.

The algorithm for the n-point correlation can be derived straightforwardly by associating a *kd*-node with each volume element in the definition of the n-point correlation we gave. Note that the input to the recursive function is now an n-tuple of nodes $Q_1, .., Q_n$ rather than simply a pair. The algorithm also holds for $n = 2$, however.

Bounds.

c^{lo} and c^{hi} represent the running global bounds on the final answer count $\hat{c}$. A function maxtuples() returns the maximum possible number of n-tuples of points contained within the set of nodes in question which could match the matcher, stored as c^{max}. This can be determined analytically based on the known numbers of points in each node and whether the nodes overlap or contain each other.

The bounds c^{lo} and c^{hi} begin at 0 and the maximum possible number of n-tuples of points contained within the entire dataset (which corresponds to maxtuples() called on the n-tuple of root nodes). Because every recursive branching implicitly represents a *disjoint partitioning* of the set of possible n-tuples of points, c^{lo} and c^{hi} can be thought of as decomposing additively across all node sets encountered during the search. Hence the update rules, which correspond to compensating for the initial pessimistic setting of each contribution of each node set to 0 and maxtuples() for that node set.

Pairwise pruning.

To determine whether the entire n-tuple of nodes can be pruned (either by inclusion or exclusion), every *pair* of nodes in the set is tested by a function test_pair(), with the possible outcomes of 'exclude', 'include', or 'inconclusive'. If there is even one exclusion within the n-tuple of nodes, the contribution of this set of nodes to the overall count is known to be zero, and the flag is_zero is set to true. We also track whether every pair of nodes returns 'include', in which case the flag all_include is set to true, indicating inclusion for the entire node set. In this case we know the contribution of this set of nodes is exactly c^{max}.

```
Npt(Q_1, .., Q_n)
  c^max = maxtuples(Q_1, .., Q_n).
  if c^max == 0, return.
  all_include = 1, all_leaves = 1, is_zero = 0.
  for i = 1..n,
    if !leaf(Q_i) all_leaves = 0.
      for j = i+1..n,
        s = test_pair(Q_i,Q_j).
        if s == exclude, is_zero = 1, goto L.
        if s != include, all_include = 0, goto L.
  L:
  if is_zero == 1,
    c^hi -= c^max, return.
  if all_include == 1,
    c^lo += c^max, return.
  else
    if all_leaves == 1,
      c = NptBase(Q_1, .., Q_n).
      c^lo += c, c^hi -= (c^max - c).
    else
      Q* = choose(Q_1, .., Q_n).
      Npt(Q_1, .., Q*.left, .., Q_n).
      Npt(Q_1, .., Q*.right, .., Q_n).
```

Figure 5.3: n-TREE ALGORITHM, BASIC FORM. L represents a label to which program execution can jump.

Base case and recursion.

The flag all_leaves records whether every node in the node set is a leaf. If so, the base case is called, which performs the matcher exhaustively on all n-tuples of points within the n-tuple of leaf nodes and returns the exact number of matches for that node set.

At the beginning of the function all the flags are initialized to default values.

The main thing to notice about the recursion strategy is that we have avoided exploding the recursive step into 2^n recursive calls by recursing only in a binary fashion — we choose a node using a priority heuristic (using the largest node works as well as any in the vanilla n-point problem) and recurse on its two children. The testing for pruning opportunities is also formulated in a pairwise fashion. This is what allowed us to make quick progress on this problem - we formulated everything pairwise so that we could reuse all the machinery we developed for pairwise N-body problems.

§5.3 Combinatorics.

5.3.1 Permutational redundancy elimination.

So far, we have discussed two operations which cut short the need to traverse the tree further - exclusion and inclusion. Another form of pruning is to eliminate node-node comparisons which have been performed already in the reverse order, or more generally, consideration of an n-tuple of nodes which has already been considered in a different permutation.

This can be done [Sza00] simply by (virtually) ranking the datapoints according to their position in a depth-first traversal of the tree, then recording for each node the minimum and maximum ranks of the points it owns, and pruning whenever a node Q_a whose index in the permutation is higher than that of another node Q_b has maximum rank less than Q_b's minimum rank. This is useful for all-pairs problems, but becomes *essential* for all-n-tuples problems. It should be noted that this kind of pruning is not practical for single-tree search.

§5.4 Geometric Monte Carlo

Despite great efficiencies relative to the exhaustive method, the exact shattering method at some point becomes overwhelmed by the sheer combinatorics of the problem. We will see this in the experimental results. We now turn to Monte Carlo ideas to reach higher efficiencies in many cases. This method serves as a nice complement to the exact algorithm, because its properties make it useful for cases which are extremely difficult for the exact method, and vice versa.

The basic approach is, as in Chapter **??**, to view the problem as one of numerical integration, then specialize a general integration technique to our kind of problem. Here we develop a special form of the Monte Carlo method. Shattering turns out to be an essential ingredient of this new kind of recipe — the idea is to use it to stratify the integration domain for sampling. Viewing it the other, we stop the shattering process before its completion and sample the open nodes the rest of the way to the answer.

5.4.1 Monte Carlo integration.

As we have discussed earlier, many N-body problems are *integration* problems. As such, it behooves us to consider the idea of *Monte Carlo* integration for N-body problems.

The Monte Carlo method is a general approach for integrating functions $f()$ of virtually any sort. The approach is extremely versatile and widely applicable, and has found use in a large array of important problems. It uses the simple observation that

$$\widehat{\langle f \rangle} = \frac{1}{S} \sum_{s}^{S} f(\underline{x}_s) \tag{5.4}$$

is an unbiased estimator of θ^*, the true mean of $f()$, and the $\underline{x}_s \in \underline{X}_{\mathcal{S}}$ are S independent uniform samples from the domain of $f()$. Its variance can be estimated by

$$\hat{\sigma}^2 = \langle (f - \widehat{\langle f \rangle})^2 \rangle = \frac{1}{S-1} \sum_{s}^{S} (f(\underline{x}_s) - \widehat{\langle f \rangle})^2. \tag{5.5}$$

The central limit theorem yields the standard error bound for the Monte Carlo method:

$$\langle f \rangle = \widehat{\langle f \rangle} \pm z_{\epsilon/2} \hat{\sigma} \tag{5.6}$$

where $z_{\epsilon/2}$ is the $100(1-\epsilon/2)^{th}$ percentile of the standard normal distribution. For example for 99% confidence, $z_{\epsilon/2} \simeq 3$.

This is the baseline confidence band which additional techniques are employed to tighten, though various *variance reduction* approaches, such as stratified sampling and importance sampling.

Binomial distribution.

We have already characterized the n-point correlation as a kind of counting problem, in fact a generalization of the all-n-tuples-range-count problem. It can thus be thought of exactly as estimating a binomial proportion p, the number of n-tuples matching the matcher out of the total possible number of n-tuples[1], *i.e.* $\langle f \rangle = p$. The standard mean and variance for the binomial distribution version of what we've discussed so far is

$$\hat{p} = \frac{1}{S} \sum_{s}^{S} I_f(\underline{x}_s) \tag{5.7}$$

and

$$\begin{aligned} \hat{\sigma}^2 &= \frac{1}{S-1} \sum_{s}^{S} (I_f(\underline{x}_s) - \hat{p})^2 \\ &= \hat{p}(1-\hat{p}) \end{aligned} \tag{5.8}$$

where $I_f()$ denotes the indicator for our matching function f.

Our resulting estimate for an n-point count c is $\hat{c} = E[c] = \hat{p}C$ where C is the total number of possibly-matching n-tuples in the data.

The principal upshot of the ability to specialize to the binomial distribution is that the problem of designing good confidence bounds, a rather difficult affair in general, has been well-studied for the case of binomial estimation. We will take advantage of the work that has been done in this area, which will allow us to go beyond the simple central limit theorem approach to Monte Carlo confidence bounds.

1. Note that this is not the same as the common *hit-or-miss* Monte Carlo strategy, which also becomes a binomial estimation problem.

n-tuple sample generation.

In our case there is no issue of a proposal function — samples are drawn from the finite datasets $\underline{X}_i$, with replacement. *Note that a 'sample' $\underline{x}_s$ in our case is actually an n-tuple of data points.* Given a set of n tree nodes Q_i, we generate an n-tuple by choosing one datum from each node.

One interaction occurs here with our virtual indexing mechanism for permutation redundancy elimination from 5.3.1. To ensure that all generated n-tuples are valid with respect to the permutation constraints expected by all other parts of the algorithm (*i.e.* the exact shattering algorithm), the n-tuple must meet the virtual index ordering constraint or else it is rejected and we try again to generate a valid n-tuple sample. The average number of attempts before success is typically somewhere around 2 for a 3-point problem. It should be emphasized that this is a direct multiplying factor of our runtime, *i.e.* a source of ineffiency which further thought might be able to reduce.

Computational cost of Monte Carlo.

It is important to note a Monte Carlo sampling approach of the kind we're constructing is no longer sensitive to the number of data N. Its runtime, given a user-supplied maximum error criterion, depends only the error bound $\sqrt{\hat{p}(1-\hat{p})/S}$, which depends only on the size of the true probability p to be estimated and the number of samples S.

Unfortunately for wide useful ranges of matchers, p is typically quite small, *e.g.* 10^{-5} is not uncommon and *e.g.* 10^{-10} is possible. In a case like this we are likely to wait quite a long time, in fact probably longer than it takes to run the exact algorithm, before we observe any matches at all.

S is under our control. We'll now see how we can effectively bring p under our control as well – by performing exact shattering for a number of recursion steps, we will effectively winnow down the number of non-matches from the integration domain until the proportion of matches is acceptable for sampling.

5.4.2 Stratified sampling.

In stratified sampling the domain of integration is broken into K pieces and standard Monte Carlo is applied within each piece. Then

$$\widehat{\langle f \rangle} = \sum_k^K w_k \left\{ \frac{1}{S_k} \sum_s^S f(\underline{x}_{sk}) \right\} \tag{5.9}$$

is an unbiased estimator of θ^*, where w_k is the proportion of the probability mass represented by the k^{th} piece and

$$\hat{\sigma}^2 == \sum_k^K \frac{w_k^2}{S_k} \left\{ \frac{1}{S_k - 1} \sum_s^{S_k} (f(\underline{x}_{sk}) - \widehat{\langle f \rangle}_k)^2 \right\}. \tag{5.10}$$

is the variance of the estimate.

The motivation for doing this is the following. If the stratification is such that the differences between the means $\langle f \rangle_k$ in each piece are greater than the variations $\langle (f - \widehat{\langle f \rangle}_k)^2 \rangle$ in each piece, stratified sampling will be more efficient, *i.e.* require fewer samples to achieve the same variance or error, than straightforward sampling without stratification. In fact the following can be proven:

Theorem 2 (Advantage of stratification) *As long as the sample allocations S_k are proportional to w_k, i.e. $S_k = w_k S$, then if $\hat{\sigma}_K^2$ and $\hat{\sigma}_{K'}^2$ are the variances for stratification sampling with K and K' strata, respectively, $\hat{\sigma}_K^2 \leq \hat{\sigma}_{K'}^2$ for $K < K'$.*

The proof of this theorem follows easily from standard arguments given in textbooks on the Monte Carlo method (such as given in [Rub81]) which usually aim to show the special case of $K = 1$ versus $K' > 1$, *i.e.* that stratification is at least as good as no stratification. But more generally, a larger amount of stratification is better.

We'll call this the *proportional allocation* scheme. While sufficient for the theorem to hold, it is clearly not the optimal allocation scheme as it does not incorporate any information about the parameters p_k.

At this point there are two open issues — how to break up the domain of integration and how to allocate the proportions of samples. We address these issues in a manner customized to our problem, using geometric shattering to obtain the pieces (described next) and a new adaptive strategy to allocate the samples (next Section).

Geometric stratification.

We will obtain our stratification by performing geometric shattering, stopping the process at some point before the exact result is computed. Whichever nodesets are 'open', or remaining to be searched at the stopping point will form the strata we will use for sampling.

Disjointness from recursion tree. A key point, which is perhaps subtle, is that the nodeset (n-tuple of nodes) at each recursion step in the exact algorithm represents a certain *subset of all the possible n-tuples of points in the data.* Furthermore the two branchings of the recursion tree, or recursive sub-calls, at any point are *disjoint* subsets of all the possible n-tuples of points. This is the key which allows us to obtain strata from the recursion tree of the exact algorithm.

Weights from analytic combinatorics. In our case the w_k values are obtained from our earlier mechanisms for computing the maximum possible number C_k of n-tuples (counting permutations or not, depending on the matcher specified) in the k^{th} nodeset. Note that $\sum_k^K C_k = C$, the total number of possible n-tuples in the dataset, so that $w_k = C_k/C$. Our overall estimate of the count is now $\hat{c} = E[c] = \sum_k^K \hat{p}_k C_k$.

Prioritized search.

Knowing that we will not carry the shattering procedure through to its exact conclusion creates a new constraint on the shattering search procedure — pruning (which includes both exclusion and inclusion) should be performed as early as possible. Another viewpoint is that we would like to end up with strata where the parameters p_k are as large as possible using our time spent on exact shattering as efficiently as possible. The goal then is to focus on eliminating chunks of definitely-non-matching n-tuples as quickly as possible.

To achieve this we reformulate the search in terms of a priority queue, as we alluded was possible back in Section 2.2. The priority function now becomes the maximum inter-node distance, which prefers nodesets which are more likely to be pruned.

The priority queue also serves naturally as the storage mechanism for maintaining the set of all the open nodes during the search. This allows the opportunity for reasoning globally about the current set of strata at any given time if desired, which is not afforded by a local search mechanism.

Stopping criterion.

A simple heuristic is used as a stopping criterion - a threshold τ on the fraction of the data accounted for so far by exact pruning operations. While intuitive and fairly robust in practice, it does not in general adapt optimally to the problem at runtime.

A better procedure would perhaps somehow taking into account the current estimate of the proportion $\hat{p}$ to decide when to stop. Unfortunately, then, the shattered Monte Carlo algorithm as it stands, contains a tweak parameter τ.

§5.5 Optimal sample allocation.

The optimal allocation of sample proportions is theoretically known. It is sometimes called *optimal Neyman allocation*, and it accounts for the differences in the binomial proportions p_k between the different strata:

Theorem 3 (Optimality of Neyman allocation) *If $\pi(S)$ indicates the partitioning $S = \cup_k S_k$, $\pi^*(S) = \min_\pi \sigma^2(\pi(S))$ occurs when*

$$S_k = \frac{w_k \sigma_k}{\sum_{k'} w_{k'} \sigma_{k'}} S. \tag{5.11}$$

However, because the variances are not known *a priori*, this is dismissed in textbooks (*e.g.* [Rub81]) as anything other than a theoretical observation.

5.5.1 Adaptive Neyman allocation.

We introduce a simple iterative procedure which uses existing information to best approximate optimal Neyman allocation, which we'll call *adaptive Neyman allocation*. If at iteration i we will take S_i more samples, we set

$$S_{ik} = \frac{w_k \hat{\sigma}_k}{\sum_{k'} w_{k'} \hat{\sigma}_{k'}} S_i \tag{5.12}$$

and $S_i = \lambda S_{i-1}$ for some $\lambda > 1$. In our experiments we set $\lambda = 2$ and $S_0 = 50{,}000$. It should be clear that this yields optimal Neyman allocation asymptotically, *i.e.*

Theorem 4 (Asymptotic optimality of adaptive Neyman allocation) *If $S^{tot} = \sum_{i'}^{i} S_{i'}$ is the total number of samples taken at iteration i, and $\pi_i(S^{tot})$ indicates the effective partitioning $S^{tot} = \cup_k \{\sum_{i'}^{i} S_{i'k}\}$ at iteration i, then $\pi_i \to \pi^*$ as $i \to \infty$.*

> **Proof:** This is trivially true because our procedure guarantees that every node receives at least one sample in every round of sampling. Thus asymptotically the estimates converge to the true values. The *rate* of this convergence is not necessarily optimal, however.

5.5.2 Binomial confidence bounds.

It will surprise many readers to learn that the issue of practical confidence bounds for estimation of a binomial proportion, the most basic statistical scenario forming the context in which the entire issue of confidence bounds is generally introduced in textbooks, is *not a closed problem*. We will gain insight from very recent work on this problem to obtain robustly accurate bounds for our estimation algorithm. Without careful attention to the confidence bounds a simpler version of our approach would be more likely to report bounds which do not actually enclose the true value of p than our error tolerance ϵ would predict.

Note that this issue is critical because in our setting a fair accuracy is required, unlike many Monte Carlo applications in which quick but rough estimates (*i.e.* wide or not-quite-enclosing bounds) are often adequate for making progress. In our case we typically will need to make comparative distinctions on the order of 10% or 1% for meaningful n-point correlation curves or comparisons between curves. Thus the target we will have in mind is accuracy to within 1% error.

Deficiencies of the Wald interval.

The standard confidence interval of 5.4.1, called the *Wald interval*, though in near-universal use, do not actually enclose the true value of p as supposed. This has long been a subject of investigation (see **??**), though awareness of the poor behavior of the 'textbook' interval, even within statistics, is surprisingly limited — in fact even the full extent of its poor behavior has been realized only very recently, *e.g.* [AC98, BCD01].

The main issue is *coverage* — the true probability that a sample falls within the stated confidence interval. In [BCD01] it is demonstrated that for any $0 < p < 1$ and $S < 45$, the coverage of the nominally-99% Wald interval is significantly less, in fact on average 91.4% in their experiments. The situation is amplified as p goes to 0 or 1 — coverage shockingly zooms down to 88% at the edges of their plot, which stops at $p \simeq 0.6$ and $p \simeq 0.94$.

Note that in our problems quite often $p \ll 1$, a virtual needle in a haystack. Though it might be thought that stratification creates larger values p_k, splitting a nodeset into two parts often has the effect of creating one part with a higher proportion of matches and one part with a smaller proportion. Thus *both* large and small p_k values are introduced. Also, a given stratum receives only a fraction of the overall number of samples S, so that S_k may be fairly small. We have observed that the accumulation of coverage-related errors over many strata can be unacceptably significant.

There are several sources of the problems: most obviously, the normal approximation to the binomial does not account for the discreteness and skewness of the distribution.

Approach of conservatism.

Unfortunately the various proposals for treating this problem are not directly applicable here. They constitute proposals for a binomial confidence interval for general use — *i.e.* the interval should not only provide high coverage, but also not be too conservative, be simple to express and remember, *etc.* . For example, the overall recommendation of [BCD01] is a slightly-generalized version of the *Agresti-Coull confidence interval*:

$$\tilde{p} \pm z_{\epsilon/2} \frac{\sqrt{\tilde{p}(1-\tilde{p})}}{\sqrt{\tilde{S}}} \tag{5.13}$$

which is simply the Wald interval except that $\hat{p}$ and S are modified by pretending that two additional successes (matches) and failures (non-matches) were observed. This cannot be used for our estimates because it implies that no matter how many samples were taken, a minimum of two matches were observed. While this procedure is shown to be preferable asymptotically, for a finite set of samples this can mean massive over-estimation of p_k values, which are multiplied by possibly huge values of C_k when taking the expectation $E[c]$.

While conservativism means additional computational cost, for adequate scientific use we ultimately require that the coverage be as close as possible to the purported confidence, even if it carries some computational expense. The simple approach of convervatism we use to counter problems of poor coverage is to merely threshold p_k at some *forcing probability* p_F, *i.e.* for the purposes of computing confidence bounds and sample allocations, p_k is treated as p_F if $p_k < p_F$ or $1 - p_F$ if $p_k > 1 - p_F$. We use the value $p_F = 0.2$ in our experiments. This seems to provide a fairly robust shield against coverage problems, though it comes at a certain computational cost.

In principle one could allow p_F to relax slowly over time, *i.e.* $p_F \to 0$ as $S \to \infty$, though we did not experiment with such approaches.

5.5.3 Defragmentation.

The aggressive nature of the adaptive Neyman allocation strategy can exacerbate the confidence issues surrounding estimating tiny proportions. For tiny values of p_k, accurate estimates are not available until a large number of samples have been taken. But early iterations of adaptive Neyman allocation, observing zero or very few matches for many nodesets, will assign zero or few samples to them for successive iterations. Fortunately a non-zero forcing probability prevents forever-zero allocation to a nodeset, so eventually all probabilities will be estimated correctly by the adaptive allocation strategy.

We experimented with another mechanism we call *defragmentation*. Here the idea is to pool together all the nodesets $\square_k$ having small p_k into a single super-nodeset. This *small-fries* nodeset $\square_{SF} = \cup_k \square_k$, $\forall k | \hat{p}_k < p_U$. The *union probability* p_U is set to 0.1 in our experiments. (Likewise the *big-cheeses* nodeset $\square_{BC} = \cup_k \square_k$, $\forall k | \hat{p}_k > 1 - p_U$.) The values of $\hat{p}_k$ are based on the first iteration of sampling.

Thus, rather than estimating a large number of small probabilities, each requiring a large and separate number of samples, we track a single small probability for the entire union of the small-fry nodesets, effectively sharing the same chunk of samples across the union of strata.

5.5.4 Shattered Monte Carlo algorithm.

Exact shattering phase.

First **Npt'**(), a modified version of the **Npt**() algorithm which is driven by priority search, is called. The set of K open node sets $\{T_1, .., T_K\}$ when that algorithm ceases shattering is used for sampling. Thus each stratum T_k corresponds to an *n-tuple* of nodes.

Sampling iterations.

Before being included in the output set, each node set T_k receives a small initial number of samples, so that an estimate for the k^{th} node set $\hat{p}_k^0$ exists at iteration 0.

Based on $\hat{p}_k$ the standard deviation $\hat{\sigma}_k$ can be computed using the standard variance formulas we have shown, and based on these standard deviations, the Neyman value for the number of samples S_k to be allocated to the k^{th} node set can be computed. These three numbers represent the collective set of parameters for each stratum at iteration i, θ_k^i. These computations of new values for $\hat{p}_k$ and $\hat{\sigma}_k$ given a number of samples S_k and the old value of the parameters during the last iteration, θ_k^{i-1}, are encapsulated by the function update_estimates().

NptMC$(Q_1, .., Q_n)$
  $\{T_1, .., T_K\} =$ **Npt'**$(Q_1, .., Q_n)$.
  `for` $k = 1, .., K$, $S_k = S_0$.
  `do`
    `for` $k = 1, .., K$,
      $\theta_k^i = \{\hat{p}_k, \hat{\sigma}_k, S_k\} =$ update_estimates(θ_k^{i-1}, S_k).
      $\{S_1, ..., S_K\} =$ update_allocation$(\theta^{i_1}, .., \theta_K^i, S)$.
      $S* = 2$.
    $\{c^{lo}, c^{hi}\} =$ global_estimate$(\theta_1^i, .., \theta_K^i)$.
  `while` $(\frac{|c^{hi} - c^{lo}|}{|c^{lo}|} < \epsilon)$.

Figure 5.4: SHATTERED MONTE CARLO ALGORITHM, BASIC FORM.

The function global_estimate() computes the global $\hat{p}$ and $\hat{\sigma}$ based on the estimates for the strata.

§5.6 Performance.

5.6.1 Mock galaxy catalog.

The first dataset is a mock galaxy catalog constructed using a simplistic halo (*i.e.* cluster) model. This involves randomly placing 1,000 halos in a volume of 141 Mpc cubed (the GIF simulation volume), with each halo having a NFW profile such that halos correspond to filament intersections. Each halo contains 1,000 galaxies placed randomly according to a certain dropoff distribution. The dataset thus consisted of 1,000,000 points. The random (R) dataset consisted of 75,000,000 objects. The 'bins', or range parameters defining the edge lengths of the triangles in the matcher (equilateral in this case) are shown below.

Bin	l - h	Width
1	0.0271644 - 0.0368129	0.0096485
2	0.0498884 - 0.0676083	0.0177199
3	0.0916221 - 0.124165	0.0325429
4	0.168267 - 0.228034	0.059767
5	0.309030 - 0.418794	0.109764
6	0.567545 - 0.769131	0.201586
7	1.04232 - 1.41254	0.37022
8	1.91426 - 2.59418	0.67992

Figure 5.5: BINS FOR MOCK GALAXY CATALOG DATA.

The important aspect of this dataset, besides its large size, is that its 2-point and 3-point functions could be computed analytically. This dataset thus served as a test of the code's correctness. The measurements indeed confirmed that the code computes the n-point functions accurately, as shown in Figure 5.8.

Bin	DD		DR		RR	
	Count	Time	Count	Time	Count	Time
1	218246	8	3500	23	133295	517
2	701879	12	20567	26	804617	917
3	4.14357e+06	23	128697	33	4.89675e+06	1786
4	1.23413e+07	36	802011	47	3.00546e+07	3162
5	3.51576e+07	58	4.91733e+06	79	1.84932e+08	4038
6	7.62079e+07	94	3.03017e+07	179	1.1389e+09	7944
7	1.16947e+08	93	1.86085e+08	512	7.00223e+09	20085
8	2.53535e+07	44	1.13945e+09	1820	4.28598e+10	62455

Figure 5.6: 2-POINT RESULTS FOR MOCK GALAXY CATALOG DATA.

Results for the 3-point correlation, using equilateral triangles, is shown next.

Bin	DDD		DDR		DRR		RRR	
	Count	Time	Count	Time	Count	Time	Count	Time
1	37573	60	1.05631e+08	58	9.23194e+09	87	1.00721e+13	3596
2	354534	109	1.46384e+08	91	1.50617e+10	123	2.18128e+13	6891
3	6.20726e+06	434	2.6161e+08	241	2.36021e+10	187	2.68654e+13	16131
4	4.26785e+07	1478	5.56273e+08	682	3.50885e+10	382	2.74388e+13	37777
5	2.44103e+08	5502	1.58497e+09	2833	5.90041e+10	1143	2.74675e+13	69563
6	7.9345e+08	14760	5.78509e+09	14541	1.31034e+11	8229	2.75566e+13	450325
7	1.62056e+09	20417	2.00724e+10	68306	7.33894e+11	97239		

Figure 5.7: 3-POINT RESULTS FOR MOCK GALAXY CATALOG DATA.

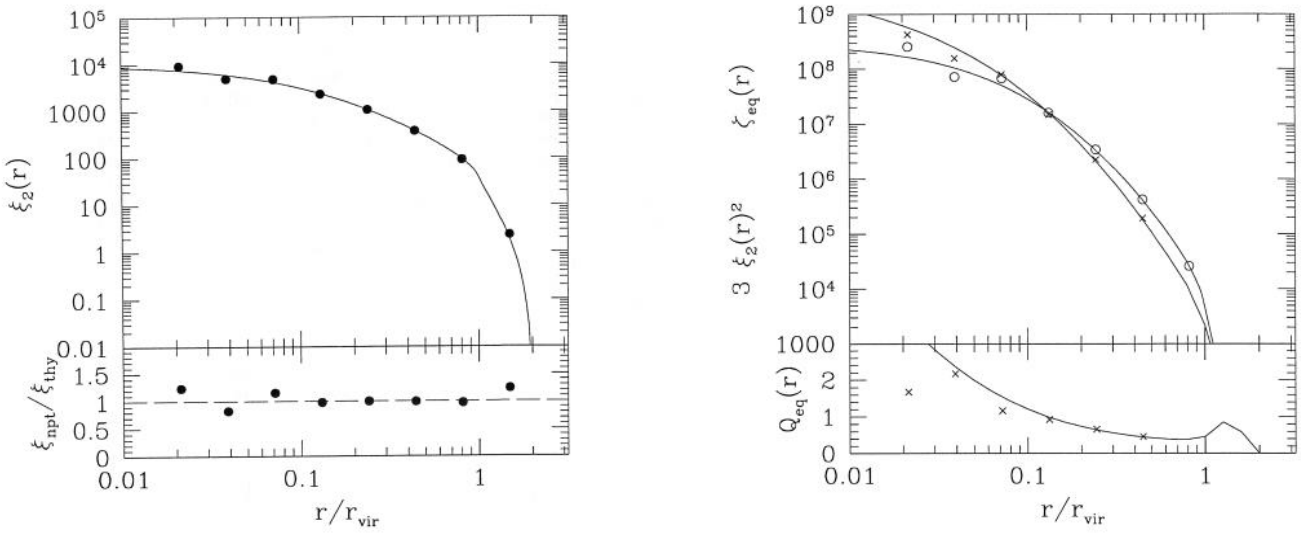

Figure 5.8: VALIDATION USING ANALYTIC CURVES. The small deviations from the analytic fall within the expected noise due to the finite sample size.

5.6.2 Galactic simulation.

The second dataset we show represents a scientific experiment of the kind we have described, in action. Our experiments on this dataset with our code will result in the first calculations ever performed on this scale.

Our astrophysicist collaborators created a sophisticated synthetic dataset which is designed to test the morhological statistics of the 'standard model' (*i.e.* the inflationary Gaussian universe hypothesis with currently agreed-upon parameters) versus the morphological statistics of observational data obtained from measurements of the true universe.

They used the VIRGO dark matter simulations, specifically the Lambda CDM simulation (LCDM at z=0.0, Hubble Volume). Then they took each matter particle in the simulations and 'pasted' a galaxy onto it. The luminosity and color of that galaxy was randomly drawn from the color-density relatuion as observed in observed data from the SDSS, and from the luminosity function seen in the real data. This process is iterated until the lower-order correlations, *i.e.* the *2-point* correlation function of the synthetic data agrees with that of the observed data. The product of this is a catalog of galaxies with the right cluster in space and color, with the addition that we know know where all the mass is due to the dark matter component. The SDSS selection function is then superimposed, *i.e.* the synthetic data reflects the flux limit and redshift limit of the real data. This is a 90-degree wedge with 10-degree thickness, going out to the magnitude limit of the survey. In other words, this data should look like the real data in all aspects to the best current knowledge of astrophysics.

The bins used in the experiments (all such parameters were selected by the astrophysicists) are shown below:

Bin	l_a	h_a	l_b	h_b	l_c	h_c
1	0.63000	0.99900	0.63000	1.99800	0.63000	2.32767
2	0.63000	0.99900	0.63000	1.99800	0.83790	2.65734
3	0.63000	0.99900	0.63000	1.99800	1.04580	2.99700
4	0.63000	0.99900	1.26000	3.99600	1.26000	4.32567
5	0.63000	0.99900	1.26000	3.99600	1.46790	4.65534
6	0.63000	0.99900	1.26000	3.99600	1.67580	4.99500
7	0.99900	1.58300	0.99900	3.16600	0.99900	3.68839
8	0.99900	1.58300	0.99900	3.16600	1.32867	4.21078
9	0.99900	1.58300	0.99900	3.16600	1.65834	4.74900
10	0.99900	1.58300	1.99800	6.33200	1.99800	6.85439
11	0.99900	1.58300	1.99800	6.33200	2.32767	7.37678
12	0.99900	1.58300	1.99800	6.33200	2.65734	7.91500
13	1.58300	2.51000	1.58300	5.02000	1.58300	5.84830
14	1.58300	2.51000	1.58300	5.02000	2.10539	6.67660
15	1.58300	2.51000	1.58300	5.02000	2.62778	7.53000
16	1.58300	2.51000	3.16600	10.04000	3.16600	10.86830
17	1.58300	2.51000	3.16600	10.04000	3.68839	11.69660
18	1.58300	2.51000	3.16600	10.04000	4.21078	12.55000
19	2.51000	3.97900	2.51000	7.95800	2.51000	9.27107
20	2.51000	3.97900	2.51000	7.95800	3.33830	10.58410
21	2.51000	3.97900	2.51000	7.95800	4.16660	11.93700
22	2.51000	3.97900	5.02000	15.91600	5.02000	17.22910
23	2.51000	3.97900	5.02000	15.91600	5.84830	18.54210
24	2.51000	3.97900	5.02000	15.91600	6.67660	19.89500
25	3.97900	6.30800	3.97900	12.61600	3.97900	14.69760
26	3.97900	6.30800	3.97900	12.61600	5.29207	16.77930
27	3.97900	6.30800	3.97900	12.61600	6.60514	18.92400
28	3.97900	6.30800	7.95800	25.23200	7.95800	27.31360
29	3.97900	6.30800	7.95800	25.23200	9.27107	29.39530
30	3.97900	6.30800	7.95800	25.23200	10.58410	31.54000
31	6.30800	10.00000	6.30800	20.00000	6.30800	23.30000
32	6.30800	10.0000	6.30800	20.0000	8.38964	26.6000
33	6.30800	10.0000	6.30800	20.0000	10.4713	30.0000
34	6.30800	10.0000	12.6160	40.0000	12.6160	43.3000
35	6.30800	10.0000	12.6160	40.0000	14.6976	46.6000
36	6.30800	10.0000	12.6160	40.0000	16.7793	50.0000

Figure 5.9: BINS FOR GALACTIC SIMULATION EXPERIMENTS. l and h refer to the low and high thresholds, respectively. a,b,and c refer to the three (unordered) legs of the 3-point correlation's triangle.

The size of the data catalog ('D') 63,678 and that of the random catalog ('R') is 191,034 points. The 2-point numbers and runtimes for the DD, DR, and RR cases are shown in the next two tables. Note that all of these timings were done on a Beowulf network of workstations (operating in serial), each of which is a Linux Pentium machine with 2x 1GhZ processor and 1Gb RAM.

The first 15 bins are shown in the first table:

Bin	l - h	DD		DR		RR	
		Count	Time	Count	Time	Count	Time
1	0.630000 - 0.99900	11795	1	4083	4	6233	2
	0.630000 - 1.99800	75195	0	42642	4	63637	3
	0.630000 - 2.32767	106991	1	68315	5	101131	5
2	0.630000 - 0.99900	11795	0	4083	4	6233	2
	0.630000 - 1.99800	75195	1	42642	4	63637	4
	0.837900 - 2.65734	138373	1	100524	5	148119	4
3	0.630000 - 0.99900	11795	1	4083	3	6233	2
	0.630000 - 1.99800	75195	1	42642	4	63637	4
	1.04580 - 2.99700	174089	0	141716	6	209140	5
4	0.630000 - 0.99900	11795	1	4083	3	6233	2
	1.26000 - 3.99600	323098	1	336608	7	496894	6
	1.26000 - 4.32567	385345	1	427998	7	632579	5
5	0.630000 - 0.99900	11795	0	4083	3	6233	1
	1.26000 - 3.99600	323098	2	336608	7	496894	6
	1.46790 - 4.65534	441324	1	527545	7	780397	6
6	0.630000 - 0.999000	11795	1	4083	4	6233	2
	1.26000 - 3.99600	323098	1	336608	6	496894	6
	1.67580 - 4.99500	501677	1	645125	8	954277	6
7	0.999000 - 1.58300	31436	1	16424	4	24623	3
	0.999000 - 3.16600	199716	1	168825	6	249131	5
	0.999000 - 3.68839	280938	1	268884	6	396853	5
8	0.999000 - 1.58300	31436	0	16424	3	24623	3
	0.999000 - 3.16600	199716	1	168825	6	249131	5
	1.32867 - 4.21078	359395	1	392819	7	580165	6
9	0.999000 - 1.58300	31436	0	16424	4	24623	3
	0.999000 - 3.16600	199716	1	168825	5	249131	5
	1.65834 - 4.74900	448817	1	552705	8	818133	6
10	0.999000 - 1.58300	31436	1	16424	3	24623	3
	1.99800 - 6.33200	818013	2	1.30373e+06	10	1.92887e+06	8
	1.99800 - 6.85439	972692	1	1.65669e+06	11	2.45091e+06	8
11	0.999000 - 1.58300	31436	0	16424	4	24623	3
	1.99800 - 6.33200	818013	2	1.30373e+06	9	1.92887e+06	7
	2.32767 - 7.37678	1.10679e+06	1	2.03799e+06	12	3.01662e+06	8
12	0.999000 - 1.58300	31436	0	16424	4	24623	3
	1.99800 - 6.33200	818013	2	1.30373e+06	10	1.92887e+06	7
	2.65734 - 7.91500	1.2525e+06	2	2.48549e+06	13	3.67714e+06	9
13	1.58300 - 2.51000	83647	0	65546	5	95986	5
	1.58300 - 5.02000	513686	2	659009	8	975251	6
	1.58300 - 5.84830	717199	1	1.04561e+06	9	1.54725e+06	7
14	1.58300 - 2.51000	83647	1	65546	5	95986	4
	1.58300 - 5.02000	513686	1	659009	8	975251	6
	2.10539 - 6.67660	909050	2	1.52279e+06	10	2.25352e+06	8
15	1.58300 - 2.51000	83647	1	65546	5	95986	5
	1.58300 - 5.02000	513686	1	659009	8	975251	6
	2.62778 - 7.53000	1.12396e+06	2	2.13797e+06	12	3.1644e+06	8

Figure 5.10: 2-POINT RESULTS FOR GALACTIC SIMULATION DATA, BINS 1-15. l and h are the low and high thresholds for the matcher, respectively. The results are grouped according to the corresponding *3-point* bins.

The remaining bins are shown in the next table:

Bin	l - h	DD Count	DD Time	DR Count	DR Time	RR Count	RR Time
16	1.58300 - 2.51000	83647	0	65546	4	95986	5
	3.16600 - 10.0400	2.03585e+06	3	5.00106e+06	18	7.37772e+06	10
	3.16600 - 10.8683	2.4211e+06	2	6.33633e+06	20	9.34247e+06	13
17	1.58300 - 2.51000	83647	1	65546	5	95986	4
	3.16600 - 10.0400	2.03585e+06	2	5.00106e+06	18	7.37772e+06	11
	3.68839 - 11.6966	2.75858e+06	3	7.77651e+06	21	1.14521e+07	16
18	1.58300 - 2.51000	83647	1	65546	5	95986	4
	3.16600 - 10.0400	2.03585e+06	2	5.00106e+06	17	7.37772e+06	11
	4.21078 - 12.5500	3.13323e+06	3	.458e+06	24	1.3907e+07	8
19	2.51000 - 3.97900	217093	1	255940	6	378258	5
	2.51000 - 7.95800	1.28496e+06	2	2.54331e+06	13	3.76208e+06	9
	2.51000 - 9.27107	1.78854e+06	2	4.01873e+06	16	5.93476e+06	10
20	2.51000 - 3.97900	217093	2	255940	7	378258	6
	2.51000 - 7.95800	1.28496e+06	1	2.54331e+06	13	3.76208e+06	9
	3.33830 - 10.5841	2.25988e+06	3	5.82627e+06	19	8.5915e+06	11
21	2.51000 - 3.97900	217093	1	255940	6	378258	5
	2.51000 - 7.95800	1.28496e+06	2	2.54331e+06	13	3.76208e+06	9
	4.16660 - 11.9370	2.80006e+06	3	8.14484e+06	22	1.19895e+07	17
22	2.51000 - 3.97900	217093	1	255940	7	378258	5
	5.02000 - 15.9160	5.22008e+06	4	1.87613e+07	34	2.7467e+07	26
	5.02000 - 17.2291	6.27787e+06	5	2.36394e+07	39	3.45753e+07	29
23	2.51000 - 3.97900	217093	1	255940	6	378258	6
	5.02000 - 15.9160	5.22008e+06	5	1.87613e+07	35	2.7467e+07	25
	5.84830 - 18.5421	7.23449e+06	4	2.88138e+07	45	4.2114e+07	32
24	2.51000 - 3.97900	217093	1	255940	7	378258	6
	5.02000 - 15.9160	5.22008e+06	5	1.87613e+07	34	2.7467e+07	25
	6.67660 - 19.8950	8.31268e+06	5	3.47907e+07	51	5.08111e+07	35
25	3.97900 - 6.30800	542345	2	989332	10	1.46519e+06	7
	3.97900 - 12.6160	3.21419e+06	3	9.67024e+06	24	1.42178e+07	19
	3.97900 - 14.6976	4.54174e+06	4	1.51608e+07	30	2.22299e+07	22
26	3.97900 - 6.30800	542345	1	989332	10	1.46519e+06	7
	3.97900 - 12.6160	3.21419e+06	3	9.67024e+06	24	1.42178e+07	18
	5.29207 - 16.7793	5.84016e+06	5	2.17777e+07	37	3.18574e+07	28
27	3.97900 - 6.30800	542345	2	989332	10	1.46519e+06	7
	3.97900 - 12.6160	3.21419e+06	3	9.67024e+06	24	1.42178e+07	18
	6.60514 - 18.9240	7.37907e+06	5	3.01075e+07	46	4.39888e+07	33
28	3.97900 - 6.30800	542345	2	989332	10	1.46519e+06	7
	7.95800 - 25.2320	1.42425e+07	8	6.71475e+07	75	9.79219e+07	58
	7.95800 - 27.3136	1.72618e+07	9	8.37457e+07	86	1.22064e+08	66
29	3.97900 - 6.30800	542345	1	989332	9	1.46519e+06	7
	7.95800 - 25.2320	1.42425e+07	8	6.71475e+07	75	9.79219e+07	58
	9.27107 - 29.3953	2.01054e+07	10	1.00942e+08	98	1.47025e+08	72
30	3.97900 - 6.30800	542345	2	989332	10	1.46519e+06	8
	7.95800 - 25.2320	1.42425e+07	8	6.71475e+07	75	9.79219e+07	57
	10.5841 - 31.5400	2.33149e+07	11	1.20427e+08	110	1.75293e+08	83
31	6.30800 - 10.0000	1.34302e+06	2	3.78333e+06	18	5.57542e+06	10
	6.30800 - 20.0000	8.52661e+06	6	5568e+07	50	5.1948e+07	36
	6.30800 - 23.3000	1.22449e+07	7	5.48613e+07	65	8.00559e+07	49

Figure 5.11: 2-POINT RESULTS FOR GALACTIC SIMULATION DATA, BINS 16-31.

The following tables are for the 3-point function, showing the DDD, DDR, DRR, and RRR cases.

Bin		DDD		DDR		DRR		RRR
	Count	Time	Count	Time	Count	Time	Count	Time
1	97781	10	20894	24	4951	41	4890	55
2	116108	12	25399	27	6154	46	6032	60
3	117707	13	25999	30	6335	51	6209	64
4	415960	29	133437	66	41625	90	41321	110
5	444286	32	145556	75	45855	100	45686	117
6	443171	36	146361	83	46404	110	46189	126
7	681755	23	220187	50	77241	75	76326	100
8	803707	27	265661	60	95733	88	94496	111
9	814283	32	271419	72	98913	102	97537	124
10	2.53756e+06	82	1.37222e+06	203	644789	275	638029	315
11	2.69453e+06	93	1.49259e+06	237	710837	330	703454	367
12	2.67603e+06	105	1.49965e+06	272	718247	389	711042	423
13	4.1752e+06	60	2.21048e+06	139	1.20088e+06	184	1.16949e+06	204
14	4.87871e+06	75	2.65971e+06	180	1.4876e+06	250	1.4469e+06	270
15	4.92414e+06	90	2.71432e+06	227	1.53538e+06	322	1.49333e+06	341
16	1.38298e+07	250	1.37365e+07	823	9.8069e+06	1263	9.50644e+06	1325
17	1.46283e+07	292	1.49707e+07	985	1.0798e+07	1625	1.04646e+07	1743
18	1.44916e+07	332	1.50412e+07	1147	1.09092e+07	1995	1.05698e+07	2159
19	2.18104e+07	185	2.16221e+07	539	1.8147e+07	752	1.77617e+07	766
20	2.53432e+07	242	2.60308e+07	738	2.24161e+07	1090	2.19372e+07	1085
21	2.54728e+07	308	2.65704e+07	963	2.31137e+07	1601	2.26198e+07	1703
22	7.24667e+07	941	1.36847e+08	3836	1.43839e+08	7693	1.39167e+08	8633
23	7.72252e+07	1074	1.4907e+08	4573	1.57963e+08	9339	1.52778e+08	10302
24	7.66309e+07	1207	1.49795e+08	5376	1.59442e+08	11088	1.54184e+08	11962
25	1.02721e+08	713	2.10564e+08	2695	2.67053e+08	4833	2.59646e+08	5198
26	1.20043e+08	952	2.54094e+08	3780	3.28887e+08	7300	3.19427e+08	7817
27	1.208e+08	1168	2.59419e+08	4963	3.38729e+08	9915	3.28895e+08	10329
28	3.88227e+08	3737	1.30643e+09	19216	1.97906e+09	42939	1.90915e+09	44082
29	4.14812e+08	4253	1.4196e+09	22665	2.16325e+09	50885	2.08632e+09	51065
30	4.13015e+08	4770	1.42565e+09	25842	2.17992e+09	58901	2.1021e+09	58694
31	5.3883e+08	2991	2.05484e+09	15637	3.73461e+09	34756	3.60121e+09	33658
32	6.34098e+08	3927	2.47426e+09	21745	4.5661e+09	50217		

Figure 5.12: 3-POINT RESULTS FOR GALACTIC SIMULATION DATA, BINS 1-32.

The empty slots correspond to runs which were stopped because they were too time consuming. Thus the remaining handful of bins were essentially left for uncomputable. We applied the shattered Monte Carlo algorithm to these bins and were able to achieve these computations.

Bin		RRR
	Count	Time
31	3.61275e+09	96
32	4.40414e+09	68
33	4.5193e+09	82
34	2.37094e+10	61
35	2.55316e+10	83
36	2.56899e+10	87

Figure 5.13: 3-POINT RESULTS FOR GALACTIC SIMULATION DATA USING MONTE CARLO, BINS 32-36.

5.6.3 Scaling with dataset size.

To study the scaling of the exact algorithms with dataset size N, we created subsets of the mock galaxy catalog of increasing size. To test the exact shattering algorithm, one of the smaller bin sizes (bin 3, 0.0271644 - 0.0368129) was used for the subsequent experiments. We tested the scaling for $n = 2$, 3, and 4.

Data = Mock catalog, DD	
N	Time
15625	0.3
31250	0.5
62500	1
125k	2
250k	5
500k	10
1M	23

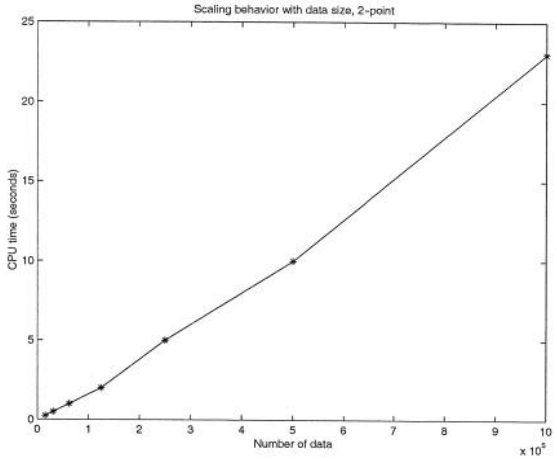

Figure 5.14: SCALING WITH DATASET SIZE, 2-POINT. Numerical values and plot.

Data = Mock catalog, DDD	
N	Time
15625	0.5
31250	1
62500	3
125k	13
250k	46
500k	169
1M	434

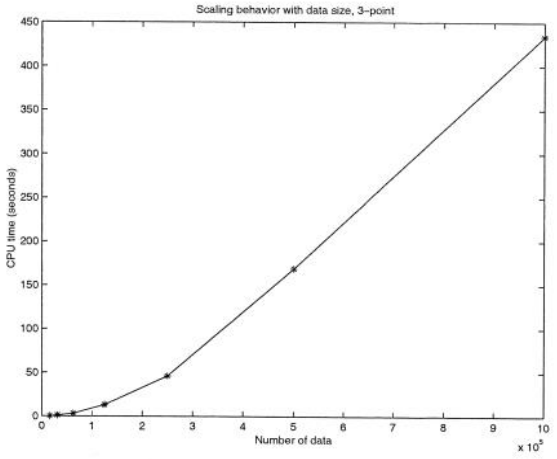

Figure 5.15: SCALING WITH DATASET SIZE, 3-POINT.

Data = Mock catalog, DDDD	
N	Time
15625	1
31250	3
62500	18
125k	88
250k	359
500k	1024
1M	4574

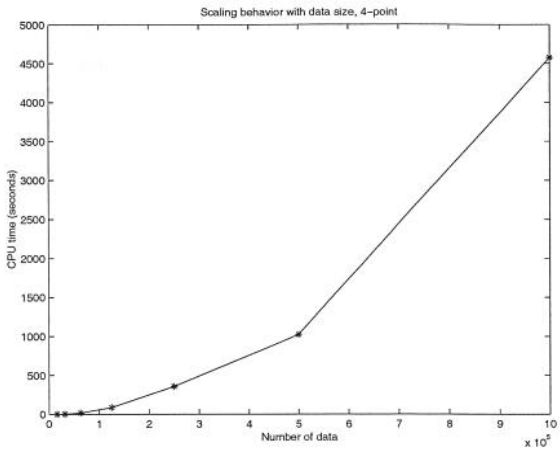

Figure 5.16: SCALING WITH DATASET SIZE, 4-POINT. Numerical values and plot.

The independence of the Monte Carlo method's runtime from the dataset size is demonstrated in Figure 5.17, where we compare 3-point runs on two datasets: the random ('R') galactic simulation dataset and a randomly-chosen 10%-size subset of it. The bin setting is 'bin 31', one that was particularly time-consuming for the exact shattering method.

N	$\hat{p}$	Exact Shattering Time	Monte Carlo Time 10% err	Monte Carlo Time 5% err	Monte Carlo Time 1% err
19103	0.003813	164	64	179	2585
191034	0.003877	33658	65	183	2470

Figure 5.17: EFFECT OF DATASET SIZE FOR MONTE CARLO.

Although the data is one-tenth the size, for the same bin setting its *proportion* of matching n-tuples is the same — in fact the answer count in both cases is nearly identical (about 3.6×10^6). We observe that the runtimes are comparable, demonstrating that the effect of $\hat{p}$ dominates for the Monte Carlo method, rather than N.

For the shattered Monte Carlo algorithm, only its exact shattering phase is affected by the dataset size. Thereafter the runtime, as we explained earlier, is only dependent on $\hat{p}$. The dependence on N observed in the graph is due to the use of the data-fraction heuristic used to choose the number of node expansions at which to stop the exact algorithm. If left as a user parameter and kept constant, the CPU time will be constant as we've seen.

Data = Mock catalog, DD		
N	$\hat{p}$	Time
15625	0.0069567	4
31250	0.0086097	6
62500	0.0109024	7
125k	0.0099195	11
250k	0.0098757	14
500k	0.0099057	20
1M	0.0099417	23

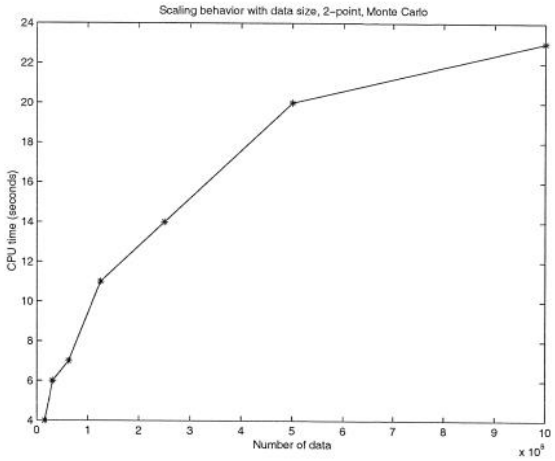

Figure 5.18: SCALING WITH DATASET SIZE, 2-POINT, MONTE CARLO. Numerical values and plot.

5.6.4 Scaling with tuple order.

To test the scaling of the shattering algorithm with tuple order n, we used the smallest bin size (bin 1, 0.63000 - 1.99800) and varied n from 2 to 5.

n	Count	Time
2	75195	1
3	126042	8
4	241459	74
5	3.96537e+13	6107

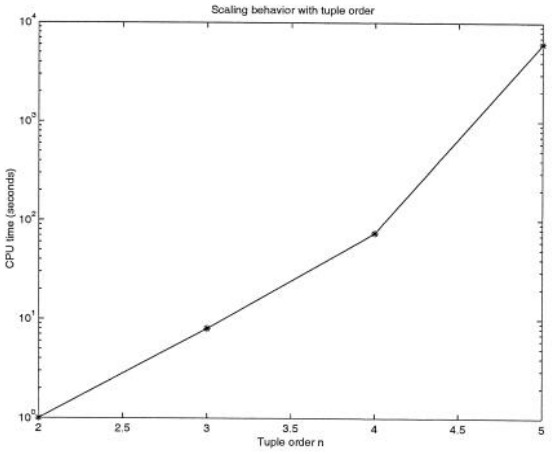

Figure 5.19: SCALING WITH TUPLE ORDER.

The next figure re-plots the experimental timings of 5.6.3 to display the dataset-size-scaling curves for $n = 2$, 3 and 4 on the same axes for comparison of their growths with N.

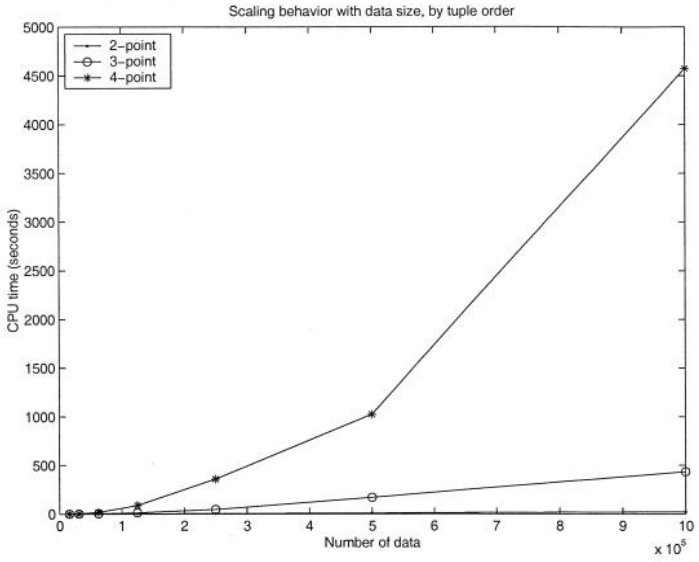

Figure 5.20: SCALING WITH NUMBER OF DATA, BY TUPLE ORDER.

5.6.5 Scaling with bin size.

We have seen that the size of the bin, which is also related to the size of the true count to be computed, has a significant on the computational cost of the algorithm. For the mock galaxy catalog, we plot the runtime as a function of the bin size.

First the DD and RR cases of the 2-point are shown:

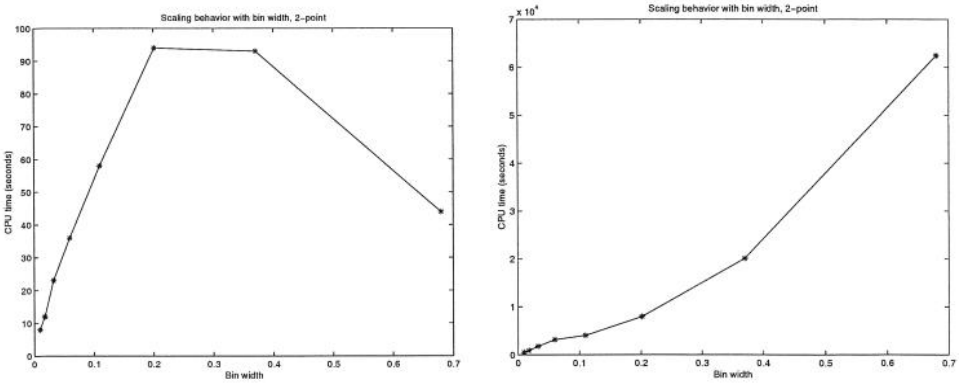

Figure 5.21: SCALING WITH BIN WIDTH, 2-POINT, DD CASE AND RR CASE.

In general a larger bin size implies larger counts, and thus higher computational cost for the algorithm. However, we see that at some point the count begins to drop, due to the edge effects of a limited sample for very large bin sizes.

A similar effect can be seen in the 3-point results:

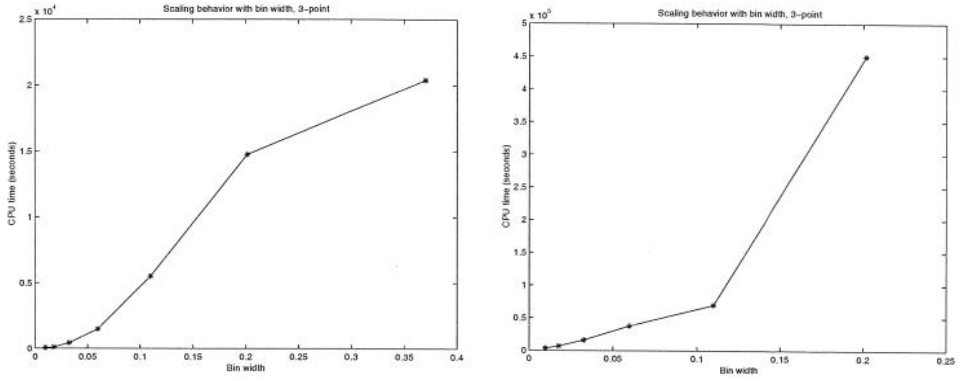

Figure 5.22: SCALING WITH BIN WIDTH, 3-POINT, DDD CASE AND RRR CASE.

To consider the effect of bin size for the galactic simulation data, recall that each bin was defined by a fully-parametrized triangle in those experiments. Thus we can no longer easily consider bin size as a function of one variable, as we could in the previous one-parameter equilateral triangle case. Thus we use a bar graph in the next figure, to emphasize that the horizontal axis does not correspond to a usual ordering. The up-and-down structure in the bars is due to the cyclic way in which the triangle parameters were chosen.

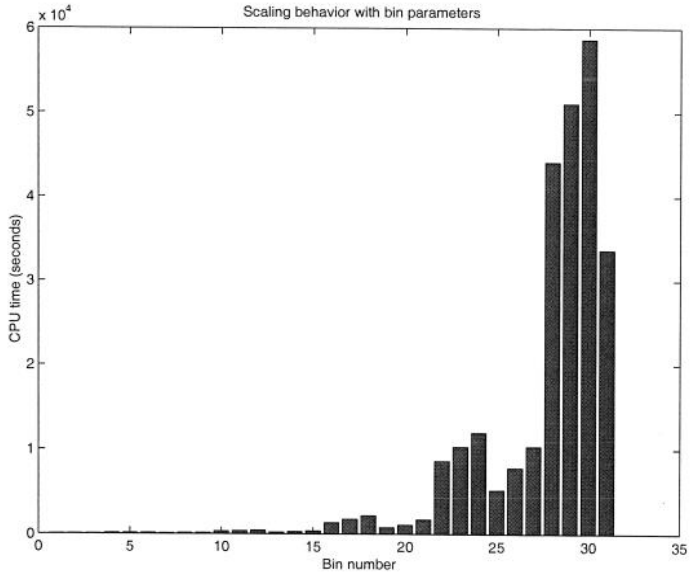

Figure 5.23: SCALING WITH BIN PARAMETERS, 3-POINT, GALACTIC SIMULATION. Shown as a bar graph to display the pattern more clearly.

5.6.6 Scaling with error tolerance.

For the approximate shattered Monte Carlo algorithm, we test the scaling of runtime with the user-supplied error tolerance ϵ. We again consider a 3-point run using the random ('R') galactic simulation catalog, and bin 31. The true answer for this example is 3.60121×10^9. The plot shows the maximum error of the algorithm, with a 99% confidence interval, reported each round of sampling as the algorithm runs.

$\hat{c}$	$[\hat{c}^{lo}, \hat{c}^{hi}]$	Error	Time
4.66e+011	[0.00, 9.32e+011]	4.65e+011	33
3.60e+09	[3.03e+09, 4.16e+09]	0.186	34
3.60e+09	[3.04e+09, 4.16e+09]	0.183	35
3.61e+09	[3.06e+09, 4.16e+09]	0.178	36
3.62e+09	[3.09e+09, 4.15e+09]	0.170	39
3.62e+09	[3.12e+09, 4.12e+09]	0.158	43
3.61e+09	[3.16e+09, 4.06e+09]	0.141	51
3.61e+09	[3.22e+09, 3.99e+09]	0.118	66
3.61e+09	[3.30e+09, 3.92e+09]	0.093	95
3.60e+09	[3.36e+09, 3.84e+09]	0.070	153
3.59e+09	[3.41e+09, 3.76e+09]	0.051	266
3.58e+09	[3.45e+09, 3.71e+09]	0.036	487
3.58e+09	[3.49e+09, 3.67e+09]	0.025	921
3.58e+09	[3.51e+09, 3.64e+09]	0.018	1793
3.58e+09	[3.53e+09, 3.62e+09]	0.012	3514

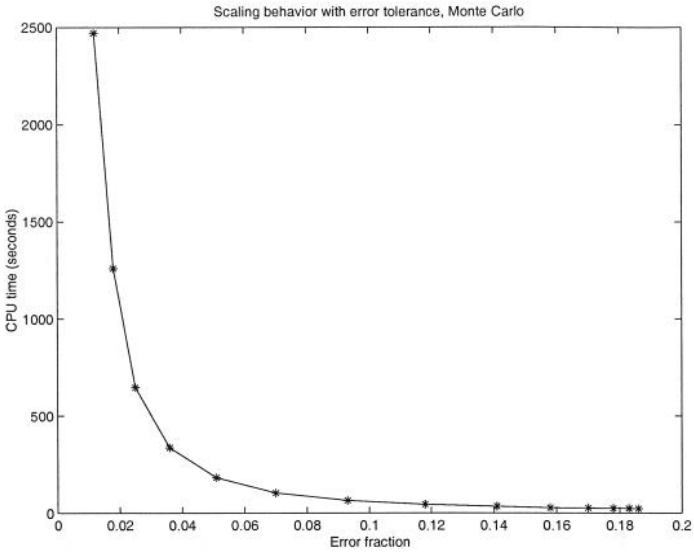

Figure 5.24: SCALING WITH ERROR TOLERANCE.

Figure 5.25 depicts the progression of the lower and upper bounds versus time.

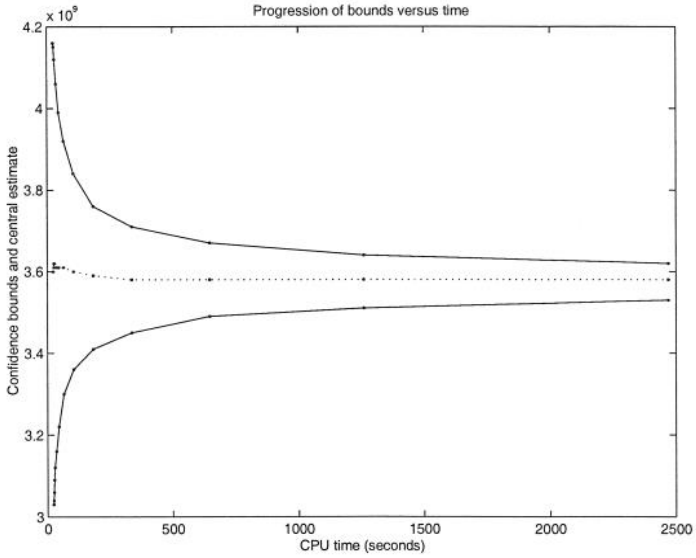

Figure 5.25: PROGRESSION OF BOUNDS VERSUS TIME.

5.6.7 Effect of stratification. We test the effect of increased stratification (described in 5.4.2) on the runtime using the same example. Figure 5.26 plots the runtime, at 10% error, versus the number of strata.

Number of Expansions	Number of Strata	Exact Shattering Time	Total Time 10% err
4263	707	1	1696
15790	5204	2	833
77239	19656	7	218
231636	60787	33	95

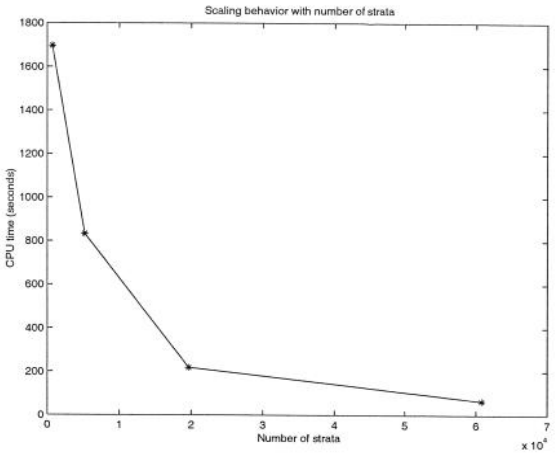

Figure 5.26: EFFECT OF STRATIFICATION.

5.6.8 Effect of defragmentation. In the same example we test the effect of defragmentation (described in 5.5.3). We observe roughtly a factor of 2 in speedup over the method without defragmentation.

$\hat{c}$	$[\hat{c}^{lo}, \hat{c}^{hi}]$	Error	Time
4.66e+11	[0.00, 9.32e+11]	4.65e+11	22
3.82e+09	[3.48e+09, 4.16e+09]	0.097	23
3.83e+09	[3.49e+09, 4.17e+09]	0.096	24
3.84e+09	[3.51e+09, 4.17e+09]	0.094	25
3.84e+09	[3.52e+09, 4.16e+09]	0.091	26
3.83e+09	[3.53e+09, 4.14e+09]	0.086	29
3.81e+09	[3.53e+09, 4.09e+09]	0.078	35
3.76e+09	[3.52e+09, 3.99e+09]	0.067	45
3.71e+09	[3.51e+09, 3.90e+09]	0.055	65
3.65e+09	[3.50e+09, 3.79e+09]	0.042	106
3.62e+09	[3.50e+09, 3.73e+09]	0.031	184
3.59e+09	[3.51e+09, 3.67e+09]	0.022	339
3.59e+09	[3.53e+09, 3.65e+09]	0.016	651
3.58e+09	[3.54e+09, 3.62e+09]	0.011	1280

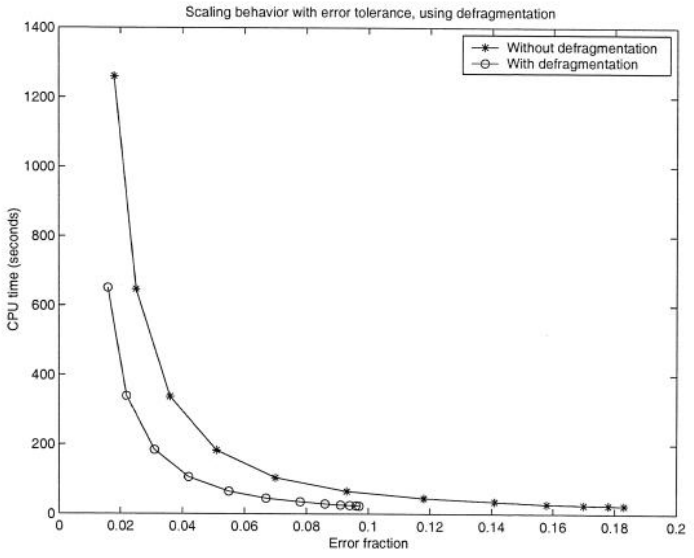

Figure 5.27: EFFECT OF DEFRAGMENTATION.

§5.7 Related problems and approaches.

5.7.1 Grid-based methods.

The chief non-FFT computational approaches to the n-point problem *e.g.* [Sza97] have been grid-based approaches, suffering all of the problems of grids which we have already discussed. Statistically these approaches technically are not even the same as the direct n-point counts – instead, they compute 'counts-in-cells', which are different quantities with different (and less desirable) statistical properties.

5.7.2 The power spectrum method.

With only these grid-based approaches in hand, the prospect of making progress using the n-point functions has seemed fairly hopeless. For this reason, practical and theoretical attention in astrophysics turned to the Fourier-space, or *power spectrum* versions of the correlation functions so that the FFT could be applied to the computational problem. However, the Fourier representation is wracked by severe problems making its use limited and difficult at best, most notably the debilitating Gibbs phenomenon (ringing in Fourier space) caused by edge effects, on top of the accuracy issues we have discussed in previous chapters. The power spectrum approach only exists because tractability has not seemed to be forthcoming for the true correlation functions.

5.7.3 Computational Geometry with n-Tuples

There does not appear to be any previous work in computational geometry concerning problems which involve n-tuples, for $n > 2$.

5.7.4 Geometric Monte Carlo

The idea of using space-partitioning trees to perform stratification has been used before. The two previous instances of this kind of algorithm [PF90, FW81] were both proposed in the general setting of Monte Carlo integration. In these approaches the process of stratification did not contribute directly to the solution, as it does in our case through the exact tightening of bounds by exact pruning methods.

5.7.5 Binomial confidence bounds

Many authors have considered the fundamental question of designing confidence bounds for the binomial distribution *e.g.* [BS83, AC98, BCD01].

This has been an instance in which work on confidence bounds has interacted with work on a Monte Carlo method; in general we conjecture that other such interactions could prove fruitful.

§5.8 Chapter summary

Let us now summarize the activities of this Chapter:

- **n-tuples with exact computation.** We developed an approach to n-tuple problems, the most daunting of all of our N-body problems, using the natural extension of the shattering principle. The increase in sophistication and efficiency of this approach over previous solution attempts is larger than we saw in the pairwise case.
- **Monte Carlo for N-body problems.** In the most difficult n-tuple context, we showed how to use shattering as the basis for a new kind of Monte Carlo approach which is also applicable to many other N-body problems.
- **General sampling techniques.** Some of the techniques we developed for this problem extend to the most general Monte Carlo context, most notably adaptive Neyman allocation and its various associated mechanisms.

- **n-point correlation problems and real solutions.** We reviewed the key theoretical aspects making the n-point correlation problems fundamental and the key practical constraints under which a solution attempt must operate efficiently. We then developed a solution meeting these criteria for the first time, via a number of specialized techniques.

What's next?

Now we are done! So it's time to summarize everything we did, in the next and last chapter.

6 Summary and Outlook

A Retrospective and Prospective.

Ah Love! could thou and I with Fate conspire
To grasp this sorry Scheme of Things entire,
Would we not shatter it to bits — and then
Re-mould it nearer to the Heart's Desire!
— Edward FitzGerald (1809 - 1883), in *Omar Khayym* (1859) xcix.

Now let's take a final overarching look at what we've done, how it affects the landscape of the fields and problems we've treated, and therefore what remains to be done.

§6.1 New understanding of important problems.

Much of research, particularly engineering research, is about carving out the right problems for researchers to focus intellectual resources upon. This is what we did along these lines:

1. **Delineating a subspace of computational problems.** We defined for first time a class of problems which covers many disparate fields but can be treated by similar methods — this problem subspace includes several well-known or 'high-profile' problems. This unified approaches and knowledge across several fields regarding these kinds of problems, and points out how progress in one field might implicitly mean progress in another, or how progress on one problem might mean progress on a whole set of problems at the same time.

2. **Some new combinatorial problems.** Generalizing from the n-point correlation problems, we defined a new class of computational geometry problems which is interesting and natural in its own right, with several potential applications. The fact that it has not appear to have been discussed previously is perhaps due to the fact that it was too difficult to have been realistically considered to date.

§6.2 New tractability for important problems.

Since the beginning of this thesis we kept our eye on developing general mechanisms for solving the entire range of N-body problems we listed in Chapter 1. We chose a few key problem aspects to focus on, and associated them with specific open computational problems - one from each of four different fields.

For two of these problems, our methodology represented a dramatic advance over the previous state-of-the-art solutions:

1. **Efficient practical kernel estimation.** The leap over the previous gridding and FFT methods is directly analogous to the leap made by the Barnes-Hut

and FMM methods in the N-body problems of computational physics. Our solution is the first to be able to treat all of the practicalities of the kernel estimation problem.

2. **Efficient practical n-point correlations.** For this set of problems the gain in efficiency afforded by our methods is substantial. Our solutions allows both exact and even-faster probabilistic computations on unprecedented scales which have made new astrophysical investigations possible for the first time.

For the other two focus problems, our methodology yielded new solutions with certain advantages over previous ones, though useful solutions already existed for these problems:

1. **Derivative-free linear-time smoothed particle hydrodynamics.** Tree-based methods based on the $O(N \log N)$ Barnes-Hut monopole method as well as $O(N)$ multipole methods are applicable to this problem, but both have key disadvantages. Our method is the first approach which is both derivative-free, thus avoiding the severe implementation and applicability barriers of multipole methods, and linear-time unlike the monopole methods.

2. **Arbitrary-dimension linear-time all-nearest-neighbors.** We give a pure-recursive or priority-queue-based method which runs in expected $O(N)$ time in arbitrary dimensions. It extends [Cal95, HS98] by avoiding the unnecessary restrictions of an intermediate data structure (thus a constant factor more efficient) and a priority queue formulation, respectively. Since these are relatively minor improvements, the main contribution in this case is insight into the advantage of the node-node approach over the point-node approach, or higher-order divide-and-conquer over standard divide-and-conquer.

§6.3 New algorithmic techniques.

Of course, as per our original intention, it is possible to transfer our methods for these focus problems to other N-body problems sharing the same key aspects. We developed a number of new algorithmic techniques which generalize to other problems to various extents:

1. **Higher-order divide-and-conquer.** We extended the computer science toolbox with a major new variant of the divide-and-conquer strategy which does not seem to have been delineated previously. This new design principle has yielded some new algorithms which are conceptually unlike any existing algorithms. We explored and demonstrated its computational-geometric form, **geometric shattering**, in particular, but anticipate that its more general application will find utility in problems far beyond the N-body scope.

2. **Finite-difference approximation.** We designed a finite-difference approach which can be used in the context of geometric divide-and-conquer approach for nearly any continuous kernel function arising in typical N-body problems.

3. **Multiple and optimal scales.** We showed how to extend our methods so that N-body problems over multiple scales ('bandwidths') can be solved efficiently. We also gave an approach using this technique for efficiently finding the optimal scale.

4. **Variable and time-varying scales.** We developed approaches for these issues, which appear in other N-body problems, based on an additional application of the higher-order divide-and-conquer principle.

5. **Up-down propagation.** We designed an additional methodology based on asynchronous dynamic programming which accelerates basic shattering.

6. **Multipole-geometric hybrids.** We suggested new algorithmic possiblities which can be realized by combining the geometric insights developed here with the standard multipole methods currently used in Coulombic N-body problems. These hybrid approaches would have certain advantages which are not possible with current approaches. Note, however, that these methods remain to be implemented and experimentally validated.

7. **n-tuple shattering.** We showed to extend the geometric shattering principle beyond pairwise N-body problems to the case of general n-tuples, opening up the possiblility of approaching an entire new class of problems.

8. **Shattered Monte Carlo.** We showed how shattering can be used in concert with Monte Carlo integration methods to obtain very fast probabilistic solutions to even the hardest N-body problems.

9. **Adaptive Neyman allocation.** We demonstrated a new approach to sample allocation which asymptotically achieves optimal allocation, and is applicable in the most general Monte Carlo integration setting.

10. **Defragmentation.** We developed a mechanism for accelerating stratified sampling in probability ranges where binomial confidence bounds are poor.

As a whole, this collection of algorithmic techniques, along with a collective understanding of generalized N-body problems in terms of key computational variations, might be aptly called an '**algorithmic N-body theory**': a constructive theory giving the existence, analysis, and implementation of efficient multi-recursive algorithms for generalized N-body problems. In practical terms it forms the beginning of a 'design handbook' for algorithms for such problems, which we hope will find utility even beyond the moderate sample of N-body problems we have listed in this thesis.

	Grids	FFT	Barnes-Hut	FMM	ASPT+HODC
Large N?	–	+	++	+++	+++
Arbitrary D?	–	–	–	–	++
Control error?	+	+	+	++	+++
Known error?	–	–	–	+++	+++
Not obtuse?	+++	–	+++	–	+++
General?	+++	–	+++	–	+++

Table 6.1: FINAL COMPARISON OF NEW ALGORITHMIC FRAMEWORK TO EXISTING SOLUTION CLASSES FOR GENERALIZED N-BODY PROBLEMS. The number of '+' symbols indicates the extent to which the property in question is satisfied. '–' indicates failure to satisfy the property.

Finally, our deliberate focus on key practical properties allows us to now return to the solution scorecard from Chapter 1, revealing a favorable comparison to the array of existing major solution classes.

§6.4 Where to go next?

At least the following handful of directions are immediately suggested by the work in this thesis.

- *SPH implementation.* The SPH algorithm remains to be implemented and validated in the context of astrophysical fluid dynamics simulations with our VIRGO Project collaborators.
- *Projected n-point.* The *projected* n-point correlation is a variant which works in the projected plane of observation rather than the three-dimensional sky coordinates. An extension of our method to this case would provide a significant benefit to astrophysicists.
- *Geometric structures.* It may still be possible that some of the many lesser-known variants of space-partitioning trees or new extensions of, say, ball-trees might provide some efficiency advantage. Also, structures which have good properties in low dimensions might be useful in physics problems.
- *Hilbert-space kernels.* The extension of our methods from metric spaces to Hilbert space could open up several new applications in the area of machine learning known as 'kernel methods', which include the well-known support vector machines.
- *Automatic algorithm derivation.* In this thesis we constructed a unified and generalized view of a certain class of problems and algorithmic solutions for them. The next level of generalization would formalize the problems and the solutions to the extent that given a problem specification, the appropriate algorithm from the geometric shattering class of solutions could be automatically derived and implemented. A brief attempt at this was already begun in work leading to [GFSB03].
- *New theoretical ideas.* Lastly, with their newfound computational feasbility, we anticipate that the mathematical tools we have sped up, especially the fundamental ideas of kernel estimation and n-point correlation, will find new application as sub-routines within other techniques. The increased utility of the basic tools will in turn provoke theoretical extensions to the ideas themselves.

References

NIPS abbreviates the proceedings volumes titled *Advances in Neural Information Processing Systems*.

[AC98] A. Agresti and B. A. Coull.
Approximate is better than 'exact' for interval estimation of binomial proportions.
The American Statistician, 52:119–126, 1998.

[AE97] P. K. Agarwal and J. Erickson.
Geometric Range Searching and its Relatives.
Technical report, Duke University, 1997.

[AED01] R. Duraiswami A. Elgammal and L. S. Davis.
Efficient Kernel Density Estimation Using the Fast Gauss Transform with Applications to Segmentation and Tracking.
In *Proceedings of the Second International Workshop on Statistical and Computational Theories of Vision*, 2001.

[AM93] S. Arya and D. Mount.
Approximate Nearest Neighbor Queries in Fixed Dimensions.
In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 271–280, 1993.

[AM95] S. Arya and D. Mount.
Approximate Range Searching.
In *Proceedings of the Eleventh Annual ACM Symposium on Computational Geometry*, pages 172–181, 1995.

[AMN+94] S. Arya, D. Mount, N. Netanyahu, R. Silverman, and A. Wu.
An Optimal Algorithm for Approximate Nearest Neighbor Searching Fixed Dimensions.
In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 573–582, 1994.

[AMN+98] S. Arya, D. Mount, N. Netanyahu, R. Silverman, and A. Wu.
An Optimal Algorithm for Approximate Nearest Neighbor Searching Fixed Dimensions.
Journal of the ACM, 45(6):891–923, 1998.

[And92] C. R. Anderson.
An implementation of the fast multipole method without multipoles.
SIAM Journal of Scientific and Statistical Computing, 13(4):923–947, 1992.

[App81] A. W. Appel.
An Asymptotically Fast Algorithm for N-Body Simulations.
Senior Thesis, Princeton University, Computer Science Department, 1981.

[App85] A. W. Appel.
An Efficient Program for Many-Body Simulations.
SIAM Journal on Scientific and Statistical Computing, 6(1):85–103, 1985.

[AS72] M. Abramowitz and I.A. Stegun.
Handbook of Mathematical Functions.
Dover Publications, 1972.

[AY00] C. C. Aggarwal and P. Yu.
The IGrid Index: Reversing the Dimensionality Curse for Similarity Indexing in High Dimensional Space.
In *Proceedings Sixth International Conference on Knowledge Discovery and Data Mining*. ACM, 2000.

[BCD01] L. D. Brown, T. Cai, and A. DasGupta.
Interval estimation for a binomial proportion.
Statistical Science, 16(2):101–133, 2001.

[Ben75] J. L. Bentley.
Multidimensional Binary Search Trees used for Associative Searching.
Communications of the ACM, 18:509–517, 1975.

[Ben80] J. L. Bentley.
Multidimensional Divide and Conquer.
Communications of the ACM, 23(4):214—229, 1980.

[Ber92] F. Bernardeau.
The gravity induced quasi-gaussian correlation hierarchy.
Astrophysics Journal, 392, 1992.

[BF96] G. J. Babu and E. D. Feigelson.
Astrostatistics: Interdisciplinary Statistics.
Chapman and Hall, 1996.

[BH86] J. Barnes and P. Hut.
A Hierarchical $O(NlogN)$ Force-Calculation Algorithm.
Nature, 324, 1986.

[Bis95] C. Bishop.
Neural Networks for Pattern Recognition.
Oxford University Press, Oxford, 1995.

[BKS93] T. Brinkhoff, H. P. Kriegel, and B. Seeger.
Efficient Processing of Spatial Joins Using R-trees.
In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*. ACM, 1993.

[BN97] Guy Blelloch and Girija Narlikar.
A practical comparison of n-body algorithms.
In *Parallel Algorithms*, Series in Discrete Mathematics and Theoretical Computer Science. American Mathematical Society, 1997.

[BN98] R. K. Beatson and G. N. Newsam.
Fast evaluation of radial basis functions: Moment-based methods.
SIAM Journal of Scientific Computing, 19(5):1428–1449, 1998.

[Boh00] Christian Bohm.
A Cost Model for Query Processing in High Dimensional Data Spaces.
ACM Transactions on Database Systems, 25(2):129–178, 2000.

[Bot73] C. J. F. Bottcher.
Theory of Electric Polarization, 2e.
Elsevier, 1973.

[Bow85] A.W. Bowman.
A Comparative Study of Some Kernel-Based Nonparametric Density Estimators.
Journal of Statistical Computation and Simulation, 21:313–327, 1985.

[BS76] Jon Louis Bentley and Michael Ian Shamos.
Divide-and-Conquer in Multidimensional Space.
In *Proceedings of the Eighth Annual ACM Symposium on Theory of Computing*, pages 220–230, 1976.

[BS78] Jon Louis Bentley and Michael Ian Shamos.
Divide-and-Conquer in Multidimensional Space.
Information Processing Letters, 7:87–91, 1978.

[BS83] C. R. Blyth and H. A. Still.
Binomial confidence intervals.
Journal of the American Statistical Association, 78(381):108–116, March 1983.

[Cal95] P.B. Callahan.
Dealing with Higher Dimensions: The Well-Separated Pair Decomposition and its Applications.
PhD thesis, Johns Hopkins University, Baltimore, Maryland, 1995.

[CB90] G. Casella and R. L. Berger.
Statistical Inference.
Duxbury Press, 1990.

[CE85] B. M. Chazelle and H. Edelsbrunner.
Optimal Solutions for a Class of Point Retrieval Problems.
Journal of Symbolic Computation, 1:47–56, 1985.

[Cha99] B. Chazelle.
Application Challenges to Computational Geometry.
Advances in Discrete and Computational Geometry, Contemporary Mathematics, 223:407–463, 1999.

[CHH85] R. C. Y. Chin, G. W. Hedstrom, and F. A. Howes.
Considerations on Solving Problems with Multiple Scales.
Academic Press, 1985.

[Cla83] K. Clarkson.
Fast Algorithms for the All Nearest Neighbors Problem.
In *Proceedings of the Twenty-fourth Annual IEEE Symposium on the Foundations of Computer Science*, pages 226–232, 1983.

[Cla87] K. L. Clarkson.
New Applications of Random Sampling in Computational Geometry.
Discrete and Computational Geometry, 2:195–222, 1987.

[Cla88] K. L. Clarkson.
A Randomized Algorithm for Closest-point Queries.
SIAM Journal of Computing, 17:830–847, 1988.

[Cla02] K. Clarkson.
Nearest Neighbor Searching in Metric Spaces: Experimental Results for sb(S).
2002.

[CNBYM01] E. Chavez, G. Navarro, R. Baeza-Yates, and J. L. Marroquin.
Proximity Searching in Metric Spaces.
ACM Computing Surveys, 33:273–321, 2001.

[CPZ98] P. Ciaccia, M. Patella, and P. Zezula.
A Cost Model for Similarity Queries in Metric Spaces.
In *Proceedings of the Seventeenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems : PODS 1998*. ACM Press, 1998.

[CS89] K. L. Clarkson and P. W. Shor.
Applications of Random Sampling in Computational Geometry II.
Discrete and Computational Geometry, 4:387–421, 1989.

[CT92] Yi-Jen Chiang and Roberto Tamassia.
Dynamic algorithms in computational geometry.
Proceedings of IEEE, Special Issue on Computational Geometry, 80(9):362–381, 1992.

[dBvKOS99] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf.
Computational Geometry: Algorithms and Applications, 2e.
Springer-Verlag, 1999.

[DG85] Luc Devroye and Laszlo Gyorfi.
Nonparametric Density Estimation: The L_1 View.
Wiley, 1985.

[DH73] R. O. Duda and P. E. Hart.
Pattern Classification and Scene Analysis.
John Wiley & Sons, 1973.

[DKG92] C. Ding, N. Karasawa, and W. A. Goddard.
Atomic level simulations of a million particles: The cell multipole method for coulomb and london interactions.
Journal of Chemical Physics, 97:4309–4315, 1992.

[DL76] D. Dobkin and R. J. Lipton.
Multidimensional Searching Problems.
SIAM Journal of Computing, 5:181–186, 1976.

[DL01] Luc Devroye and Gabor Lugosi.
Combinatorial Methods in Density Estimation.
Springer-Verlag, 2001.

[DM95] K. Deng and A. W. Moore.
Multiresolution Instance-based Learning.
In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 1233–1239, San Francisco, 1995. Morgan Kaufmann.

[DVJ72] D. J. Daley and D. Vere-Jones.
A Summary of the Theory of Point Processes.
In P. A. W. Lewis, editor, *Stochastic Point Processes*, pages 299–383. John Wiley & Sons, 1972.

[Epa69] V. A. Epanechnikov.
Nonparametric Estimation of a Multidimensional Probability Density.
Theory of Probability and its Applications, 14:153–158, 1969.

[Epp98] David Eppstein.
Fast hierarchical clustering and other applications of dynamic closest pairs.

In *SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*, 1998.

[Epp99] D. Eppstein.
Bibliography for ICS 280.
UCI, 1999.
`http://www.ics.uci.edu/ eppstein/280/bib.html`.

[EPY00] D. Eppstein, M. S. Paterson, and F. F. Yao.
On Nearest-Neighbor Graphs.
2000.

[Ewa21] P. P. Ewald.
Die berechnung optischer und elektrostatischer gitterpotentiale.
Ann. Physik, 64, 1921.

[FBF77] J. H. Friedman, J. L. Bentley, and R. A. Finkel.
An algorithm for finding best matches in logarithmic expected time.
ACM Transactions on Mathematical Software, 3(3):209–226, September 1977.

[Fef00] C. L. Fefferman.
Existence and Smoothness of the Navier-Stokes Equation.
2000.

[FH91] S. Fortune and J. E. Hopcroft.
metric trees.
Information Processing Letters, 40:175–179, 1991.

[FK94] C. Faloutsos and I. Kamel.
Beyond Uniformity and Independence: Analysis of R-trees Using the Concept of Fractal Dimension.
In *Proceedings of the 13th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems : PODS 1994*. ACM Press, 1994.

[FO02] C. S. Frenk and Others.
The Santa Barbara Cluster Comparison Project: A Comparison of Cosmological Hydrodynamics Solutions.
The Astrophysical Journal, 525(2):554–582, 2002.

[Fre81] M. L. Fredman.
A Lower Bound on the Complexity of Orthogonal Range Queries.
Journal of the ACM, 28:696–705, 1981.

[Fry84] J. N. Fry.
The galaxy correlation hierarchy in perturbation theory.
Astrophysics Journal, 279, 1984.

[Fry86] J. N. Fry.
On Statistical Searches for Filaments.
The Astrophysical Journal, 306:366–373, 1986.

[FW81] J. H. Friedman and M. H. Wright.
A nested partitioning procedure for numerical multiple integration.
ACM Transactions on Mathematical Software, 7(1):76–92, 1981.

[GBT84] H. N. Gabow, J. L. Bentley, and R. E. Tarjan.
Scaling and Related Techniques for Geometry Problems.
In *Proceedings of the Sixteenth Annual ACM Symposium on the Theory of Computing*, pages 135–143, 1984.

[GFSB03] A. Gray, B. Fischer, J. Schumann, and W. Buntine.
Automatic Derivation of Statistical Algorithms: The EM Family and Beyond.
In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems 15 (December 2002)*. MIT Press, 2003.

[GG98] V. Gaede and Oliver Gunther.
Multidimensional Access Methods.
ACM Computing Surveys, 30(2), 1998.

[GH03] M. Girolami and C. He.
Probability Density Estimation from Optimally Condensed Data Samples.
IEEE Transactions Pattern Analysis and Machine Intelligence, 2003.

[GHWS91] H. Grubmller, H. Heller, A. Windemuth, and K. Schulten.
Generalized Verlet Algorithm for Efficient Molecular Dynamics Simulations with Long-Range Interactions.
Molecular Simulation, 6:121–142, 1991.

[GIM99] A. Gionis, P. Indyk, and R. Motwani.
Similarity Search in High Dimensions via Hashing.
In *Proc 25th VLDB Conference*, 1999.

[Gio97] N. Giordano.
Computational Physics.
Prentice-Hall, 1997.

[GM77] R. A. Gingold and J. J. Monaghan.
Smoothed Particle Hydrodynamics: Theory And Application to Non-Spherical Stars.
Monthly Notices of the Royal Astronomical Society, 181:375–389, 1977.

[GP83] P. Grassberger and I. Procaccia.
Measuring the Strangeness of Strange Attractors.
Physica D, pages 189–208, 1983.

[GR87] L. Greengard and V. Rokhlin.
A Fast Algorithm for Particle Simulations.
Journal of Computational Physics, 73, 1987.

[GS91] L. Greengard and J. Strain.
The Fast Gauss Transform.
SIAM Journal of Scientific and Statistical Computing, 12(1):79–94, 1991.

[Ham50] J. M. Hammersley.
The Distribution of Distances in a Hypersphere.
Annals of Mathematical Statistics, 21:447–452, 1950.

[HE88] R. W. Hockney and J. W. Eastwood.
Computer Simulation Using Particles.
Institute of Physics/Adam Hilger, 1988.

[HHv74] J. D. F. Habbema, J. Hermans, and K. van der Broek.
A Stepwise Discrimination Program Using Density Estimation.
In G. Bruckman, editor, *Computational Statistics*, pages 100–110. Physica Verlag, 1974.

[HJR97] Y. W. Huang, N. Jing, and E. A. Rundensteiner.
Spatial Joins Using R-trees: Breadth-first Traversal with Global Optimizations.
In *Proc 23rd VLDB Conference*, pages 396–405, 1997.

[HK89] Lars Hernquist and Neal Katz.
Treesph: A Unification of SPH with the Hierarchical Tree Method.

Astronomy and Astrophysics Supplemental Series, 70:419–446, June 1989.

[HL56] J. L. Hodges and E. L. Lehmann.
The Efficiency of Some Nonparametric Competitors of the t-Test.
Annals of Mathematical Statistics, 27:324–335, 1956.

[HMS01] David J. Hand, Heikki Mannila, and Padhraic Smyth.
Principles of Data Mining.
MIT Press, 2001.

[HS98] G. R. Hjaltason and H. Samet.
Incremental Distance Join Algorithms for Spatial Databases.
In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*. ACM, 1998.

[JMS96] M.C. Jones, J.S. Marron, and S.J. Sheather.
A Brief Survey of Bandwidth Selection for Density Estimation.
Journal of the American Statistical Association, 91:401–407, 1996.

[Joh98] D. E. Johnson.
Applied Multivariate Methods for Data Analysis.
Duxbury Press, 1998.

[Kad00] L. P. Kadanoff.
Statistical Physics: Statics, Dynamics, and Renormalization.
World Scientific, 2000.

[Kle97] J. Kleinberg.
Two Algorithms for Nearest Neighbor Search in High Dimension.
In *Proceedings of the Twenty-ninth Annual ACM Symposium on the Theory of Computing*, pages 599–608, 1997.

[Kut95] R. Kutteh.
A simpler and more efficient formulation of the cell multipole method.
CCP5 Quarterly, February 1995.

[LHB+99] C. G. Lambert, S. E. Harrington, N. D. Bronson, C. R. Harvey, and A. Glodjo.
Efficient Online Non-Parametric Density Estimation.
Algorithmica, 1999.

[LQR97] G. Lake, T. Quinn, and D. C. Richardson.
From Sir Isaac to the Sloan Survey: Calculating the Structure and Chaos Owing to Gravity in the Universe.
In *ACM-SIAM Symposium on Discrete Algorithms*, pages 1–10, 1997.

[LS93] S. D. Landy and A. Szalay.
Bias and Variance of Angular Correlation Functions.
The Astrophysical Journal, 412, 1993.

[LT77] R. J. Lipton and R. E. Tarjan.
Application of a Planar Separator Theorem.
In *Proceedings of the Eighteenth Annual IEEE Symposium on the Foundations of Computer Science*, pages 162–170, 1977.

[Luc77] L. B. Lucy.
A Numerical Approach to the Testing of the Fission Hypothesis.
Astronomical Journal, 82:1013–1024, 1977.

[M. 91] M. van Kreveld and M. H. Overmars.
The divided k-d tree.
Algorithmica, 6:840–858, 1991.

[Man82] B. Mandelbrot.
Fractal Geometry of Nature.
W. H. Freeman & Company, 1982.

[Mei93] S. Meiser.
Point Location in Arrangements of Hyperplanes.
Information and Computation, 106:286–303, 1993.

[ML98] Andrew W. Moore and M. S. Lee.
Cached Sufficient Statistics for Efficient Machine Learning with Large Datasets.
JAIR, 8, March 1998.

[MM01] S. Maneewongvatana and D. M. Mount.
The Analysis of a Probabilistic Approach to Nearest Neighbor Searching.
In *Proceedings of WADS 2001*, 2001.

[Mon92] J. J. Monaghan.
Smoothed Particle Hydrodynamics.
Annual Review of Astronomy and Astrophysics, 30:543–74, 1992.

[Moo99] A. W. Moore.
Very fast mixture-model-based clustering using multiresolution kd-trees.
In M. Kearns and D. Cohn, editors, *Advances in Neural Information Processing Systems 10*, pages 543–549, San Francisco, April 1999. Morgan Kaufmann.

[Moo00] A. W. Moore.
The Anchors Hierarchy: Using the Triangle Inequality to Survive High-Dimensional Data.
In *Twelfth Conference on Uncertainty in Artificial Intelligence*. AAAI Press, 2000.

[MPT93] T. J. Martin, F. R. Pearce, and P. A. Thomas.
An Owner's Guide to Smoothed Particle Hydrodynamics.
1993.

[MS00] K. R. Mecke and D. Stoyan, editors.
Statistical Physics and Spatial Statistics: The Art of Analyzing and Modeling Spatial Structures and Pattern Formation.
Springer, 2000.

[MSK96] J. Mecke, D. Stoyan, and W. Kendall.
Stochastic Geometry and its Applications.
John Wiley & Sons, 1996.

[Nic03] R. Nichol.
Personal Communication.
, 2003.

[NT94] C. Niedermeier and P. Tavan.
A structure adapted multipole method for electrostatic interactions in protein dynamics.
Journal of Chemical Physics, 101:734–748, 1994.

[Omo87] S. M. Omohundro.
Efficient Algorithms with Neural Network Behaviour.
Journal of Complex Systems, 1(2):273–347, 1987.

[Omo89] S. M. Omohundro.
Five Balltree Construction Algorithms.
Technical Report TR-89-063, International Computer Science Institute, 1989.

[Omo90] S. M. Omohundro.
Geometric Learning Algorithms.
Physica D, 42:307–321, 1990.

[Omo91] S. M. Omohundro.
Bumptrees for Efficient Function, Constraint, and Classification Learning.
In R. P. Lippmann, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems 3*. Morgan Kaufmann, 1991.

[Pee80] P. J. E. Peebles.
The Large-Scale Structure of the Universe.
Princeton University Press, 1980.

[PF90] W. H. Press and G. R. Farrar.
Recursive stratified sampling for multidimensional monte carlo integration.
Computers in Physics, 4:190–195, 1990.

[PH77] F. P. Preparata and S. J. Hong.
Convex Hull of Finite Sets of Points in Two and Three Dimensions.
Communications of the ACM, 20(2):87–93, 1977.

[PM97] A. Papadopoulos and Y. Manolopoulos.
Performance of Nearest Neighbor Queries in R-trees.
In *Proceedings of the Zixth International Conference on Database Theory*, 1997.

[PM99] D. Pelleg and A. W. Moore.
Accelerating Exact k-means Algorithms with Geometric Reasoning.
In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*. ACM, 1999.

[PP62] W. K. H. Panofksy and M. Phillips.
Classical Electricity and Magnetism, 2e.
Addison-Wesley, 1962.

[Pri94] C. Priebe.
Adaptive Mixtures.
Journal of the American Statistical Association, 89:796–806, 1994.

[PS85] F. P. Preparata and M. Shamos.
Computational Geometry.
Springer-Verlag, 1985.

[Rab76] M. O. Rabin.
Probabilistic Algorithms.
In J. F. Traub, editor, *Algorithms and Complexity: New Directions and Recent Results*, pages 21–39. Academic Press, 1976.

[Rao83] B.L.S. Prakasa Rao.
Nonparametric Functional Estimation.
Academic Press, 1983.

[Ras99] Frederic A. Rasio.
Particle Methods in Astrophysical Fluid Dynamics.
In *Proceedings of the Fifth International Conference on Computational Physics*, 1999.

[Rip76] Brian D. Ripley.
Locally Finite Random Sets: Foundations for Point Process Theory.
Annals of Probability, 4:983–994, 1976.

[Rip81] Brian D. Ripley.
Spatial Statistics.
John Wiley & Sons, 1981.

[Rip88] Brian D. Ripley.
Statistical Inference for Spatial Processes.
Cambridge University Press, 1988.

[Rot91] D. Rotem.
Spatial Join Indices.
In *Proceedings of the Seventh International Conference on Data Engineering*, pages 500–509, 1991.

[Rub81] R. Y. Rubinstein.
Simulation and the Monte Carlo Method.
John Wiley & Sons, 1981.

[Rud82] M. Rudemo.
Empirical Choice of Histograms and Kernel Density Estimators.
Scandinavian Journal of Statistics, 9:65–78, 1982.

[Sam90] H. Samet.
The Design and Analysis of Spatial Data Structures.
Addison-Wesley, 1990.

[SC03] S. Shekhar and S. Chawla.
Spatial Databases: A Tour.
Prentice-Hall, 2003.

[Sco85] D. W. Scott.
Averaged Shifted Histograms: Effective Nonparametric Density Estimators in Several Dimensions.
Annals of Statistics, 13:1024–1040, 1985.

[Sco92] D. W. Scott.
Multivariate Density Estimation.
Wiley, 1992.

[SDS] SDSS.
The Sloan Digital Sky Survey Project Book.
`www.astro.princeton.edu/PBOOK/welcome.htm`.

[SH75] Michael Ian Shamos and D. Hoey.
Closest-point Problems.
In *Proceedings of the Sixteenth Annual IEEE Symposium on the Foundations of Computer Science*, pages 151–162, 1975.

[Sha75] Michael I. Shamos.
Geometry and Statistics: Problems at the Interface.
In *Proc. Symposium on Algorithms and Complexity.* Carnegie-Mellon University, 1975.

[Sil82] B.W. Silverman.
Kernel Density Estimation using the Fast Fourier Transform.
Journal of the Royal Statistical Society Series C: Applied Statistics, 33, 1982.

[Sil86] B. W. Silverman.
Density Estimation for Statistics and Data Analysis.
Chapman and Hall/CRC, 1986.

[Spr91] R. F. Sproull.
Refinements to Nearest-neighbor Searching.
Algorithmica, 6:579–589, 1991.

[SS98] I. Szapudi and A. Szalay.
A New Class of Estimators for the n-point Correlations.
The Astrophysical Journal, 494:L41–L44, 1998.

[ST85] D. Sleator and R. Tarjan.
Self-adjusting Binary Search Trees.
Journal of the ACM, 32(3):652–686, 1985.

[SW99] P. Smyth and D. Wolpert.
Linearly Combining Density Estimators via Stacking.
Machine Learning, 36:59–83, 1999.

[Sza97] I. Szapudi.
A New Method for Calculating Counts in Cells.
The Astrophysical Journal, 1997.

[Sza00] A. Szalay.
Personal Communication.
, 2000.

[SZM99] J. Shepherd, X. Zhu, and N. Megiddo.
A Fast Indexing Method for Multidimensional Nearest Neighbor Search.
In *SPIE Conference on Storage and Retrieval for Image and Video Databases VII*, pages 350–355, 1999.

[TJ03] M. Takada and B. Jain.
The three-point correlation function in cosmology.
Monthly Notices of the Royal Astronomical Society, 340, 2003.

[TS92] G. Terrell and D. W. Scott.
Variable Kernel Density Estimation.
Annals of Statistics, 20(3):1236–1265, 1992.

[Uhl91] J. K. Uhlmann.
Satisfying general proximity/similarity queries with metric trees.
Information Processing Letters, 40:175–179, 1991.

[Vai89] P. M. Vaidya.
An O(NlogN) Algorithm for the All-Nearest-Neighbors Problem.
Discrete and Computational Geometry, 4:101–115, 1989.

[Ves94] Franz Vesely.
Computational Physics: An Introduction.
Plenum Press, 1994.

[Wan94] M. P. Wand.
Fast Computation of Multivariate Kernel Estimators.
Journal of Computational and Graphical Statistics, 1994.

[WB97] R. Weber and S. Blott.
A simple vector-approximation file for similarity search in high-dimensional vector spaces.
Technical Report 19, ESPRIT project HERMES, 1997.

[Wei78] B. W. Weide.
Statistical Methods in Algorithm Design and Analysis.
PhD. Thesis, Carnegie Mellon University, Computer Science Department, 1978.

[Whi79] Simon D. M. White.
The Hierarchy of Correlation Functions and its Relation to Other Measures of Galaxy Clustering.
Monthly Notices of the Royal Astronomical Society, 186:145–154, 1979.

[Yao82] A. C. Yao.
Space-time Tradeoff for Answering Range Queries.
In *Proceedings of the Fourteenth Annual ACM Symposium on the Theory of Computing*, pages 128–136, 1982.

[ZRL96] T. Zhang, R. Ramakrishnan, and M. Livny.
BIRCH: An Efficient Data Clustering Method for Very Large Databases.
In *Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems : PODS 1996*. ACM Press, 1996.