Content-Based Indexing of Spatial Objects in Digital Libraries

SU-SHING CHEN

Department of Computer Science, University of North Carolina, Charlotte, North Carolina 28223

Received July 6, 1995; accepted December 4, 1995

Digital libraries are being built upon internet-based systems of heterogeneous information bases which contain texts, images, audio, video, and graphics. A relatively unexplored but important research area for digital libraries is the indexing of spatial objects and 3-D geometric information. This paper provides some new content-based approaches to this area. Experience gained from building a digital molecular biology library including spatially indexed biological objects is described.

1. INTRODUCTION

Digital libraries have emerged from the building of information infrastructures on the internet [11, 16, 17]. The least common denominator of digital libraries is a networked system of heterogeneous information bases which contain texts, images, audio, video, and graphics. A number of indexing methods have been proposed for digital libraries [3, 4, 12]. Although the defining method for text indexing was established by Salton and his students [24, 25], there has not been any dominant method for the indexing of spatial objects. This is a relatively unexplored but important research area. Traditionally, spatial objects are understood to be the contents in image and video databases. This paper considers a wide variety of spatial objects and describes a digital molecular biology library which includes 2-D visualization and 3-D DNA and protein structures, e.g., entries of the Brookhaven Protein Data Bank [1, 13, 29]. Content-based indexing of such spatial objects requires complex search algorithms which are not commonly available in networked information services [6, 30, 31].

The basic functions of a digital library are generating, storing, processing, transmitting, accessing, and presenting information [3]. Large collections of spatial objects, such as remote sensing images, medical images, movies, and protein structures, occupy storages of the size in the range of petabytes [3]. Therefore, efficient content-based indexing and organization of spatial objects is one of the key issues of digital libraries. If a digital library provides only subjects, titles, and metadata for indexing, then it does not support well the user need and utilization in the content-based searching and retrieval for spatial digital library objects. This paper studies ways to index spatial objects so that their contents and semantics are represented in the searching and retrieving mechanisms.

Spatial objects differ from text objects in many ways. There is an old saying: “A picture is worth a thousand words.” A picture contains a lot of information. There are multiple things situated at different locations in a picture. How do we interpret these things? There are multiple interpretations of a single thing, say, a basketball, in a picture. If we are interested in geometric figures, the interpretation would be a circle or a ball. If we are interested in sport events, the interpretation would be a basketball game or a basketball team. If we are interested in materials, the interpretation would be the elastic rubber. The rich interpretations of spatial objects constitute multiple semantic meanings—leading to multiple representation schemes that digital libraries need to possess as their human users do.

Content-based indexing of spatial objects leads to the issue of information complexity of digital libraries. If there are too many semantic meanings that digital libraries must maintain and store, the information complexity will become hard to manage and the system performance and interoperability will degrade and suffer as some AI systems do (see book reviews of [21]). Consequently, content-based indexing should be implemented judiciously so that the information complexity would not hinder too much the system performance and interoperability of digital libraries. This paper brings content-based indexing into the digital library organization only at critical places so that the user utilization will be enhanced and the system performance and interoperability will be maintained. Of course, a quantitative study must be made to evaluate the correlations among content-based indexing, user serendipity, and system performance and interoperability. This paper will not address the issues here.

Spatial knowledge has been studied earlier for other purposes (e.g., [7, 8, 26]). However their results remain useful to the development of content-based indexing of spatial objects in digital libraries, because content-based
indexing depends on the semantics of spatial objects which constitute the spatial knowledge in total. Spatial knowledge consists of knowledge hierarchies of different kinds. For examples, we have the static and dynamic image (perceptual) knowledge, static and dynamic world knowledge, and task knowledge—knowledge hierarchies of at least five kinds [8]. Diverse applications have different knowledge representation schemes and thus different knowledge hierarchies. Examples of such applications are geographic information systems, arts and entertainment, environmental management, engineering CAD systems, education and learning systems, and medical information systems. All these applications have different spatial informations ranging from remote sensing satellite images to CAD designs, to multimedia educational material, and to MRI/PET images. This paper explores general spatial indexing methods which are applicable to these applications.

This paper contributes to the research on digital libraries in two aspects. First, the BC tree (binary convex tree) of indexing geometric figures (including points and regions) and the incorporation of semantics to points and regions with their indexing are two novel extensions of previous indexing results of spatial objects which are only concerned with objects in the coordination systems without any high level spatial knowledge. Our approach combines pure indexing which originates from databases [20] with semantic knowledge which originates from artificial intelligence [7]. This combination is a fruitful direction because it enhances significantly the power of indexing, search, and retrieval for user utilization. Due to the complexity and heterogeneity of information bases, in particular digital libraries, this combination cuts down a great deal of overhead and serves the users in a better way. Second, the indexing of 3-D geometric information exhibits a new indexing and search algorithm for capturing spatial knowledge in complex application domains, such as molecular biology. Due to the density of spatial points in those geometric configurations, a purely discrete data representation (in digitalized form) is not efficient. Simply the number of pixels and voxels becomes too large, occurring often in dynamic molecular simulations even on supercomputers. Our approach transforms discrete spatial points into invariant geometric features which assume a linear feature vector form. However, discrete data becomes continuous data, and the floating point representation becomes necessary.

After introducing general indexing methods of spatial objects, we describe a digital molecular biology library, called the DMBL (digital molecular biology library), which is under construction. The DMBL includes texts, sequences, and spatially indexed biological structures. Semantics of objects are presented in spatial data structures as well as biological knowledge. Molecular biology is a very rich application domain which has static and dynamic information of complex types (e.g., [1, 6, 9, 10, 13–15, 18, 19, 22, 23, 27–31]). We have chosen this domain to develop our learning and knowledge of digital libraries, and to test our digital library indexing and search algorithms described above.

The organization of this paper is as follows. Section 2 describes a general framework of digital libraries in terms of abstract concepts—objects and functions. In this framework, we consider issues regarding content-based indexing of 2-D and 3-D objects in Section 3. The novel BC tree based on convex sets for indexing spatial objects will be introduced. The BC tree offers a single algorithm for point and region files. Thus, it can handle mixed point and region files. In Section 4, spatial knowledge hierarchies will be discussed, and semantics will be appended to the BC trees as semantic BC trees. In Section 5, we present a novel approach to content-based indexing of 3-D geometric objects. Invariant geometric feature vectors will be defined and calculated. How they are used in the indexing and search of geometric objects will be described. Section 6 presents a digital molecular biology library, called the DMBL, which is under construction. A similarity measure and an indexing scheme which are described in Section 5 will be applied to the DMBL. Further, we discuss a cross-referencing scheme relating character strings of amino acid sequences and 3-D structures of proteins. The scheme which stems from a learning mechanism illustrates the way two seemingly irrelevant classes of objects are cross-referencing each other through learning.

2. A DIGITAL LIBRARY FRAMEWORK

A general framework is needed for the analysis, design, evaluation, implementation, standardization, and maintenance of digital libraries. However, a widely accepted framework has not existed [12]. Here we attempt to formulate such a framework.

The framework of digital libraries contains a collection of networked, heterogeneous multimedia information bases which may be databases, knowledge bases, information repositories, bulletin boards, and other information resources. The union of all the information bases is called the information space of a digital library. The users of a digital library may browse and navigate through the information space to access information and to solve problems using the accessed information.

An information base contains information objects and hyperobjects connected via links and hyperlinks. These are multimedia objects. Recursively, objects contain subobjects, subobjects contain subs ubobjects. Objects are characterized by their names, types, authors, key words, metadata fields (e.g., authenticity, availability, sources, object preprocessing, network addresses, and copyrights), and contents which are used for indexing and retrieving. Informa-
library functions, including: (1) information organization, (2) information transformation, (3) information transmission, (4) information integration, (5) information access—query, search, and retrieval, and (6) information presentation. Even though this list is by no means complete, these basic functions support digital libraries essentially. A pure object-oriented system builder would argue that these basic functions can be implemented as objects, i.e., basic library function objects, leading to two kinds of objects, information objects and basic library function objects. This situation is similar to other object-oriented applications to business enterprises which have database objects and business process objects [2]. In contrast, this paper will distinguish the two by naming the latter as basic library functions. The reason is that these basic library functions must deal with different levels of semantics arising from content-based indexing. The handling of knowledge hierarchies demands that they are implemented in a more intelligent module, such as intelligent agents [5].

Now, let us briefly describe these basic library functions. Information organization defines the structure of digital libraries, and includes information indexing and other tasks, such as capturing, storing, and selecting of library objects. Content-based indexing is a key feature of information organization, because it implies how we carry out the remaining steps as well, from (2) to (6). Information transformation supports the format exchange of information objects. It allows various formats to be interoperable. Information transmission is concerned with communication protocols and bandwidth requirements. While information transformation deals with the interoperability of information objects, information transmission requires the interoperability of communications networks and computer hardware. Information integration combines information objects into more useful objects. It is closely related to information transformation, because it integrates multiple objects into a single object. The above four functions are internal library functions, because they deal with internal system issues. The next two functions are external library functions, because they deal with external user access issues. Information access includes query, search, and retrieval. Query supports and interacts with user requests of information objects. Search and retrieval finds the requested objects. Finally, information presentation delivers the objects to the user in desired formats. The access function—query, search and retrieval—depends on how content-based indexing is defined. This paper addresses the issues about indexing and therefore query, search, and retrieval as well.

These basic library functions can be implemented as intelligent agents [5]. Intelligent agents are intelligent software systems which possess several integrated capabilities, such as I/O, feature extraction, classification, reasoning, learning, and decision making. The agents interact with
This section is concerned with content-based indexing of spatial objects in digital libraries. First we distinguish content-based features from noncontent-based features of spatial objects. Then we define what is content-based indexing, search, and retrieval of these objects. If an information access library function only supports a character string query of names, titles, and identification fields of spatial objects, these fields are definitely not content-based features. If some other fields contain information about the spatial content, they will not be considered as content-based features either, but as metadata about the spatial content. Metadata about the spatial content are descriptors of general nature, such as metadata about an image and a video. Metadata will not contain specific content descriptors or features at the point or region level. Only intrinsic features in points and regions will be treated as content-based. Thus, content-based features are those directly extracted from the content of a spatial object. The features extraction and classification components of an information organization agent (thus, including indexing) described in Section 2 can extract meaningful features—often in points and regions—of spatial objects and classify them into specific categories for indexing. The search component of an information access agent uses pattern recognition techniques to find those categories and the retrieval component obtains the objects for the user’s consumption. This defines a general scheme of the content-based indexing, search, and retrieval in the basic library functions. In this section, we will focus on how to index points and regions and how semantics are incorporated into the indexing in digital libraries without describing how feature extraction and classification is performed. There are many image processing books on feature extraction and classification; however, the connection has not been made clear to digital libraries yet.

In digital libraries, the main issue of indexing of spatial objects lies in how to represent points and regions and how to organize point and region files in the coordinate plane $\mathbb{R}^2$ and coordinate space $\mathbb{R}^3$. Indexing higher dimensional points in general $\mathbb{R}^n$ is useful for not only spatial objects in digital libraries but also multiple keys in ordinary databases and file structures. Therefore, point indexing is of fundamental importance. In nonstandard (or spatial) databases, such as geographic information systems and CAD systems, region indexing of spatial objects is required in accessing their contents in secondary disk or tape storage systems. In digital libraries, this problem is more acute because of possibly tertiary storage systems of digital libraries. Region files are topologically complicated, and include many different kinds of geometric regions. Region indexing has still many open research issues.
For simplicity, one can enclose complex geometric regions by simple ones, such as 2-D rectangles and 3-D parallelopipes. They are called the MBRs (minimum bounding rectangles) in the database literature and the MBPs (minimum bounding parallelopipes) in this paper. Also point files are decomposed into MBRs for storing in pages of storage systems. There have been several useful algorithms for dealing with this kind of data structures [20, 26], including interpolation hashing, multidimensional linear hashing, quantile hashing, quadtree, octree, MD tree, GDB tree, k-d-B tree, grid file, interpolation-based grid file, buddy tree, BANG file, hB tree, B+ tree with z-order, R tree, and R+ tree. All these algorithms were originally developed for spatial databases. In spatial databases, the indexing of points and regions depends on an efficient binary tree algorithm, called the B+ tree. The main thinking of spatial databases lie in the tradeoffs between memory and disk accesses, and between key indices and data pages. In digital libraries, we need to think somewhat differently. Digital library objects are not as regular as database objects. The database organization into pages could be modified to reflect the reality of digital library access functions, involving heterogeneous information types and user semantics. We propose an additional approach, the BC (bounding convex) tree, which addresses these issues and enhances the traditional indexing methods of spatial databases.

The traditional indexing of spatial databases can be described essentially in the organization of point and region files. Points and regions, in particular, are enclosed by MBRs (Most spatial databases deal only with 2-D regions, although 3-D MBPs can be easily defined). Both point and region files are organized in B+ trees as follows (See [20] and the references quoted there): In a B+ tree, the data points and regions are stored in the leaf nodes, and a nonleaf node stores a fixed number of the MBRs which contain the data points and the region MBRs at the leaf level, or other MBRs at the next lower level. The MBRs at the same level may overlap (e.g., the R trees for region files) and may overlap only at their boundaries (e.g., the buddy trees for point files). When a new point or region is inserted, the insertion algorithm finds an appropriate leaf node into which it stores the point or the region according to the coordinate system. If the leaf node which is designated is full, the leaf node will split into two leaf nodes, and the node splitting will be propagated from the leaf node level to the root node level until the splitting node is stored. A similar process holds for the deletion algorithm. This has worked well for point files. For region files, the overlapping in rectangles (e.g., the R trees) makes possible unique representation and efficient storage utilization (at least 50%). However a range query will degrade the performance significantly.

In addition to traditional indexing methods, we enhance the spatial indexing of digital library objects by introducing the BC (binary convex) tree indexing. The BC tree allows more flexible indexing, search, and retrieving by constructing MBCHs (minimum bounding convex hulls) of points and regions instead of MBRs and MBPs. Thus, our digital library organization consists of both the B+ trees of MBRs and MBPs and the B+ trees of MBCHs. The two kinds of B+ trees are connected by linking the MBRs and MBPs to their MBCHs. Each MBCH is contained in a MBR or a MBP, and conversely, each MBR or MBP contains several MBCHs. The technical detail will be presented in a forthcoming paper.

The BC tree structure is more flexible and supports user-centered access of digital libraries. The BC tree indexing of point files is as follows. The neighboring points in the coordinate plane R2 should be stored in a storage page. A point file covers several such pages. A directory tree is constructed for such a file, and then a super directory tree is constructed for a group of file directories. First let us enclose all the points of a file in a spatial object in a large rectangle with the left lower corner (Lx, Ly) and the right upper corner (Rx, Ry). Then Lx and Rx are the lower and upper bounds of the x coordinates, and Ly and Ry are the lower and upper bounds of the y coordinates. The first step is to decompose the rectangle spanned by the intervals (Lx, Rx) and (Ly, Ry) into disjoint subsets containing an equal number of points. We use the convex hulls of neighboring points to decompose the rectangle into disjoint convex subsets. For a fixed number of neighboring points {p1, p2, ..., pn}, we form the convex hull C(p1, p2, ..., pn) of these points. We require all the points to be decomposed into disjoint point subsets such that their convex hulls (MBCHs) are disjoint also. A finite number of MBCHs are stored in a page. All the pages are disjoint, but do not cover the coordinate plane completely. The remaining vacant lots will be used for further point insertion, but they are not to be stored in the storages. The MBRs enclosing those convex hulls serve as reference frames with linking pointers. When users know how and what to query exactly for in the reference frame, the actual MBRs may then be searched and retrieved.

The BC tree contains all the points in the leaf nodes. The points are organized into MBCHs which are contained in the lowest nonleaf level. For a fixed number of neighboring MBCHs {C1, C2, ..., Ck}, we form the MBCH C(C1, C2, ..., Ck) of the convex sets {C1, C2, ..., Ck}, which is in the next nonleaf level. Recursively, the convex hulls of convex sets form convex hulls which are in the higher nonleaf levels. This generates a B+ tree-based directory of point pages in storage. The number of nodes in the directory B+ tree grows linearly with N/P, where N is the total number of points and P is the number of points in each convex page. The directory B+ tree grows to the logL(N/P) level, where L is the number of convex hulls in each bounding convex hull. This method extends the
rectangle partition method in [20]. In Fig. 4, corner points
on convex polygons and black dots inside the convex polygons
are the points in a file. The points are organized into
MBCHs, such as A1, A2, B1, and B2. Then these MBCHs
are further organized into MBCHs, such as A and B. The
$B+$ directory tree is described on the right side of the
figure.

If new points are inserted into or existing points are
deleted from the file, the MBCHs must be reconstructed
into new disjoint MBCHs. Thus, some convex hulls are
split up and reorganized into other convex hulls. Because
insertion and deletion do not occur as frequently as search
and retrieval in digital libraries, the reconstruction of
MBCHs does not burden much the basic digital library
functions. The search is efficiently performed by testing
the bounding hyperplane signs of a convex hull in a point
query, and it may include the handles for potential search
of the rectangular reference frames in the storage systems
by keeping pointers to the rectangular projections of the
convex MBCHs. The two intervals ($L_x$, $R_x$) and ($L_y$, $R_y$)
are partitioned into (overlapping) $X$ subintervals on the $x$
axis and $Y$ subintervals on the $y$ axis by projections of
convex hulls onto the $x$ axis and the $y$ axis. A point query
finds the indices of the projected $X$ and $Y$ intervals, looks
up the pointer address in the BC directory tree, and re-
trieves the rectangular reference frame containing the
point and the MBCH. This capability is optional and may
not be necessary to all the BC trees.

The BC tree indexing of region files is as follows. Tradition-
ial spatial indexing methods, such as the $R$ tree, enclose
regions in their MBRs whose sides are vertical and hori-
zontal. The enclosing MBRs are stored in the leaf nodes
of the directory tree. The nonleaf nodes store larger MBR
rectangles which contain several leaf MBR rectangles in
a recursive fashion. The higher level, nonleaf MBR rect-
angles contain leaf-level MBR rectangles or MBR rectangles
of lower levels. In most cases, the directory tree has over-
lapping MBR rectangles, due to the rigidity of MBR rect-
angles, so that the search becomes inefficient. The BC
tree is efficient by enclosing arbitrary geometric regions
in MBCHs instead. Geometric regions are included in ap-
propriate convex hulls which are included in higher level
convex hulls. The construction of a BC directory tree for
regions is the same for points. If convex hulls are split and
reorganized due to insertion and deletion, then they and
their higher level convex hulls are split and reorganized
into other convex hulls in the updated BC tree. In the
propagation of splitting, convex hulls should never overlap.
In Fig. 5, regions are organized into MBCHs, such as A1,
A2, B1, and B2. Then these MBCHs are organized into
MBCHs, such as A and B.

The BC tree indexing works well for arbitrary geometric
regions (including points). Therefore, our BC trees can
handle mixed point and region files. However, the BC tree
indexing prefers relatively small and compact geometric
rectangles. This means that regions should not be very large
and occupy the whole coordinate plane, such as a large,
long spiral figure in $R^2$. The protein structures are of this
type (see Section 5). No matter how complicated the geo-
metric regions are, as long as they are compact, we can
bound a fixed number of them by the smallest convex hull
(MBCH) in the coordinate plane. We can further set up
additional requirements for the construction of convex
hulls in our BC trees, such as a fixed two-dimensional area
$A$ for the areas of regions and MBCHs in $R^2$. Based on a
fixed two-dimensional area $A$ as the upper limit, regions
and points (in a mixed manner) form convex hulls, and
their convex hulls form further convex hulls of higher lev-
els. The BC tree can be extended easily to the 3-D coor-
dinate space $R^3$, because convex hulls in $R^3$ are constructed
in the same way.

4. SEMANTICS AND CONTENT-BASED INDEXING

The incorporation of semantics to points and regions
and their indexing is a novel extension of previous indexing
schemes of spatial objects which are only concerned with
the coordination systems without any high-level spatial
knowledge. Our approach combines indexing with semantic knowledge. This leads to the new concept of semantic BC trees which enhances significantly the power of indexing, search, and retrieval. Due to the complexity and heterogeneity of multimedia information bases in digital libraries, this combination cuts down a great deal of overheads in search by pruning not so useful branches of a semantic BC tree. Semantics in spatial knowledge are annotated to the indexed points and regions. Consequently, annotated semantics are also used in the search.

The semantics of digital library objects can be captured and appended to BC trees. This entails the construction of spatial knowledge hierarchies in BC trees [5, 7, 8]. As we have discussed earlier, metadata are not considered to be content-based features in our framework. They are extensions of the traditional library catalogue cards. Metadata are mostly in text format and can make use of existing indexing schemes. In contrast, digital library semantics depend on the feature extraction, classification, reasoning, learning, and decision-making components of the intelligent agent architecture described in Section 2. As the basic digital library functions are implemented by intelligent agents, the semantics, features and classes of digital library objects will depend on the capabilities of the intelligent agent architecture (see Section 2).

The feature extraction component of the intelligent agent architecture extracts spatial features of objects. The selection of features depends on the context of spatial objects, library functions, and user needs. Basically, the features should represent the contents of digital library objects. The features are building blocks of an object. Key words are good examples of features in a text. Image features include pixels, edges, blobs, corners, and their classifications. The classification component divides the objects up into categories based on their features. The classes are connected by semantic relations and inheritance relations. This generates a set of hierarchies of content-based objects, organized as spatial knowledge hierarchies. The hierarchies are further connected to each other through the connections among semantics which are appended to content-based objects.

There are many possible spatial knowledge hierarchies. Basic spatial knowledge hierarchies are those originating from static and dynamic (perceptual) image spaces, static and dynamic world spaces, and task space. The static image space includes various perspective projections of the world space through various imaging mechanisms. The dynamic image space includes various perspective projections of the dynamic world space through the filming mechanism. The static world space represents the 3-D environment and things in it at one time instant. The dynamic world space represents the 3-D environment and things in it over a time interval. The task space describes various high-level activities of humans and things with semantic meanings.

It is built over the dynamic world space of various states in space and time coordinates. Within each space, there are many spatial knowledge hierarchies. Other spatial knowledge hierarchies, like geographic, cultural, and geometric, are derived from the five spaces. It is not difficult to see that these spaces are often interrelated.

The spatial knowledge hierarchies are connected through linkages between semantics, features, objects, and classes in different hierarchies. Figure 6 shows an example of combined hierarchies—geographic, cultural, geometric, and image space hierarchies. Spatial relations can be both topological and metrical. Topological relations include intersection, union, complement, betweenness, and to-the-right-of in the world spaces and the image spaces. Metrical relations include has-equal-distance-as, is-the-shortest-distance-between, and has-one-acre-area. We allow both kinds of relations in spatial knowledge hierarchies.

The BC tree hierarchy reflects the granularity similar to multiresolution quadtrees and octrees for digitized images [26]. The granularity in spatial knowledge hierarchies is provided by the decomposition of the image and world spaces into BC trees. Furthermore, a local region surrounding things of interest can be decomposed locally into nonoverlapping subregions in BC trees such that things are represented by subregions of suitable resolution. This provides the granularity in spatial knowledge hierarchies. By decomposing the MBCHs in BC trees, we create additional tree levels but smaller areas of the MBCHs. Figure 7 shows how a one-level decomposition into a11, a12, a21, a22, b11, b12, b21, and b22 of the MBCHs A1, A2, B1, and B2 is performed.

The BC tree decomposition scheme facilitates the efficient access to spatial objects in digital libraries. Its efficiency is further enhanced by appending spatial knowledge hierarchies to BC trees. We can construct semantic BC
trees (Fig. 8) which represent both the semantics and the decomposition in appropriate spaces of spatial knowledge hierarchies. The search of semantic BC trees can use pruning to eliminate the search on certain branches which are not useful. For example, if we are searching for a highway and a certain convex set node is labeled as farmland, then the branches of the farmland node should be pruned. In the following semantic BC tree, only one level of 16 semantic labels is provided to the MBCHs at the leaf nodes. For example, they may be highway, farmland, airport, bridge, harbor, hospital, school, military base, town, railroad, river, lake, housing, shopping center, office complex, and park. Other semantics may be provided to this leaf level and all other nonleaf levels. The linking of semantics to nodes and to themselves is feasible by pointers.

5. DISCRETE GEOMETRY OF SPATIAL STRUCTURES AND INDEXING OF 3-D OBJECTS

Some spatial objects are more regular than others. Digitized images have intrinsic regularity in pixels due to digitization. Other spatial objects are more irregular than digitized images. For example, DNA and protein structures in molecular biology are spatial structures consisting of chains of 3-D coordinates in $R^3$. They are very large, discrete, point sets which are not conveniently handled by ordinary spatial decomposition schemes, including the BC tree. Their spatial representation is best described by the discrete geometry—in terms of invariant geometric feature vectors—characterizing the spatial structures of DNAs and proteins. A spatial structure $S$ is first decomposed into several linearly ordered curve structures $S_1$, $S_2$, ..., $S_k$, and a linearly ordered curve structure $S_i$ $(1 \leq i \leq k)$ is represented by a linear list of invariant geometric feature vectors.

A linearly ordered curve structure $S$ is a discrete collection of distinct linearly ordered points in $R^3$, called vertices $P(i)$ $(i=\ldots,-2,-1,0,1,2,\ldots)$. These points can be joined by linear segments $[P(i),P(i+1)]$ over the interval $[i,i+1]$, forming a polygonal curve. A right-handed orthonormal basis $(T_i,N_i,B_i)$ is attached to each vertex $P(i)$. $T_i$ is the vector $P(i)P(i+1)$, called the tangent vector at $P(i)$. The binormal unit vector

$$B_i = T_{i-1} \times T_i/T_i|T_{i-1} \times T_i|$$

is normal to the plane through $P(i-1)$, $P(i)$, $P(i+1)$. The unit normal vector is the cross product $N_i = B_i \times T_i$.

The curvature $k_i$ at $P(i)$ is the undirected angle between $T_{i-1}$ and $T_i$. The torsion $t_i$ along $P(i)P(i+1)$ is the directed angle between $B_i$ and $B_{i+1}$. The torsion $t_i$ is positive if $B_i \times B_{i+1}$ is the same vector as $T_i$, and is negative if $B_i \times B_{i+1}$ is the opposite vector as $T_i$. The segment length $[P(i),P(i+1)]$ is denoted by $d_i$.

A mathematical argument shows that two curve structures are equivalent up to a translation in $R^3$ provided that they have the same segment lengths, curvatures, torsions, and initial orthonormal frame. If they do not have the same initial orthonormal frame, then they differ by a rotation in $R^3$. Therefore, segment lengths, curvatures, and torsions characterize curve structures [6].

Translations and rotations do not change the intrinsic properties of curve structures. The invariance under translations and rotations reduces the information complexity of spatial objects. Thus, two curve structures with the same segment lengths, curvatures, and torsions are considered the same. The list

$$\langle [d_1,k_1,t_1],[d_2,k_2,t_2],\ldots,[d_{N-2},k_{N-2},t_{N-2}] \rangle$$

called the linear list of invariant geometric feature vectors of a curve structure consisting of $N$ points. An efficient content-based indexing scheme of curve structures can now be defined based on invariant geometric feature vectors. The overall spatial structure $S$ is indexed first by its linearly ordered curve structures which are then indexed by invariant geometric feature vectors. This indexing is content-based, because it represents completely the geometric content of the spatial structure $S$. The constraints and de-
pendencies among the linearly ordered curve structures are represented by pointers at relevant cells in the linearly ordered link lists.

The search through a collection of curve structures in the 3-D space \( R^3 \) for a specific curve structure is accomplished by comparing the similarity measure, called the root mean square error (RMSE) distance. Let \( X \) and \( Y \) be two curve structures, each of which consists of \( N \) invariant geometric feature vectors—in curvature, torsion, and segment length—respectively:

\[
X = \{(x_{11}, x_{12}, \ldots, x_{N1}, x_{N2}, x_{N3})\},
\]

\[
Y = \{(y_{11}, y_{12}, y_{13}), \ldots, (y_{N1}, y_{N2}, y_{N3})\}.
\]

An invariant RMSE distance \( R(X, Y) \) of \( X \) and \( Y \) represented by \( N \) invariant geometric feature vectors—in segment length, curvature, and torsion—is defined as

\[
R(X, Y)^2 = (1N) \sum_{i=1}^{N} \sum_{j=1}^{3} (x_{ij} - y_{ij})^2,
\]

where \( x_{ij} \) and \( y_{ij} \) \((i = 1, \ldots, N \text{ and } j = 1, 2, 3)\) denote the segment length, curvature, and torsion features (i.e., \([d_1, k_1, t_1], [d_2, k_2, t_2], \ldots, [d_N, k_N, t_N])\) of \( X \) and \( Y \) respectively. The distance \( R(X, Y) \) is invariant under rotations and translations. The search for a spatial curve segment which most resembles \( X \) will yield all the \( Y \)'s which have the minimal RMSE distance from \( X \). The invariant RMSE distance search looks for intrinsic geometric proximity independent of locations and orientations. This significantly reduces the information complexity. Locations are compensated by translations, and orientations are compensated by rotations.

6. A DIGITAL MOLECULAR BIOLOGY LIBRARY—DMBL

Molecular biology is the scientific discipline that studies life on earth by combining biology, physics, and chemistry [1, 13, 29]. The richness of knowledge in molecular biology requires a very complex design of its digital library. In this ongoing work, we begin by capturing its essential knowledge and representing the knowledge in digital formats. Then we derive algorithms which can manipulate the information objects efficiently. There are many representations (in digital formats) and computational algorithms (to manipulate them) in the literature (e.g., [9, 10, 14, 15, 18]). In this paper, no attempts are made to cover them completely. Only a few algorithms of the DMBL will be exemplified.

The two most important objects in molecular biology are the genes and the proteins. A short story can be told about DNA (deoxyribonucleic acids) and the proteins [13]. In 1953, J. Watson and F. Crick discovered the double helix structural model of DNA which carries the genetic information [1, 13]. The DNA molecule consists of two polymer chains consisting of four types of residues—\( A \) (Adenyl), \( G \) (Guanyl), \( T \) (Thymidyl), and \( C \) (Cytidyl). The sequence of residues can be entirely arbitrary, but the sequences in both chains are strongly interconnected in accordance with the complementarity principle—\( A \) is opposite to \( T \), and vice versa; \( G \) is opposite to \( C \), and vice versa. While the structural model of Watson and Crick has been well established, the spatial structures of all proteins are much less known. Proteins are the cell’s main building blocks. They are complex macromolecules with 22 amino acids as building bricks. The amino acid sequences of proteins are called the primary sequences. The 22 amino acids are as follows: Ala, Arg, Asn, Asp, Asx, Cys, Gln, Glu, Glx, Gly, His, Ile, Leu, Lys, Met, Phe, Pro, Ser, Thr, Trp, Tyr, and Val (see [29] for more details). The difference among all living things lies in their amino acid sequences, the primary sequences, of proteins. These sequences are predetermined by the genes which are part of the DNA. Thus, each cell must have a dictionary to translate the four-letter DNA text into the 22-letter protein text. The process consists of transcription first and translation next. The transcription is to extract an RNA (ribonucleic acid) copy of the gene’s DNA, serving as the messenger, called the mRNA. The translation is to use a molecular computer, the rRNA, and a computer program, the genetic code, to translate the DNA text into protein text. For more details, we refer to the excellent introductory book, “Unraveling DNA,” by M. D. Frank-Kamenetski [13].

Intuitively, there are a very large number of varieties of spatial structures, called protein conformations, arising from the amino acid sequences. X-ray crystallography has solved the structures of several hundred protein structures which are recorded in the Brookhaven Protein Data Bank [1, 29], and this number is growing day by day. However, this number is still far too small compared with the primary structures—amino acid sequences—which have been processed. DMBL is currently under construction. It collects information about DNAs and proteins. For example, the following information about proteins will be included: (1) meta information—authors, dates, histories, and references; (2) the amino acid sequence; (3) 3-D coordinates of atoms; (4) 3-D graphics of the protein; (5) biological, chemical, and physical information—e.g., hydrophobicity, hydrogen bonding, polypeptide, side chains, and the solvent environment; and (6) the geometric structural elements—secondary structures and the packing of these structural elements. The library is useful for both theoretical molecular biology research and practical drug design and genetic engineering. The Brookhaven Protein Data Bank is an excellent digital collection of protein data, literature, and documentation, and can be considered as an existing digital library of proteins.
Secondary structures in proteins are described in three main classes. The \( \alpha \) helices are patterns of main chain hydrogen bonding with repeating curvature and torsion values. In most cases, \( \alpha \) helices are right-handed, but there are some left-handed \( \alpha \) helices. Extended \( \beta \) strands are called \( \beta \) sheets. The primitive unit of a \( \beta \) structure is not an individual \( \beta \) strand but the \( \beta \) strand pair—hydrogen bonded in either a parallel or an antiparallel setting. Nonrepetitive structures are turns, connections, and compact loops. Large portions of protein structure are well ordered but nonrepeating coils and random coils. A nonrepetitive structure straps one piece of repetitive structure to another. While secondary structures are defined and stabilized by their backbone interactions, side chains define and stabilize compact loops, which are highly ordered parts of a protein and are not random in any sense. In [14], a dictionary of secondary structures is given. However, the geometric analysis was not clearly made there. The DMBL will revisit the secondary structures described in [14].

There are several kinds of search algorithms in the DMBL (see [30, 31] for further references). We only describe the spatial curve segment search here. The RMSE distance \( R(X, Y) \) is used to search for protein 3-D structures and secondary structures of a fixed length \( L \) in proteins. In the secondary structure search, there are two kinds. The first is an internal search for secondary structures within a given protein 3-D structure \( X' \) by discovering similarities among its local secondary structures of length \( L \). The second is an external search for similar secondary structures of two proteins \( X \) and \( Y \). For example, in the internal search of \( X \), if the invariant RMSE distance \( R \) of consecutive 3, 4, and 5 segments in a protein 3-D structure \( X \) is the same (or remains smaller than a prescribed threshold), the consecutive segments form an \( \alpha \) helix or an \( \alpha \) secondary structure in \( X \).

The 3-D protein structures are essential to the understanding of the relationship between amino acid sequences and their functions. The problem of predicting the 3-D protein structures from a primary sequence—called the "protein folding problem"—has yet to be resolved. Although a given string of amino acids contains all the data needed to determine the final 3-D structure, the number of geometric possibilities is enormous which makes this search problem extremely difficult. Thus far, the scientific community has relied on the database of secondary structures accumulated over the past decades as a useful guide to the modeling of new proteins. We have suggested a computational model for the cross-referencing between the two categories—amino acids and proteins [6]. The cross-referencing is defined by a set of learnable mappings between amino acid sequences and their protein 3-D structures. This illustrates an application of spatial indexing to the cross-referencing of amino acids and protein secondary structures.

The protein 3-D structures are represented by their invariant geometric feature vectors. The set of inputs and the set of outputs of the learnable mappings are the amino acids and their invariant geometric feature vectors respectively. The mappings are constructed from certain training sets of known amino acids and their invariant geometric feature vectors using a backpropagation learning mechanism [6]. A training set consists of several library entries (e.g., the Brookhaven Protein Data Bank). The learnable mappings were used first by Qian and Sejnowski [22]. The learning architecture is as follows. The input layer is a window of say 13 (or any number) amino acids. Each amino acid is represented by 21 input units—20 naturally occurring amino acids and a special spacer. Each amino acid places a 1 in one of the 20 input units and 0 in the rest of the input units. Therefore, the total number of input units is \( 13 \times 21 = 273 \). The input units have binary values. The output layer is the set of invariant geometric feature vectors of the backbone atomic structures associated with the 13 amino acids in the input layer. Since each amino acid has 3 atoms in the backbone, N, CA, and C, there are 3 pairs of curvature and torsion numbers and 3 segment lengths. Thus, the invariant geometric feature vector, associated to each amino acid, has dimension 9. The total number of output units is \( 13 \times 9 = 117 \). The output units have analog values.

The cytochrome c superfamily is chosen for illustration. The secondary structures of this superfamily are well identified. This training set contains 13 proteins: (155C), (1CCR), (1CYC), (2C2C), (2CDV), (3C2C), (451C), (1CC5), (1CY3), (2CCY), (351C), (3CYT), and (5CYT). The protein (1CC5) has at least two library objects attached to it:

---

**TABLE 1**

Amino Acids in Protein (1CC5) [1CC5.aacid]

<table>
<thead>
<tr>
<th>GLY</th>
<th>GLY</th>
<th>GLY</th>
<th>ALA</th>
<th>ARG</th>
<th>SER</th>
<th>GLY</th>
<th>ASP</th>
<th>ASP</th>
<th>VAL</th>
<th>VAL</th>
<th>ALA</th>
<th>LYS</th>
<th>TYR</th>
<th>CYS</th>
<th>ASN</th>
<th>ALA</th>
</tr>
</thead>
<tbody>
<tr>
<td>CYS</td>
<td>HIS</td>
<td>GLY</td>
<td>THR</td>
<td>GLY</td>
<td>LEU</td>
<td>LEU</td>
<td>ASN</td>
<td>ALA</td>
<td>PRO</td>
<td>LYS</td>
<td>VAL</td>
<td>GLY</td>
<td>ASP</td>
<td>SER</td>
<td>ALA</td>
<td>TRP</td>
</tr>
<tr>
<td>LYS</td>
<td>THR</td>
<td>ARG</td>
<td>ALA</td>
<td>ASP</td>
<td>ALA</td>
<td>LYS</td>
<td>GLY</td>
<td>LEU</td>
<td>ASP</td>
<td>GLY</td>
<td>LEU</td>
<td>LEU</td>
<td>ALA</td>
<td>GLN</td>
<td>SER</td>
<td>LEU</td>
</tr>
<tr>
<td>CYS</td>
<td>HIS</td>
<td>GLY</td>
<td>THR</td>
<td>GLY</td>
<td>LEU</td>
<td>LEU</td>
<td>ASN</td>
<td>ALA</td>
<td>MET</td>
<td>PRO</td>
<td>PRO</td>
<td>LYS</td>
<td>GLY</td>
<td>THR</td>
<td>CYS</td>
<td>ASP</td>
</tr>
<tr>
<td>ASP</td>
<td>ASP</td>
<td>GLU</td>
<td>LEU</td>
<td>LYS</td>
<td>ALA</td>
<td>ALA</td>
<td>ILE</td>
<td>GLY</td>
<td>LYS</td>
<td>MET</td>
<td>SER</td>
<td>GLY</td>
<td>LEU</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The First Five Invariant Geometric Feature Vectors of Protein (1CC5) [1CC5.igfvector]

<table>
<thead>
<tr>
<th>Feature Vector</th>
<th>0.648992</th>
<th>0.449930</th>
<th>0.715397</th>
<th>0.678521</th>
<th>0.068471</th>
<th>0.758039</th>
<th>0.685837</th>
<th>0.024795</th>
<th>0.676455</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.676844</td>
<td>0.131074</td>
<td>0.718891</td>
<td>0.644293</td>
<td>0.492683</td>
<td>0.716175</td>
<td>0.655140</td>
<td>0.013529</td>
<td>0.655010</td>
</tr>
<tr>
<td></td>
<td>0.685764</td>
<td>0.309422</td>
<td>0.751197</td>
<td>0.692313</td>
<td>0.775112</td>
<td>0.752650</td>
<td>0.685031</td>
<td>0.970944</td>
<td>0.635578</td>
</tr>
<tr>
<td></td>
<td>0.647029</td>
<td>0.771834</td>
<td>0.740595</td>
<td>0.722068</td>
<td>0.351191</td>
<td>0.767686</td>
<td>0.701482</td>
<td>0.987822</td>
<td>0.662212</td>
</tr>
<tr>
<td></td>
<td>0.659553</td>
<td>0.966814</td>
<td>0.727900</td>
<td>0.686052</td>
<td>0.205364</td>
<td>0.755272</td>
<td>0.685788</td>
<td>0.024042</td>
<td>0.635832</td>
</tr>
</tbody>
</table>

(1) amino acid sequence object [1CC5.aacid],
(2) invariant geometric feature vector object [1CC5.igfvector].

The protein (1CC5) has the amino acid sequence given in Table 1. The first five invariant geometric feature vectors for the protein (1CC5) are given in Table 2.

In this backpropagation learning architecture, the parameters—learning rate and smoothing factor—were chosen to be 0.6 and 0.8 respectively. In a training session of 10,000 cycles, the total error converged to 0.004954. That is, within this set of 13 proteins and their homologous classes, the error rate of prediction is less than 0.5% for the resulting learnable mappings for the cytochrome superfamily. This implies that the cross-referencing constructed above can identify the entries in object [1CC5.igfvector] from object [1CC5.aacid] almost correctly.

7. CONCLUSION

We have presented a general framework for digital libraries and several content-based indexing schemes for spatial objects. In 2-D and 3-D point and region files, we have considered content-based indexing from the database and knowledge representation viewpoints. We have described the BC trees and the semantic BC trees. A content-based indexing scheme of 3-D geometric objects has been developed with applications to molecular biology. A digital molecular biology library, called the DMBL, is under construction. It serves as a good example of indexing spatial objects for our research.

ACKNOWLEDGMENT

The author thanks Ed Fox for his valuable comments on this paper.

REFERENCES

11. FCCSET/OSTP, HPCC: Toward a national information infrastructure, A report by the Committee on PMES, 1994.
22. N. Qian and T. J. Sejnowski, Predicting the secondary structure of DNA sequence analysis, in


SU-SHING CHEN has been with the University of North Carolina at Charlotte since 1986. He served as the Chairman of Computer Science from 1986 to 1989. From 1991 to 1994, he was the Program Director of Knowledge Models and Cognitive Systems. From May 1994 to August 1995, he was the Program Director of Information Tech and Organizations. During that period, he was responsible for the establishment of the NSF/ARPA/NASA Research on Digital Libraries Initiative and was the Program Director of that Initiative.