Database Support to Data Fusion Automation

RICHARD T. ANTONY

Invited Paper

Because the development of robust, context-sensitive data fusion algorithms can require the evaluation of sensor-derived data with respect to large scale domain knowledge bases, database management systems can be expected to play an increasingly important role in machine-based reasoning. This paper offers a top-down view of key algorithm and database management system requirements associated with advanced data fusion applications. A database kernel design is outlined that seeks to achieve an effective compromise across a range of algorithm performance and database efficiency requirements.

I. INTRODUCTION

Traditional tactical data fusion algorithms have relied on database management systems (DBMS) primarily to store and retrieve sensor-derived parametric and text-based data, fusion products, and algorithm components such as templates and exemplar sets. With the increased use of multimedia data sources, including imagery, video, and graphic overlays, DBMS have taken on an expanded role in data fusion applications. In an effort to achieve the problem-solving proficiency of human analysts, future data fusion algorithms will incorporate “deep” problem domain knowledge that can, in turn, be sensitive to the underlying domain “context.” In tactical applications, such context can include the current friendly force disposition, existing weather conditions, natural domain features (terrain/elevation, surface materials, vegetation, rivers, drainage regions), and cultural features (roads, airfields, mobility barriers).

While the benefits of fusing information-rich sensor data, such as imagery and video with more conventional tactical data sources is readily apparent, the potential benefits of incorporating nonsensor-derived problem and domain context knowledge can be illustrated by a simple example. Radar systems often employ a single statistical-based algorithm for tracking air targets, regardless of whether an aircraft is flying at an altitude of 20 km or just above treetop level, and regardless of whether the target is a high performance fighter aircraft or a relatively low speed helicopter. Suppose a high speed reconnaissance aircraft is flying 100 m above a river through a mountainous region. Even when terrain masking does not occur, airborne radar systems may have difficulty tracking the target due to the high acceleration turns that can be required by an aircraft that is following highly irregular surface features. Manual intervention would be required to correct the inevitable track fragmentation errors that occur. Because helicopters can hover, fly behind tree lines, and execute rapid directional changes, helicopter tracking can be problematic, as well.

Target tracking performance can potentially be improved by making the analysis sensitive to target class-specific behavior patterns, as well as to constraints posed by the domain. For example, the determination that the aircraft is flying at a low altitude suggests that surface features are likely to influence the target trajectory. When evaluated with respect to various “terrain feature following models,” the trajectory would be discovered to be highly consistent with a “river following flight.” By anticipating that the target will continue to follow the river, track association algorithms could favor future radar detections that supported this hypothesis. In addition to potentially improving tracking performance, the interpretation of sensor-derived data within context permits more abstract interpretations. In this case, since the aircraft could be flying at a low altitude in an attempt to avoid radar detection by nearby friendly surface-to-air missile batteries, hostile intent can be inferred. Successively more global interpretations of the data might permit even higher level-of-abstraction interpretations. Suppose, for instance, that a broader view of the situation revealed a second unidentified aircraft operating in the vicinity. By analyzing the apparent coordination between the two aircraft, their relationships can be hypothesized. If the second aircraft begins jamming friendly communication channels just as the first aircraft reaches friendly airspace, its role can be inferred to be “standoff protection for the collection aircraft.” Thus the incorporation of relevant domain knowledge and physical domain constraints can permit the development of both more effective, as well as higher level-of-abstraction interpretations of sensor-derived information.

Although the development of sophisticated data fusion algorithms holds considerable promise, such algorithms can place heavy demands on the supporting database manage-
ment system. The implications of the expanding role of DBMS in data fusion applications and key design principles associated with the development of more effective and efficient database management system support are explored herein. Fig. 1 identifies a number of high level algorithm and database issues that are associated with sophisticated data fusion algorithms. The principal algorithm issues that drive the database system design requirements are discussed in Section II. Section III offers a brief database management system tutorial. Section IV discusses relationships between algorithm effectiveness and the characteristics of supporting DBMS. Section V discusses the relationship between DBMS characteristics and database performance efficiency. Because the storage, manipulation, and retrieval requirements associated with spatially-organized information tends to eclipse those imposed by nonspatially organized information, the focus of the design is on spatially organized data. Section VI reviews key characteristics of spatial data structures. Section VII discusses a high level database kernel architecture that achieves an effective compromise among the algorithm support and database efficiency requirements. Section VIII provides a brief summary.

II. ALGORITHM ISSUES

A. Sensory Fusion

Sensory fusion in biological systems provides a valuable metaphor for studying the characteristics and requirements of tactical sensor fusion systems. The very survival of many biological species depends on their ability to develop and maintain high levels of situational awareness. Small mammals, for example, fuse visual and auditory data to warn of approaching predators. The stealthy predator, in turn, attempts to prevent its prey from becoming aware of its presence until escape is unlikely.

Humans routinely perform sophisticated single and multiple sensory fusion to generate a comprehensive situational understanding. The spatial direction of a sound is determined by evaluating the amplitude difference of acoustical signals at the two ears. Three-dimensional (3-D) visual perception is accomplished by stereo processing of pairs of two-dimensional (2-D) images. The fusion of sound and vision over time permits the estimation of the speed of an oncoming vehicle. Biological systems fuse information from the primary senses of sight, hearing, touch, taste, and smell, as well as from internal senses. Motor skills, for instance, rely on goal planning and dynamic replanning that are supported by feedback from both vision and balance sensing subsystems.

In general, humans successfully develop and maintain a dynamic (temporal reasoning) situational awareness with respect to 3-D space\(^1\) (spatial reasoning) by fusing sensory-derived information with a priori domain knowledge using highly-focused, multiple level-of-abstraction analysis (hierarchical reasoning). Support to these three reasoning classes will be seen to represent key requirements of artificial situation awareness systems, as well.

B. Sensor Fusion

Traditional data fusion automation has tended to rely on relatively rigid, domain-insensitive algorithms. In addition, many of the reasoning paradigms employed, such as those that involve combinatorial optimization or that require the correlation between data sets and a large set of templates, possess high computational requirements. Thus for many data fusion tasks, traditional algorithms can be both fundamentally inadequate, as well as highly inefficient. Human analysts, on the other hand, successfully solve numerous complex data fusion tasks seemingly effortlessly. Consider the interpretation of the time-stamped radar detections shown in Fig. 2. Although the data set contains multiple closely-spaced targets, some with potentially crossing trajectories, humans effectively deal with such ambiguity by using powerful reasoning models that are sensitive to relevant domain constraints. When alternative interpretations of the observations exist, competing hypotheses are maintained until adequate information becomes available to validate the correct hypothesis.

Although traditional statistical radar tracking algorithms typically associate the “closest” new detection to an existing

\(^{1}\)To simplify the discussion, this paper will focus on 2-D spatial reasoning; extensions to 3-D space are straightforward.
track using a relatively simple evaluation metric, human radar analysts typically employ multiple context-sensitive target behavior “models.” By interpreting the time-stamped detections with respect to local topographic features shown in Fig. 3, tracks 1–3 are seen to be highly consistent with road-following vehicles, tracks four and five are consistent with a minimum terrain-gradient trajectory, while track six is not consistent with any class of ground-based vehicle. By evaluating tracks 1–3 with respect to road class, vehicle speed, and intertarget spacing, the first three targets can be deduced to be wheeled vehicles traveling in a convoy along a secondary road. Based on the vehicle speeds and the associated surface conditions along their trajectories, tracks four and five can be deduced to be tracked vehicles. Finally, because of its relatively high speed and the rugged terrain in the vicinity, track six is most consistent with a low-flying airborne target. Because the velocity of target six is too low to be a fixed-wing aircraft, the target can be deduced to be a helicopter.

The principal difference between traditional track assignment algorithms and the more robust, context-sensitive analysis just outlined is that humans tend to perform decision making in a higher-dimensional space. With a conventional tracker, the dimensionality of the decision process is limited largely to the dimensionality of the sensor-derived information. Humans, on the other hand, apply a wide range of relevant a priori information that effectively increases the dimensionality of the decision process. As a consequence, algorithms that attempt to automate such higher dimensional analysis must be capable of effectively exploiting large a priori knowledge bases.

**C. Algorithm Requirements**

To be effective, data fusion algorithms must be *robust, context-sensitive,* and *efficient.* Robustness measures the quality of performance of an algorithm. Because there can exist complex, and possibly dynamic dependencies among features, achieving robust algorithm performance virtually demands some form of model-based reasoning. The efficaciousness of the model employed depends on the complexity of the process being modeled. A problem that intrinsically exhibits few critical degrees of freedom would be expected to require a simpler model than one that possesses many highly correlated attributes.

Consider the problem of automating handwritten character recognition. Although handwritten characters possess a large number of degrees of freedom such as the writer’s style, position, and orientation on the page, character size, color, contrast-ratio, line thickness, and so forth, a simple model can capture the salient attributes of many characters. The character “H,” for instance, consists of two parallel, nearly vertical line segments connected at their approximate centers by a third line segment. Therefore, although a handwritten “H” intrinsically possesses many degrees-of-freedom, most of these dimensions are not critical for distinguishing it from other characters. Whereas an effective model-based approach would utilize only the key discriminating features, a nonmodel-based algorithm must typically compare each new character with a complete set of representative exemplars for the entire symbol set. Thus, nonmodel-based approaches tend to require the consideration of all combinations of both critical and noncritical problem dimensions.
Context-sensitivity is a measure of a problem’s dependency on domain knowledge and domain constraints. In the broadest sense, context includes both static constraints (natural and cultural features, physical laws) and dynamic constraints (current location of all enemy air defense batteries, time of day) that influence the problem solution. There exist four general classes of context-sensitivity. Context-sensitive knowledge is conditional knowledge that must be specialized before it can be applied. Context-insensitive knowledge is generic, absolute, relatively immutable knowledge that is effectively domain-independent (terrain can obscure radar coverage, wide rivers are obstacles to ground-based vehicles); such knowledge is fundamentally unaffected by the underlying context. Context-specific knowledge is reasoning knowledge that applies only in a given, fixed context. Context-independent knowledge simply ignores the effect of any underlying context.

Efficiency measures the relative performance of algorithms with respect to computational requirements. For complex applications, although exceptions exist, the following generalizations are offered.

1) Model-based reasoning approaches tend to generate both more effective and more efficient solutions than nonmodel-based reasoning.
2) Multiple level-of-abstraction reasoning approaches tend to generate both more global, as well as more efficient solutions than single level-of-abstraction reasoning.

III. DATABASE SYSTEMS

Data fusion applications can require access to many forms of supporting data, including tables (equipment characteristics, logistical databases), entity-relationship graphs (organizational charts, functional flow diagrams), maps (natural and cultural features), images (optical, forward-looking infrared radar, synthetic aperture radar), and 3-D physical models (terrain, buildings, vehicles). Perhaps the simplest data representation form is the file (often referred to as a flat file); due to the lack of organizational structure, data access normally involves exhaustive search.

Analogous to the index of a book, database indexing attempts to overcome the inefficiency of exhaustive search. Much like a subject index at the end of a book, the database index dimensions can provide access to the spectrum from very general to very specific information. Just as a book might provide multiple index dimensions, such as a subject index organized alphabetically and a figure index organized numerically, database systems can employ multiple data set indices. Numerous data representation schemes, including hierarchical, graph, and relational models exist that offer some form of indexing.

Since the development of the relational data model in 1970, relational database management systems (RDBMS) have experienced explosive growth, currently dominating the marketplace. In the relational model, data are maintained in tables. Each row of a table stores one occurrence of an entity, while each column maintains a separate attribute of that entity. To facilitate rapid search, tables can be indexed with respect to either a particular attribute or a linear combination of attributes. Multiple tables that share a primary key (a unique attribute of an entity) can be viewed together as a composite table (linking personnel data and corporate records through an employee’s social security number). Because the relational model fundamentally supports only linear search dimensions, it affords inefficient access to data that exhibit significant dependencies among
Fig. 4. Database supporting efficient spatial query.

multiple dimensions. Consequently, a RDBMS tends to be sub optimal for managing 2-D or 3-D spatially organized information.

Geographic Information Systems (GIS), on the other hand, were developed specifically to manage and manipulate spatially-organized information [1]. Most GIS employ either vector and/or raster-based 2-D representations of points, lines, and regions, as well as 3-D representations of surfaces stored in triangulated irregular networks (TIN) [2]. A limited number of GIS support 3-D spatial data structures such octrees.2 In addition to commercial GIS systems, numerous research prototypes are under development [4]. Because data fusion applications can require extensive spatially organized data, algorithm developers often maintain the required spatial data in a GIS and the nonspatial data in a RDBMS. Such “hybrid” database systems tend to be both inefficient, as well as difficult to maintain. To overcome such shortcomings, some relational database systems have begun integrating true GIS functionality into an existing RDBMS.

Object-oriented reasoning is a popular problem-solving paradigm that extends the notion of reasoning about physical objects to reasoning about more abstract entities. In the object-oriented reasoning paradigm: 1) objects can contain data, knowledge bases, and procedures, 2) objects inherit properties from parent objects based on an explicitly represented object hierarchy, and 3) objects can communicate with, as well as control other objects. The wide spread use of the object paradigm has led to the development of numerous object-oriented database management systems (OODBMS). A significant number of research-oriented systems exist as well. OODBMS allow users to define new data types as needed by an application (extensibility), while hiding implementation details from the user (encapsulation) [5]. In addition, such databases permit explicit relationships to be defined between objects. As a result of these characteristics, OODBMS offer more flexible data structures and more “semantic expressiveness” than the table-based relational data model [6]. Numerous extensions to the relational data model have been proposed to capture desirable attributes of the object model [7], [8].

IV. DATABASE SUPPORT TO ALGORITHMS

This section discusses database management system requirements to support the development of more robust, context-sensitive, and efficient data fusion algorithms.

A. Spatial, Temporal, and Hierarchical Reasoning

Insight into a human’s spatial reasoning ability can be gained by consideration of the following query:

Find all roads west of River 1, south of Road 10, and not in Forest 11.

Given the data representation in Fig. 4, humans can readily identify all roads that are within the query window. However, if the map-based data set was provided to the human in a vector-based form,3 the query would be considerably more difficult to answer. Although the vector representation might be technically “equivalent” to the representation in Fig. 4, the map-like presentation explicitly portrays all relevant spatial relationships within the data set. With a vector represented data set, on the other hand, all such relationships must be discovered by computational means. In addition, human perception permits boundary-only representations of regions to be interpreted as the boundary plus the enclosed area. This “areal-based” perceptual facility permits humans to perform 2-D set operations virtually “by inspection.” Thus the choice of data representation can significantly affect the efficiency of machine-based reasoning.

Manual approaches to tactical data fusion capitalize on the human’s natural spatial reasoning ability by providing analysts with fully registered, 2-D topographic maps, acetate overlays, and other spatially organized products. Sensor reports and analysis products are typically plotted on clear acetates that are overlaid on appropriately scaled topographic maps. In addition to supporting highly intuitive spatial reasoning, the “acetate overlay reasoning paradigm” effectively avoids search and analysis of irrelevant data. For

2A region octree is the recursive decomposition of 3-D space into octants which can be represented as a tree of dimension eight [3].

3In a vector form, points are stored as tuples, lines as the ordered list of the end point tuples of a piecewise line segment approximation to the lineal feature, and regions as the ordered tuple list of the vertices of the enclosing polygon boundary.
Fig. 5. Multiple level-of-abstraction depiction of information classes viewed conceptually as acetate overlays.

analysis that depends only on “primary road networks,” for example, acetates depicting other feature classes are simply ignored. Because spatial features that result from Boolean set operations among existing feature sets possess the same fully-registered form, all analysis products are readily usable in subsequent problem solving. Fig. 5 illustrates the temporal sensitivity, as well as the multiple level-of-abstraction character of several classes of spatially organized information.

Perhaps the most fundamental database search operation required by fusion algorithms is the retrieval of data that is “close” in both space and time to new sensor-derived detections. If both stationary and moving entities are maintained as discrete 3-tuples, indexing or sorting along any individual dimension is straightforward. However, most data fusion applications demand search across all three dimensions. A simple example illustrates the shortcomings of both single dimension indexing and linear indexing (as supported by the relational model) of these 3-tuples. Suppose that the existing database includes the following temporally sorted target detections

\[(x_{a1}, y_{a1}, t_{a1}) = (1, 1.1)
\]
\[(x_{c1}, y_{c1}, t_{c1}) = (3, 3.5)
\]
\[(x_{b1}, y_{b1}, t_{b1}) = (5, 2.10)
\]

and that a new report \(d = (x_{d1}, y_{d1}, t_{d1}) = (3, 1.5, 5)\) is obtained. Based on the temporal index, query point \(d\) appears to be “near” point \(c\) (since \(t_{c} = t_{d}\)) and “not near” either \(a\) or \(b\). Alternatively, if the data set were indexed along all three dimensions, with the \(y\)-coordinate as the primary index dimension, point \(d\) would appear to be “close” to point \(a\). In reality, as observed in Fig. 6, query point \(d\) is actually “far” from both points \(a\) and \(c\) and “close” to the line segment \(a-b\).

Although moving target data is often collected at discrete times (at the radar scan rate, for instance), the actual trajectory of a physical object is a continuous function of both space and time. While proximity-based search of target tracks can be performed on list-oriented tuple data, much more efficient data retrieval is possible if the database directly supports search along continuous dimensions of both space and time. With a 2-D spatially organized data representation that explicitly depicts the track segment \(a-b\), highly localized search about point \(d\) would reveal all spatially local target tracks. The distance from \((x_{d1}, y_{d1})\) to the estimated position along track segment \(a-b\) at time \(t_{d}\) can be readily computed.

The representation of moving targets by their continuous, time-referenced lineal trajectories matches the characteristics of the physical phenomenon, as well as supports highly effective database search dimensions. Equally important, the representation preserves spatial relationships between the target tracks and other database elements. Based on these considerations, there exists a strong justification for subordinating temporal indexing to spatial indexing. Because of the effectiveness of multiple resolution spatial representations, there also exists a strong justification for subordinating hierarchical indexing to spatial indexing. Since temporal reasoning can be treated as a special case of hierarchical reasoning, a well-defined relationship exists among these three reasoning classes.

As a natural consequence, reasoning can be classified as either spatial or nonspatial (semantic). As depicted in Fig. 7, each of these reasoning classes can be handled using either hierarchical or nonhierarchical approaches. Hierarchical spatial reasoning and hierarchical nonspatial reasoning are facilitated by multiple resolution spatial representations and tree-structured nonspatial representations, respectively. Each of these classes, in turn, may or may not involve temporal reasoning. If temporal reasoning is referred to as dynamic reasoning and nontemporal reasoning is referred to as static reasoning, four classes of spatial reasoning and four classes of nonspatial reasoning can be defined: 1) dynamic, hierarchical, 2) static, hierarchical, 3) dynamic, nonhierarchical, and 4) static, nonhierarchical. Therefore, a DBMS designed to support data fusion automation must provide effective support to
each of these reasoning classes. Since hierarchical reasoning technically subsumes nonhierarchical reasoning, and dynamic reasoning effectively subsumes static reasoning, data fusion applications are adequately served by a database management system that supports dynamic hierarchical spatial reasoning and dynamic hierarchical nonspatial reasoning. Support to these two reasoning classes represent the two principle database design requirements. Since conventional OODBMS adequately support hierarchical semantic reasoning, the balance of this paper focuses on database support to dynamic, hierarchical spatial reasoning.

B. Development of Intuitive Algorithms

The development of sophisticated data fusion algorithms is often facilitated by selecting effective data representation forms. The above discussion of spatial/temporal reasoning highlighted the advantage of data representations that provide natural search dimensions and that preserve relevant relationships among data sets. In addition to supporting database search efficiency, such representations can support the development of powerful and highly intuitive algorithms. Table-based representations, for example, are much more natural than tree-based data structures for analyzing personnel data. Hierarchical data structures tend to be more natural for capturing the organization of a large company than spatially organized representations. Raster representations of an image support the development of much more intuitive image processing algorithms than do vector-based representations of the associated polygons. In addition to matching data types to representation structures, intuitive algorithm development is enhanced by the full integration of all data types. For example, since a river possesses both semantic attributes (class, width, flow-rate) and spatial attributes (linear and/or region representations), reasoning about rivers is greatly simplified if the respective semantic and spatial data are fully integrated.

C. Efficient Algorithm Performance

For problems that require access to large databases, data representation selection tends to have a significant impact on algorithm performance efficiency. For algorithms that require access to table-based data sets, RDBMS can be highly efficient. Two-dimensional map-like spatial representations tend to support more efficient spatial templating algorithms than do vector-based data structures. Because top-down problem solving often yields both more global, as well as more efficient solutions, multiple level-of-abstraction semantic representations and multiple resolution spatial representations tend to support the development of more efficient algorithms than do nonhierarchical representations. Complex problem solving that requires 3-D object matching would be expected to be more efficient if the inherent 3-D character of the data sets are preserved by the database.

D. Data Representation Accuracy

In general, database representation accuracy must be adequate to support the full range of data fusion applications. For finite resolution representations, accuracy is related to the data sampling scheme. In general, data sampling can be either uniform or nonuniform. Uniformly sampled spatial data are normally maintained as integers, while nonuniformly sampled spatial data are normally represented using floating point numbers. A pixel-based representation of a region boundary is a uniformly sampled representation, while a vector representation is a nonuniformly sampled representation. For a fixed memory size, nonuniformly sampled representations tend to provide higher accuracy than uniformly sampled representations.

V. DATABASE PERFORMANCE EFFICIENCY

In addition to supporting the development of powerful, highly intuitive, and intrinsically efficient data fusion algorithms, overall algorithm performance efficiency can be enhanced by maximizing the internal efficiency of the supporting database management system. For real-time applications that require sophisticated search and manipulation of large databases, database performance efficiency can be a significant factor in the overall execution speed of an algorithm. Six key classes of database efficiency are considered.

Storage efficiency refers to the relative storage requirements of alternative data representations. Fig. 8 depicts two polygons and their associated vector, raster, pixel boundary, and quadtree representations. With a raster representation, approximately \( A/\Delta^2 \) nodes are required to store the region, where \( A \) is the area of the region and \( \Delta \) is the spatial width of a (square) resolution cell (or pixel). In order to accurately replicate the region shown in Fig. 8(a), the resolution cell size must be four times smaller than that required to replicate Fig. 8(b). Because the pixel boundary representation maintains only the boundary pixels of the region, the required cell count is proportional to \( P/\Delta \), where \( P \) is the perimeter of the region. For fixed \( \Delta \), the

\[ \frac{A}{\Delta^2} \]

Although vector representations are not normally treated as sampled representations, doing so will permit a number of useful observations.

The term quadtree is sometimes used to refer to a large class of hierarchical, 2-D spatial data structures [3]. In this paper, we will use it to refer specifically to the regular region quadtree. The regular region quadtree is a recursive decomposition of 2-D space into four equal-sized quadrants. Each level in the representation possesses twice the resolution of the preceding level. Quadtrees can be represented using either a pointer-based or nonpointer-based representation. In the nonpointer-based linear quadtree representation, leaf nodes are represented using a string notation that encodes the top-down path through the tree to the leaf node. Thus, in addition to providing a reasonably efficient variable resolution representation of spatial data, the linear quadtree effectively provides a multiple resolution representation, as well.
ratio of the node count for the raster representation relative to the pixel boundary representation is $A/(\Delta P)$. Although like the raster representation, a quadtree representation stores both the region boundary and its interior, for large regions, the required node count tends to be related to the region perimeter rather than its area.\footnote{Since the region quadtree encodes the interior of a region in a pseudo maximal block fashion, the representation can be considerably more storage efficient than a comparable raster encoding. For large regions, the worst-case storage requirement can be shown to be approximately $1.5P/\Delta$ \cite{10}.} Thus for uniform decompositions, the region node count depends principally on the required maximum resolution cell size and on the overall region “size” (either its area or perimeter).

Because nonuniform decompositions are data-dependent, no simple relationship exists between the region “size” and its storage requirements. For example, two regions of

![Fig. 8. Comparison of vector polygon, raster, pixel boundary, and regular region quadtree representations for two similar polygonal shapes.](image)

the same size that possess identical uniform decomposition storage requirements might have significantly different storage requirements under a nonuniform decomposition representation. Vector-based polygon representations are the most common form of nonuniform decomposition. When the complexity in the region boundary is high, a large number of polygon vertices are required to represent the region; when the complexity is low, few vertices are required. Because the complexity of the boundaries in Fig. 8(a) and (b) are the same, both require the same number of polygon vertices. In general, for a given representation accuracy, nonuniform decompositions tend to have significantly lower storage requirements than uniform decompositions.

Achieving search efficiency requires effective control of the search space size for the range of typical database queries. Search efficiency is enhanced by storing
or indexing data sets along effective search dimensions. Bounding boxes\(^8\) are a simple indexing scheme that affords significant search reduction potential for vector-represented spatial data sets. Many kinds of data possess a “natural” representation form, that, if preserved, facilitates search. As mentioned previously, 2-D representations that preserve the essential character of maps, images, and topographic information permit direct access to data along very natural 2-D search dimensions. Multiple resolution spatial representations support efficient top-down spatial search.

Database overhead efficiency includes both indexing efficiency and data maintenance efficiency. Indexing efficiency refers to the cost of creating data set indices, while data maintenance efficiency refers to the efficiency of reindexing and reorganization operations, such as tree-balancing that can be required following insertions and deletions. Because natural data representations do not require separate indexing structures, such representations tend to enhance overhead efficiency. Although relatively insignificant for static data sets, data maintenance efficiency becomes a significant consideration for highly dynamic data.

Association efficiency refers to the relative cost of evaluating relationships, such as inclusion and proximity, among data sets. Natural data representations tend to enhance association efficiency by preserving the inherent organizational characteristics of data sets. Two-dimensional spatially organized representations, for example, tend to be much more efficient than vector-based spatial representations for inclusion testing (“inside an error ellipse”) or spatial proximity evaluation (“within 2 km of”). As already discussed, the relational model supports efficient association of simple table-based data, but does not support the efficient association of data sets that are implicitly associated through their combined spatial and temporal proximity.

Complex query efficiency includes both set operation efficiency, as well as complex clause evaluation efficiency. Set operation efficiency demands efficient Boolean and fuzzy set operations among point, line, and region features. Complex clause evaluation efficiency requires query optimization for compound query clauses that can include mixed spatial and semantic constraints.

Implementation efficiency is enhanced by a system architecture and associated data structures that support the effective distribution of data, processing, and control. Fig. 9 provides a high-level view of these design considerations.

**VI. SPATIAL DATA REPRESENTATION CHARACTERISTICS**

A large number of spatial data structures have been developed and investigated over the years. Fig. 10 depicts a taxonomy that can be used to organize spatial data structures according to their underlying properties. At the highest level-of-abstraction, sampled representations of 2-D space employ either uniform (regular) or nonuniform (nonregular) decompositions. At a low resolution, spatial features tend to be uniformly distributed, while at a high resolution, spatial data tends to be nonuniformly distributed. Thus uniform decompositions are most appropriate for maintaining relatively low resolution spatial representations. Because of their uniformly sampled character, such decompositions inherently support registration among data sets, as well as efficient spatial indexing. Nonuniform decompositions, on the other hand, provide the most memory-efficient representations of high resolution spatial data.

**Hierarchical decompositions of 2-D spatial data support the continuum from highly global to highly local spatial reasoning.** Humans routinely employ top-down reasoning to solve complex problems [11]. Multiple resolution spatial representations support low resolution global problem solving followed by recursive top-down refinement. By analogy, the development of plans for a cross country automobile trip would logically begin with a low resolution map of the United States. Once a coarse resolution plan is identified, appropriate state, county, and city maps would be used to refine elements of the high level plan.

Areal-based representations maintain the boundary plus the interior area of regions, while nonareal-based representations store only region boundaries. Because Boolean

---

\(^8\) The bounding box of a region is defined as \( [(x_{\text{min}}, y_{\text{min}}), (x_{\text{max}}, y_{\text{max}})] \), where the values are the maximum and minimum values of the actual \( x \) and \( y \) coordinates of that region. The region is completely enclosed by this bounding rectangle.
set operations among 2-D spatially organized data sets inherently involve both the region interior, as well as its boundary, areal-based representations tend to support efficient set operations among all classes of spatial features. Set operation efficiency can be especially important for operations involving large multiply-embedded regions.

Data decompositions that preserve the inherent 2-D character of spatial data are defined as 2-D data structures. Two-dimensional data structures provide natural spatial search dimensions, eliminating the need to maintain separate search indices. In addition, 2-D representations preserve Euclidean distance metrics and other spatial relationships. Thus a raster representation is a 2-D data structure, while list-oriented representations of lines and region boundaries are non-2-D data structures.

Explicit data representations literally depict a set of data, while implicit data representations store information that permits reconstruction of the data set. A raster represent-
tation is an explicit representation because it explicitly depicts line segments, while a vector representation is an implicit representation, storing only the end-point pairs of the associated line segments. In general, implicit representations tend to be more memory-efficient than explicit representations.

Specific feature spatial representations maintain individual point, line, and region features, while composite feature spatial representations store the presence or absence of multiple features. A raster representation that stores the value of all features in a given cell is a composite spatial representation, while a pixel boundary list of a particular region is a specific spatial representation. A specific spatial representation tends to be most effective for performing spatial operations with respect to individual features, while a composite representation tends to be most effective for operating on classes of spatial features.

Data representations that are both regular and hierarchical support efficient tree-oriented spatial search. Since regular decompositions utilize a fixed grid size at each level of the decomposition and possess a fixed resolution relationship between levels, the resultant fully-registered data sets can be readily associated at all resolution levels. Fully-registered data representations that are both regular and areal-based support efficient set operations among all classes of 2-D spatial features. Because a raster representation is both regular and areal-based intersection can be computed by simple Boolean AND operations between respective raster cells in the regions; union is generated by computing the Boolean OR between the respective cells. Spatial representations that are both regular and 2-D preserve spatial adjacency among all classes of 2-D spatial features. Because they use a data-independent, regular decomposition, regular 2-D data structures do not require extensive database rebuilding operations following insertions and deletions. In addition, reindexing operations are not required because the representation does not require a separate spatial index. Because spatially local changes tend to remain local within the representation, spatial decompositions that are both regular and 2-D tend to be relatively dynamic data structures.\footnote{Data-independent regular decompositions are considered to be relatively dynamic data structures because local changes to the data remain relatively local in the data structure. In some data-dependent, nonregular decompositions, on the other hand, spatially local changes to the data can require extensive changes to the representation.} As a result, regular 2-D spatial representations support efficient database maintenance operations.

Because data fusion involves the composition among both dynamic (sensor data, data fusion products) and static (tables of equipment, natural, and cultural features) data sets, data fusion algorithm efficiency is enhanced by using compatible data representation forms. For example, if an application required intersecting target track files that were maintained in a regular 2-D data structure with the static road network that was maintained in a nonregular representation, association efficiency would tend to be poor due to the disparate nature of the representations. Thus both database association efficiency and maintenance efficiency are enhanced by storing static and dynamic information in identical spatial data structures.

Spatial representations that are explicit, regular, areal, and 2-D support efficient set operations, offer natural search dimensions among fully registered spatial data, as well as represent relatively dynamic data structures. Because they possess the key characteristics of analog 2-D spatial representations, such as paper maps and photographs, these representations are defined to be true 2-D spatial representations. Nontrue 2-D spatial representations, on the other hand, possess one or more of the following properties:

1) spatial data is not stored along 2-D search dimensions (list or table-based representations);
2) region representations are nonareal-based (bounding polygons);
3) nonregular decompositions are employed (vector representations).

True 2-D spatial representations tend to support both intuitive algorithm development and efficient spatial reasoning. Because all 2-D spatial relationships are preserved, the representation supports highly intuitive spatial search (“Northwest of point A,” “behind the front line,” and “inside a staging area”). Because raster representations store both the region boundary and its interior, intersection and union operations among regions require only simple pixel-by-pixel set operations, regardless of region complexity (e.g., embedded holes). Set operations among regions that are maintained in nontrue 2-D spatial representations, on the other hand, can involve highly computationally intensive operations. For example, the intersection of two simple polygons has a computational complexity of order \( n^2 m \) (since all combinations of line segment pairs may need to be tested for intersection), where \( n \) and \( m \) are the number of vertices in the two polygons \cite{12}. The computational complexity increases considerably when one or more of the regions have included holes. For real-time applications that require the intersection of multiple polygons, each containing thousands of vertices, intersection generation can have a significant impact on algorithm performance. Although the effective use of a bounding rectangle for vector-represented lineal and region features can enhance efficiency, bounding boxes offer ineffective search space control for multiply-connected regions and extended lineal features, such as roads, rivers, and topographic contour lines. In addition, bounding boxes provide limited support to proximity-based search.

Specific spatial representations that are explicit, regular, areal, 2-D, and hierarchical (or hierarchical true 2-D) support highly efficient top-down spatial search and top-down areal-based set operations among specific spatial features. With quadtree based region representations, for example, efficient top-down, multiple resolution intersection generation can be supported that capitalizes on two important characteristics of Boolean set operations. First, the product of the intersection of multiple regions is a proper subset of the smallest region. Second, the intersection product
does not depend on the order in which the regions are composed. Thus, rather than exhaustively ANDing all nodes in all regions, the intersection product can be more efficiently generated if the smallest region is treated as a variable resolution search “window” for interrogating the representation of the next larger region [10]. Nodes in the second region that are within this search window are potentially within the intersection product. Since nodes in the second region that are not within the first region can not be within the intersection product, they need not be tested. The product of the intersection of the two smallest regions becomes the search window for interrogating the next larger region. The recursive application of this simple optimal search strategy produces the final intersection product. Thus, while the computational complexity of polygon intersection for multiple simple regions depends on the product of the number of vertices in all polygons, with a regular grid-based spatial representation, optimal search-based intersection has computational complexity that is on the order of the number of cells in the smallest region. Composite spatial representations that are explicit, regular, areal, 2-D, and hierarchical support efficient top-down spatial search among classes of spatial features.

Based on the relationships between the database efficiency issues and the classes of spatial data decomposition just discussed, a number of generalizations can be formulated.

1) Hierarchical, true 2-D spatial representations can support the development of highly intuitive algorithms.
2) For problems that require relatively low resolution spatial reasoning, algorithm efficiency can be enhanced by using hierarchical, true 2-D spatial representations.
3) Representation accuracy is supported by implicit, nonregular spatial representations.
4) Specific spatial representations are most appropriate for reasoning about individual spatial features; conversely, composite spatial representations are preferable for reasoning about classes of spatial features. The determination of the closest road to a given location, for example, would be more efficient using a true 2-D composite road network representation rather than a representation that maintained only individual roads. In the worst case, the latter representation could require search of the entire road network database.
5) Finite resolution data decompositions that are implicit, nonregular, nonhierarchical, nonareal, and non-2-D tend to be highly storage-efficient.
6) Spatial indexing efficiency is enhanced by hierarchical, true 2-D representations that support natural, top-down search dimensions.
7) Because spatially-local changes require only highly local changes to the underlying representations, regular, dynamic, 2-D spatial representations can enhance database maintenance efficiency.
8) For individual spatial features, database search efficiency is enhanced by spatial representations that are specific, hierarchical, true 2-D, and that support distributed search; search efficiency for classes of spatial features is enhanced by composite hierarchical spatial feature representations.

Table 1

<table>
<thead>
<tr>
<th>Spatial data structure characteristics</th>
<th>Spatial database requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intuitive algorithm development</td>
<td><img src="image" alt="Table 1" /></td>
</tr>
<tr>
<td>Spatial reasoning accuracy</td>
<td></td>
</tr>
<tr>
<td>DB storage efficiency</td>
<td></td>
</tr>
<tr>
<td>DB overhead efficiency</td>
<td></td>
</tr>
<tr>
<td>DB search efficiency</td>
<td></td>
</tr>
<tr>
<td>DB association efficiency</td>
<td></td>
</tr>
<tr>
<td>DB complex query efficiency</td>
<td></td>
</tr>
<tr>
<td>DB implementation efficiency</td>
<td></td>
</tr>
</tbody>
</table>

9) Because true 2-D spatial representations preserve all spatial relationships among data sets, for specific spatial features, database association efficiency is enhanced by hierarchical true 2-D spatial representations of individual spatial features; for classes of spatial features, association efficiency is enhanced by hierarchical, true 2-D composite feature representations.

10) For specific spatial features, complex query efficiency is enhanced by hierarchical, true 2-D representations of individual features; for classes of spatial features, complex query efficiency is enhanced by hierarchical, composite, true 2-D spatial representations.

11) Finally, database implementation efficiency is enhanced by data structures that support the distribution of data, processing, and control.

These general observations are summarized in Table 1.

VII. DATABASE DESIGN TRADEOFFS

Because object-oriented reasoning can facilitate the construction of robust, context-sensitive data fusion algorithms, the development of an effective OODBMS could be expected to provide significant support to data fusion automation. Based on the design principles outlined above, this section explores the characteristics of an object-oriented database required to achieve an effective compromise among the algorithm support and database efficiency issues indicated in Fig. 1. As discussed in Section IV, support for dynamic, hierarchical spatial reasoning and dynamic, hierarchical nonspatial reasoning represent the primary database requirements. To satisfy these requirements, the database
must employ data structures that facilitate the storage of and access to time-sensitive multiple resolution spatial and hierarchically-organized nonspatial information.

Nonspatial (or semantic) declarative knowledge can be represented as $\eta$-tuples, arrays, tables, transfer functions, frames, trees, and graphs. Because object-oriented representations permit the use of very general internal data structures and because the object-oriented reasoning paradigm fully embraces hierarchical representations at the semantic object level, conventional object databases intrinsically support hierarchical semantic reasoning. Consequently, semantic object databases provide support for solving complex, multiple level-of-abstraction problems that possess extensive parent/child relationships, that lend themselves to problem decomposition, or that employ global problem-solving strategies. Because objects are highly modular, changes over time to a specific object or its attributes tends to remain within a single object or to affect only closely related objects. Thus semantic objects tend to be relatively dynamic data representations.

Analogous to a semantic object database, an effective spatial object database must support top-down, multiple level-of-abstraction reasoning with respect to classes of spatial objects, as well as permit efficient access to specific spatial objects. Thus just as the conventional semantic object paradigm requires an explicitly represented semantic object hierarchy, a spatial object database requires an analogous hierarchical object representation of 2-D space. Just as specific entities in conventional semantic object databases possess individual semantic object representations, specific spatial objects in a spatial object database require individual spatial object representations.10

A. Object Representation of 2-D Space

Consider the query: Identify the class-1 road that is closest to query point $(x_1,y_1)$. For a database that maintains only the representations of individual spatial features, the above feature-class query could require the interrogation of all class-1 road representations. Just as a conventional semantic object hierarchy permits efficient class-oriented queries, a hierarchical representation of 2-D space supports efficient spatial object class queries. At the highest level-of-abstraction, the object representation of 2-D space consists of a single spatial object that characterizes the entire area of interest (the Persian Gulf region, a single map sheet, a Division Area of Interest). At each successively lower-level in the spatial object hierarchy, 2-D space is decomposed into progressively smaller regions. Higher-order spatial objects characterize the general properties of their offspring nodes just as higher-order semantic objects possess more general properties than their offspring.

Based on the principles outlined in the last section, an object representation of 2-D space must possess the properties shown in Table 2. As previously mentioned, a true 2-D spatial representation possesses properties (1)–(5). With the addition of property (6), the object representation of 2-D space would provide a hierarchical, true 2-D spatial representation of classes of point, line, and region features. A pyramid data structure possesses properties (1)–(6). Since the existence of multiple classes of spatial objects (roads, waterways, soil type) can be maintained within each cell, the pyramid permits the representation of composite features. Because a pyramid is a true 2-D spatial representation, it is a relatively dynamic data structure. In particular:

1) spatially local changes require only relatively local changes to the data structure;
2) no reindexing is required following updates;
3) and because the pyramid is a data-independent representation, extensive tree-balancing operations are not required following insertions or deletions.

Finally, due to both its hierarchical and grid-based character, a pyramid data structure readily accommodates data distribution. Therefore, a pyramid data structure satisfies all the requirements for an object representation of 2-D space.

B. Hybrid Spatial Feature Representation

The traditional approach to spatial database design involves the selection of a single spatial data structure for point, line, and region features. However, a more effective solution is to employ a hybrid spatial representation because it introduces additional degrees of freedom into the design process. A simple, yet effective hybrid spatial data structure that achieves an effective compromise among all the preceding design considerations involves the integration of a multiple resolution, low resolution spatial representation with a memory-efficient, high resolution spatial representation.

1) Low Resolution Spatial Representation: Based on the representation principles previously developed, the low resolution spatial representation should possess the properties shown in Table 2. With the exception of the composite representation property, the low resolution spatial representation requirements are identical to those of the object representation of 2-D space. Whereas a composite-feature-

<table>
<thead>
<tr>
<th>Representational Charecteristics</th>
<th>Key spatial data types</th>
<th>Object representation of 2-D space</th>
<th>Low resolution spatial representation</th>
<th>High resolution spatial representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Plural resolution</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>Regular / Non-regular</td>
<td>Regular</td>
<td>Regular</td>
<td>Non-regular</td>
</tr>
<tr>
<td>3</td>
<td>Areal / Nontoreal</td>
<td>Areal</td>
<td>Areal</td>
<td>Nontoreal</td>
</tr>
<tr>
<td>4</td>
<td>2-D / Non-2-D</td>
<td>2-D</td>
<td>2-D</td>
<td>Non-2-D</td>
</tr>
<tr>
<td>5</td>
<td>Explicit / Implicit</td>
<td>Explicit</td>
<td>Explicit</td>
<td>Implicit</td>
</tr>
<tr>
<td>6</td>
<td>Hierarchical / Nonhierarchical</td>
<td>Hierarchical</td>
<td>Hierarchical</td>
<td>Non-Hierarchical</td>
</tr>
<tr>
<td>7</td>
<td>Specific features / Composite features</td>
<td>Composite features</td>
<td>Specific features</td>
<td>Specific features</td>
</tr>
<tr>
<td>8</td>
<td>Relatively dynamic data structure</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>Distributed representation property</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>Low / High precision</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
</tr>
</tbody>
</table>

10 A number of authors have proposed the use of object-oriented representations of spatial data [12]–[14].
based representation supports efficient spatial search with respect to classes of spatial features, a specific feature-based representation supports effective search and manipulation among specific point, line, and region features. The regular region quadtree possesses all the properties identified in Table 2 [10].

2) High-Resolution Spatial Representation: Based on the design recommendations from Section VI, the selected high-resolution spatial representation should possess the ten properties indicated in Table 2. Vector-based spatial representations clearly meet the first four criteria. Because they offer a nonhierarchical representation of specific features based on an implicit piecewise linear representation of lines and region boundaries, vector representations satisfy properties (5)–(7). Because changes to a feature involve the modification of the property list of a single feature, vector representations are relatively dynamic data structures. Since the representation of individual features is self-contained, vector represented data sets can be readily distributed. Finally, vector representations of spatial data inherently provide high precision. Thus vector-based representations satisfy all the requirements of the high resolution spatial representation.

The highlighted branches in Fig. 10(b) and (c) indicate the two key branches of the spatial data structure taxonomy that comprise the hybrid spatial feature representation. The quadtree-based low resolution representation supports efficient multiple resolution spatial reasoning, as well as provides a hierarchical index to the high resolution vector-based representation.

C. Composite Data Representation

Effective database support to data fusion applications requires full integration of the semantic and spatial objects. Fig. 11 depicts a high level view of the semantic/spatial database kernel showing the implicit links between the various data structure.11 Because a pyramid can be represented as a complete quadtree, the object representation of 2-D space and the low resolution representation of individual spatial features are intrinsically linked. In addition, the quadtree representation supports efficient low resolution spatial reasoning and provides a hierarchical spatial index to high resolution vector-represented point, line, and region-boundary features. Table 3 summarizes key characteristics of the spatial object representation and demonstrates that it embraces the spectrum of spatial data representation properties. Table 4 compares the effectiveness of the vector, raster, pixel boundary, and the hybrid spatial representation with respect to the database design criteria.

Fig. 11. High level view of the semantic/spatial object database kernel depicting the implicit linkages between the various data structures.

Table 4

[Table content]

11 A more detailed discussion of the integration of the semantic and spatial object representations, as well as query optimization for mixed spatial/semantic object queries is found in [10].
embedded within a complex domain. By establishing the context within which to interpret the sensor-derived information, relevant domain knowledge can be used to enhance the analysis process. Thus data fusion automation can be viewed as the object-centered analysis of both sensor-derived and nonsensor-derived information. Due to the potentially large volume of both sensor-derived data (imagery, overlays, and video) and nonsensor derived data (topographic features, cultural features, and past, present, and future weather conditions) that must be processed, the effectiveness and efficiency of machine-based reasoning can be expected to be influenced by the character of the supporting database management system.

The top-down design of an object-oriented database kernel that supports both the development and efficient execution of robust, context-sensitive data fusion algorithms has been outlined. At the highest level of abstraction, the database kernel consists of both semantic and spatial objects. Because conventional OODBMS provide adequate support to semantic object representation, the focus of our discussions were design criteria for the spatial object representation.

A spatial object realization that consists of an object representation of 2-D space integrated with a hybrid spatial representation of individual point, line, and region features was shown to achieve an effective compromise across all design criteria. The object representation of 2-D space provides a spatial object hierarchy that is metaphorically similar to a conventional semantic object hierarchy. Just as a semantic object hierarchy supports top-down semantic reasoning, a spatial object hierarchy supports top-down spatial reasoning. A hybrid spatial representation, consisting of a regular region quadtree data structure integrated with a vector representation of individual point, line, and region boundaries, supports efficient top-down search and analysis, as well as high precision refined analysis of individual spatial features. Because both the object representation of 2-D space and the multiple resolution representation of individual spatial features use the identical quadtree decomposition, the object representation of 2-D space can be viewed as a composite of the spatial indices of the individual spatial objects.

**REFERENCES**


Richard Antony received the B.S. and M.S. degrees in electrical engineering from the University of Maryland, College Park, in 1968 and 1972, respectively.

He is a Senior Computer Scientist at the U.S. Army CECOM Intelligence and Electronic Warfare Directorate, Warrenton, VA. He has been the Army member of the Joint Directors of Laboratories Data Fusion Group since 1986. During 1991–1992, he was a Visiting Research Fellow at the Center of Excellence in C3I at George Mason University, Fairfax, VA, where he worked on the book *Principles of Data Fusion Automation*. His background includes work in digital simulation, systems analysis, radar target classification, and electronic attack. His primary interests include hierarchical, temporal, and spatial reasoning, spatial database design, and machine-based reasoning.