ABSTRACT

Title of dissertation: ISSUES IN VISUAL QUERYING AND INDEXING FOR THE EFFICIENT RETRIEVAL-BY-CONTENT OF ARRANGEMENTS OF POINT OBJECTS

Charles Bentley Cranston
Doctor of Philosophy, 2007

Dissertation directed by: Professor Hanan Samet
Department of Computer Science

In the context of computer science, an *index* is a data structure that enables the efficient retrieval of specific items from a collection of data. An *image database* is a collection of digitized map images, with relevant objects or features represented by *symbols*. One search method for such a database is to determine the set of images that contain a desired arrangement of symbols. Examples are presented of index data structures to support the *position-independent search* of an image database, where the absolute positions of the symbols within the image are unimportant, and only the relative spatial relationships of the symbols are significant. By separating the *size*, *shape*, and *orientation* attributes of an arrangement, these index structures support efficient searching that is either size-dependent or size-independent, and either orientation-dependent or orientation-independent. The visual language of an existing retrieval-by-content image database system is extended to allow intuitive control of the additional search flexibility.

ISSUES IN VISUAL QUERYING AND INDEXING FOR THE EFFICIENT RETRIEVAL-BY-CONTENT OF ARRANGEMENTS OF POINT OBJECTS

by

Charles Bentley Cranston

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2007

Advisory Committee:
Professor Hanan Samet, Chair/Advisor
Professor Shunlin Liang, Dean's Representative
Professor Amitabh Varshney
Professor Dave Mount
Professor V. S. Subrahmanian

© Copyright by
Charles Bentley Cranston
2007

Portions of this work were previously published in [12, 13, 10, 11].

Preface

When I first studied the visual query paradigm of the MARCO system of Samet and Soffer [39, 48], I was struck by the novelty of the problem it poses: while indexing database objects by their properties is well understood, an index to support MARCO searches should be based not on the properties of any single object, but instead on the gestalt embodying the mutual arrangement of the multiple objects in any one database image.

Because the database objects are *tagged* (as hotels, airports, etc), a simple inverted-file approach could be used to find the database images containing the desired set of objects. However, such a design makes no use of the spatial interrelationships of the objects. The techniques described in this dissertation arose from a desire for a *spatial* index, that is, an index in which the spatial arrangements of the objects would play a primary part.

In researching the literature in this area, the work of Gudivada and Raghavan [21, 22] stood out as addressing the same kind of problem. Their use of the slopes of the $n(n-1)/2$ line segments between the n point objects presages my use of both the slopes and lengths in the interpoint method, and their attempts to support rotation-independent search were an inspiration for the work described in this dissertation.

Dedication

This work is dedicated to a long line of teachers, both formal and informal, who have aided my lifelong pursuit of understanding. A good place to start is with Mrs. Jean Richstone, a seventh-grade mathematics teacher who had her 12-year-old students paste construction paper into Möbius strips and who quoted from the "Worm Runner's Journal". It includes Patrick E. Hagerty, who ran the Systems Programming group at the old Computer Science Center, the late John Gannon, who served as chair of the Computer Science Department and who was instrumental in my decision to pursue an advanced degree, and a set of dedicated undergraduate and graduate teachers along the way.

My dissertation advisor, Hanan Samet, has been an outstanding teacher, supporter, and friend, and has been the source of much good advice over the years. I consider myself truly fortunate to have been part of his circle.

Acknowledgements

I am grateful for having participated in many extensive technical discussions with a number of graduate students over the years, including but not limited to Gísli R. Hjaltason, František Brabec, and Jagan Sankaranarayanan.

I would also like to thank Lindley Darden for a number of discussions on topics and current work in the Philosophy of Computer Science, and for having provided me with the unique opportunity of contributing technical and proofreading support in the production of her recent book.

Finally, I would like to acknowledge the efforts of Dorothea F. Brosius, who developed and maintained the package of LaTeX macros used in the typesetting of this dissertation.

Table of Contents

List of Figures

Chapter 1

Introduction

Maps provide a means of maintaining a database of cartographic information. Although maps are usually thought of as capturing positional information, they are also used to capture spatial relation information. In particular, relevant objects or features are represented by symbols, and both the topological layout of these objects and their inter-object distances are significant. Examples of such maps include floor plans, blueprints, and satellite images. Examples of features include doorways and windows, component parts of an assemblage, and rivers, roads, and buildings.

A collection of digitized map images is an example of an *image database*. Searching such a database includes the determination of which images within the database contain a desired arrangement of symbols, such as a beach north of a city or a youth hostel near a train station. For such a search, the absolute position of the symbols within the database image is unimportant, only the relative spatial relationship of the symbols is significant. This is called a *position-independent search*. Search for symbols with a particular spatial relationship has applications in mobile computing, mobile data management, moving-object and moving-framework databases, and location-based services.

While such queries can be posed using conventional database techniques (such as SQL [41, chapter 4]); constructing a *pictorial query* by dragging icons (represent-

ing the objects) into an approximation of the desired relationship is a more intuitive interface. But even when the basic icon-dragging paradigm is extended with additional menus and controls, the complexity of any query given by a single picture is limited. A more complex query can be formed by using the Boolean operators AND, OR, and NOT to join query pictures together into a Boolean expression. The parse tree for such an expression forms a *pictorial query tree* [49].

If database images can contain multiple instances of a particular kind of icon (such as multiple hotels in the same geographic area), an icon in a query image actually designates (a member of) an *icon class*, to be satisfied by a corresponding specific icon *instance* in database images matched. In addition to a topological arrangement of icon class designators, a query must also specify which spatial relationships between the classes of the query image are to be considered relevant in the search.

One important task in matching a query picture to a database image is to determine the correspondence between query picture icons and database image symbols. When more than one such mapping is possible, a naive evaluation of the query tree can result in generation of an anomalous result. If the query tree does not contain negation (i.e., has only AND and OR operators but no NOT operators) these anomalous situations can be resolved by introducing *binding* between icons in different query pictures, thereby requiring both to match the same database image symbol. When the query tree does contains negation, however, anomalies arise that cannot be addressed by icon binding.

This describes a spatial version of *retrieval by content*, which has been inves-

tigated in both the spatial and non-spatial domains. The MARCO (MAp Retrieval by Content) system of Samet and Soffer [39, 48] is an example of this search method.

While a minimal database search capability can be constructed using little more than a basic similarity measure (as a brute-force search would simply compare the query image against each database image in turn), efficient database access usually depends upon the existence of an *index*, an auxiliary data structure that is maintained by the database system to allow the efficient determination of (and thus access to) only the portion of the database relevant to the current task. Indexes often embody *abstraction* (with only a subset of the data being replicated into the index), and *structure* (supporting access to the index entries in more sophisticated ways than simply in sequential order of the database proper). The simplest form of such structure is an *ordering*. For example, in a book index, only certain key words are abstracted into the index, which is then alphabetically ordered. However, more complicated structuring methods are also possible.

As it is inefficient to regenerate the index for each new task, the information in the index must be in some sense *invariant*, that is, useful (perhaps to a greater or lesser extent) for all supported tasks. While many methods of generating an invariant may be possible, it is also necessary that the method chosen be a useful one. For example, in a database of two-dimensional line segments, one choice for an invariant would be a single point in four-dimensional space, with the two-dimensional coordinates of both end points supplying the four required coordinates. This is known as the *corner transformation* [40], which is a two-dimensional variant of an interval representation (e.g., [16]) and used by a number of researchers

(e.g., [17, 24, 25, 28, 34, 50]). However, while this invariant is useful for queries involving the line segments themselves, it is not always useful when the queries also involve the space in which the line segments are embedded (e.g., proximity queries involving points not on the line segments themselves). In this particular case, an invariant involving partitioning underlying two-dimensional space and then indexing the partitions intersected by each line segment is preferable for answering such queries.

There have been a number of studies of pictorial queries for image databases in recent years [44]. Although much of the image database research has examined matching a single query object to a database object, some prior work [3, 5, 7, 14, 15, 20, 21, 22, 26, 27, 29, 32, 30, 31, 42, 43] does address queries composed of several objects. Spatial ambiguity is addressed in [7, 14, 15, 29, 32]. In particular, PQBE [32] can express complex queries as compositions of query images joined by conjunction and disjunction.

Query in spatial retrieval by content is complicated by the fact that a query may contain icons corresponding to only a subset of the symbols in the desired database images, thus comparison between query image and database images must involve partial matching. Indexing schemes for these databases must allow for query with differing subsets of the symbols present in database images.

One approach [7, 27, 29] is to extract from each database image a linearization of the symbols while scanning along one coordinate axis, plus a linearization of the symbols along the orthogonal axis. These *2D strings* are independent of the icons in any given query, so they can be precomputed and stored in a separate index file.

When a query is presented to the system, the two corresponding linearizations can be extracted from the query image, and similarity can be evaluated as substring match. Typically the index data retains topological but not metric information. In the absence of any further categorization of these image linearizations, the entire index (representing all the images in the database) must be accessed for every query. Thus this is the type of index that reduces the information in each entry but does not reduce the set of entries that must be accessed. Furthermore, while this matching is inherently invariant with respect to both linear translation and uniform scaling, tying characterization of the database image to the coordinate axes makes rotationally invariant queries quite difficult, if not impossible.

Berretti, Del Bimbo, and Vicario [4, 5] have addressed the problem of indexing tagged spatial arrangements of objects, modeling an arrangement of image objects as an Attributed Relational Graph (ARG). In [5] a metric-space index is described, using a distance metric function based on the *optimal error-correcting (sub)graph isomorphism problem*. While this is an example of an index structure that is more complex than simple ordering, the algorithm given for the computation of the exact distance between two ARGs requires (according to the authors) "polynomial time of the fifth order". Furthermore, as the ARG used to place each database image into the index is evaluated using all objects present in that image, the efficiency of the method given degrades when the query ARG represents fewer objects than are present in database images. However, their approach does support fuzzy identification of database objects.

Alt, Aichholder, and Rote [1] describe a method to determine a *pseudo-optimal*

matching between two point sets, based on alignment of their *Steiner points*. Because the Steiner point is invariant under translation, rotation, and scaling, it provides the basis for a "center to point" method that is an alternative to the one described in Section 5.3.6.3.

Gudivada and Raghavan [22] describe a similarity function SIM that is designed to compute the degree of closeness between a query picture and a database image. The algorithm first determines the n icons common to the query and database images, and then compares the slopes of the $n(n-1)/2$ lines between icon pairs in each database image to the slopes of the lines between corresponding icon pairs in the query image, thus abstracting away the absolute positions of the icons and considering only the relative spatial relationships, as captured by these line slopes. The resulting similarity measure is invariant with respect to both linear translation and uniform scaling. To incorporate rotational invariance, algorithm SIM_R clusters the lines by degree of rotation, uses the mean rotation of the largest cluster to attempt to undo the rotation, then uses algorithm SIM to compute similarity. A computational proof of the irrelevance of the center of rotation is given. Alternatively, this can be demonstrated via an application of the familiar "alternate interior angles" theorem from classical geometry. Their approach does generate a measure of the similarity between a particular query image and a particular database image. However, because the common icon set depends on the particular query image, their algorithm does not generate an invariant that could be used as the basis of a database index.

Without any provision for the user to explicitly control "translational ambi-

guity" and "scale ambiguity", non-translationally-invariant and non-scale-invariant queries cannot be specified. Consequently, the values of these parameters are hard coded. If the user interface were to permit a choice of using SIM or SIM_R for any particular query, it could be said that this system permits explicit control of "rotational ambiguity".

It should also be noted that these algorithms do not provide an indexing method per se, but merely a function to compute the similarity between a query picture and any given database image. In the absence of any independent way to either abstract information from the database entries or limit the subset of database entries that must be examined, every database entry must be accessed. Thus the time to complete a query grows linearly with the total number of images in the database. Practical databases aspire to sub-linear scale performance. Later work by Gudivada [21] describes a scheme where a $\Theta\Re$ string is generated and stored for each database image. This scheme can be viewed as the polar coordinate analog to 2D strings, instead of scanning along a coordinate axis to generate a string, a radius is swept around the center of the image and the objects encountered are ordered, first by angle Θ and then by distance $\Re$ from the center. Because this sorting is topological, metric information is not retained. The corresponding string can be computed for a query image, and an algorithm SIM_G then computes a similarity value based on the strings. This computation is parameterized by three user-specifiable constants. Contextual ambiguity is addressed by ObjectFactor, which determines the relative importance of the same objects appearing in both the query picture and the database image. Spatial ambiguity is addressed by SpatialFactor, which deter-

mines the relative importance of the objects having the same spatial relationships to each other. If SpatialFactor is zero, then relative orientation is unimportant. Spatial ambiguity is also addressed by ScaleFactor, which determines the relative importance of inter-object distance in the similarity computation. If ScaleFactor is zero, then the computation is invariant with respect to uniform scaling. Providing explicit control over these search parameters is a major advance over the previous work.

Like the 2D string, the $\Theta\Re$ string is query independent, and thus can be precomputed and stored in a separate index file. Such an index stores the topology but not the metrics of the database image. However, every entry in the index file must still be accessed for every query.

The 2D-string is not unambiguous; it is possible for different arrangements of symbols to generate identical 2D-strings. The Virtual Image system [33] characterizes the symbols within an image not as a pair of strings but instead as a set of binary relations between symbols pairs. Each pair of symbols ob_i and ob_j and the relationship γ between them forms a tuple $(ob_i\, \gamma\, ob_j)$ and the resulting set of tuples is used to characterize the database image.

A variant of Virtual Image called Quantitative Virtual Images (QVI), which incorporates metric as well as topological information, has recently been proposed [8]. A metric value δ is added to each binary spatial relationship. Intuitively, this describes how far to the left (or above) object i lies with respect to object j.

Other schemes for characterizing database images as strings or tuples retain only the topological relationships between database symbols, thus it is clear that the

metric information is being abstracted away. It would appear that the index entries generated by QVI contain all the information in the original database image (except perhaps for a linear offset) and thus this scheme represents not an indexing system per se, but more an advanced means of compressing the database information to minimize the number of accesses necessary for an exhaustive scan.

In contrast, the search algorithms described in this dissertation use an abstract index space whose basis is a set of separable attributes, such as size, shape, and orientation. For every image in the database, groups of icons are mapped into a single point in the index space. This is a *point-based* indexing method [35, Section 3.4]. The goal is to support position-independent search, on either a size-dependent or size-independent, and either an orientation-dependent or orientation-independent basis.

Although it was expected that shape would be the most difficult attribute to characterize, the characterization of orientation is actually the most challenging. Efficiently dealing with the phenomenon of *rotational symmetry* generates much of the complexity associated with this attribute.

The remainder of this dissertation is organized as follows: Chapter 2 introduces the visual language of the MARCO system. Chapter 3 describes extensions and modifications to the MARCO visual language to support the new search flexibility inherent in the new index designs. Chapter 4 analyzes some anomalies that have been discovered in the current implementation of MARCO's "NOT" primitive, and lists some strategies for resolving them. Chapter 5 briefly discusses some issues in the history and philosophy of indexing, and presents some concepts that are particularly

applicable to spatial indexing. Chapter 6 describes methods to efficiently index pairs of point objects, including a scheme to compress or *prune* the resulting index. Chapter 7 continues with methods to index groups of three objects, by indexing the triangle associated with each triple. Chapter 8 extends the discussion to groups of four or more points, which exhibit as-yet unencountered properties. Chapter 9 contains a summary and concluding remarks.

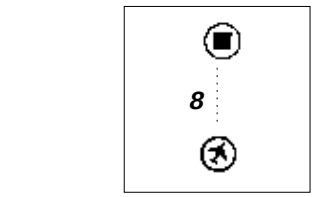
Chapter 2

Visual Language: Introduction

In this chapter we provide background information for some of the discussion to follow. A brief mention is made of some difficulties encountered when applying traditional database techniques to query by content in symbolic image databases. The important topic of *ambiguity* is discussed, and three types of ambiguity (matching, contextual, and spatial) are defined. The degree of explicit control of search parameters is discussed. The problem of multiple mappings is discussed both from the point of view of spatial databases and from a more general database perspective. Observations are made as to some qualitative differences between the Boolean NOT operator and the other two logical operators (AND and OR). The distinction made will be useful in Chapter 4, when interpretation and binding are discussed.

2.1 Conventional Database Techniques

Conventional database techniques (such as SQL) can be applied to spatial retrieval by content, with varying degrees of success. Data items that occur only once per database image (such as display images stored as Binary Large Objects) can simply become attributes in a row-per-image table. However, data items that occur more than once per image must either be packed into a single per-image attribute or stored in a secondary table. The latter approach is workable for data items that



Context: Both symbols must be found
Spatial: Distance and orientation are both significant

Hotel Airport

(a)

```
select distinct image.map
from image, symbol H, symbol A
where image.id = H.im and
      H.im = A.im and
      H.type = "hotel" and
      A.type = "airport" and
      dist(A.loc,H.loc) < 8 and
      dir(A.loc,H.loc) = "north"
```

(b)

Figure 2.1: Query expressed in Pictorial and Conventional Form

occur only once per symbol, such as instance-specific symbol annotation information. However, the number of inter-symbol relationships (such as inter-object distance and relative spatial orientation) grows as the square of the number of symbols present.

As an example, if the usual 8 cardinal directions (North, Northeast, East, ...) are recognized and 30 symbol types are distinguished, each instance of a symbol in the database can be represented by a row of a table with 240 attributes (distance to nearest symbol type 1 in North direction, distance to nearest symbol type 2 in North direction,...distance to nearest symbol type 1 in Northeast direction,...).

With this database structure, queries involving particular symbols within a given distance can be specified by reference to the appropriate field. However, the addition of a single new symbol type to the query system requires that 8 new fields must be added to every row in the table Such a scheme scales quite badly.

In general it is not possible for a symbolic image database to *explicitly* represent all information necessary to support the full range of desirable queries. Instead of storing the information explicitly (as all the $N(N-1)/2$ possible inter-object distances and orientations) the information is usually stored *implicitly*, as the spatial X

and Y coordinates of each of the N symbols. In Figure 2.1 the query "Find a hotel within 8 miles north of an airport" is shown in (a) pictorial and (b) conventional (SQL) form. For the pictorial query, it is assumed that the ambiguities inherent in pictorial specification (described more fully in the next section) have been resolved. For the conventional query, it is assumed that the conventional database model is extended by the addition of database procedures for the distance and direction operators. The relational database model depends strongly on successful optimization to achieve reasonable query execution times, and the success of such an implementation will depend on the extent to which queries containing these distance and direction primitives can be optimized.

As a result of the implicit storage of information, spatial queries expressed in a conventional query language (such as SQL) become complicated, sometimes requiring a nested query structure. The construction of such a query is not intuitive and therefore difficult for the naive user. Pictorial query trees were developed as a more intuitive way to accomplish retrieval by contents in image databases.

2.2 Ambiguity in Pictorial Queries

There are many different ways to interpret the meaning of any particular query picture. Consider the query picture in Figure 2.1 which contains a hotel icon and an airport icon. Such a picture could be interpreted as a request for all database images that contain both a hotel and an airport. Under this interpretation, the spatial positions of the icons in the query picture are unimportant. Under a differ-

ent interpretation, some aspect of the spatial positions of the icons (such as relative orientation) could be considered important. A query picture with a hotel icon positioned north of an airport icon could be interpreted as a request for all database images in which a hotel lies north of an airport. Under yet another interpretation, such a query picture could be interpreted as a request for all database images that contain *either* a hotel *or* an airport. Because of this lack of an unambiguous interpretation, query pictures are said to possess *ambiguity*. Three kinds of ambiguity are distinguished [48]:

1. If the database images were created by a pattern recognition algorithm, each instance of the recognition of a symbol may possess an associated confidence factor. Icons in a query picture are either present or not; a query picture by itself gives no knowledge of the required matching confidence required for each of its icons. Such *matching ambiguity* can be resolved by specifying a required matching confidence factor for each query icon.

2. The differing ways of interpreting the meaning of the spatial relationship between the icons in the query give rise to *spatial ambiguity*. For some queries, such as finding a hotel near an airport, only inter-object distance is important, and relative spatial orientation is irrelevant. For other queries, such as finding a hotel north of an airport, orientation may be the only significant spatial relation. In both of these cases, the query picture contains both a hotel icon and an airport icon. The ambiguity is in determining what (if anything) is important about the spatial relationships of the icons in the query picture.

3. If a query picture contains several icons, it is not clear if the intent is that a desired database image contain *every one* of them, or merely that *any one* of them be present. For example, to find a place to spend the night, a query picture might be constructed containing a campground icon, a hotel icon, and a youth hostel icon, with the interpretation that *any one* of them is desired. This distinction of any versus all is one form of *contextual ambiguity.*

   It is also not clear if the appearance of a certain set of icons in a query picture is meant to preclude the appearance of additional symbols in the database images to be found. This distinction of others versus no others is another form of contextual ambiguity.

Some query construction systems resolve these ambiguities by imposing a fixed, hard coded interpretation. Others provide explicit controls to vary the interpretation of a query picture on a query by query basis.

2.3 Explicit Control of Query Execution Parameters

One axis along which query construction systems differ is the degree of explicit parametric control provided to the user. With no explicit control, the interface is fully automatic; thus the user need not expend the effort to understand any explicit controls, but a given query may or may not produce the desired results. If it does not, no further recourse exists. For example, the similarity metric of PQBE [32] is tied to spatial directions; a rotationally-invariant query cannot be specified. On the other hand, the similarity measure of Gudivada and Raghavan [22] is inherently

rotationally-invariant; hence a non-rotationally-invariant query cannot be specified.

The decision to provide explicit controls can require more user training, but yields a higher probability that any particular ad-hoc problem can be addressed by the interface. For example, in SQBS [15] the "Cardinal Directions" spatial relation is only considered if the user explicitly specifies an orientation (for instance, by drawing a North arrow as part of the sketch). This allows both rotationally-invariant and non-rotationally-invariant queries to be issued by the same interface.

2.4 Multiple Mapping in Pictorial Query By Content

More than one feasible mapping from query icons to database image symbols can exist in any of these three cases: when a query picture contains more than one instance of an icon, when it contains any wild card icons (icons that match *any* symbol), or when a database image contains more than one instance of a symbol. This multiple mapping situation has only been partially addressed in symbolic image databases. In many cases it is not permitted; the assumption is made that only one instance of any particular symbol may occur in any database image, and that only one instance of any particular icon may occur in any query picture. Under these restrictions multiple mapping cannot occur.

When multiple mappings do occur, it can sometimes be the case that some mappings satisfy the query constraints but others do not. In such cases it is difficult to say that a particular database image either matches or does not match a particular query. When there are several pictures in a pictorial query expression, a naive

evaluation of the expression can yield anomalous results. The query evaluation engine must deal carefully with such cases.

2.5 Multiple Mapping in Other Domains

The multiple mapping problem is not unique to the domain of pictorial queries. It can occur any time an item from a query can match one of several items in a database entry.

Consider a database of textbooks and authors. A user might search for textbooks of which the author is both a college professor and a medical doctor. However, books often have more than one author, so the single query term "author" can have multiple mappings to the database entry term(s) "author". Now, how can a query be written to retrieve books with the *same* author as professor and medical doctor, without also retrieving books of which *one* of the authors is the professor and *another* is the medical doctor?

This query does not contain negation, thus the anomaly introduced by the presence of multiple mapping is of a type that can be addressed by *binding*, to ensure that the same object is matched by different parts of the query. Binding the two instances of the search target "author" so that the *same* author must be both a professor and a medical doctor specifies the correlation desired between the evaluations of the two clauses in the target expression. It is expected that situations in which binding is not a sufficient solution (analogous to those presented later in this chapter for spatial databases) will occur in non-spatial databases as well.

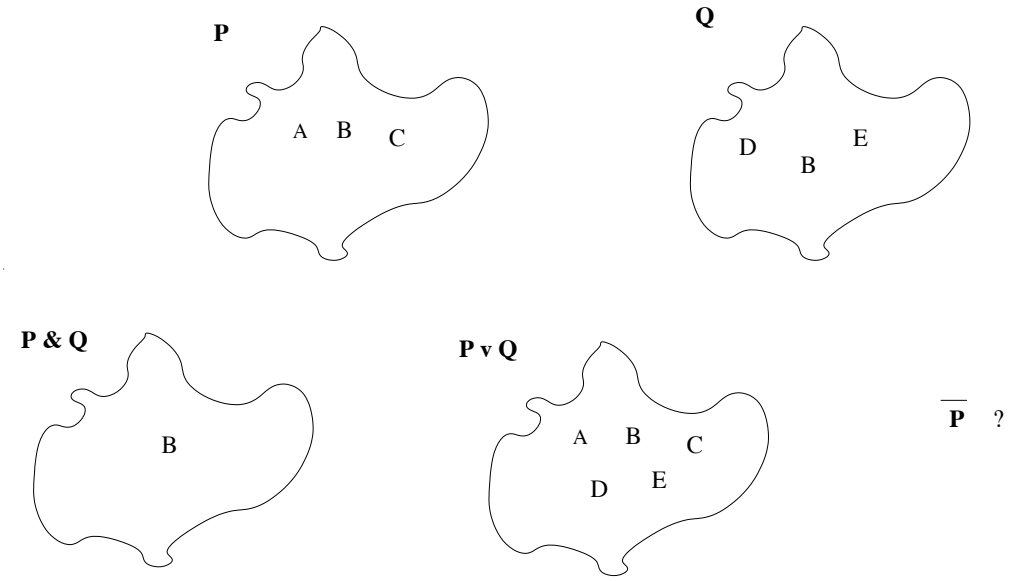


Figure 2.2: Universe needed to evaluate Boolean NOT

2.6 NOT is Qualitatively Different from AND and OR

An important qualitative difference between the Boolean NOT operator and the AND and OR operators is that AND and OR can be evaluated without reference to the universe of objects, whereas the evaluation of NOT requires such a universe. Evaluation of the AND and OR operations requires only an equality operation between the objects under consideration. Figure 2.2 shows two sets P and Q and the results of AND and OR operations between them. For the AND operator, the objects in one set can be examined in turn, and each one can be matched against every member of the other set. If an object from the first set matches any object of the second set, that object is part of the result of the AND. Evaluation of an OR operation can also be done using only an equality operation. Objects from both the first and second sets become part of the result, with the equality operation being used to ensure that no object occurs in the result more than once. In neither case

is a definition of the universe of objects required. Evaluation of the NOT operator requires such a definition. As the objects in the input set are defined *not* to occur in the result set, the only possible source of result objects is such a universe definition. Thus, it is quite likely that data structures designed to support the evaluation of AND and OR operations may require redesign to support NOT operations.

This is the reason the algorithm described in Section 4.4 preserves both the matching and non-matching interpretation sets. The union of these two sets is precisely the interpretation universe, and the NOT operation can easily be implemented by exchanging the two sets.

2.7 MARCO

Our research group has developed a pictorial query specification technique for image databases that addresses the issues of matching, contextual, and spatial ambiguity inherent in pictorial queries. A brief description is presented here. More details, including a description of the semiautomatic methods used to build the test database and illustrations of the user interface in action, can be found in [48, 49].

SAND (Spatial And Nonspatial Data) [2] is a prototype spatial database system, developed to be a research vehicle for work in spatial indexing, spatial algorithms, interactive spatial query interfaces, etc. The basic notion of SAND is to extend the traditional relational database paradigm by allowing row attributes to be *spatial* objects (e.g., line segments or polygons), and by allowing spatial indexes (such as quadtrees [35] or R-trees [23]) to be built on such attributes, just

as traditional indexes (such as B-trees [41, chapter 11]) can be built on nonspatial attributes.

MARCO (MAp Retrieval by COntent) [39, 48] is a map image database system developed as a SAND application. MARCO image processing is done in two phases. In an initial phase a scanned map image is semiautomatically parsed and the locations of various spatial features are stored in a database. Once the database is built, pictorial queries (expressed as pictorial query trees) can be run against the stored database. Icons representing the objects of interest are dragged from an icon menu to a position in the query picture being constructed. Pictorial query trees can be constructed from query picture leaf nodes and internal nodes corresponding to Boolean logic operations.

In the MARCO approach, matching, contextual, and spatial ambiguity are resolved by providing explicit control over each kind of ambiguity. The *matching similarity level* (MSL) determines a lower bound on the certainty required to match a database image symbol to a query icon. A MSL of 0.5 requires at least a 50% confidence in recognition of a symbol for the icon to match that symbol.

The *contextual similarity level* (CSL) specifies how completely the overall collection of symbols in a database image must match the collection of icons in the query picture. There are four possible settings for CSL. When set to "All symbols; no others", every icon in the query picture must match a unique symbol in a database image, and no other symbols may be present in that database image. When set to "All symbols; maybe others", every icon in the query picture must match a unique symbol in a database image, but additional symbols in that database image are

permitted. When set to "At least one symbol; no others", at least one of the icons in the query picture must match a symbol in a database image, and the database image cannot contain any symbols that are not in the query picture. Finally, when set to "At least one symbol; maybe others", at least one of the icons in the query picture must match a symbol in a database image, but additional symbols in that database image are permitted.

The *spatial similarity level* specifies how completely the spatial relationships (inter-symbol distance and relative orientation) of the symbols in a database image must match the spatial relationships of the icons in the query. There are five possible settings for SSL. When set to "Exact same location", the relative positions of database image symbols and of the corresponding icons in the query picture must be identical. When set to "Within distance; same direction", each pair of database image symbols must have the same relative orientation and be no farther from each other than the corresponding query picture icons. When set to "Any distance; same direction", each pair of database image symbols must have the same relative orientation as that of the corresponding picture query icons, but no constraint exists on inter-symbol distances. When set to "Within distance; any direction", each pair of database image symbols must be no farther from each other than the corresponding query picture icons, but no constraint exists on relative orientation. When set to "Any distance; any direction", there are no spatial constraints at all.

The current version of the MARCO query construction system provides on screen menus to set the MSL, CSL, and SSL values independently for each query picture in a query tree. Thus, one SSL setting affects all pairs of icons in a particular

picture, even though it would be desirable to set different SSL values for each pair of icons. Similarly one MSL setting affects all icons in a particular picture, although it is recognized that the ability to set different MSL values for each icon would be desirable. If the user interface is changed to support a separate MSL specification for each individual icon, the value of MSL to be used for testing the "no others" CSL constraint would no longer be well specified. One suggestion is to remove the "no others" semantic from the CSL menu and embed it into a new icon, perhaps with a death's head motif. Presence of that icon in a query picture would invoke a semantic in which that icon matches the "other" symbols (those not matched by other icons in the query picture), and such matching causes the database image to be considered undesirable for the query picture. This death's head icon would then be an intuitive place to specify the MSL value to be used for this "no others" constraint. These limitations in specifying SSL and MSL values are solely in the user interface code, there are no such limitations in the actual database search algorithm.

Database images deemed to match the query are presented in a separate response window, with boxes draw around symbols that correspond to query icons. Experimental extensions of the user interface incorporate additional interface elements for stepping between the different mappings returned by the query evaluation engine.

Queries for a specific symbol and for pairs of symbols constrained by their mutual distance were first described in [38] and [47] but the full user interface was not yet present. The pictorial query user interface was described in [46], which also described contextual and spatial ambiguity and similarity. In [45] the case of

more than one instance of a symbol in a database image was discussed, but the case of more than one instance of an icon in a query picture was explicitly left to future research. In [48] Boolean combinations of query pictures were described, and examples were presented that included Boolean NOT. In [49] the first explicitly tree-structured queries were presented.

In [19] Folkers describes a graph theory based algorithm for computing the full set of mappings produced at a single query picture. The computation of subgraph isomorphism is known to be NP-complete, but with the addition of contextual constraints the worst case time complexity is (depending on the exact contextual constraint), either $O(m\,2^m)$ or $O(m\,2^{n+m})$, where the query picture has n icons and the database image has m symbols. The addition of spatial constraints can result in significant further reductions in running time.

Chapter 3

Visual Language Modification for New Capabilities

The visual language used in the MARCO system needs to be enhanced to allow access to the additional search capability provided by the index structures devised to support the formats of the queries discussed in Chapters 6 through 8. This chapter discusses visual language elements ("widgets") to allow explicit control of the sensitivity to the search of the size and orientation of the icons in the query.

3.1 Control of Size-Independent Search

As the new database index structures support both size-dependent and size-independent search, it must be possible for the user to control this facet of the database search capability. The following sections describe two proposed forms of a user-language element (or "widget"), to provide the user such control.

One candidate for a size stringency control widget is shown in Figure 3.1(a). The small triangle icon at the left and large triangle icon at the right are meant to be visual cues to suggest the widget's function. The calibration line in the center indicates the control setting corresponding to an exact size match with the associated query picture. The left and right thumbs set the minimum and maximum scale factors allowed for the search. The gray bar is a visual cue to the range of sizes selected. The open circles on the thumbs are meant to be an application-wide visual

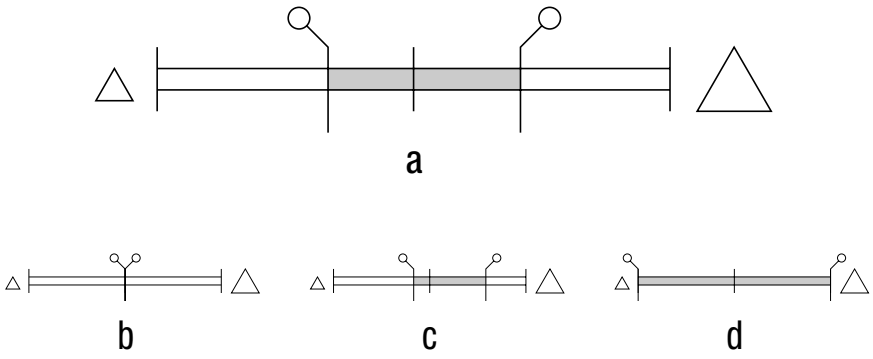


Figure 3.1: Linear Search-Size Widget

cue that the associated interface element is movable and a control, and to provide an unambiguous place to click when the thumbs are close to overlapping (as in Figure 3.1(b)). Control programming for this widget should constrain the thumbs so they are not allowed to overlap.

Figure 3.1(b) shows a size-dependent search, in which images are sought that match the given arrangement exactly. Figure 3.1(c) shows a search allowing the arrangements found to be anywhere from somewhat smaller to quite a bit larger. Figure 3.1(d) shows a completely size-independent search.

Another candidate for a size stringency control widget is shown in Figure 3.2(a). The four calibration ticks show the control setting corresponding to an exact size match with the query picture (use of a third circle for this function seems to result in a somewhat cluttered widget). It is possible that a better alternative might be the use of different colors, or some other differentiating circle attributes, such as dotted or dashed circles, or the use of thicker and thinner strokes.

The inner and outer circles comprise the movable thumbs to set the minimum and maximum scale factors allowed for the search. The gray annulus is a visual cue

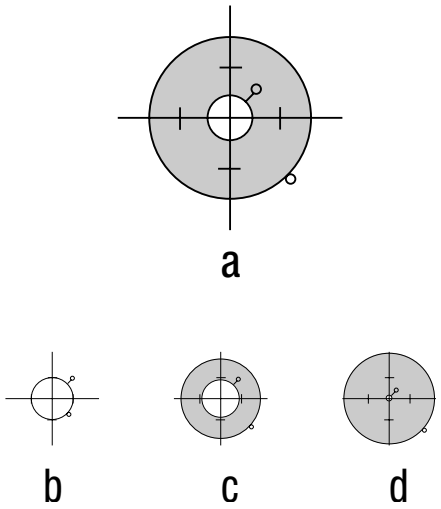


Figure 3.2: Circular Search-Size Widget

to the range of sizes selected. As with the design previously described, the open circles on the thumbs are a visual cue. Control programming for this widget should constrain the circles so they are not allowed to overlap.

Figure 3.2(b) shows a size-dependent search, in which images are sought that match the given arrangement exactly. Figure 3.2(c) shows a search allowing the arrangements found to be anywhere from somewhat smaller to quite a bit larger. Figure 3.2(d) shows a completely size-independent search.

3.2 Control of Orientation-Independent Search

As the new database index structures support both orientation-dependent and orientation-independent search, it must be possible for the user to control this facet of the database search capability. An proposed design for an orientation stringency control widget is shown in Figure 3.3(a). The vertical calibration line indicates the control setting corresponding to an exact orientation match with the associated

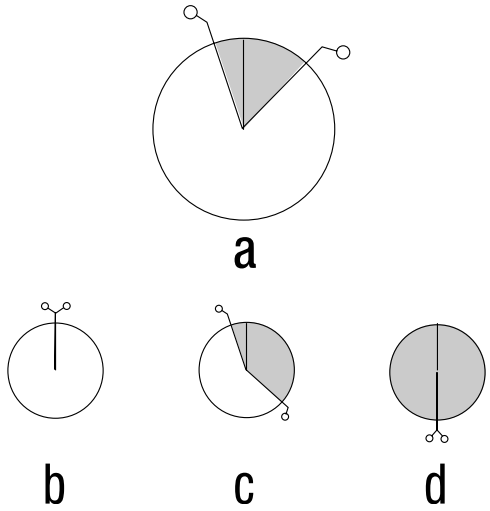


Figure 3.3: Search Orientation Widget

query picture. The two thumbs control the degree of orientation variation to be allowed for the search, and the gray pie wedge is a visual cue for the current setting. The control programming should prevent the thumbs from crossing; however, it is not necessary that the pie wedge always contain the calibration line, as the shape of the thumbs provides a visual cue for the thumb polarity.

Figure 3.3(b) shows an orientation-dependent search, in which images found must contain the desired arrangement in exactly the orientation given, Figure 3.3(c) shows a search that allows the images found to be rotated anywhere from somewhat to the left to quite a bit more to the right. Figure 3.3(d) shows a completely orientation-independent search.

Other widget designs are possible, and constitute one area for further research.

3.3 Control of Mirror-Image Search

As described in Section 7.2, in order to accommodate inline point arrangements the new database structures map both the left-handed and right-handed versions of configurations to the same database index point. As a byproduct, mirror-image-invariant searches are easily supported, simply by omitting some of the filtering normally performed after the index is accessed. However, it must be possible for the user to control use of this feature. As only a simple binary go-nogo control is required, the required visual language extensions are not expected to be particularly complicated.

Chapter 4

Negation

This chapter presents examples of the various kinds of anomalies that the presence of multiple mapping introduces into the pictorial query tree evaluation process. These examples show that when the query tree contains only Boolean AND and OR operators these anomalies can be addressed by adding symbol binding, (as is present in the current MARCO system), but when the tree contains NOT operators the simple addition of symbol binding is insufficient to resolve the anomalies. One solution to this problem is then presented: refinement of the semantics of the visual language to specify that, instead of computing a match or no match value for the database image as a whole, a separate value is computed for each unique mapping of query icons to database image symbols. It is shown that this refinement (in combination with symbol binding), is sufficient to address an extended class of anomalies. A novel notational scheme is described that can be used to hand-evaluate pictorial query trees with respect to specific candidate database images. This evaluation method is intuitive, self consistent, and produces both a final yes or no answer for any given database image and a set of feasible mappings that can be useful in displaying the final results of the query to the end user. An algorithm to evaluate the scheme using a bottom-up traversal with partial results represented by bitmaps is presented.

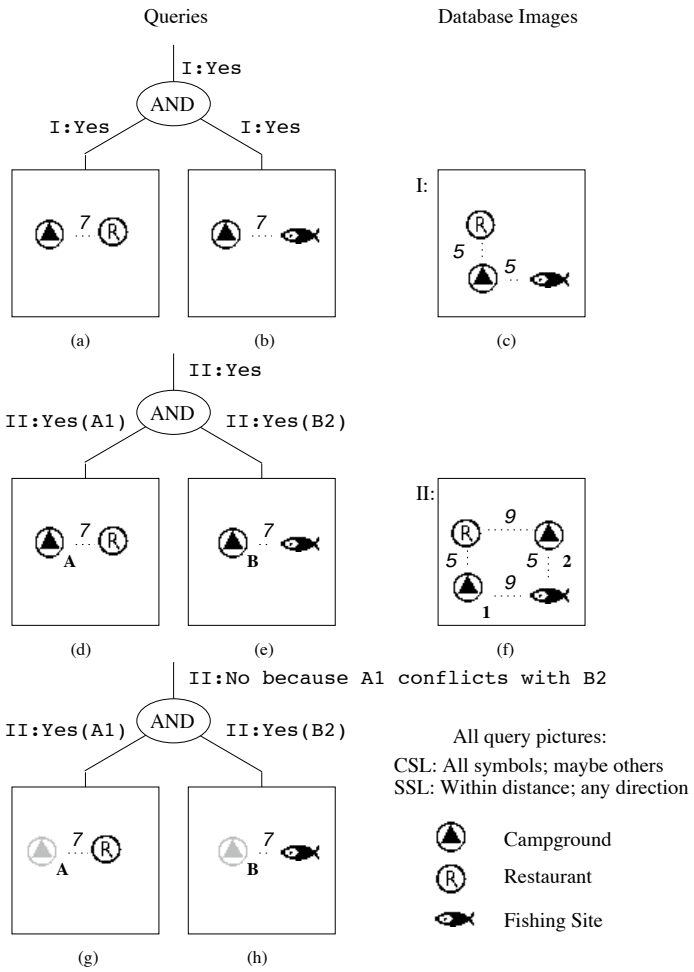


Figure 4.1: Binding resolves type 1 multiple mapping anomaly

4.1 Binding as a Partial Solution to Multiple Mappings

Figure 4.1 shows (a,b) a pictorial query tree constructed to find a campground close to a fishing site and a restaurant. The contextual similarity level (CSL) for both query pictures is specified as "All Symbols, Maybe Others", so query picture (a) must find both a campground and a restaurant, and query picture (b) must find both a campground and a fishing site. The spatial similarity level (SSL) for both query pictures is specified as "Within Distance, Any Direction", so the campground and restaurant found by picture (a) must be no farther than 7 distance units from each

other, and the campground and fishing site found by picture (b) must be no farther than 7 distance units from each other. Candidate database image I is shown in (c). Normally a database system such as MARCO would display actual map frames, which would not necessarily show inter-symbol distances. These diagrams present candidate database images in a schematicized form, with distance annotations added for clarity. As can be seen, for each database image a yes or no value for each query picture can be derived, and these values can be combined logically to give a useful answer.

This query produces an intuitively satisfying result as long as no database image contains more than one camping site. When a database image contains more than one camping site, however, as in the case of (d,e) the same query and (f) candidate database image II, it can return a result that seems intuitively anomalous. This can happen whenever there are multiple distinct mappings between the query icons and the database image symbols. The following notation is used to uniquely name the mappings. Query picture icons are labeled by bold upper case letters, and database image symbols are labeled with bold numbers. These labels will always appear to the lower right of the objects they label. Inter-object distances are shown in light italic numbers, which in database images will be placed so as not to be easily confused with the numeric labels. Mappings are named by a string of letters and numbers denoting the icon to symbol mappings. In Figure 4.1, the two mappings for query picture (d) are `A1` and `A2`. In some cases, a dash is used to indicate that no database image symbol matches a given query picture icon, such as in `A1B-C3`. This signifies that query icon A matches symbol 1, query icon B does not match

any symbol in the database image currently under consideration, and query icon C matches symbol 3. This notation is not part of the MARCO user interface. It was developed as a pencil and paper notation to facilitate conceptualizing mappings as independent entities. These names are dependent on both the query tree and the database image; even within the context of the same query tree there is no correspondence between mapping names for one database image and mapping names for a different database image.

In this example, query (d,e) reports database image II because the two camping site icons are not required to match the same database symbol. To remedy this, icons A and B of this query can be specified as being mutually *bound*, as in query (g,h). In the actual MARCO user interface, unbound icons are drawn in black and white and display color is used to draw bound icons, that is, all icons drawn in red are part of a bound group, all icons drawn in blue are part of a different, disjoint bound group. To make black and white copies of this chapter more readable, these diagrams show bound icon groups in differing shades of gray. The fishing sites in query tree (g,h) are shown in gray to signify that they are mutually bound.

When the query with binding (g,h) is processed, details of the mappings at each query picture are preserved, so that binding can be checked when the AND operator is evaluated. Because icons A and B are mutually bound and match different database symbols, it can be determined that database image II does not match the query.

4.2 Multiple Mappings with Negation

Figure 4.2 shows (a) a simple query picture specifying a picnic site and a fishing site that are no more than 2 distance units apart, and (b,c) a *range query*, constructed to match a fishing site and a campground whose inter-object distance is between 2 and 6 units. It does this by finding pairs whose distance is less than 6 units, then excluding those whose distance is less than 2 units.

The actual MARCO query construction system displays negation somewhat differently than in these diagrams. Negation of a pictorial query (leaf) node is indicated by drawing a horizontal bar over the node, while negation of an operator (internal) node is indicated by changing the operator name, such as from AND to NAND or from OR to NOR. In this chapter NOT is displayed as a separate node so that both of these cases can be treated in a unified way, and so that the intermediate results between the operator and the negation can be shown in the diagrams.

The simple query picture (a) is identical to component (c) of the range query. When there is only one mapping from the query icons to the database symbols (as is the case with (d) candidate database image I), it is possible to give a single, consistent yes or no answer for this picture.

In the following discussion it will be shown that it is not possible to consistently assign a value for this query picture in the presence of multiple mappings.

Candidate database image II (h) contains the symbols from image I and also a second picnic site symbol. This second picnic site introduces a second mapping for icon A. When the same queries (e) and (f,g) are applied to database image II, the

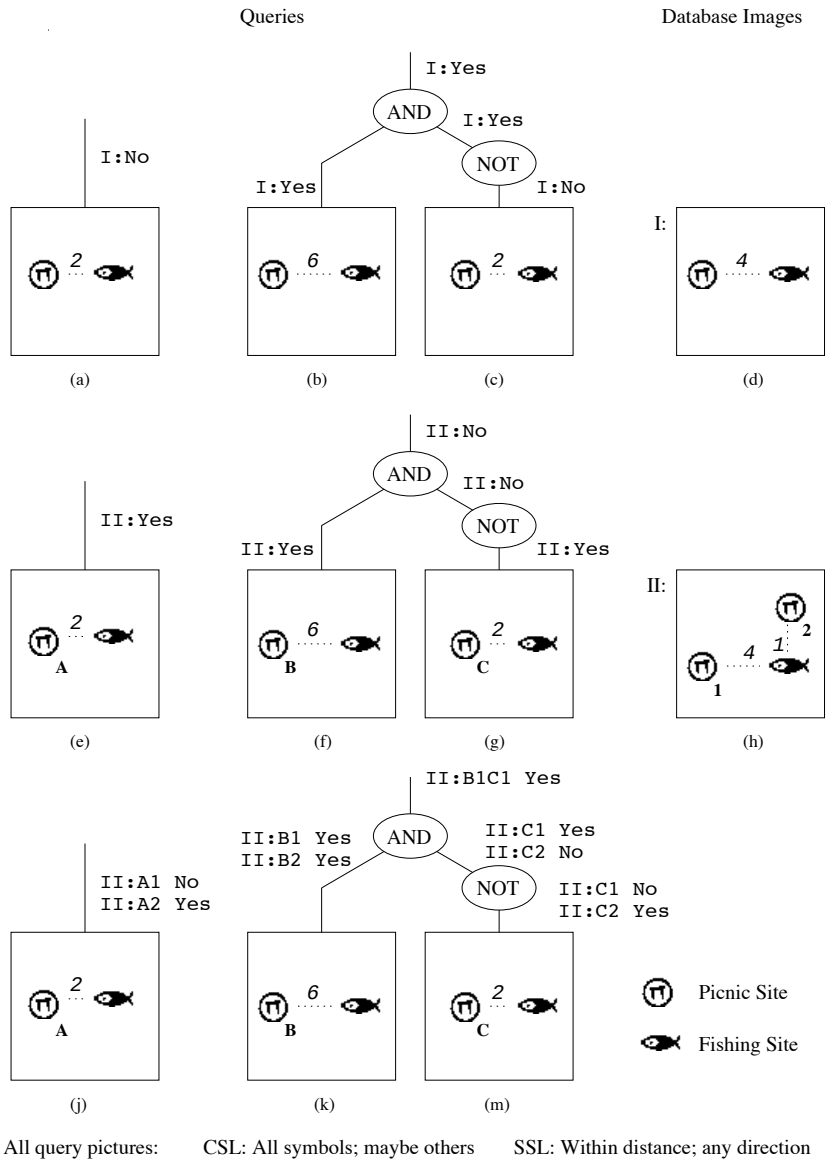


Figure 4.2: Type 2 multiple mapping anomaly caused by negation

anomaly caused by negation and multiple mappings can be demonstrated. It is not possible to give a consistent answer for the query picture represented by (e) and (g). In the case of (e), the answer must be yes, or database image II will not be found by the query. However, if yes is reported in (g), then database image II will not be found by query (f,g). The Boolean logic that leads to this false negative result is shown in the figure. The presence of the second mapping causes the range query to fail. This situation arises whenever at least one mapping satisfies the constraints (e.g., `A2`) but at least one other mapping fails to satisfy the constraints (`A1`). In such cases it is not possible to assign a consistent yes or no value to the query picture.

If the database image is the unit of reporting, the match between a given query picture and a given database image must have a consistent yes or no value, regardless of the position of that query picture in the expression tree. By showing examples in which the same query picture, matching the same database image, should generate inconsistent answers, these examples demonstrate that it is not possible to do so.

One solution to this problem is not to report yes or no for each database image, but instead to report yes or no for each individual mapping. When the mappings are reported as separate entities (as in j,k,m), both the simple query (j) and the composite query (k,m) can be made to yield an intuitively satisfying result. Note that both successful and failed mappings must be reported, because failed mappings become successful mappings if the subtree is negated. Note also that the space complexity of the intermediate results increases as the computation proceeds.

One way of describing this situation is by reference to the predicate calculus. The actual evaluation of the spatial constraints in a query picture is done on the

basis of the mapping. For example, in order to evaluate the distance between two symbols, it must be known precisely which symbols are being considered, in order for their spatial positions to be used in the distance formula. Yet an answer is required, yes or no, for the database image as a whole. A single answer is actually derived using an operator from the predicate calculus, one that is usually called "exists" and is written as $\exists$ in logic formulae. If any one mapping exists that satisfies the query constraints, the database image satisfies the query.

This approach is sufficient only when negation is not present in the query. When a predicate calculus expression is negated, instances of the $\exists$ operator must be changed to a different operator, one that is usually called "forall" and is written as $\forall$ in logic formulae. The clear implication is that systems that adequately process negation must at some times collectivize with the exists operation and at other times must collectivize with the forall operation.

The solution described in this section corresponds to preserving the mappings as separate entities during the entire evaluation of the query tree, and only then collectivizing with the exists operation after the tree has been evaluated.

4.3 Multiple Mappings with Negation and Binding

When multiple mappings occur in a query tree containing negation, a third type of anomaly can arise, one that can be resolved by specifying symbol binding. In Figure 4.3 pictorial query tree (a,b) returns a false positive for database image (c) because the fishing site icons from query pictures (a) and (b) match different

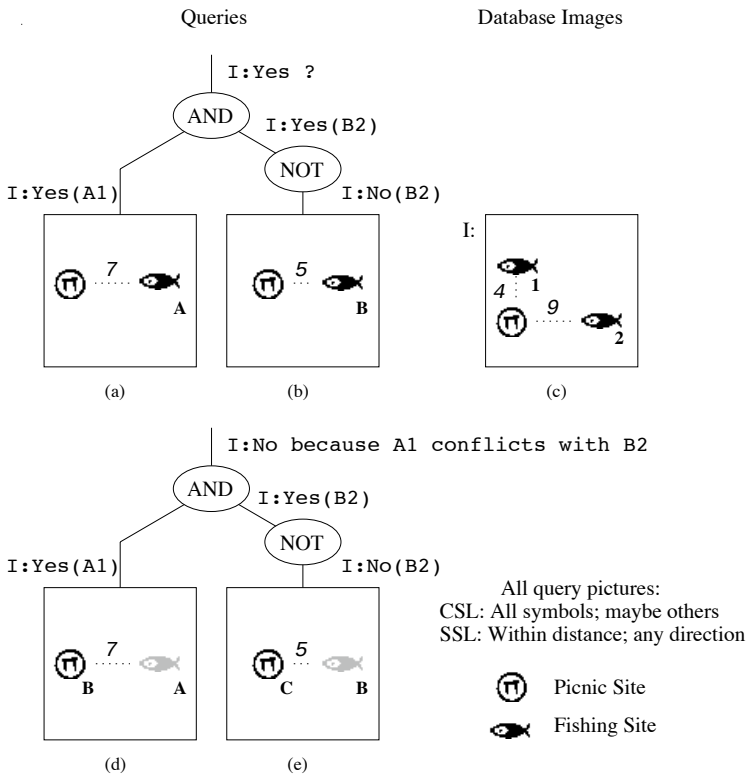


Figure 4.3: Binding resolves type 3 anomaly

database image symbols. So, even though a symbol pair is found that are within 7 units of each other, and a symbol pair is found that are not within 5 units of each other, they are different symbol pairs. This situation can be addressed by (d,e) *binding* the fishing site icons so they must match the same database image symbol. In a well specified range query both symbol pairs should be bound, to ensure consistency in all multiple mappings situations. In this case, the two picnic site icons should also be mutually bound, to protect against a database image containing more than one picnic site.

4.4 Pencil and Paper Algorithm

The following pencil and paper algorithm can be used to evaluate a candidate database image against a pictorial query tree. As a final notational shortcut all the mappings in both the accepted and unaccepted sets are compressed into a single string, with accepted mappings before a slash (/) and unaccepted ones after. Mappings are separated by commas, and if either set is empty a dash is used as a placeholder. Some examples using this notation are `D2E4,D3E5/D2E5,D3E4` and `A2B-C1/-`.

First, for every icon in every query picture, a list of matching database image symbols is constructed. This process takes into consideration matching similarity level but not contextual nor spatial similarity levels. Any query picture icons that do not match any database image symbol are given an empty (dash) matching.

Next, for each query picture, a set of all possible composite mappings is constructed. These mappings are then evaluated with respect to the contextual and spatial similarity levels, and are partitioned into accepted and unaccepted sets.

Finally, a value for the entire query tree is computed using the mapping sets generated at the query pictures and these rules for the Boolean operators at the internal nodes: If the accepted and unaccepted mapping sets for operand P are designated as P^+ and P^- respectively, and similarly for operand Q as Q^+ and Q^-, the operation NOT P returns $P^- \ / \ P^+$ (that is, the accepted and unaccepted sets are interchanged). The operation P OR Q returns $P^+ \times Q^+ \ \cup \ P^+ \times Q^- \ \cup \ P^- \times Q^+ \ / \ P^- \times Q^-$, that is, the Cartesian product of the operand mappings is

generated and the result accepted set consists of product mappings containing at least one accepted component, while the result unaccepted set contains only product mappings composed of two unaccepted components.

The operation P AND Q returns $P^+ \times Q^+$ / $P^+ \times Q^- \cup P^- \times Q^+ \cup P^- \times Q^-$, that is, the result accepted set consists of only product mappings containing two accepted components, while the result unaccepted set consists of product mappings containing at least one unaccepted component, with the additional provision that product mappings that cannot be unified are moved from the accepted set to the unaccepted set.

When the root of the tree is finally evaluated, the result consists of the two sets of mappings, an accepted set and an unaccepted set. If the accepted set is not empty, it contains successfully unified mappings which can be used to justify the matching between the pictorial query tree and the database image being examined.

Figure 4.4 shows a pictorial query tree (a-h) and a candidate database image (j). The intuition for this query is that it finds desirable places (hotels, scenic views, sites of interest, and campgrounds near restaurants), while excluding undesirable places (sports institutions, holiday camps, and campgrounds near airports). Query picture (a) specifies a contextual similarity level of "Any symbol; maybe others", so a match for any one of the three desirable icons is sufficient for a match, and a spatial similarity level of "Any distance; any direction", so there are no constraints on the inter-object distance or orientation. Query picture (b) specifies a contextual similarity level of "All symbols; maybe others", so both a campground and a restaurant are required, and a spatial similarity level of "Within distance; any direction".

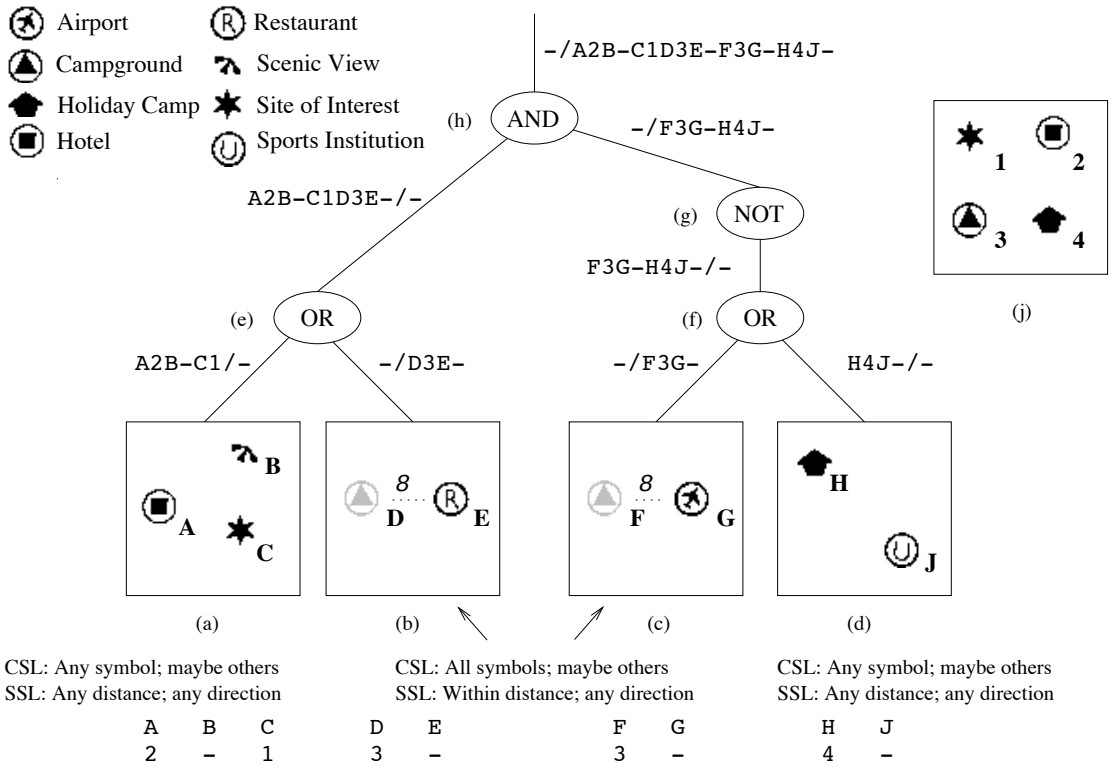


Figure 4.4: Example evaluation without multiple mappings

So, a further requirement is that they be within 8 distance units of each other. Query picture (c) specifies the same two similarity levels. So, both a campground and an airport must be found, and they must be within 8 distance units of each other. Query picture (d) has a contextual similarity level of "Any symbol; maybe others", so a match for either of the two undesired symbols is sufficient. As in the case of picture (a), there are no constraints on inter-object distance or orientation.

The mapping named by the string `A2B-C1D3E-F3G-H4J-` is the only one that exists between this query and database image (j). The combined mappings produced at the OR nodes (e and f) are in the accepted set because the logical operation is OR and at least one of the component mappings came from an accepted set. At the NOT node (g), the accepted and accepted sets are interchanged. At the AND node (h),

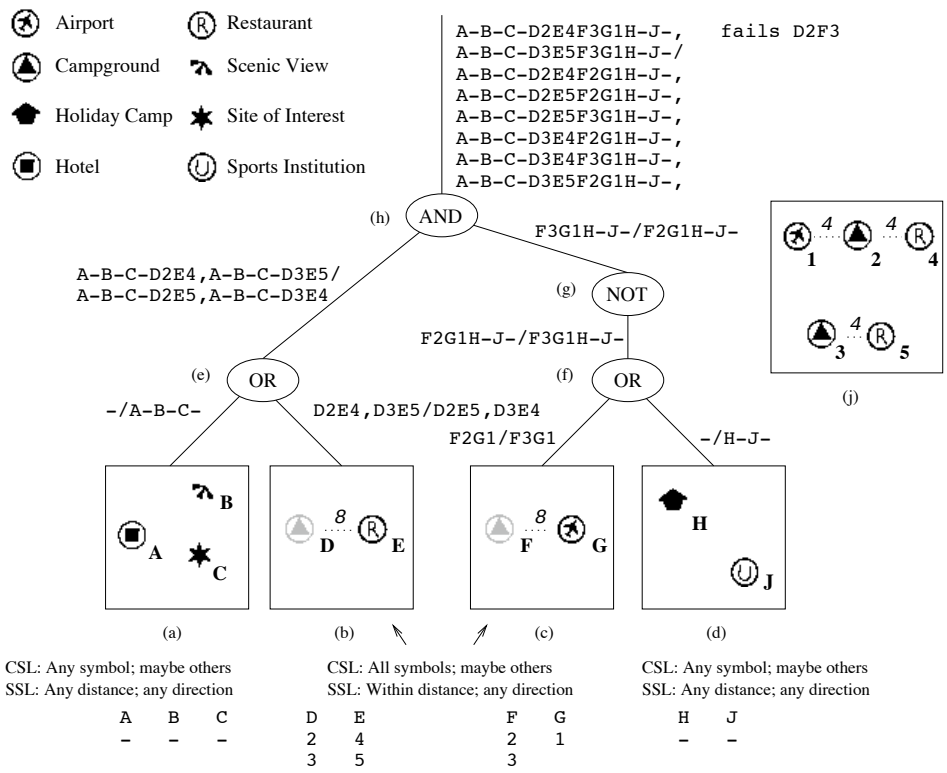


Figure 4.5: Example evaluation with multiple mappings

the final mapping is in the unaccepted set because at least one of the components came from an unaccepted set. The final result is that the query does not match this database image. Notice that if the holiday camp (4) were not present, the result of query picture (d) would be `-/H-J-` and the result of the OR operation (f) would be `-/-F3G-H-J-`. The result of the NOT operation (g) would then be `-F3G-H-J-/-`, and the result of the AND operation (h) would be `A2B-C1D3E-F3G-H-J-/-`, and the database image (j) would matched the query.

Figure 4.5 shows the same pictorial query tree (a-h) with a candidate database image (j) containing multiple mappings. The restaurant icon E matches both symbols (4) and (5), and each of the two instances of campground icon D and F matches both symbols (2 and 3). None of the icons in query picture (a) is matched, so it

returns an empty mapping `A-B-C-` in the unaccepted set. Each of the icons in query picture (b) can match one of two symbols, so there are four possible mappings in all, of which two are accepted (within 8 distance units of each other) and two are not accepted (not within 8 distance units of each other). The campgrounds icon in query picture (c) could match either (2) or (3), so there are two possible mappings, of which one is accepted. Matching at query picture (d) is analogous to that at query picture (a). At OR node (e) four combined mappings are produced. Two are accepted (those containing accepted picture (b) mappings) and two are not accepted (containing unaccepted picture (b) mappings). At OR node (f) two combined mappings are produced, of which one is accepted (from the accepted picture (c) mapping). At NOT node (g) the accepted and unaccepted sets are interchanged.

At AND node (h) a set of eight mappings are produced. Initially two mappings are accepted (those containing accepted mappings from each of the sub-nodes) but binding processing is done at AND nodes, and icons D and F are mutually bound. Mapping `A-B-C-D2E4F3G1H-J-` fails the binding test, because it assigns different symbols (2 and 3) to icons D and F. Because of this, the mapping is moved from the accepted set to the unaccepted set.

Only mapping `A-B-C-D3E5F3G1H-J-` survives into the final result. Intuitively this makes sense because symbols (3) and (5) are the paired camping site and restaurant for which the camping site is *not* close to airport (1). The existence of a mapping in the accepted set means this database image matches the query, and that mapping gives the correspondence between query icons and database symbols that a user interface would need to produce an explanation, if requested.

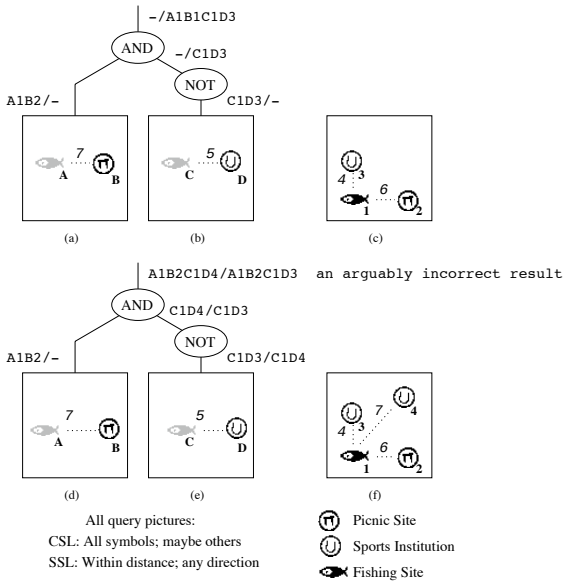


Figure 4.6: Loss of ability to exclude based on distance

4.5 Remarks on Negation

By maintaining *all* the mappings generated at each query picture, partitioning them into accepted and unaccepted sets, and combining those sets at AND and OR nodes in the way described, the union of the accepted and unaccepted sets in the intermediate results data structure is exactly the complete universe of mappings for the subtree those results represent. This provides the very information necessary for correct implementation of the negation operation.

4.6 Loss of Expressive Power

A certain amount of expressive power is lost, however, by the decision to treat mappings as independently reported units. Consider again the textbook-author example. One might issue a query for "A book on human sexuality that was *not* authored by Johnson" but be returned Masters and Johnson nevertheless, because

there exists an author (i.e., mapping from query-author to one of the database-authors) that is not Johnson, e.g., Masters. This can also occur in visual queries. Figure 4.6 shows (a,b) a query that excludes the presence of a sports institution within a threshold distance, (c) a database image with only one sports institution, (d,e) the same query, and (f) a database image with two sports institutions, one within and the other farther than the distance threshold. Reporting the mapping that matches the distant sports institution is a direct analogy to the two-author textbook anomaly.

H. Samet has observed that this anomaly can be addressed by defining both a *not all* operator and a *not any* operator [36]. Such an ability to parameterize the NOT operation adds a powerful explicit control to the user interface. However, the cognitive load imposed on the user by negation is already large, with both the over bar notation for query pictures and negation of subtrees by specifying negated operators (NAND, NOR etc). To add to this cognitive loading by essentially doubling the number of ways negation can be specified would seem to work against the intuitive nature of the pictorial interface.

4.7 Implementation

This section describes an algorithm to perform a postorder traversal of the query tree, building an intermediate data structure based on a bitmap representation of the mappings at each node.

Efficient algorithms for query by content will use some form of database index-

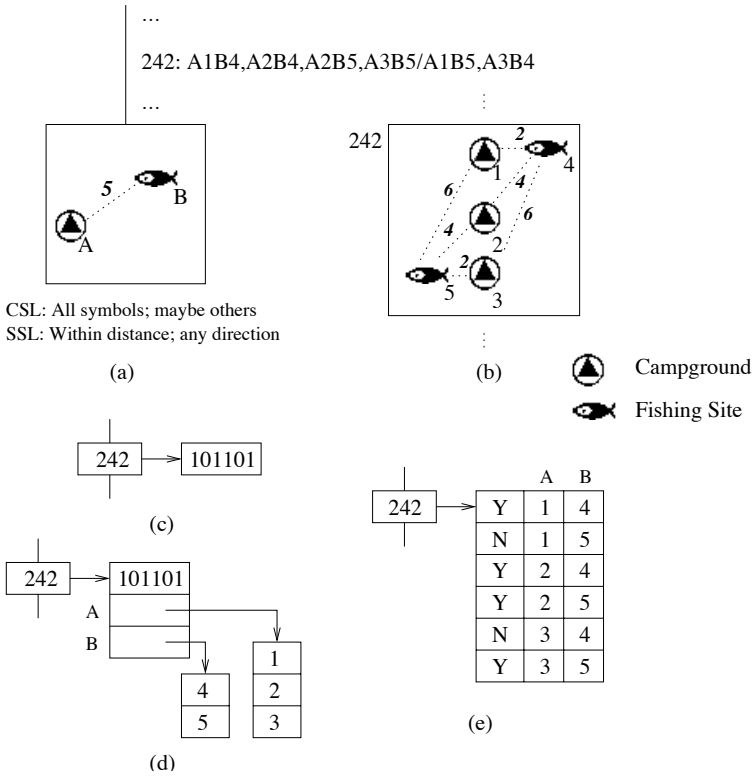


Figure 4.7: Options for intermediate data structure

ing. The current MARCO code uses an index of symbols, their containing database images, and the relative x-y coordinates of the each symbol within its database image. Because not all symbols occur in all database images, an enumeration of the database images in which a particular symbol exists forms a sparse list. The sparse lists for the symbols that match icons in the query picture are used to generate an intermediate result list containing all database images with symbols mapping to query icons, and enumerating all the different query icon to database symbol mappings. These lists become input to the Boolean logic operators at the internal nodes of the tree.

Figure 4.7 shows several different ways the icon to symbol mappings can be represented in the intermediate data structure. While there are six possible ways to

map the icons in query (a) to the symbols in database image 242 (b), only four satisfy the spatial constraint of the query. In (c) the intermediate data structure contains only a bitmap, with one bit representing each possible mapping. This requires the smallest amount of memory, but requires additional IO to the index (to reacquire the symbol mappings) every time an AND node must process binding, plus one more time when presenting the final query results to the user. In (d) the bitmap and a minimal amount of core data are maintained. The mappings corresponding to the bits in the bitmap can be regenerated without requiring redundant reads from the database index. In (e) the information is represented in explicit form. Alternative (d) requires somewhat less space than alternative (e), (in this example five entries rather than six), because the number of entries required by alternative (d) is the sum of the degrees of multiple mapping represented, while for alternative (e) it is the product of the multiple mapping degrees. This space savings comes at the cost of the time necessary to regenerate the mappings when binding must be processed.

The current MARCO code uses representation (e), but maintains only the satisfying mappings (such as those marked by Y in the figure), and does not maintain list entries for database images with only non-satisfying mappings. This is sufficient to process queries containing only AND and OR, but is the root cause of the current difficulties with negation.

Data structure (c) in Figure 4.7 is the least memory intensive choice for the organization of the intermediate data structure. To demonstrate that this minimal data structure can support both AND and OR operations as the tree is evaluated, Figure 4.8 shows how both operations can be accomplished. The data being com-

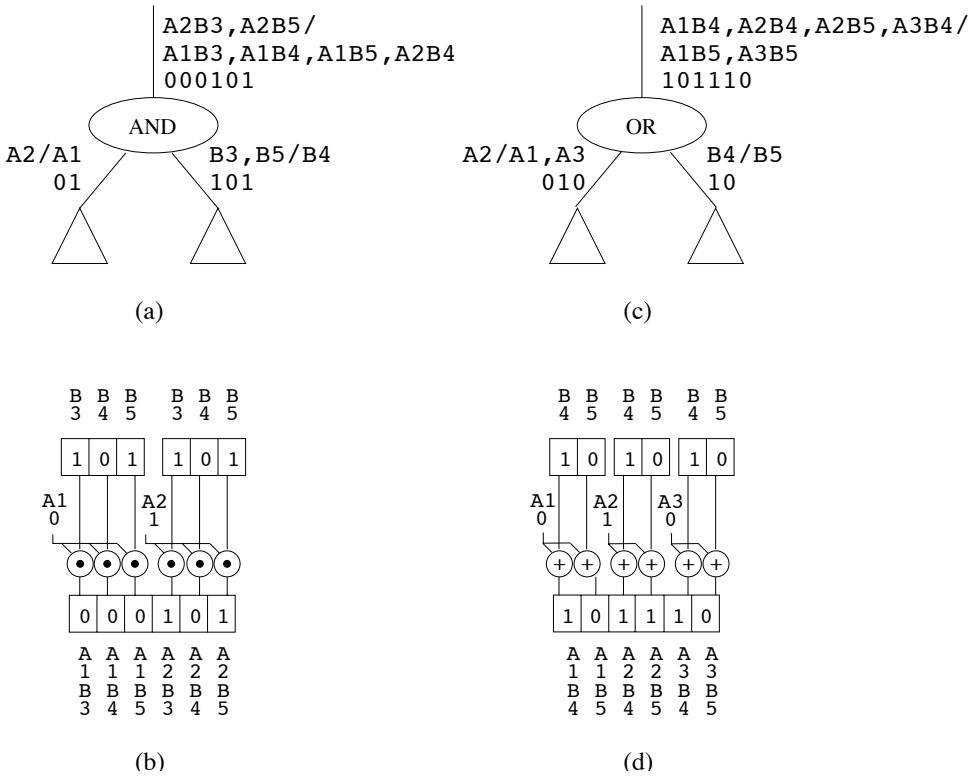


Figure 4.8: Combining bitmaps at internal nodes

bined in this example is not related to the example data from Figure 4.7, only the data structure itself is being considered.

The input data is a bitmap, with each bit corresponding to one of the possible mappings. The ordering convention used in these examples is that simple mappings are numbered from left to right, while composite mappings are ordered with the rightmost mapping varying most rapidly. If the bit is set, that mapping is currently in the accepted set (that is, exists to the left of the / in a mappings list). If the bit is not set, that mapping is currently in the unaccepted set (to the right of the / in a list). So the mapping set `A2/A1` is represented by the bitmap 01, because the left bit corresponds to A1 which is not in the accepted set while the right bit corresponds to A2 which is in the accepted set.

In (a) the bitmap `01` (representing a mapping set `A2/A1`) is combined with the bitmap `101` (representing a mapping set `B3,B5/B4`) at an AND node to produce the bitmap `000101`, which represents the mapping set `A2B3,A2B5/A1B3,A1B4,A1B5,A2B4`. The details of the computation are shown in (b). The Cartesian product of the two bitmaps is produced, with an output bit set only if both the input bits contributing to it are set. This computation does not rely on the individual icon mappings, so it cam be accomplished using nothing more than the two input bitmaps. In (c) the bitmap `010` (representing a mapping set `A2/A1,A3`) is combined with the bitmap `10`, (representing a mapping set `B4/B5` at an OR node to produce the bitmap `101110` which represents the mapping set `A1B4,A2B4,A2B5,A3B4/A1B5,A3B5`. The details of the computation are shown in (d). The Cartesian product of the two bitmaps is produced, with an output bit set if either of the input bits contributing to it are set. Again, this computation requires nothing more than the two input bitmaps.

A "holy grail" research problem for the minimal data structure (illustrated by (c) in Figure 4.7) is defining how to process binding unification in this data structure without explicitly regenerating the full mapping information. The bits for a particular mapping association (such as `B3`) fall in bands within the full bitmap. The position and size of these bands depend on three factors, the cardinality of the mappings to the left (such as `Ax`), the cardinality of the particular mapping itself, and the cardinality of the mappings to the right (such as `Cx`).

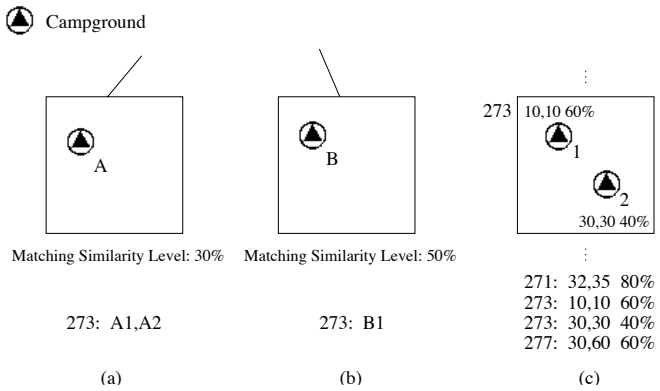


Figure 4.9: Sharing of matching symbols list

4.8 Optimizations

Several small optimizations are possible in this algorithm. If an icon is used more than once in a given query tree, some reuse of already computed data is possible. The large data structures produced by the negation operation can in some cases be represented in a compressed form, thus reducing the amount of memory used during evaluation.

4.8.1 Reuse of Matching Symbols Data

If a particular icon appears more than once in the query tree, the sparse lists (of matching symbol instances) for each icon instance are generated independently, involving redundant references to the index file. One fertile area for optimization is the possible reuse of such lists. The complication in doing so is that the various icon instances may have differing matching similarity levels, and thus generate different sparse lists. However, the list for a high matching similarity level is a strict subset of that for a lower MSL, so a single shared list can be used provided care is taken

to omit processing of symbol instances for which the confidence factor is less than the MSL value for that icon instance. Figure 4.9 shows that both mappings `A1` and `A2` are generated for query picture (a) while only mapping `B1` is generated for query picture (b). This is because the index (c) contains entries for a 60% confidence for a campgrounds at relative coordinates 10,10 but for only a 40% confidence for a campgrounds at relative coordinates 30,30. Mapping `B2` is not generated because the confidence level of 40% for this symbol is less than the 50% confidence required by the MSL specified for icon instance B.

The suggestion has been made that an entire set of mutually bound icons can be processed as if they constitute a single entity. That this is not the case can be shown by considering a variant of this example in which icons `A` and `B` are bound. If two icons with different MSL specifications are mutually bound, a database image symbol could be encountered with a recognition confidence level that satisfies one MSL specification but not the other. In such a case, the possible mappings for the two bound icons will differ. Because they must be treated differently in this case, the data structure used must be capable of distinguishing one icon from the other. Therefore, they may not be treated as a single entity.

4.8.2 Compressing Negated Gap Information

To the extent that not every symbol exists in every database image, icon to symbol matching lists and intermediate results lists from query pictures are sparse. However, intermediate results lists produced by a negation operator are

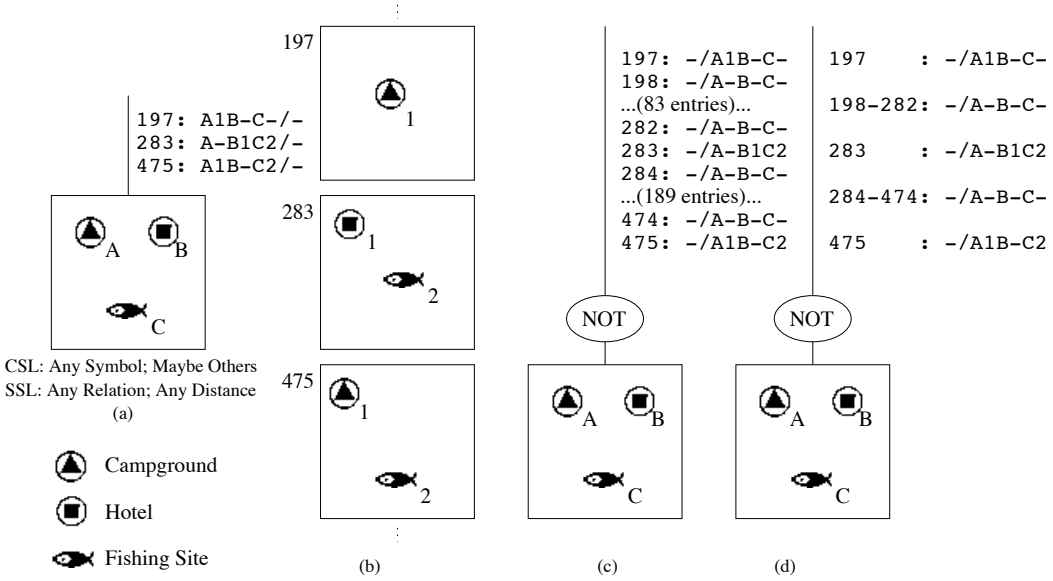


Figure 4.10: Compression of negated gap information

not sparse. An alternative view is that the missing (or *gap*) images represent an implicit (`-/P-Q-R-...`) mapping where P, Q, R ... is the complete list of icon instances in the query subtree described by that intermediate results list, while negation produces (`P-Q-R-.../-`) mappings, which must now have an explicit representation.

Some efficiency can be gained by compressing the negated gap information. In Figure 4.10 query (a) reports only database images 197, 283, and 475. In (b) it is assumed that database images 198-282 and 284-474 do not contain any campgrounds, hotels, or fishing sites. When this query picture is negated (c), all database images from the gap regions appear in the results. Since in this case the mappings are the same (the default mapping) some space can be saved by compressing the partial results for each gap region into one list entry (d). Routines which take these lists as input must be made aware of this optimization.

4.9 Summary

This chapter has presented the evidence supporting the argument that mapping sets, not merely database images, are the natural basis upon which to report matches between query images and database images, and has described an algorithm for process queries and reporting the mapping sets (along with the associated database images) which satisfy them.

Chapter 5

Indexing Concepts

An *index* is a data structure that enables the efficient retrieval of specific items from a collection of data. After a brief introduction to the history of indexing, this chapter covers some of the elementary concepts involved in indexing, followed by some specific observations about indexing for spatial databases.

5.1 History of Indexing

As soon as large collections of data were developed, the problems of effectively accessing and managing them arose. Given the human propensity for invention, it is not surprising that structures for dealing with these problems were soon developed. Patrons of ancient Roman libraries found it difficult to distinguish between particular papyrus scrolls without unrolling them to read their contents. A letter from Cicero to Atticus reveals their eventual solution: to attach to each scroll a small slip of papyrus, upon which was written that scroll's title (and sometimes author). These papyrus slips were called *indices*:

> "...ut [librarioli] sumant membranulam, ex qua *indices* fiant, quos vos Graeci ... $\sigma\iota\lambda\lambda\acute{\upsilon}\beta o\upsilon\varsigma$ appelatis" (so that [the copyists] may take some bits of parchment to make *title slips* from them, which you Greeks call *sillybus*) (Cicero, Atticus, 4.41.1)

The organized study of religion served to inspire a number of collection management problems. In her study of ancient religious texts Weinberg observes

> ...many scientific indexing structures thought to have originated in the computer era were invented as much as a millennium earlier, in the domain of religion.

and describes *concordances* (alphabetical indexes) developed as early as the eighth century, by Christians for the Latin Bible and by the Masoretes for the Hebrew Bible [51].

5.2 Philosophy of Indexing

An index is any data structure which improves the performance of lookup. To index an information collection is to opt for a trade-off of space for time. A secondary, redundant copy of some or all of the information is kept, in order to facilitate access to the collection.

A price must be paid in both volume and effort. Not-inexpensive papyrus must be provided, and scribes must be detailed to write the title slips. If a book is 100 pages, with the addition of an index it may become 110 pages. If a database is 100 gigabytes, with the addition of an index it may become 110 gigabytes. The cost in additional effort is somewhat less obvious: not only is effort required to create an index, but if the information in the collection can change over time, extra effort will be required to update the index whenever the information does change.

What is the payback for this cost? An effective index enables access to the

pertinent parts of a collection, without the need to access the entire collection. The specific means of doing so usually involves *abstraction* or *structure* (or both).

An index based on abstraction alone must still be exhaustively searched, but may yet be advantageous, in cases where the abstracted information is much smaller than the entirety of the information base from which is is drawn. Such an index might be erected on an information attribute that cannot easily be ordered or otherwise structured.

Structuring is the more powerful of the two techniques, as it offers the possibility of reducing not only the fraction of the information collection that need be accessed, but also the fraction of the index itself that need be accessed. Ordering is a particular kind of structuring (specifically, a linear structuring), which can be done whenever the values of an information attribute can consistently be placed into a sorted order. For example, access by account number, date, or name falls into this category.

The familiar "book index" is an example that embodies both abstraction and an ordered structure. Key words are abstracted from the text of the book itself, and are then alphabetically ordered. Were the index not sorted, it would still be usable, but to find any particular item it would be necessary to sequentially read through the entire index (at least until the desired item is encountered).

When an information collection has an innate structure or order, such as with a dictionary or an encyclopedia, it may be possible to access it without an explicit index. This is sometimes referred to as *self-indexing*. This could be considered an implicit form of indexing, based on structure but not abstraction.

When creating an information collection, the ordering of the entries is often unconstrained, but usually a collection can be maintained in only one order. Once the preferred order has been determined, accessing the collection in an alternative order can be accommodated by erecting an additional index, sorted in the desired order, and abstracting only the information necessary to establish the item's proper sequential position within the actual collection order. This scheme is commonly called an *inverted-file index*.

A papyrus scroll is a strictly linear storage device; any given location cannot be accessed without scrolling through (potentially) the entire storage space. Furthermore, a scroll is not divided into pages, making it difficult for any putative index to refer such locations. In contrast, a book is a storage device that can be randomly accessed via a linear ordering of its pages; the linkage between an entry in a book index and the text it refers to is established via those page numbers. Automated information storage devices such as hard disks are also accessed via a linear ordering of data bytes. Filesystems, which reside on such devices, structure linear sequences of such data bytes into higher-level structures, such as database entries, files, and file directories.

The situation becomes more complicated when the attribute values to be indexed cannot usefully be organized as a linear list. Various techniques have been developed to address these situations. For example, when a method exists to determine a "distance" measure between any two attribute values, and when these distances obey the "triangle inequality" $\forall A, B, C : \text{dist}(A, C) \leq \text{dist}(A, B) + \text{dist}(B, C)$, the values are said to embed in a *metric space*, and can be indexed via a structure called

a metric space tree [37]. This technique has been used to index biological strings, such as DNA base sequences or the amino acid sequences of protein, with the distance measure defined as the minimal number of single item deletions and insertions necessary to make one string into the other.

Another approach often taken is to extract from the attribute values an *invariant*, some signature quantity which will be useful for all types of access desired, and around which an efficient index can be structured. To access the collection, the target attribute value is use to derive the corresponding invariant, which is then used to access the designated item in the collection.

One of the most difficult kinds of access to provide is in situations where the target value is in some sense a subset of the indexed attribute value. For example, in an information collection of picture images, with each image containing several objects, and access is desired based on varying subsets of the objects. If the objects can be easily discerned, an inverted-file index can be set up for each object individually; however, selectivity is reduced because the spatial relationships between the various objects of an image are not captured in the index.

A brute-force approach to this problem is to index all possible subsets of objects separately, with each index entry containing the spatial arrangement of its corresponding subset. For example, a collection item with k objects would generate k entries in the single-object index (this would essentially be an inverted-file index), $\binom{k}{2}$ entries in the two-object index, $\binom{k}{3}$ entries in the three-object index, etc. While this approach is sound and supports a very quick access, the amount of space dedicated to index storage grows very quickly with the number of objects in

an arrangement.

5.2.1 Stability

Many applications require access based not on an exact match, but instead on similarity. (or equivalently, based on a range of attribute values). For similarity-based access, a *stability* property is required. This is similar to the mathematical concept of *continuity*; information items whose index attribute values differ only slightly should be stored "close to" each other in the collection, thus minimizing the effort required to access a specific range of values. (The sense of the word stability used here is "stability with respect to small variations in the input data", the property falsified in chaotic systems.) Some "hashing" schemes do not possess this stability property, and are thus inappropriate for similarity-based access.

While any information representable in bits (binary digits) is inherently orderable (and any information representable on a computer is represented in bits), access based on bit pattern ordering may not possess the stability property needed for similarity-based access. For example, indexing stored pictures in this way is probably not useful, because a change to a single pixel will in general throw the picture into an entirely different area of the collection.

Multiple linear dimensions can be folded into a single, ordered dimension using such techniques as diagonalization, row-ordering, or Morton (also called N, Z, or digit-folding) ordering. Generally, in these cases, stability cannot be preserved globally. However, the departure from stability can be statistically quantized [35,

page 15, exercises 1.13 and 1.14].

5.2.2 Selectivity

Another important property of an indexing scheme is the degree of *selectivity* provided. A collection of screw-fasteners could easily be organized into three cardboard boxes marked "small", "medium", and "large". Alternatively, they could be organized as a much larger, multidimensional array of boxes, along one dimension for size ("#2" for smallest through "#10" for largest), another dimension for cap style ("hex cap", "truss head", "pan", et al.), and yet another dimension for application ("sheet metal", "wood", "bolt with nut", etc.). The greater selectivity provided by this multidimensional indexing scheme allows access to focus quickly to the desired items only. Furthermore, accessing strategically-chosen hyperplanes of such a hypercube provides the ability to do access based on subsets of the criteria, for example size-independent access ("find all flat-head screws") or function-independent access ("find all #4 screws").

This example illustrates an important point: to index a collection so that it can be accessed based on subsets of item properties, functionally segregate the properties, then organize the index as a hypercube based on multiple, orthogonal dimensions corresponding to the separated properties. When this is done, access based on an arbitrary subset of object properties can be achieved by accessing the appropriate hyperplane of the index hyperspace. For example, a triangle in two-dimensional space consists of three independent points, each one of which has

two degrees of freedom in its x and y coordinates, for a grand total of six degrees of freedom. Alternatively, these six degrees of freedom can be characterized as a spatial position (two linear dimensions corresponding to a representative point such as a centroid), a spatial orientation (one angular dimension), a size (perimeter, as one linear dimension), and a (size-independent) shape (two angular dimensions, since the third angle is fully determined). If a set of triangles is organized in a three-dimensional index space, with two dimensions of shape and one dimension of size, size-independent access (say, access all 3-4-5 right triangles) corresponds to accessing a line of this index space parallel to the size dimension, while a partially shape-independent access (say, all right isosceles triangles with a perimeter of 10 units) corresponds to a line parallel to the shape index plane.

5.3 Considerations for Spatial Indexing

The index structures described in this dissertation follow the attribute-separation principle. By deriving attribute values for size, shape, and orientation, and structuring the index as a hypervolume, many different modes of search can be supported.

5.3.1 Size

Of all the attributes to be extracted for spatial indexing, *size* would seem the least complicated. For example, size could be derived as the maximal point distance from a center (i.e., the radius of a hypersphere enclosing the points), the maximum of the distance between any two points, or the sum of all $n(n-1)/2$ interpoint

	Points x, y	-2 pos'n	-1 scale	-1 rot'n	remain "shape"
Pair	4	-2	-1	-1	0
Trio	6	-2	-1	-1	2
Quartet	8	-2	-1	-1	4
Quintet	10	-2	-1	-1	6

Figure 5.1: Shape Degrees of Freedom in Two Dimensions

distances. Once a measure of the size has been determined, the arrangement can be scaled to a standard size to facilitate the extraction of values for the other index attributes. Size-independent search can then be supported by projecting away this attribute axis from the index space (i.e., searching all hash buckets along the rows of this attribute).

It should also be mentioned that of all the attributes to be extracted, size is the only attribute value that is theoretically unbounded. Orientation (rotation) is inherently limited to 360°, and shape can be defined using a set of bounded measures, particularly if the arrangement is scaled as described above. While the application domain of the index may impose an independent limit on the maximum size arrangement to be indexed (and thus the maximum value of the size attribute), the size attribute is the only one that is inherently unbounded.

5.3.2 Shape

For point arrangements that correspond to plane polygons, the concept of *shape* is fairly intuitive. We use the term shape in a somewhat more abstract form, as essentially that which remains after size and orientation are extracted.

Figure 5.1 contains an analysis of the degrees of freedom that are involved in

the characterization of the "shape" of arbitrary planar configurations of n points, and demonstrates that it is inherently $2n - 4$ dimensional. This can be seen by observing that each of the n points in the configuration introduces two degrees of freedom, in the form of x and y coordinates. Two degrees of freedom are subsumed by the absolute spatial position of the configuration as a whole (e.g., the x and y coordinates of the center of gravity), one degree of freedom is subsumed in uniform scaling, and one in rotation (e.g., orientation) in the plane. For example, the six initial degrees of freedom of a trio of points (e.g., a triangle) are reduced to two inherent shape dimensions. This makes sense, as the "shape" of a triangle is completely determined by two vertex angles. A similar analysis can be made for spaces of more than two dimensions; however, special care must be taken to account for *all* possible orientational rotations.

The number of dimensions inherent to the shape space determines the number of vectors necessary to serve as a basis for that space. Any method that extracts that number of independent values generates a valid basis; however, for any given database some methods may work more efficiently than others.

5.3.3 Orientation

In the general d-dimensional case, characterization of the spatial *orientation* of an object can be accomplished in three steps, as shown in Figure 5.2. First, a reference direction for the object itself is assigned. This can be visualized as a unit vector (a) that is rigidly attached to the object. If the object possesses

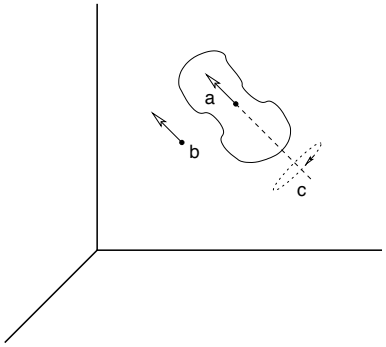


Figure 5.2: Orientation in d-dimensional space

any rotational symmetries an unambiguous assignment may not be straightforward. Second, a direction (b) in the embedding space is identified, in parallel with which the object's reference direction will be aligned. Third, the final remaining degrees of freedom, that of the object rotating (c) around the parallel line, are characterized. This requires additional reference direction assignments, which may be complicated by any rotational symmetries around the axis that the object might possess. And for cases involving more than three dimensions, characterizing this rotation is more complex than one might intuit. In four-dimensional space, for example, there are *two* ways for an object to rotate around a line in the embedding space.

In the general case, $d-1$ parameters are required to define (b), the direction in the embedding space, and $d-2$ additional parameters are required to characterize the rotation (c). Thus $2d-3$ parameters suffice to characterize the spatial orientation of the object. In three dimensions the convention for designating the direction in space (b) is by specifying two angles θ and ϕ. In two dimensions, however, orientation simplifies to rotation within a plane, and no further degree of freedom (c) is allowed (as the object cannot rotate up out of the plane). In this special case, a single

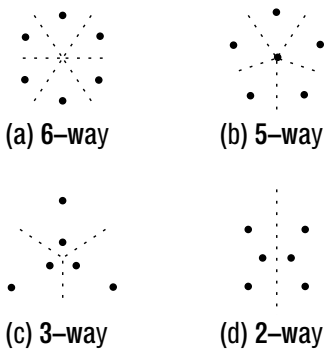


Figure 5.3: Possible Rotational Symmetries of 6 Points

circular dimension is sufficient to characterize the orientation of an object.

5.3.4 Rotational Symmetry

A set of pointlike objects exhibits *rotational symmetry* when a rotation of the set through an angle less than 360° transforms the set into one isomorphic to the original set, for example, a 60° rotation for the set shown in Figure 5.3(a) or a 120° rotation for the set in Figure 5.3(c). In this section we describe the rotational symmetries that could possibly be exhibited by a configuration of n points.

The configurations depicted in Figure 5.3 show all rotational symmetries possible for a configuration of six points. For example, all six points could lie at the vertices of a regular hexagon (a), resulting in a 6-way rotational symmetry. Alternatively, five could lie at the vertices of a regular pentagon, with the sixth at the center (b), resulting in a 5-way rotational symmetry. No configuration of 6 points exhibits a 4-way symmetry.

The configurations exhibiting 6-way and 5-way rotational symmetry are unique (down to rotation and uniform scaling), because the two degrees of freedom exhibited

by the single point per sector are subsumed within the rotation and scaling of the configuration as a whole. In configurations exhibiting 3-way (c) and 2-way (d) symmetry, the additional degrees of freedom made possible by multiple points per sector allow additional variation. In (c), for example, the triangle formed by the inner three points can scale and rotate independently of the triangle formed by the outer three, exhibiting two unconstrained degrees of freedom. In (d) there are four unconstrained degrees of freedom, as each of the outside points is free to move arbitrarily, provided that the diagonally opposite point moves correspondingly.

The rotational symmetries that can be exhibited by a configuration of n points fall into into one of two classes. Members of one class are generated on the basis of the distinct factorizations of n, and do not have points at their centers, as in all cases in Figure 5.3 except (b). Members of the other class are generated on the basis of the distinct factorizations of $n-1$, and contain center points, as in (b).

There are several possible approaches for handling objects possessing a potential k-fold rotational symmetry within the orientation dimension of a point-based index.

1. Each indexed object may result in several index entries, with differing index values. For example, a triangle could be entered in three different places, corresponding to directions from the center to each of the three vertices. The drawback of this technique is that the index must contains three times as many entries.

2. The search algorithm can search more than one area of the index space. For ex-

ample, the directions of rays from the center of the target triangle towards each of its three vertices can be determined, and the index space can be searched for all three directions. The drawback of this technique is that three times the area of the index must be accessed.

3. The symmetry can be broken by point tagging. If the points that form the triangle are tagged (say, one represents a hotel, another a railroad station, and the third a youth hostel), a predefined ordering of the tag values can be used to identify a distinguished member of the vertex set, and orientation can then be defined relative to that vertex. However, when a distinguished member cannot be determined (such as in cases when all points happen to be tagged identically) this technique fails to determine a reference direction.

4. The symmetry can be broken geometrically. Various schemes are possible, including largest-angle, smallest-angle, and for triangles, schemes based upon the Euler line (see Section 7.1). However, when the figure to be indexed actually does possess a k-fold rotational symmetry, this technique fails to determine a reference direction.

5. The orientation dimension itself can be folded by a factor that is a multiple of k, thereby bringing all points at which the object could potentially be mapped into a single position along the folded dimension. For example, the set of all possible triangles includes both figures with a 3-fold rotational symmetry (equilateral triangles) and figures with a 2-fold rotational symmetry (symmetric collinear pseudotriangles, see Figure 7.2(d)). Folding the orienta-

tion dimension six times (that is, reducing the orientation angle modulo 60°) allows a direction to be defined for equilateral triangles (for example, the direction from the center to *any* vertex), with the property that regardless of which vertex is chosen, the triangle will map to the same coordinate in the orientation dimension. The same is true for symmetric collinear pseudotriangles. No matter which of the two possible senses is chosen, the pseudotriangle will map to the same coordinate in the orientation dimension. For example, the directions from the center of an equilateral triangle to its vertices might be 40°, 160°, and 220°, but when reduced modulo 60° all three values become 40°, providing an unambiguous characterization of the triangle's orientation. Similarly, a collinear pseudotriangle with an Euler line at an orientation of 30° or 210° (depending on the assigned sense) generates an unambiguous value of 30° when reduced modulo 60°. The drawback of this scheme is that only $1/k$ of the inherent orientational selectivity is used in the index. Furthermore, this sacrifice is made not only for the ambiguous triangles, but instead for every triangle in the index. This technique is called *normalization* and is discussed in the next section.

6. The orientation dimension can be combined with the other index space dimensions in a structure with the property that objects with ambiguity of reference direction map to regions of the index space where the orientation dimension is suppressed. The drawback of this technique is that *none* of the inherent orientational selectivity of such objects is used in the index. This technique is

n	$\mathrm{lcm}(n, n-1)$	$\frac{360^\circ}{\mathrm{lcm}(n,n-1)}$
2	2	180°
3	6	60°
4	12	30°
5	20	18°
6	30	12°
7	42	$8\frac{4}{7}^\circ$

Figure 5.4: Modulo Reduction Factors

called *elimination* and will be discussed in a later section.

5.3.5 Normalization

Because of rotational symmetry, any putative value for the orientation attribute of such configuration is plausibly multiple-valued. The multiple values can be collapsed to a single value by *normalizing* the value of the orientation attribute, by reducing it modulo 360° divided by a number that is a multiple of all the possible symmetry orders. Because the symmetry orders are submultiples of n and $n-1$ (as discussed in Section 5.3.4), the smallest such number is the *least common multiple* of n and $n-1$, conventionally written as $\mathrm{lcm}(n, n-1)$. For example, normalization of the rotation attribute value for a configuration of 6 points can be done by reduction of the value modulo $360^\circ \div 30 = 12^\circ$.

Unfortunately, as shown in Figure 5.4, the required degree of reduction rises quickly with the number of points. For this reason, the normalization technique is mainly useful for small values of n, such as in a database that is expected to contain large numbers of square ($n = 4$) or equilateral triangle ($n = 3$) configurations.

For the remainder of this dissertation, reduction of the orientation attribute

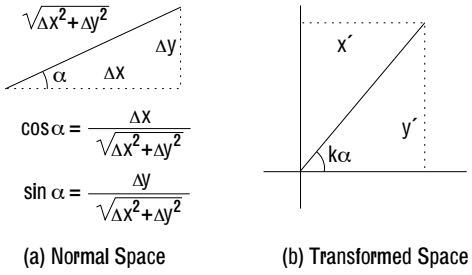


Figure 5.5: Normalizing Transform

value modulo $360° \div k$ will be shown by multiplication by a factor of k, as shown in Figure 5.5. The natural modulo reduction at 360° will then allow diagrams to illustrate the normalization in a more intuitive way, as well as allowing mathematical formulas involving trigonometric functions to be simplified to formulas containing at worst radicals. The numeric values produced by these two methods are always interrelated by a factor of k.

5.3.6 Capturing Orientation

Orientation within a d dimensional space can be captured as a $d-1$ dimensional quantity, so for 2-dimensional (planar) point configurations one can derive a scalar value by using an orientation-independent extraction of a salient set of features, followed by the combination of these features into a vector. The "orientation" component of this vector then supplies the value to be used for indexing. It is not strictly necessary that the vector represent any particular geometric property of the configuration, but only that it be a continuous function of the configuration's "shape" that rotates along with the configuration, and that it spread the indexed configurations as evenly as possible across the orientation axis of the index.

5.3.6.1 Previous Alternatives

In [13], which concentrated on configurations of three points, we relied mainly upon two techniques for capturing orientation. One technique was use of the *Euler line* of the triangle formed by the points. However, for symmetrical configurations this feature is somewhat badly behaved, as it is indeterminate for an equilateral triangle, and approaches infinite length (and therefore undefined polarity) as the three points approach collinearity. Another technique involved designation of the single triangle vertex with the largest or smallest interior angle. However, such a designation is not always stable (in the sense of mathematical continuity). Furthermore, neither of these techniques generalizes well to configurations of more than three points (see Section 8.1).

Two alternative methods for deriving a rotation value are presented below, one using the vectors from a suitably-defined "center" to each point in the configuration, and another using the line segments that interconnect the points themselves. In each case, selected "features" are *normalized* (transformed) to generate two-dimensional vectors, which are then combined via a conventional vector sum. The result is a two-dimensional *eccentricity vector* whose direction component is an absolute characterization for the orientation of the point configuration. Because the "features" used are a continuous function of the shape, and the transformation and vector sum are both continuous functions, the function defined by this overall process is also continuous. However, when the vector sum is zero, the process fails to yield an orientation value.

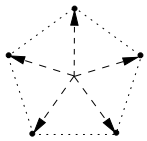


Figure 5.6: Center-to-point Vectors

5.3.6.2 Eccentricity Vector

Discovery of the key role played by rotational symmetry in the derivation of reference direction led to the development of the concept of an *eccentricity vector*, a continuous vector function of an arrangement's shape that measures the deviation of the arrangement from symmetry. Because it is a vector function, it has a direction, and that direction can be used as a characterization of the arrangement's orientation.

5.3.6.3 Center-to-Point Vectors

One method to derive a value for the rotation attribute of a planar point configuration is as the direction component of the vector sum of the set of center-to-point vectors (such as the five dashed lines in Figure 5.6), normalized to accommodate some set of rotational symmetries. Any convenient definition of center, such as the numerical average of x and y coordinates, can be used. If the vectors were not normalized, configurations in the shape of a regular n-gon would yield a vector sum of zero, and a value for the orientation attribute would not be determined. However, normalization of the vectors by n (or any multiple of n) causes the normalized vectors to all be identical, and hence the vector sum to be n times this value.

Figure 5.7 shows the center-to-point vectors of an equilateral triangle, when transformed by reduction modulo 120°. All three vectors transform to the same

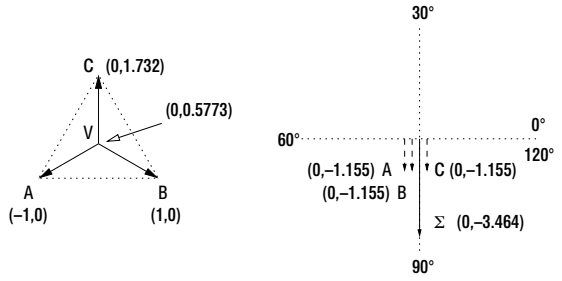


Figure 5.7: Equilateral Triangle

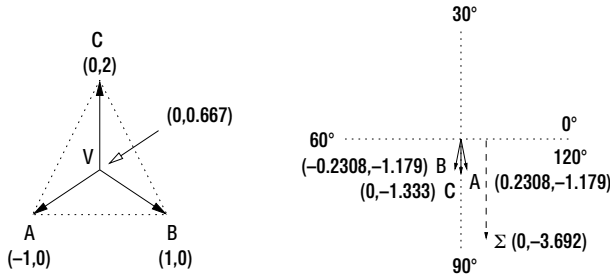


Figure 5.8: Tall Isosceles Triangle

vector (as a convention, dashed lines indicate vectors drawn off-origin for clarity). The vector sum is three times that vector, and therefore nonzero, so a value of 90° for the orientation attribute (within the 120° space) is determined. Note that as the triangle rotates through a full 360° range, the orientation attribute value traverses its full range of 0° to 120° three times.

For configurations that differ slightly from regular n-gons, such as in Figure 5.8, the vectors begin to diverge, but continue to yield a non-zero sum.

From these examples it can be seen that normalization can be used to accommodate a subset of the possible symmetries, if other techniques (such as mapping to the axis of a hypercylindrical index, as described in Section 5.4) are used to accommodate the remaining symmetries.

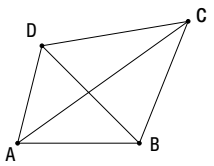


Figure 5.9: Six Interpoint Vectors

5.3.6.4 Interpoint Line Segments

Another method of deriving a value for the rotation attribute of a planar point configuration is as the direction component of the vector sum of the set of all $n(n-1)/2$ interpoint line segments. In Figure 5.9 the interpoint vectors are AB, AC, AD, BC, BD, and CD.

However, unlike the case of center-to-point vectors, the *polarity* of the interpoint line segments is not inherent, that is, it is undefined as to which of the two vertices serves as the head and which one serves as the tail. In fact, each interpoint line segment possesses a $360° \div 2$ rotational symmetry. There are two ways to break this symmetry and allow the interpoint line segments to be treated as vectors.

One way is normalization by $360° \div k$ where k is even. It is always the case that one of the two numbers $n-1$ and n is even. Therefore $\text{lcm}(n, n-1)$ will always be even, and the normalization will suppress the 2-way ambiguity (i.e., symmetry) normally involved in defining the direction of a line segment. Because of this, a value for the orientation attribute can be derived as the direction component of the vector sum of all of the normalized interpoint line segments. Research is currently underway on how to use this derivation of a value for the orientation attribute for the general case of n point configurations.

For the special case of a convex polygon in a two-dimensional space, an alter-

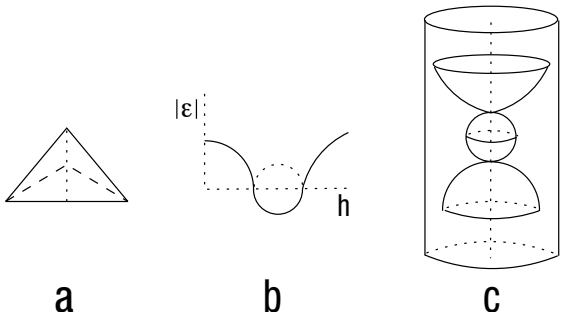


Figure 5.10: Zero eccentricity maps to hypercylinder axis

native way to break the symmetry is to impose a clockwise (or counterclockwise) assignment of direction. However, in spaces of more than two dimensions, clockwise and counterclockwise make no sense, as they are reversed when the configuration is examined from the opposite direction. Furthermore, when some of the points in the configuration are interior to the configuration's convex hull, it is not even possible to unambiguously define a polygon (see Section 8.1.2). For these reasons, this form of symmetry breaking has not been investigated further. However, it is used in one of the examples in the next section, to allow investigation of one case of $360° \div 3$ normalization.

The major drawback of the interpoint method, other than the need to break the polarity-assignment symmetry, is that the number of computations required rises as the square of the number of points in the configuration (as opposed to the center-to-point method, in which the number of computations rises only linearly). However, these are simple computations, so this is not expected to be important in practical terms.

5.4 Elimination

An index space can be organized as a hypercylinder, with the magnitude of the eccentricity vector for each arrangement specifying the distance of that arrangement's index point from the hypercylinder axis. Figure 5.10 shows an arrangement of points (a), the variation in the eccentricity vector as one vertex moves up and down (b), and organization of an index space where the magnitude of the eccentricity vector determines the distance of the index point from the hypercylinder axis. Note that the magnitude of the eccentricity vector does not really become negative (dotted line in (b)), but instead maps to points on the "other side" of the index. Note that arrangements with a zero value for the eccentricity vector map onto the axis, and therefore the orientation of such figures does not play a role in the index. In other words, all such arrangements, no matter what their actual orientation, map to the same point in the index space, resulting in a loss of index selectivity. However, this penalty is paid only for the problematic arrangements, unlike the situation with normalization, where the penalty is paid for every arrangement in the index.

It is possible to use both normalization and elimination in the same index structure. Various examples of doing so, in the case of point triples, are given in Section 7.4.

5.5 Summary of Indexing Concepts

An index is a data structure to make retrieval more efficient, using a process involving abstraction, ordering, or both. In order to support similarity search, an

index must possess the property of stability (continuity). One objective for an efficient index is to maximize selectivity, an even distribution of the indexed objects across the index space.

Arrangements of point objects can be characterized by size, orientation, and "shape" (defined as "that which remains after size and orientation are removed"). For applications that involve only orientation-independent retrieval, an index space can be constructed on the basis of size and shape alone.

For indexes designed to support both orientation-dependent and orientation-independent retrieval, the fact that orientation dimensions are closed and circular requires index spaces involving orientation to typically be hypercylinders.

While ad-hoc data structures can be designed for particular situations, the eccentricity-vector approach is valid across all point arrangements. Unless specifically addressed by the technique of normalization, arrangements that possess a rotational symmetry yield a zero eccentricity vector, and thus map to the axis of a hypercylindrical index space. For particular situations in which a large number of symmetric arrangements are expected, normalization can be used, but at a significant cost to the selectivity of the resulting index.

Chapter 6

Indexing of Pairs of Point Objects

In this chapter we describe a scheme for indexing pairs of point objects, including an optional method of compressing or *pruning* the resulting index structure to reduce its storage requirements.

One approach to the indexing of arrangements of point objects is to decompose the arrangement into a number of object pairs. The search algorithms to be described here use an index created by examining every image in the database, and mapping each pairing of icons into a single point in an abstract space consisting of the Cartesian product of separation (inter-icon distance) and relative orientation. We call this an r-θ space. The absolute positions of the icons are abstracted away, the icon separation of the pair determines the r coordinate, and the relative orientation determines the θ coordinate. All pairs with the same separation and relative orientation (regardless of absolute position) map to the same point in this space. All pairs with the same separation but different relative orientations map to points on a line parallel to the θ axis, while all pairs with different separations but the same relative orientation map to points on a line parallel to the r axis. This r-θ index can be organized using well-known spatial database techniques, such as quadtrees [35] or R-trees [23]. In the general model used for this dissertation, the icon-separation r corresponds to the attribute of size, and because every point pair has the same

shape, the basis for the index space is only size and orientation.

Because an image of n icons contains $n(n-1)/2$ pairs, the size of such an index grows quadratically with the number of icons in an image (although only linearly with the number of images in the database). For databases with only a sparse population of icons this may not be a severe problem. Another approach is to *prune* some of the pairs from the index, at the cost of requiring some additional work while searching the index.

The remainder of this chapter is organized as follows: Section 6.1 shows the structure and construction of an unpruned version of an index. Section 6.2 discusses various kinds of queries and the spatial constraints they imply, and how these constraints determine the part of the area spanned by the index to access. Section 6.3 provides a method of pruning the index to reduce its size. Section 6.4 outlines modifications to the search algorithm required to support such pruning. Section 6.5 gives the results of Monte-Carlo simulation experiments designed to determine the efficacy of index pruning. Section 6.6 describes a variant where separate indexes are kept for each pair of icon types. Section 6.7 discusses considerations for updating an index when the database is modified. Section 6.8 discusses how such an index can be used for non-position-independent search. Section 6.9 briefly shows how such an index can be used for purely existential searches, which do not invoke any spatial constraints. Section 6.10 contains concluding remarks.

6.1 Non-pruned Index Construction

The relative spatial orientation of two icons can be characterized by the slope of the line connecting them, or alternatively it can be characterized by an angle. Because the slope of a vertical line is mathematically infinite, it is more tractable to characterize this orientation as an angle. Therefore, the θ axis of the r-θ space is most conveniently calibrated in some angular measure.

Pairs of icons possess a reflection symmetry: if, with respect to icon A, there exists an icon B at a particular distance and relative orientation, there also exists, with respect to icon B, an icon A at the same distance and the "opposite" orientation. This symmetry can be used to halve the number of entries in the index, if it is possible to determine, when searching for an icon pair of classes A and B, whether to search for an A with a particular orientation towards a B or instead for a B with the "opposite" orientation towards an A. One way to address this (except for the special case of searching for an A with respect to another A) would be to impose a total order on the icon classes. However, in the work described here, the spatial orientation of two icons is limited to the range 0° to 180° by exchanging the icons if their orientation lies outside this range.

The choice between these two methods of halving the size of the index is another trade-off between index size and search time. Roughly speaking, throwing the same number of points into a smaller space results in an increased density of points, so in the pruning algorithm of Section 6.3 this higher density increases the probability that evidence to enable pruning will be found, thereby decreasing the

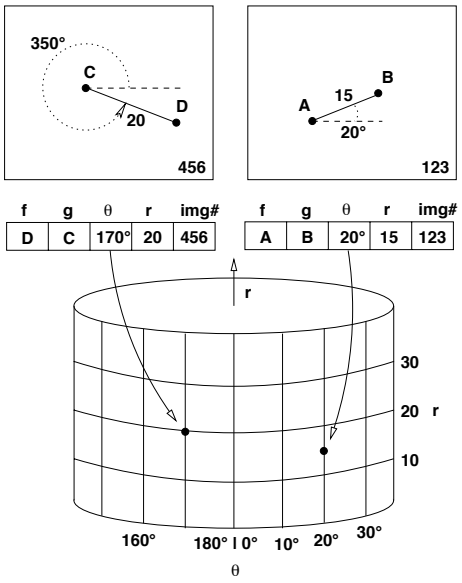


Figure 6.1: Mapping of icon pairs into r-θ space

size of the resulting index. However, when searching with such an index, twice the number of entries must be examined.

The r-θ space that we use is not a traditional Euclidean space because it is a two-dimensional space that is half-open in one dimension (r) but circularly closed in the other (θ) dimension. More precisely, the r dimension starts at zero and theoretically runs to infinity, although in any particular database there will be a maximally separated icon pair, which will then determine the maximum r coordinate value for that database. The θ dimension is topologically closed, that is, 0° and 180° are logically identical. Thus the space is a half-cylinder running from zero to infinity.

Figure 6.1 illustrates the mapping of two icon pairs from database images 123 and 456 into r-θ space. The relative orientation of icons A to B from database image 123 is 20°, which is less than 180°. Since icons A and B are 15 units apart, the pair

maps into the point $r = 15$, $\theta = 20$. In database image 456, icons C and D are 20 units apart, but because the relative orientation of C toward D of 350° is greater than 180° the two icons are exchanged and the pair maps into the point $r = 20$, $\theta = 170$.

Construction of the non-pruned index is not complicated. For each symbolic image in turn, each pairing of icons is considered, and an index entry for that pairing is added. This entry must explicitly contain identifiers for the two icons (perhaps including icon class), and an identifier for the particular symbolic image, which is used to indicate the images found by a successful query. In addition, given the index entry, it must be possible to recover both the separation and relative orientation of the pair, although this data may be implicit in the spatial data structure that is used. One possibility is to explicitly keep separate Δ_x and Δ_y (relative distance in x and y) for the pair, as computation and comparison of separation can be done in the r^2 domain and relative orientation can be recovered by using a four-quadrant arctangent function such as Fortran's ATAN2.

6.2 Non-pruned Index Search

When searching the index, the set of applicable spatial constraints determines an area of the r-θ index that must be examined. The 9 nontrivial boxes in Figure 6.2 represent different combinations of separational and orientational constraints. The boxes labeled A represent distance constraint, while the boxes labeled B represent a combination of both distance and orientational constraints. For the orientational

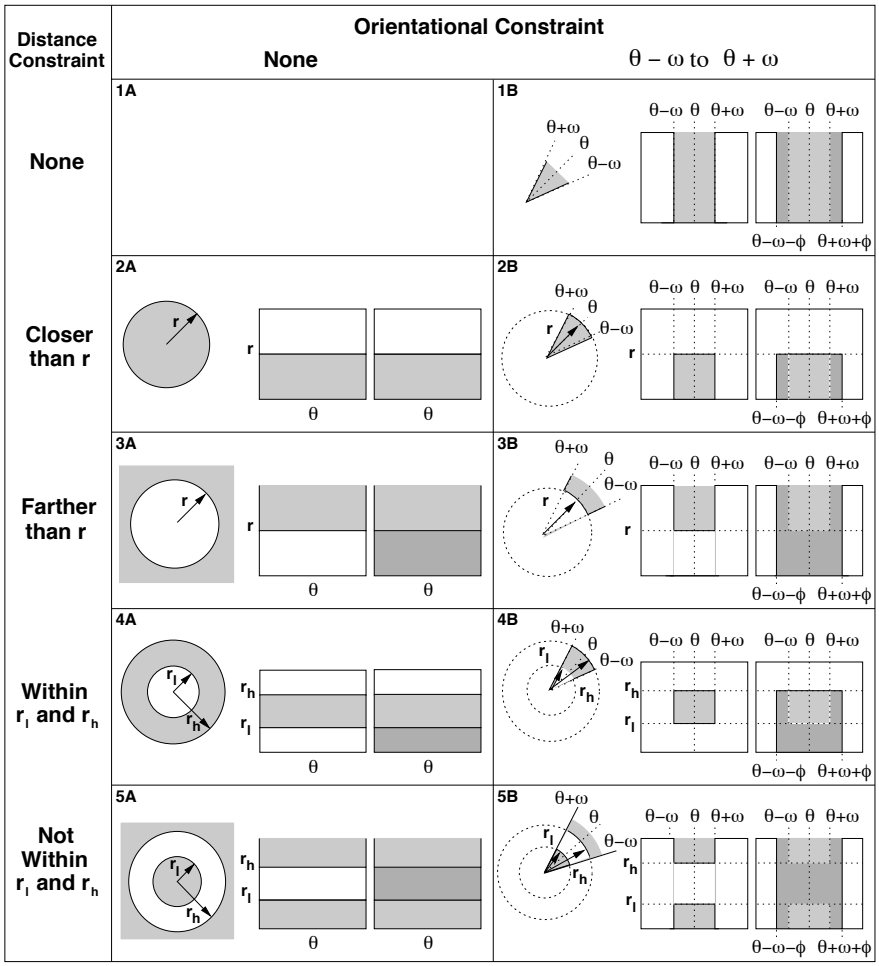
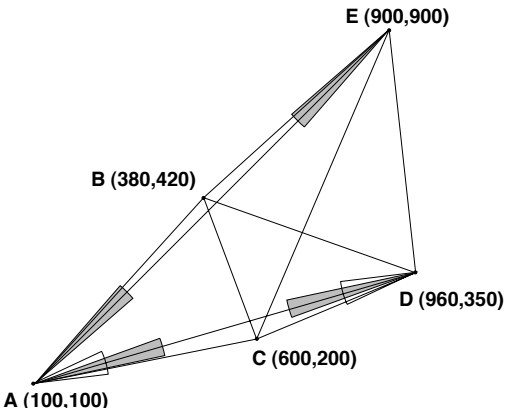


Figure 6.2: Areas of r-θ index space to be searched



Pair	r	Raw θ	Effective θ
AB	425	48.8°	
AC	509	11.3°	
AD	895	16.2°	
AE	1131	45.0°	
BC	311	315.0°	135.0°
BD	584	353.1°	173.1°
BE	707	42.7°	
CD	390	22.6°	
CE	761	66.8°	
DE	553	96.2°	

Figure 6.3: AB and BE constitute evidence for AB

constraints described in this figure, ω represents the range of orientations desired, that is, icon pairs with orientations between $\theta - \omega$ and $\theta + \omega$ are sought. In each box, the left image shows a representation in normal 2-D space of the target of such a query, and the middle image shows the area of r-θ space examined. (The right image shows the search area for a pruned index, described below). For example, in case 5A, the search is for icon pairs either closer together than r_l or farther apart than r_h. In this case, the area of r-θ space examined is the band below r_l and the band above r_h. In case 4B, where separation and orientation are both constrained, the examined area is a rectangle (or window), and various well-known methods can be used for such a window query.

6.3 Pruned Index Construction

The intuition for index pruning is the observation that some entries, if included in the index, can act as proxies or *evidence* for others, allowing the latter to be omitted from the index. However, upon search within such a pruned index, the area

of r-θ space that must be examined is generally larger than the area required for an unpruned index. Postprocessing of the retrieved index data is also required.

Figure 6.3 shows an example database image containing five icons, and the ten corresponding icon pairs along with their r and θ values. The θ values of pairs BC and BD are above 180°, so both pairs will be reversed, yielding effective θ values of 135.0° and 173.1°.

The two gray cones at points A and E are both 8° wide, corresponding to a ϕ value of 4°. In an index pruned to such a ϕ value, pairs AB and BE constitute evidence for pair AE, because their orientations (θ values) of 48.8° and 42.7° differ from the 45° orientation of pair AE by less than this value of the pruning parameter ϕ. Neither pairs AD and DE nor pairs AC and CE constitute evidence for AE. Pairs AC and CD do not constitute evidence for AD. However, for an index pruned to a ϕ value of 10° (open cone), AC and CD do constitute evidence for AD.

In the general case, determining the minimal subset of pairs providing complete evidence for all pairs (and thus comprising the smallest possible index for that ϕ value) may be of high-order time and space complexity, and therefore best addressed by the techniques of dynamic programming.

Alternatively, there are a wide range of heuristic evidence strategies that could be employed. Because the search algorithm itself is used during index pruning, soundness of this class of indexing schemes is guaranteed if the evidence examined at search time is a superset of the evidence considered at the time the index is pruned.

The strategy used to build and search the indexes described in this chapter

is based on two design choices. First, evidence for an entry with a larger r value (representing an icon pair that is farther apart) will be sought only among entries with smaller r values (pairs that are closer together). Second, evidence for an entry with a given θ value will be sought only among entries whose θ values (relative orientation) differ from it by at most a search width parameter ϕ. For an index pruned to larger values of ϕ, evidence is sought among a larger number of entries, (in general) more entries can be pruned, and eventually a smaller index is generated.

The intuition for these design choices is as follows. For any given icon pair that a particular query might accept, that pair may or may not have been pruned from the index. The area of the index space where it would (if not pruned) be present comprises the minimal area that must be accessed. One way to reduce the total area of the index space accessed is to limit the area examined for evidence to as small an extension of this minimal space as possible. Under these design choices, the area accessed is extended down to the $r = 0$ axis and (as we shall see) widened by the pruning parameter ϕ. Other choices are possible. For example, if evidence for pairs with a separation of r is sought only within pairs with separations from r to $r/3$ then the search area need only be extended down to $r/3$. However, in this case less pruning may be possible.

An index entry for a particular icon pair may be pruned if and only if implicit evidence for that pair can be found by the search algorithm. In Figure 6.4 the evidence for an icon pair AB is shown. This evidence consists of a set of explicit pairs in the index comprising a sequence $A-P_1, P_1-P_2, \ldots, P_k-B$ that connects A and B; and that the relative orientation (angle) of each of these pairs is within ϕ of

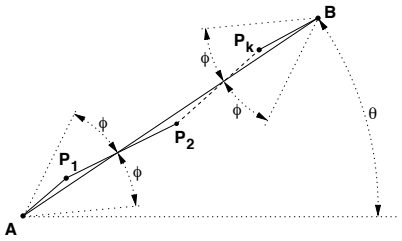


Figure 6.4: Evidence for pair AB. Note all slopes lie between $\theta - \phi$ and $\theta + \phi$.

r	θ	Unpruned Index	Pruned $\phi = 4°$	Pruned $\phi = 10°$
311	135.0°	CB	CB	CB
390	22.6°	CD	CD	CD
425	48.8°	AB	AB	AB
509	11.3°	AC	AC	AC
553	96.2°	DE	DE	DE
584	173.1°	DB	DB	DB
707	42.7°	BE	BE	BE
761	66.8°	CE	CE	CE
895	16.2°	AD	AD	
1131	45.0°	AE		

Figure 6.5: Unpruned and Pruned Indexes

that of the original pair AB (this relative orientation constraint limits the extent in the θ dimension of index space that must be accessed). The Union-Find algorithm, operating on the pairs found in the index, is used to make the association between A and B in slightly more than linear time.

The pruned index is constructed by the following greedy algorithm. Independently for each image in the database, all icon pairings are generated and sorted on ascending r (closer to farther). For each pairing AB considered in this order, the entries already included in the index are filtered according to the ϕ constraint and the Union-Find algorithm is run on the result. If evidence for AB is not found, an entry for AB is added to the growing index. In Figure 6.5, the entries of pruned indices for the icons of Figure 6.3 are shown for pruning factors of $\phi = 4°$ and

r	θ	Indexed	Selected
311	135.0°	CB	
390	22.6°	CD	CD
425	48.8°	AB	AB
509	11.3°	AC	
553	96.2°	DE	
584	173.1°	DB	
707	42.7°	BE	BE
761	66.8°	CE	
895	16.2°	AD	

Figure 6.6: Search of a 4° index for pairs at $34° \pm 12°$

$\phi = 10°$. In either case, when pair AE is considered by the algorithm, evidence for AE (consisting of pairs AB and BE) is found, consequently AE is not added to the index. Similarly, when the $\phi = 10°$ index is built, AD is not added to the index, as AC and CD serve as evidence.

Generating the full set of pairings and sorting them in ascending order of separation requires $O(n^2)$ storage space and $O(n^2 \log n)$ execution time. As each of the $O(n^2)$ pairs is considered, the subset of pairs already in the index and within the orientational constraint must be examined. As there will be between $O(n)$ and $O(n^2)$ of them, construction of the index requires between $O(n^3)$ and $O(n^4)$ time,

6.4 Pruned Index Search

When searching the index, all pairs in an appropriately expanded area of the index are retrieved. In order to ensure that a pruned pairing can always be found, the pairs retrieved by the search algorithm must include all evidence that might have been considered by the original pruning decision. This requires accessing a somewhat larger area of r-θ space, relative to that examined by the same search on

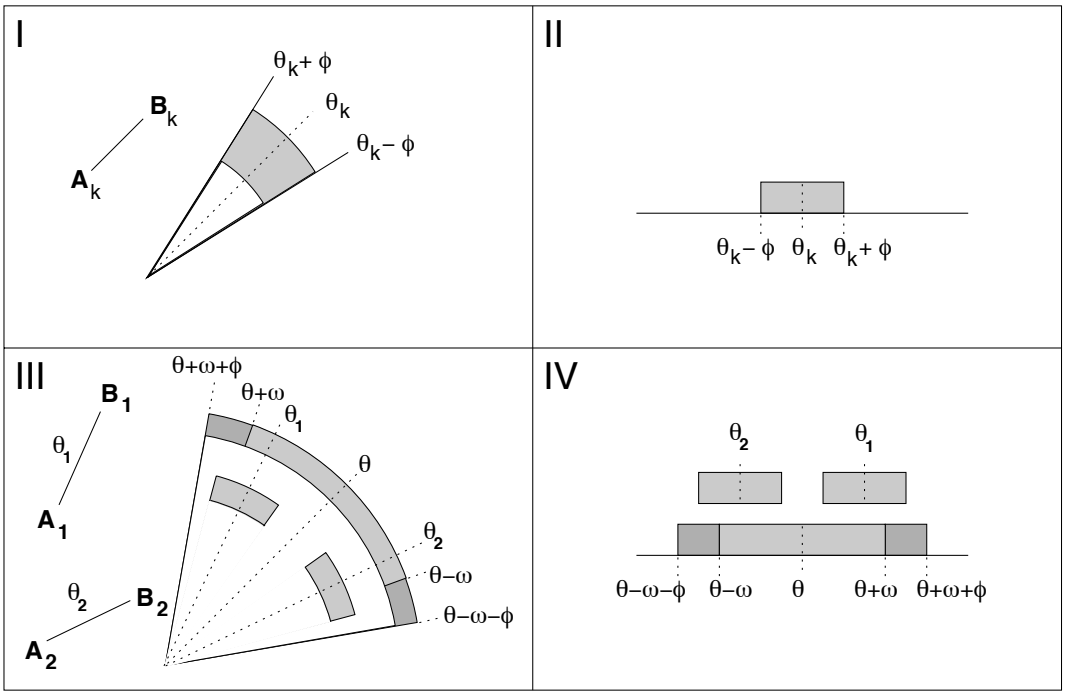


Figure 6.7: Search in θ dimension broadened by ϕ

an unpruned index.

In the index of Figure 6.6, pair AE should be found by a search for pairs with orientations between 22° and 46°($\theta = 34°$ $\omega = 12°$), even though it has been pruned, and therefore no longer appears explicitly in the index. Because the evidence for AE includes AB at 48.8°, the area of the index retrieved must be expanded in the θ dimension to the range 18° ($\theta - \omega - \phi$) to 50° ($\theta + \omega + \phi$). (Note that CB is not selected. Although its 135.0° orientation appears to be a variant of 45.0°, it is in reality directly opposite.)

In Figure 6.7 (I) the evidence considered by the original pruning decision for pair $A_k - B_k$ includes pairs whose relative orientations lie between $\theta_k - \phi$ and $\theta_k + \phi$. This is shown in the θ dimension of r-θ space (II). When searching for a pair with relative orientation within ω of θ (III) the evidence for pairs close to the edge of the

search criterion may protrude as far as ϕ beyond that space, so the area accessed in r-θ space must be widened by the index pruning factor ϕ in order to include this evidence. This is shown in r-θ space in (IV). The additional area that must be accessed is shown in dark gray.

In each of the five boxes 1B through 5B on the right half of Figure 6.2, the rightmost illustration shows widening of the accessed space in the θ dimension by ϕ, the pruning factor of the index being searched.

In addition, because closer pairs act as evidence for farther pairs, the area of r-θ space accessed must be extended down to the $r = 0$ axis. In cases 1B, 2A, 2B, 5A, and 5B in Figure 6.2 this area down to the $r = 0$ axis is already accessed, so the extension of search in the r dimension imposes no additional cost, while in cases 3A, 3B, 4A, and 4B some additional cost is incurred. In all boxes of this figure the additional area that must be accessed due to considerations of index pruning are shown in dark gray.

The evidence retrieved from the implicated area of r-θ space is processed by the Union-Find algorithm. Each icon pair retrieved is interpreted as signifying an equivalence relation between its two constituent icons, thus a set of equivalence classes of icons (or *clusters*) is the final result.

If, at index creation time, an icon pair A-B was pruned from the index, the evidence examined at that time must have contained the chain of pairs $A - P_1, P_1 - P_2, \ldots, P_k - B$, all within the constraints on relative orientation. Because of the extension of the accessed space in both the r and θ axes, the retrieved entries are guaranteed to contain all this evidence, and the chain of pairs will cause the Union-

Find algorithm to place icons A and B together in the same output cluster.

Each cluster generated by the Union-Find algorithm is examined separately for pairs satisfying the search constraints. After filtering the icons in a particular cluster by icon class, each pairing of the remaining icons is examined. In the example of Figure 6.6 clusters (ABE) and (CD) are generated, and the final results AE, BE, and CD are found.

The time complexity for this phase rises quadratically with cluster size, but is somewhat mitigated by two factors. First, the algorithm is actually quadratic on the cluster size; thus processing of a large number of small clusters will require less time than processing a small number of large clusters. Second, each cluster is filtered based on icon class, so the size of the data is greatly reduced before the quadratic phase of the algorithm.

Note that if a distance constraint $r < 400$ had been specified, pairs AB and BE would not have been retrieved, and only pair CD would be present in the result. If a distance constraint $r < 1000$ had been specified, pairs AB and BE would still have been retrieved, but AE would have been filtered out by the final pass, and again, only pair CD would be present in the result.

6.5 Simulation Results

Monte Carlo simulation is a methodology that uses a random number generator to construct a set of test cases, along with a statistical analysis of the results of those tests. A Monte Carlo simulation was undertaken in order to investigate the

		Pruning Parameter ϕ			
n	$\frac{n(n-1)}{2}$	2	4	6	8
10	45	40	38	35	35
20	190	160	142	126	105
30	435	328	277	234	194
40	780	552	428	347	286
50	1225	776	588	464	385
60	1770	1036	749	573	477
70	2415	1299	907	688	578
80	3160	1567	1068	809	666
90	4005	1844	1244	937	758

Figure 6.8: Sizes of Unpruned and Pruned Indexes

effectiveness of pruning in reducing index size, and the degree to which clustering reduces the execution time of the search algorithm.

If the full circle is divided into n units (such as 360°), then modulo-n arithmetic can be used for computation on this axis. Furthermore, if n is chosen to be a power of two, a bitwise AND can be used to implement the modulo division operation. In the simulations that we ran, the circle was divided into 256 (2^8) units.

The version of the Union Find algorithm implemented for this simulation was extended to preserve the offsets Δ_x and Δ_y between each icon and the *representative* chosen by the algorithm for each equivalence class (cluster) it produces. The offsets between two icons in the same equivalence class can then be determined from their offsets with respect to their shared cluster representative. The relative orientation of the two icons can then be determined from these offsets by table lookup, while separation comparisons can be done in the r^2 domain, by comparing $\Delta_x^2 + \Delta_y^2$ to the square of the separation constraint.

Test images were generated, consisting of 10 to 90 icons apiece, randomly located on a 1024 by 1024 grid. For each test image, indexes were generated with

values of the pruning parameter ϕ of 2, 4, 6, and 8. (Angular unit values from 0 to 255 represent angles from 0° to 360°. Thus, one angular unit represents slightly more than 1.4 degrees. Values of the pruning factor investigated correspond to beam half-widths of approximately 2.8°, 5.6°, 85.°, and 11.2°, respectively.) Figure 6.8 shows the (computed) size of an unpruned index, compared to the sizes of the resulting pruned indexes. The same information is shown graphically in Figure 6.9 using a logarithmic scale.

As the pruning factor ϕ is increased, the size of the index is reduced. The trade-off for this size reduction is that the area of the index space that must be examined is broadened in the θ dimension.

Simulations of search were performed for each index produced. A search was run for every icon pair in the image (no simulations of unsuccessful searches were conducted), and the structure of the clusters produced by the Union-Find algorithm was examined.

The simulation showed that for small values of r, small values of ϕ, and sparse data loading, a larger number of smaller clusters were formed. However, as the values of r and ϕ increased, or as the data became more dense, the Union-Find algorithm trended toward production of a singular large cluster.

In a real world implementation, the index would be disk resident, and thus query costs would be dominated by disk access. For this reason the size of the index and the size of the r-θ space search area were used as a proxy for estimating query costs.

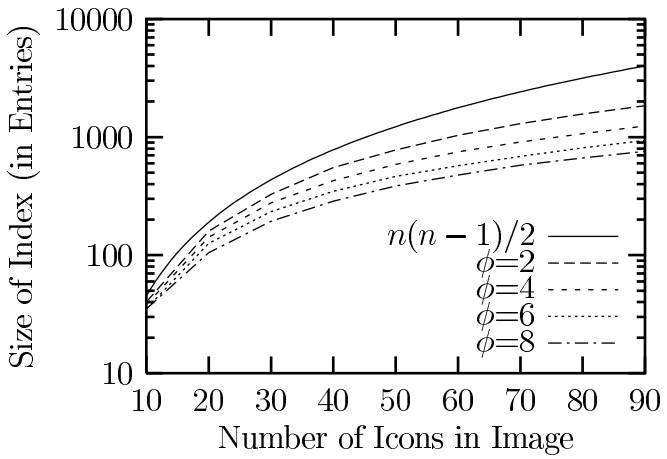


Figure 6.9: Pruning Efficiency

6.6 Separate Class Pair Indices and Wildcard Searching

While so far the discussion has been limited to the use of a single, unified index, databases storing only a small and limited number k of icon classes might advantageously maintain a separate r-θ index for each of the $k(k+1)/2$ combinations of classes. (This number includes the k homogeneous class pairs A-A, B-B, etc. To achieve the factor of two reduction in the number of indexes, a total order would be induced on the icon classes, as discussed in Section 6.1.) Either unpruned or pruned indexes might be used, depending on the expected density of icons in that particular database. However, introduction of a single instance of a novel icon class would require immediate instantiation of a relatively large number of new indexes.

In such a multiple-index system, a *wildcard* query (searching for icons belonging to any one of a specific set of icon classes) will generally require more than one of the indexes to be accessed.

6.7 Database Update

Adding a completely new database image is straightforward. The pairs for the new image are generated, pruned by the value of ϕ used by that database, and added to the index. Similarly, deletion of an entire image consists simply of removing all the pairs of that image from the index.

Adding an icon to an existing database image can be done easily by generating all the pairs composed of the new icon and the existing icons, then adding into the index only those new pairs for which evidence does not already exist in the index. This is somewhat suboptimal in that one or more of these new pairs may constitute crucial evidence for some of the other pre-existing pairs, allowing them to theoretically be pruned from the index. However, these benefits can be regained by periodically regenerating the index.

Removing an icon cannot be done by simply removing all pairs containing it, as a removed pair may be a crucial part of the evidence for other pairs. Instead, marking the pairs as "logically deleted" without physically deleting them allows their continued participation in the search algorithm but also allows for the final filtering pass to remove them from the results. Again, periodically regenerating the index allows for the eventual physical deletion of such pairs.

6.8 Position-dependent Queries

Although the algorithms described above were developed to perform position-independent search, position-dependent queries can be accommodated by adding

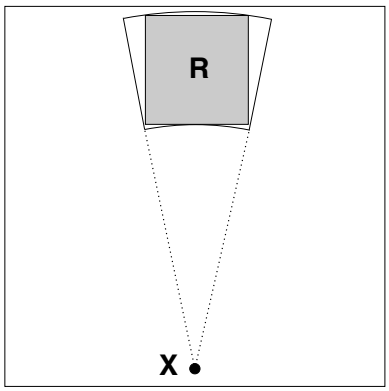


Figure 6.10: Absolute-position search relative to X

to the index one artificial icon for each database image, at a point determined by a convention of that particular database (such as "the center" or "the lower-left corner"). Query for an icon A in a given area of any image is then performed by determining a set of r-θ constraints, relative to the known conventional position of the artificial icon X, that subsumes the given search area. The search is performed for an X-A pair with the determined spatial relationship, and the result is filtered to satisfy the original position-dependent query.

In Figure 6.10, a rectangular search window in normal space can be subsumed into an r-θ query (of type 4B in Figure 6.2) with respect to artificial icon X. Images found by the algorithms described above can then be filtered to remove any "false positives" generated by this subsumption.

6.9 Non-spatial Queries

An existential query attempts to find database images containing a particular icon A, with no constraint on that icon's position. Such a query could also be satisfied using the artificial X point, by searching for an X-A pair with an unconstrained

spatial relationship, but doing so would require that the entire index be retrieved. If a database must frequently support such queries, a separate but parallel inverted-file index might be added.

6.10 Concluding Remarks

We have shown how to use a spatially organized index of icon pairs to accelerate search of an image database for icon pairs possessing a desired spatial relationship. The quadratic growth in the storage space required for such an index can be controlled by using an algorithm that prunes pairs from the index when their existence can be inferred from remaining unpruned pairs. The trade-off for this pruning is that search must access a larger fraction of the spatially organized index, and may require execution time at worst quadratic in the number of icons per image.

In the two-dimensional work described above, the spatial relationship between two icons is characterized as separation (r) plus a single angle (θ). The straightforward extension to three dimensions adds a second angle, characterizing the spatial relationship as a separation plus two angles. The closest familiar analogue might be the "Az-El" (azimuth-elevation) measurement of surveying, in which a direction is characterized as an azimuth (angular bearing in the horizontal plane) and an angular elevation above the horizon, with the range (e.g., distance or separation) supplying the required third coordinate value.

In this case, the shape of the index space would be analogous to a $2\frac{1}{2}$ D map of the earth's surface, or the skin of an orange, with two angular axes, both closed

(i.e., a sphere), and with the r dimension extending outward, along the orange skin's thickness. The general pattern for an n dimensional database space would be an index space consisting of a single linear r axis directed "outward" from the surface of a $n-1$ dimensional hypersphere, which would embody the $n-1$ remaining angular dimensions.

It is easier to envision the structure of such an index space than to imagine any real-world application for this technology. Human beings are surprisingly adept at correlating two dimensional maps with physical terrain, given that we did not evolve in an environment where looking down on a terrain from above is an everyday experience. Sadly, this ability does not extend to higher dimensionalities. The only three-dimensional analogy that comes to mind is using a database of positions in the solar system (such as space ships, space stations and planets) to find feasible links for an interplanetary relay network, while avoiding radio noise from the galactic core. Higher-order analogs would be even more opaque.

At first glance there are some similarities between the methods described here and the two-dimensional version of the Θ-graph class of spanner networks discovered independently by Clarkson and Keil [18]. However, in the Θ-graph algorithm the cones (in two dimensions, sectors) emanating from any given point are generated only once, and with orientation independent of the other points in the set. In our method a different cone is used for each pairing of points, and the orientation of that cone is directed towards that other point.

Future research includes extending the analysis from synthetic to real-world datasets, the investigation of algorithms to construct optimal pruned indexes, and

algorithms to construct useful pruned indexes in less than $O(n^3)$ time. One area for investigation would be use of the *well-separated pair decomposition*, described in [6] and [37, pages 582-583], to construct a bottom-up index generation algorithm.

The space-time tradeoff involved in the employment of a 180° space versus a 360° space is also worthy of investigation.

Chapter 7

Indexing Arrangements of Three Points

This chapter describes a scheme for indexing triples of point objects, using a point-based index based on the geometric properties of the associated triangle. There is no variation to the "shape" of a pair of points; every pair has essentially the same shape. Thus the new concepts introduced in this chapter include the characterization of shape, and the definition of "orientation" for point triples.

Any three distinct points uniquely determine a triangle. For the special case of three collinear points, a *collinear pseudotriangle* with overlapping sides (and with "angles" of 0°, 180°, and 0°) is determined. As such figures are not included in the classical definition of a triangle, this special case must be carefully considered when invoking any of the classical triangle properties. Because of this one-to-one correspondence between point triples and triangles, point triples may be indexed using methods that effectively index these associated triangles.

Indexing of the spatial position and size of a triangle is straightforward and will not be discussed in the remainder of this chapter, except to note that spatial position must be referenced to some specific point in the triangle, and that multiple candidates exist for the "center" of a triangle. This will be discussed further in Section 7.1.

The remainder of this chapter is organized as follows: Section 7.1 briefly out-

lines some elementary and classical triangle properties. Section 7.2 describes an inherently two-dimensional abstract space of triangle "shapes", and examines methods for defining a basis for this space, including methods based on interior angles and methods based on normalized side lengths. Section 7.3 discusses the definition of the spatial orientation of a triangle, including the special problems posed by triangles possessing k-fold rotational symmetry. Section 7.4 presents several alternative schemes for construction of a point-based index, showing how the problems introduced by k-fold rotational symmetry can be addressed by innovative methods of integrating the shape and orientation dimensions. Section 7.5 contains concluding remarks.

7.1 Triangle Properties

Various standard terms are used to describe the shapes of triangles. When no two triangle sides have the same length (and thus by the Law of Sines no two angles are equal) a triangle is termed *scalene*. A triangle with three equal sides (and thus three equal 60° angles) is called *equilateral*. In the remaining case two sides of the triangle are equal; such triangles are called *isosceles*.

In this chapter it will be useful to distinguish between two subclasses of isosceles triangles: *tall* isosceles, in which the third side is shorter than the other two, and *wide* isosceles, possessing a longer third side.

Classical triangle literature describes a large number of special points, some of which seem likely candidates for the "center" of a triangle. Figure 7.1 shows three

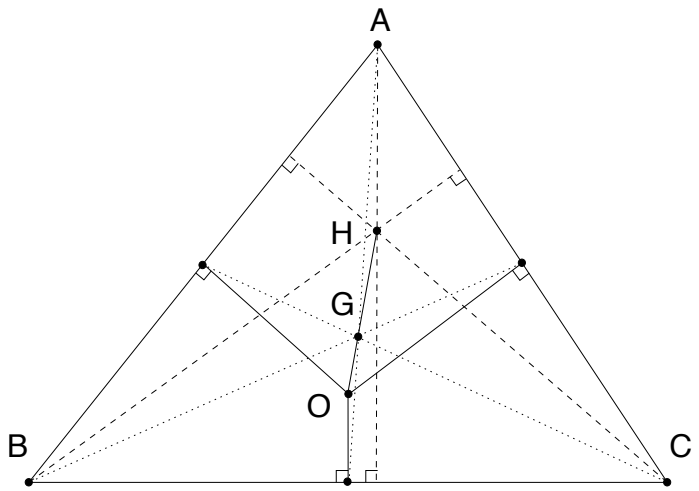


Figure 7.1: Euler line of a triangle

such special points. The *centroid* (center of gravity, barycenter), by convention labeled G, lies at the intersection of the three *medians* (lines from each vertex to the midpoint of the opposite side), which are shown as dotted lines. The *orthocenter*, labeled H, lies at the intersection of the *altitudes* (lines from each vertex perpendicular to the opposite side), which are shown as dashed lines. The *circumcenter*, labeled O, lies at the intersection of the three side perpendicular bisectors, which are shown as solid lines. These three points are always collinear, and define the *Euler line* of the triangle [9, page 17].

Figure 7.2 shows the Euler line of both tall (a) and wide (b) isosceles triangles. Note that the spatial order of the special points is reversed. While the Euler line will prove useful for defining the spatial orientation of a triangle, two anomalous cases must be considered. In an equilateral triangle (c) the three points coincide, so the classical Euler line is indeterminate. And in the degenerate pseudotriangle defined by three collinear points (d), both H and O lie at the point at infinity, so the polarity of the Euler line analogue becomes indeterminate.

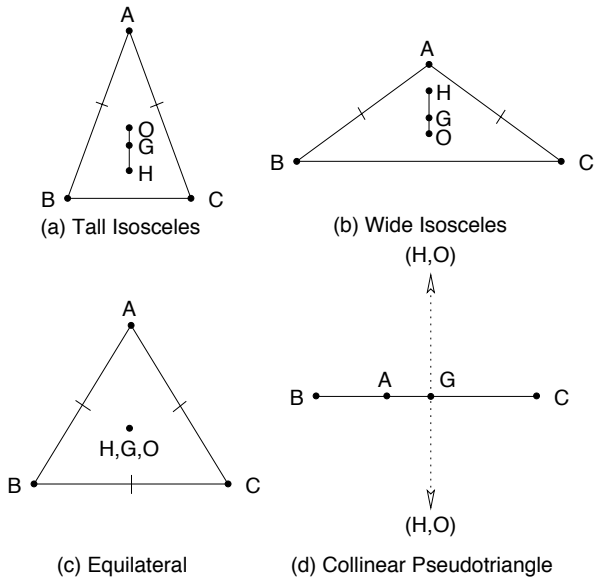


Figure 7.2: Euler lines of other triangles

Because it is always spatially enclosed within the triangle, the centroid seems the most logical candidate for being designated as the "center" of a triangle. This definition of "center" will be used in the remainder of this chapter.

7.2 Indexing on Abstract Shape

As described in Section 5.3.2 (see Figure 5.1) the shape space of a point triple is inherently two-dimensional. The following sections describe schemes based on both vertex angles and on (normalized) side lengths.

7.2.1 Shape Indexing Based on Angles

The three internal vertex angles of a (planar) triangle completely determine its shape: all triangles with the same three angles are *similar* (mathematically, all corresponding sides are in the same proportion). However, because the angles always

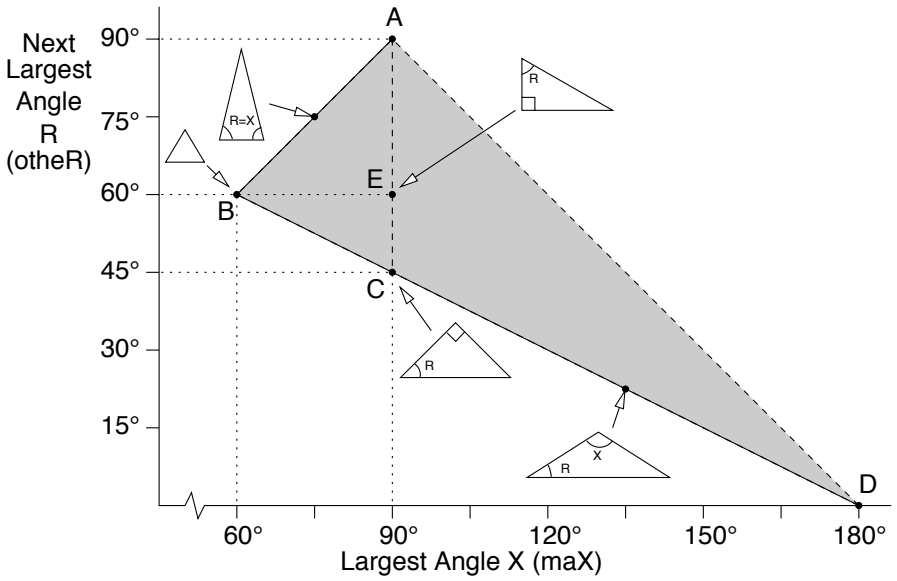


Figure 7.3: Shape space based on angles

sum to 180°, specifying any two of the angles completely determines the third. For this reason, the space in which each distinct triangle shape maps to a distinct point is inherently two-dimensional. A basis for such a space can be defined in many ways.

Figure 7.3 shows one possible basis, that of the largest angle X (maX) plotted against the next-largest angle R (otheR). The smallest angle will later be referred to as N (miN). The shaded area indicates where feasible triangles lie.

The largest angle in a triangle cannot be less than 60° (if it were, the other two angles would sum to more than 120°, one of the other two angles would then necessarily be greater than 60°, and the first angle would not then be the largest). When the largest angle is 80°, the other two angles must total 100°, but by a similar argument the next-largest angle cannot be less than 50° (if it were, the remaining angle would be greater than 50°). In general, the size of the second-largest angle is bounded from above by either the largest angle itself (line A-B) or the remainder

left by the largest angle from 180° (line A-D), and is bounded from below by half this remainder (line B-D).

All equilateral triangles map to point B, and all right triangles map to points along line A-C, with the 45°-45°-90° triangle at point C and the 30°-60°-90° triangle at point E. Wide isosceles triangles map to points along line B-D, while tall isosceles triangles map to points along line A-B. Points actually on line A-D represent infeasible triangles (two angles summing to 180° leaving no remainder for the third angle). All pseudo-triangles formed by three collinear points map to point D.

Mapping of triangles into this index space depends only on vertex angles, so it is position-independent, orientation-independent, and scale-independent. It is also invariant to a mirror reflection, as the largest and second-largest angles are selected for classification regardless of their clockwise or counterclockwise order in the triangle.

As this mapping is continuous (in the mathematical sense) the *shape stability* can only fail for the "corner cases" where a slight variation in triangle shape (for example, the situation in Figure 7.7) could cause *shape ambiguity*, a sudden shift in which angle is considered "largest". For a scalene triangle, where the three angles materially differ, designation of angles as largest, smallest, and "other" is unambiguous. In the case of near-equilateral triangles, where the three angles are almost equal, such triangles will map to the general vicinity of point B, thus expanding the search to an appropriate neighborhood of point B will ensure that all such triangles are found.

In Figure 7.3, for near-wide-isosceles triangles (when two angles are almost

equal and the third angle is larger), choice of the largest angle will be unambiguous. Slight variations in the relative sizes of the other two angles will result in a set of index space points that approaches line B-D and then "bounces back" from it. As an example, if the largest angle is 90°, the other two angles will vary in the vicinity of 45°. The trajectory of points in the index space will be the small segment of line E-C in the vicinity of point C. All such triangles will map to the vicinity of point C, and expanding the search to an appropriate neighborhood of point C will ensure that all such triangles are found.

In near-tall-isosceles triangles (two angles are almost equal and the third is smaller), one of the two large angles will be chosen as largest, and the other will be chosen as second largest. Slight variations in the relative sizes of these two angles will result in a set of index space points that approaches line A-B and then "bounces back" from it. Expanding the search to an appropriate neighborhood of the contact point with line A-B will ensure that all such triangles are found.

In both the tall and wide isosceles triangles, the invariance of the index to mirror reflection ensures stability in the mapping from triangles to the shape space. However, because the selection of which angles to index is based solely on largest vs. next-largest angle (regardless of their clockwise or counterclockwise order), no distinction is made between the left-handed and right-handed versions of a given triangle. Thus only half the inherent *shape selectivity* is used in the index. Other examples of the sacrifice of selectivity in order to secure stability will be presented later in this chapter.

Because the space of triangle shapes is inherently two-dimensional, any pair

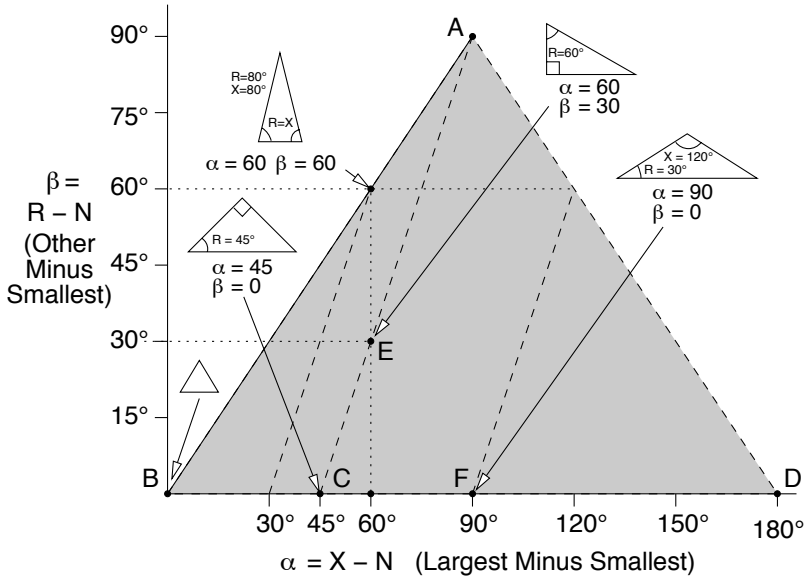


Figure 7.4: Shape space based on angle differences

of linearly independent axes that spans the space may be used as a basis. One such alternative basis, constructed using differences between pairs of angles, is shown in Figure 7.4. Coordinate α is the difference between the largest and smallest angles (maX - miN), and coordinate β is the difference between the other and smallest angles (otherR - miN). The original angles can be recovered from α and β as:

$$\begin{aligned} miN &= \frac{180° - \alpha - \beta}{3} \\ otherR &= \beta + miN \\ maX &= \alpha + miN \end{aligned} \tag{7.1}$$

Using this basis, triangles with equal largest angles map to points along the lines shown as dashed. For example, all right triangles map to points along line A-C. Note that wide isosceles triangles map to points along line B-D,

The advantage to using this shape space is that when the reference orientation is defined as the direction from the center to the vertex with the smallest angle,

all triangles with ambiguous reference orientations (equilateral, wide isosceles, and collinear) lie along line B-D. As will be seen in Section 7.4, this enables one strategy for integrating the shape space and orientation dimension into the combined index space.

7.2.2 Shape Indexing Based on Normalized Side Lengths

While angle-based methods for characterizing triangle shape are inherently size-independent, side-length-based methods can be made size-independent by basing them on appropriately normalized side lengths. For the remainder of this section it is assumed that triangle side lengths are scaled such that the total triangle perimeter length is 180 units. To compute values based on a perimeter scaled to 1.0, the numbers given should be divided by 180.

When the total perimeter is known, the triangle is completely determined by two side lengths (as the length of the third side can be directly computed). The space shapes generated by a side-based approach are similar to those generated by an angle-based approach.

Figure 7.5 shows a shape space based on normalized side lengths. The largest side of a triangle cannot be more than half the perimeter, nor can it be less than one third the perimeter. The next largest side is bounded from above by the largest side (line A-B) and from below by half the remaining perimeter (line B-D). In this space, tall isosceles triangles map to points along line A-B, wide isosceles triangles map to points along line B-D, and right triangles map to points along the dashed

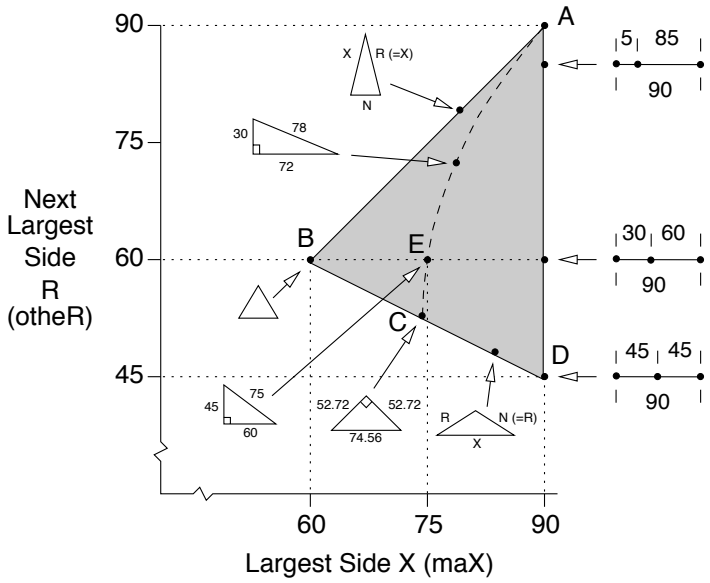


Figure 7.5: Shape space based on normalized side lengths

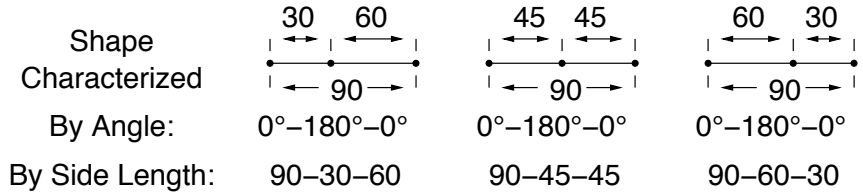


Figure 7.6: Characterization of shape for collinear point triples (triangle perimeters normalized to 180).

curve C-E-A.

There are two small advantages to using a side-length-based method over using an angle-based method. When point positions are given as x and y coordinates, side length computation is somewhat less expensive than angle computation, since evaluation of the computationally expensive arctangent function is not required.

The second advantage of side-length-based methods is that pseudotriangles corresponding to collinear point triples map to points spread out along a line in the shape space, rather than all mapping to a single point. Figure 7.6 shows characterizations of three instances of point triples. An angle-based approach does not

distinguish between these cases, while an approach based on normalized side length is capable of discriminating between them. In Figure 7.5 collinear pseudotriangles map to points spread out along line A-D, while in Figure 7.3 they all map to point D. Avoiding the sacrifice of the shape information of these pseudotriangles results in a slight selectivity advantage for side-length-based methods.

As in the angle-based method, because the selection of which sides to index is based solely on largest and second-largest sides (regardless of their clockwise or counterclockwise order), only half of the shape selectivity inherent in the indexed triangles will be used in the shape index.

7.3 Indexing on Spatial Orientation

Based on the general model of reference direction (see Section 5.3.3), and noting that in a two-dimensional space a triangle cannot rotate up out of the plane, a single circular dimension is sufficient to characterize the orientation of a triangle. The following sections discuss various ways to define the reference direction for a triangle.

7.3.1 Reference Direction Based on Distinguished Vertex

The direction from the triangle center toward a *distinguished member* of the vertex set (such as the vertex with the smallest or largest interior angle) could be a candidate for a geometrically-based definition for a triangle's reference direction. However, instability in the designation of this vertex results in the loss of *orienta-*

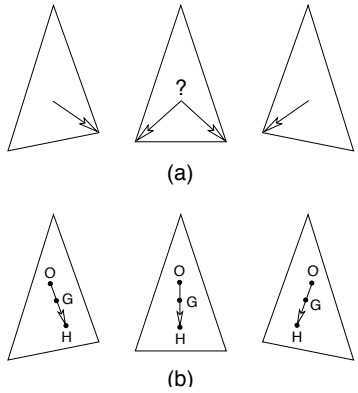


Figure 7.7: Stability of reference direction

tional stability in the index. Figure 7.7(a) shows the loss of orientational stability for triangles in the near-tall-isosceles region when the vertex with the largest angle is used as the distinguished member (In the case of designating the vertex with the smallest angle, an analogous situation arises with triangles in the near-wide-isosceles region.) As the triangle varies from near-isosceles-left to near-isosceles-right, the reference orientation snaps discontinuously from one vertex to another, and when the triangle is truly isosceles it is indeterminate.

7.3.2 Reference Direction Based on Euler Line

An alternative for defining the orientation of a triangle is to use the triangle's Euler line. As can be seen from Figure 7.7(b), the Euler line provides a definition of orientation that is stable for triangles in the near-tall-isosceles region.

As noted in Section 7.1, in the case of the collinear pseudotriangle the sense of the Euler line is indeterminate. While the definition of reference direction as the direction from the center to the vertex with the smallest angle suffers from instability in the near-wide-isosceles region, it does allow stable assignment of reference

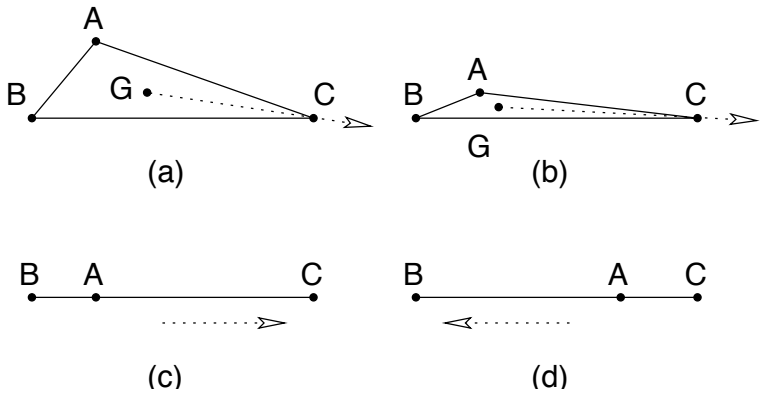


Figure 7.8: Reference direction for collinear pseudotriangles

direction for asymmetric collinear pseudotriangles. Figure 7.8 shows the behavior of a triangle in the limit (a,b) as it approaches becoming an asymmetric collinear pseudotriangle (c). When the interior point is on the other side (d) the direction is reversed. However, when the interior point is exactly equidistant from the other two (the *symmetric* collinear pseudotriangle), the smallest angle method does not assign a stable reference direction.

7.3.3 Eccentricity Vector: Center To Point

Consider the following construction, which is illustrated in Figure 7.9. For an arbitrary triangle, without loss of generality, let the midpoint of one of the triangle edges be chosen as the origin, let the x axis be aligned with that edge, let the axes be scaled such that the vertices of that edge lie at $(-1, 0)$ and $(1, 0)$, and let x and y be the coordinates of the remaining vertex. (This "without loss of generality" argument is only valid because we are not concerned with the triangle's size or orientation, but only with its "shape".) Briefly, every triangle is "similar" to a triangle with one vertex at $(-1, 0)$ and another vertex at $(1, 0)$.

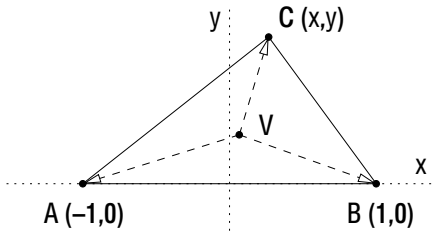


Figure 7.9: Triangle Shape

This construction clearly demonstrates that the shape space of triangles is inherently two-dimensional, by showing an explicit one-to-one correspondence between any triangle shape and the coordinates of the remaining triangle vertex. Furthermore, this analysis extends naturally to the general case of n points, which further illustrates that the configuration space for a set of n points has an innate dimensionality of $2n - 4$, as described in Section 5.3.2.

For the construction of Figure 7.9 the triangle center V lies at $(\frac{x}{3}, \frac{y}{3})$ which are the x and y averages of the point coordinates. The three center-to-point vectors are then

$$\begin{aligned} VA &= (-\frac{x+3}{3}, -\frac{y}{3}) \\ VB &= (-\frac{x-3}{3}, -\frac{y}{3}) \\ VC &= (\frac{2x}{3}, \frac{2y}{3}) \end{aligned} \tag{7.2}$$

For many configurations, including but not limited to those with 2-way or 3-way rotational symmetry, the sum of these raw center-to-point vectors is zero, and thus would not be useful for deriving a value for the orientation attribute; therefore, the vectors will generally be normalized before summing to form the value for the orientational attribute that will be used to enter the point triple into the index.

7.3.4 Eccentricity Vector: Interpoint

For a point triple, the $n(n-1)/2$ interpoint line segments are exactly the edges of the triangle determined by the points:

$$\begin{aligned} AB &= (2,0) \\ BC &= (x-1,y) \\ AC &= (x+1,y) \end{aligned} \tag{7.3}$$

As discussed in Section 5.3.6.4, the ambiguity in the definition of polarity for these line segments must be addressed before they can be used as vectors. This can be accomplished by normalization by an even subfraction of 360°, or by other methods. Generally the vectors will be normalized before summing to form the value for the orientation attribute that will be used to enter the point triple into the index. Section 7.4 discusses various normalization methods, and how they accommodate the different orders of rotational symmetry.

7.3.5 Rotational Symmetry

No matter which method for assigning a reference direction is employed, two special cases remain a significant challenge. Equilateral triangles have an inherent 3-fold rotational symmetry. For these triangles it is not possible to geometrically define an unambiguous reference direction. Similarly, symmetric collinear pseudotriangles possess a 2-fold rotational symmetry. The common factor in these two special cases is that they possess a k-fold rotational symmetry. The inability to define a reference direction is not simply a result of a particular choice of definition. In these two cases,

the figures possess an innate rotational symmetry, and do not admit an unambiguous geometric definition of reference direction.

The approach taken in this chapter is to assign the value 360° as the inherent *orientational selectivity* of a scalene triangle. Triangles with a k-fold rotational symmetry are considered as possessing a k-fold *orientational ambiguity*, and $360° \div k$ is assigned for their inherent orientational selectivity. Thus 120° of orientational selectivity is retained by an equilateral triangle, and 180° for a symmetric collinear pseudotriangle.

The following sections give a number of examples of the use of the normalization techniques from Section 5.3.4 to address the problem of orientation ambiguity in a point-based index space.

7.4 Index Space Construction

A point-based index for triangles can be constructed by mapping any chosen subset of the separable attributes of position, size, orientation, and shape onto orthogonal axes of a hyperdimensional index space. Index dimensions for position and size are straightforward. Several alternate methods of structuring the orientation and shape dimensions are presented here.

7.4.1 Structures Based on a 360° Space

Figure 7.10 shows an index space (designated A360) formed by rotating the angle-based shape space of Figure 7.4 around the B-D axis. With this structure,

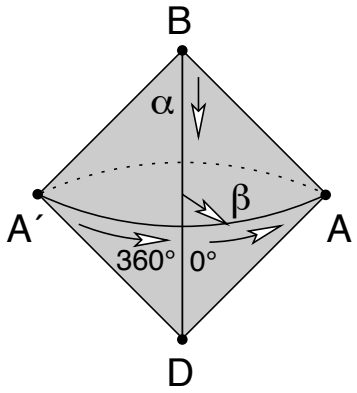


Figure 7.10: A360: Angle-based, not normalized

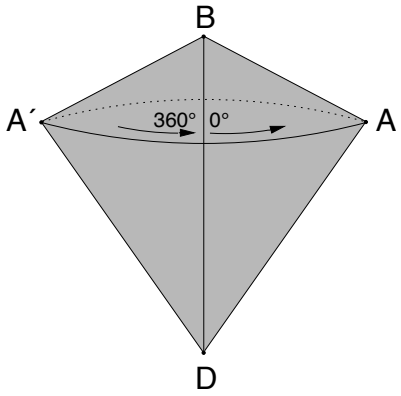


Figure 7.11: S360: Side-based, not normalized

all triangles whose reference direction (as defined by the smallest-angle method) is ambiguous lie along line B-D and thus their orientational index information is suppressed. That is, no matter what their original spatial orientation, they map along line B-D in this index space.

Figure 7.11 shows a similar figure formed by rotating the side-based shape space of Figure 7.5 around the B-D axis.

These index space structures correspond to using technique 6 from Section 5.3.4 to address both the $k = 2$ and $k = 3$ cases of orientational ambiguity. The disadvantage of doing so is that none of the inherent orientational selectivity of equilateral triangles, wide-isosceles triangles, and collinear pseudo-triangles is used in the index

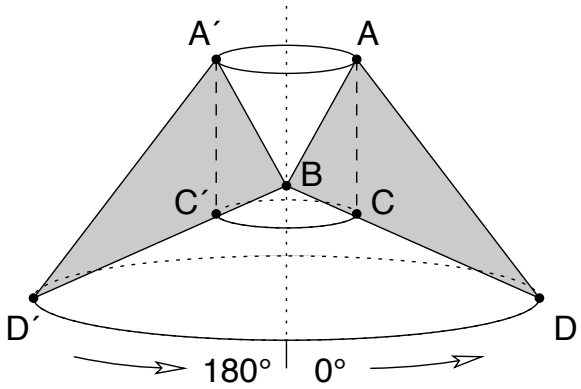


Figure 7.12: A180: Angle-based, folded at 180°

(because they all map to points along the line B-D, where the index structure does not capture orientation). However, the orientation need not be normalized, so there is no sacrifice of orientational selectivity for other triangles (which constitute the vast majority of the triangles indexed).

7.4.2 Structures Based on a 180° Space

A mixed approach would preserve some of the orientational selectivity of wide isosceles triangles and of collinear pseudo-triangles, while sacrificing less of the orientational selectivity of the vast majority of triangles that possess no ambiguity.

Figure 7.12 shows an index space (designated A180) formed by rotating the angle-based shape space of Figure 7.3 around the equilateral triangle index point (B). Mapping all equilateral triangles, regardless of orientation, to this single point in the index space suppresses the orientational ambiguity due to 3-fold rotational symmetry. The rotation is normalized at 180° so the ambiguity in collinear pseudotriangles is suppressed, and such figures map to points along circle D-D′ at the bottom of the space. Wide isosceles triangles map to the cone D-B-D′ and right

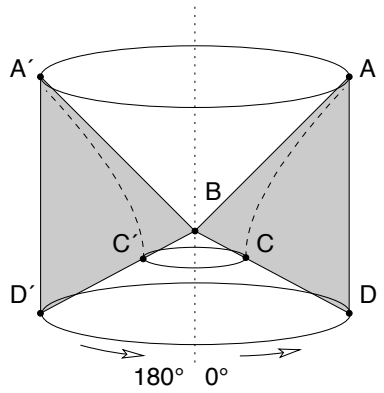


Figure 7.13: S180: Side-based, folded at 180°

triangles map to the cylinder A-A′-C′-C. Because the rotational orientation is suppressed for equilateral triangles and accommodated for collinear pseudotriangles, the Euler line method can be used to define the reference orientation.

A similar structure could be generated by rotating Figure 7.4 around point B. In fact, any formulation of triangle shape space could be rotated around the equilateral triangle point to generate an analogous index space.

This index structure corresponds to using technique 5 from Section 5.3.4 to address $k = 2$ ambiguity and technique 6 to address $k = 3$ ambiguity. The disadvantage of doing so is that the inherent orientational selectivity of equilateral triangles is not used in the index. In addition, because the orientation space is folded at 180°, only half the inherent orientation selectivity of all other triangles is used in the index.

Figure 7.13 shows an index space (designated S180) formed by rotating the normalized-side-based shape space of Figure 7.5 around point B. In this index space all equilateral triangles map to point B, thus the 3-fold orientational ambiguity inherent in these equilateral triangles is accommodated, at the cost of sacrificing

all their orientational selectivity. Right triangles map to points along the vase-shaped figure A-C-C′-A′. Tall isosceles triangles map to the cone A-B-A′ and wide isosceles triangles map to the cone D-B-D′. Collinear pseudotriangles map to the cylinder A-D-D′-A′. The 2-fold orientational ambiguity of these pseudotriangles is accommodated by normalizing the orientation axis at 180°.

In common with the A180 structure, this index space corresponds to using technique 5 from Section 5.3.4 to address $k = 2$ ambiguity and technique 6 to address $k = 3$ ambiguity. The disadvantage of doing so is that only half of the inherent orientational selectivity of all triangles is used in the index. However, the orientational selectivity of asymmetric collinear pseudotriangles is used in the index. This is the advantage of using the side-based shape space.

Alternatively, the center-to-point eccentricity vector method can be used to define similar index spaces. The following analysis employs an analytic geometry approach. Starting with the double-angle formulas $\sin 2\alpha = 2\sin\alpha\cos\alpha$ and $\cos 2\alpha = \cos^2\alpha - \sin^2\alpha$ and simplifying yields:

$$\begin{aligned} x' &= \frac{\Delta x^2 - \Delta y^2}{\sqrt{\Delta x^2 + \Delta y^2}} \\ y' &= \frac{2\Delta x \Delta y}{\sqrt{\Delta x^2 + \Delta y^2}} \end{aligned} \tag{7.4}$$

Applying this transform to each of the three center-to-point vectors from Equation 7.2 and summing yields:

$$\Sigma x' = \frac{1}{3}\left[\frac{(x+3)^2 - y^2}{\sqrt{(x+3)^2 + y^2}} + \frac{(x-3)^2 - y^2}{\sqrt{(x-3)^2 + y^2}} + 2\frac{x^2 + y^2}{\sqrt{x^2 - y^2}}\right]$$

$$\Sigma y' = \frac{2y}{3}\left[\frac{(x+3)}{\sqrt{(x+3)^2+y^2}} + \frac{(x-3)}{\sqrt{(x-3)^2+y^2}} + 2\frac{x}{\sqrt{x^2+y^2}}\right] \tag{7.5}$$

The only point triple with a 3-way rotational symmetry is the equilateral triangle, which corresponds to the point $(0, \sqrt{3})$. Substitution of these values into Equation 7.5 yields zero. So this normalization does not accommodate the 3-way rotational symmetry.

The only point triple with a 2-way rotational symmetry is the symmetric version of the collinear pseudotriangle of section 7.1, which is generated by the three coordinate values $(\pm 3, 0)$ and $(0, 0)$. Although these appear to be poles for Equation 7.5, the values in the numerators dominate the values in the denominators, so at these critical points the affected terms vanish (become zero). The geometric interpretation of this situation is that for these values the center and one of the points coincide, and so one center-to-point distance becomes zero.

Substitution shows that coordinate values $(\pm 3, 0)$ yield $(4, 0)$ and coordinate value $(0, 0)$ yields $(2, 0)$, which are all nonzero. Thus the 2-way rotational symmetries are accommodated.

Because the 180° normalization accommodates the 2-way rotational symmetry inherent in the interpoint line segments, the interpoint method could also be used for this index scheme. Transforming each of the three raw vectors of Equation 7.3 with the double-angle transform of Equation 7.4 and summing yields:

$$\Sigma x' = 2 + \frac{(x+1)^2 - y^2}{\sqrt{(x+1)^2+y^2}} + \frac{(x-1)^2 - y^2}{\sqrt{(x-1)^2+y^2}} \tag{7.6}$$

$$\Sigma y' \;=\; 2y\left[\frac{x+1}{\sqrt{(x+1)^2+y^2}}+\frac{x-1}{\sqrt{(x-1)^2+y^2}}\right]$$

The only point triple with a 3-way rotational symmetry is the equilateral triangle, which corresponds to the point $(0, \sqrt{3})$. Substitution of this value into Equation 7.6 yields zero, so this normalization does not accommodate the 3-way rotational symmetry.

The only point triple with a 2-way rotational symmetry is the symmetric version of the collinear pseudotriangle of section 7.1. There are three point coordinates that generate this figure. Substitution shows that coordinate values $(\pm 3, 0)$ yield $(8, 0)$ and coordinate value $(0, 0)$ yields $(4, 0)$. Thus the 2-way rotational symmetries are accommodated.

Under the assumption that $x = 0$ (the triangle is isosceles) the $\Sigma x'$ part of Equation 7.6 is solvable, and its zeroes are $y = \pm\sqrt{3}$. So the vector is only zero at $(x, y) = (0, \pm\sqrt{3})$, which is the case for the equilateral triangle.

As both the center to point and interpoint methods accommodate the 2-way rotational symmetry, either eccentricity vector method is appropriate for a 180° index structure.

7.4.3 Structures Based on a 120° Space

Figure 7.14 shows an index space (designated S120) formed by rotating the normalized-side-length based shape space of Figure 7.5 around line A-D along which all the collinear pseudotriangles map. This index structure corresponds to using technique 5 from Section 5.3.4 to address $k = 3$ ambiguity and technique 6 to

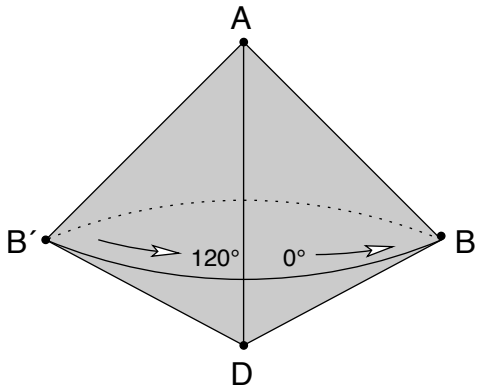


Figure 7.14: S120: Side-based, folded at 120°

address $k = 2$ ambiguity. The disadvantage of doing this is that none of the inherent orientational selectivity of collinear pseudotriangles is used in the index. Because the orientational index is wrapped at 120°, all the orientational selectivity of equilateral triangles is used in the index. However, only one third of the inherent orientational selectivity of scalene and isosceles triangles is used in the index.

The center-to-point eccentricity vector method can be used to define reference directions in this structure. Starting with the triple-angle formulas $\sin 3\alpha = 3\sin\alpha - 4\sin^3\alpha$ and $\cos 3\alpha = 4\cos^3\alpha - 3\cos\alpha$ and simplifying yields:

$$\begin{aligned} x' &= \frac{4\Delta x^3}{\Delta x^2 + \Delta y^2} - 3\Delta x \\ y' &= 3\Delta y - \frac{4\Delta y^3}{\Delta x^2 + \Delta y^2} \end{aligned} \tag{7.7}$$

Applying this transform to each of the three center-to-point vectors of Equation 7.2 and summing yields:

$$\Sigma x' = \frac{4}{3}\left[\frac{2x^3}{x^2+y^2} - \frac{(x+3)^3}{(x+3)^2+y^2} - \frac{(x-3)^3}{(x-3)^2+y^2}\right] \tag{7.8}$$

$$\Sigma y' = \frac{4y^3}{3}\left[\frac{1}{(x+3)^2+y^2}+\frac{1}{(x-3)^2+y^2}-\frac{2}{x^2+y^2}\right]$$

The only point triple with a 3-way rotational symmetry is the equilateral triangle (shown in Figure 5.7), which corresponds to the point $(0, \sqrt{3})$. Substitution of these values into Equation 7.8 yields $(0, -2\sqrt{3})$. Thus the 3-way rotational symmetry is accommodated.

The only point triple with a 2-way rotational symmetry is the symmetric version of the collinear pseudotriangle of section 7.1, and is generated by the three coordinate values $(\pm 3, 0)$ and $(0, 0)$. Although these appear to be poles for Equation 7.8, the cubic terms in the numerators dominate the quadratic terms in the denominators. So at these critical points the affected terms actually vanish (become zero). The geometric interpretation of this situation is that for these values, the center and one of the points coincide, and so one center-to-point distance becomes zero.

When this is taken into account, Equation 7.8 yields a zero vector for these configurations. Thus the normalization at $360° \div 3$ does not accommodate the 2-way rotational symmetry. In fact, this method yields a zero sum for all collinear configurations, instead of just the symmetric ones.

The interpoint method could also be used for this index structure. As discussed in Section 5.3.6.4, normalization of the orientations of the interpoint line segments by an even subfraction of 360° obviates the necessity for definition of their polarity. In a two-dimensional space, and in the special case that the line segments form a convex polygon, an alternate method is to use the clockwise/counterclockwise sense

to define the polarities.

Three points always form a two-dimensional convex triangle. Directions for the segments may be assigned in a clockwise manner:

$$\begin{aligned} BA &= (-2, 0) \\ AC &= (x+1, y) \\ CB &= (1-x, -y) \end{aligned} \tag{7.9}$$

Substituting Equation 7.9 into Equation 7.7 and summing yields:

$$\begin{aligned} \Sigma x' &= 4\left[\frac{(x+1)^3}{(x+1)^2+y^2} - \frac{(x-1)^3}{(x-1)^2+y^2} - 2\right] \\ \Sigma y' &= 4y^3\left[\frac{1}{(x+1)^2+y^2} + \frac{1}{(x-1)^2+y^2}\right] \end{aligned} \tag{7.10}$$

The only point triple with a 3-way rotational symmetry is the equilateral triangle, which corresponds to the point $(0, \sqrt{3})$. Substitution of these values into Equation 7.10 yields $(-6, 0)$. Thus the 3-way rotational symmetry is accommodated.

The only point triple with a 2-way rotation symmetry is the symmetric version of the collinear pseudotriangle of section 7.1, and is generated by the three coordinate values $(\pm 3, 0)$ and $(0, 0)$. Substitution of the values $(0, 0)$ yields a zero vector. Thus the 2-way rotational symmetry is not accommodated.

Under the assumption that $x = 0$ (the triangle is isosceles) the $\Sigma x'$ part of Equation 7.10 is solvable, and its zero is when $y = 0$. So the vector is only zero at $(x, y) = (0, 0)$, which is a case of three collinear points. As both the center to point and interpoint methods accommodate the 3-way rotational symmetry, either

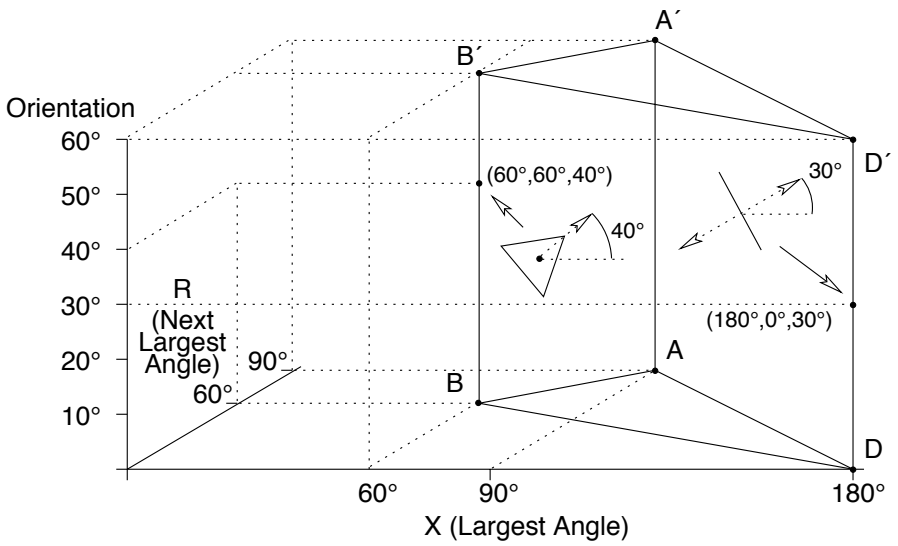


Figure 7.15: A60: Angle-based, folded at 60°

eccentricity vector method is appropriate for a 120° index structure.

7.4.4 Structures Based on a 60° Space

Figure 7.15 shows an index space (designated A60) formed by extending the angle-based shape space of Figure 7.3 along an orientational dimension with a 6-way folding (that is, the modulo 60° residue of the triangle's orientation determines the orientation component of the index space coordinates). Thus, for an equilateral triangle with possible orientations of 40°, 160°, or 280°, the triangle will be indexed at 40° up on the B-B′ line. Similarly, a collinear pseudotriangle with orientation of 30° or 210° will be indexed at 30° up on the D-D′ line. As the orientation of wide isosceles triangles is not suppressed, the Euler line method should be used to define their reference direction.

It should be noted that the orientational dimension is wrapped at 60°, that is, the top plane A′-B′-D′ is actually identical to the bottom plane A-B-D. If this figure

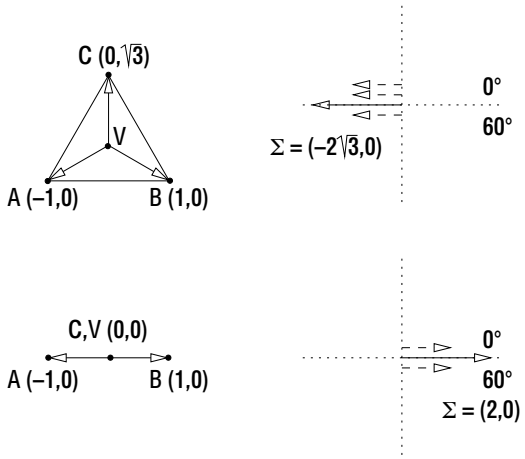


Figure 7.16: Center-to-Point Normalized for Both 2-way and 3-way Symmetries

were redrawn to be analogous to the other figures in this section, it would resemble Figure 7.12, with the exception that the locus of point B during the rotation would be a small circle instead of a point.

This index space structure corresponds to using technique 5 from Section 5.3.4 to address both the $k = 2$ and $k = 3$ cases of orientational ambiguity. Their least common multiple value of 6 is used to fold the orientational index dimension. The disadvantage of doing so is that the orientational selectivity of the index for all triangles is reduced by the same factor, resulting in the generation of a large number of false positives that must eventually be filtered out of the result set. However, the question still remains as to how to determine the reference direction when using this index.

Figure 7.16 shows that for both the equilateral triangle (with a 3-way rotational symmetry) and the symmetric collinear pseudotriangle (with a 2-way rotational symmetry) the center-to-point method with normalization to 60° yields a nonzero vector. So both the 2-way and 3-way rotational symmetries are accommodated.

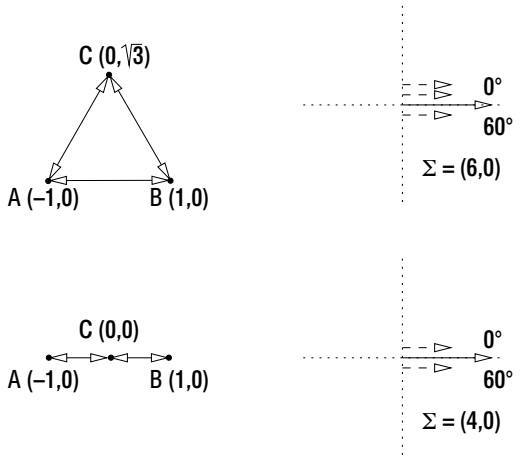


Figure 7.17: Interpoint Normalized for Both 2-way and 3-way Symmetries

However, because the x value changes sign while the y value is zero, the Intermediate Value Theorem dictates the existence of at least one intermediate shape at which the value must be zero. This result was not expected, and a search using numerical techniques identified $(0, 0.6847+)$ and $(0, 4.1291+)$ as shapes which yield a zero vector sum. Because of these zeroes, this normalization method is not suitable for implementation of the A60 and S60 indexes.

Figure 7.17 shows that for both the equilateral triangle (with a 3-way rotational symmetry) and the symmetric collinear pseudotriangle (with a 2-way rotational symmetry) the interpoint method with normalization to 60° yields a nonzero vector. So both the 2-way and 3-way rotational symmetries are accommodated.

However, the same numerical search technique identifies $(0, 0.4815+)$, $(0, 0.7265+)$ and $(0, 4.3812+)$ as shapes that yield a zero vector sum (the sign of the x component changes *twice* as the shape flattens from an equilateral triangle to a symmetrical pseudotriangle). Thus this normalization method is not suitable for implementation of the A60 and S60 indexes.

	A60 S60	A120 S120	A180 S180	A360 S360
Scalene	1/6	1/3	1/2	All
Triangle	1/6	1/3	1/2	All
Tall	1/6	1/3	1/2	All
Isosceles	1/6	1/3	1/2	All
Wide	1/6	1/3	1/2	None
Isosceles	1/6	1/3	1/2	None
Asymmetric	1/6	None	None	None
Collinear	1/6	None	1/2	All
Symmetric	1/3	None	All	None
Collinear	1/3	None	All	None
Equilateral	1/2	All	None	None
Triangle	1/2	All	None	None

Figure 7.18: Orientational selectivity used

At this time no method is known to always generate a nonzero eccentricity vector. However, the elegance of the eccentricity vector method is that zero values do not cause the indexing method to fail, but merely map the problematic triangle to the central axis of a hypercylindrical index space, as described in Section 5.4.

7.4.5 Index Structure Summary

Figure 7.18 tabulates the fraction of the inherent orientational selectivity (broken down for various classes of triangles) actually used for each of the index structures described in this section.

7.5 Concluding Remarks

In an iconic image database, point triples can be indexed based on geometric attributes of their associated triangles. A size and orientation independent characterization of the triangle "shape" attribute can be based either on interior angles

or normalized side lengths; in both cases the induced shape space is inherently two-dimensional.

In the general case of an object embedded in an n-dimensional space, characterization of spatial orientation requires n parameters, but in the special case of a planar (two-dimensional) space, orientation degenerates into simple rotation about an axis, and a single parameter suffices. Objects with rotational symmetry inherently possess ambiguity in the assignment of reference direction. No geometrical method is know to exists that resolves this ambiguity.

In integrating the shape space and the orientation dimension into a combined index space, two techniques are available for addressing the orientational ambiguity of objects possessing k-fold rotational symmetry.

1. The points in the shape space to which the problematic objects map can be located at points in the combined index space where the coordinate value of the orientation dimension is suppressed (for example, at the zero point of a polar plot). The disadvantage of doing so is that the inherent orientational selectivity of such objects will not be used in the resulting index.

2. The orientation dimension itself can be folded at an angle less than 360°, such that all ambiguous orientation values map to the same point in the index space. The disadvantage of doing so is that only some fraction of the inherent orientational selectivity for every object will be used in the index, even for those objects that do not possess any orientational ambiguity.

The A60 index addresses both ambiguous cases, equilateral triangles (3-fold ro-

tational symmetry) and symmetric collinear pseudotriangles (2-fold rotational symmetry) by folding the orientation dimension at $360° \div 6 = 60°$. The A360 index addresses both ambiguous cases by mapping them (and all wide isosceles triangles) to the zero point of a polar plot. The other example index spaces utilize combinations of these techniques, one technique to address each ambiguous case. Absent an application-specific knowledge of the distribution of triangles, the S360 index would appear to be optimal, as it uses all the available orientational selectivity of the majority of the expected triangles.

The methods described in this chapter can be used to construct a point-based index for triples, with which any combination of position-independent, orientation-independent, and size-independent search can efficiently be accomplished.

Chapter 8

Indexing Arrangements of Four or More Points

Prior chapters have dealt with configurations of two and three points. In this chapter, we focus on configurations of four or more points. Although finding a general solution to cope with configurations of arbitrary numbers n of points is desirable, in practice, configurations of more than 4 points are rare in the application for which this work has been applied (e.g., [39, 48]). In particular, the query processing engine used therein requires that the specified spatial relationships must hold for all of the objects in the query image rather than just a subset of them. When the spatial relationship only involves a subset of the objects, the query is decomposed into several subqueries and the necessary result is obtained by making use of a Boolean combination of them. One of the motivations for investigating configurations of four points is that, unlike configurations of three points, the configurations of four points do not map directly to traditional convex quadrilaterals.

However, four points always define six interpoint line segments (which may in some cases be partitioned into a set of 4 "edges" and a set of 2 "diagonals"). The possibility that either all four points or a set of three points may be collinear must also be considered.

The remainder of this chapter is organized as follows: Section 8.1 describes the emergent properties when the number of objects n goes from 3 to 4. Section 8.3

describes an inherently four-dimensional abstract space of point quartet "shapes", and examines methods for defining a basis for this space. Section 8.4 presents methods of deriving a value for the spatial orientation of a point quartet, and discusses the special problems posed by rotational symmetry. Section 8.5 presents a method for constructing a point-based index, showing how the problems introduced by rotational symmetry can be addressed by using innovative methods of integrating the shape and orientation dimensions. Section 8.6 contains concluding remarks.

8.1 Emergent Properties

It is well known that when attempts are made to generalize based on a known case, complications (termed *emergent properties*) not previously present sometimes arise. An example of such a complication is the emergence of non-Abelian properties in the group of rotations in k-dimensional space when k increases from 2 to 3. Thus it is not surprising that several such complications arise when generalizing from point triples to point quartets. For example, the four points may be non-coplanar, or may fail to unambiguously determine a convex quadrilateral. Moreover, even if a quadrilateral is determined, it will generally be non-rigid, and in some cases may scale in non-isotropic ways. In the rest of this section, we explore some of the properties which complicate the task of deriving from the quadrilateral shape a set of attribute values that can be used as components of a point-based index.

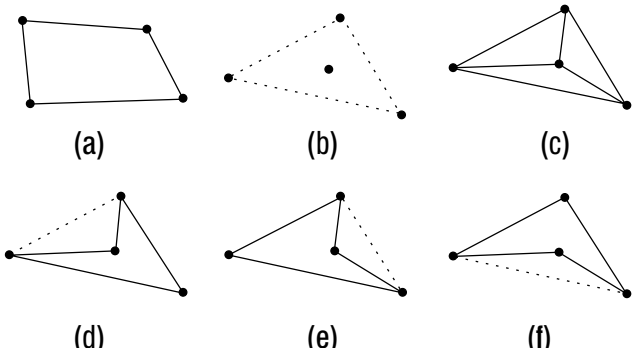


Figure 8.1: Non-convexity

8.1.1 Non-Planarity

In a Euclidean space three non-collinear points always determine a plane. Thus it is not possible for a point triple to fail to be planar. However, in a space of three or more dimensions, a point quartet may fail to be planar. Such a situation is sometimes termed a *skew quadrilateral*. In the rest of this chapter a two-dimensional space will be assumed.

8.1.2 Non-Convexity

With the exception of the one special case of collinear points (which has been suitably handled in Chapter 7), configurations of three points uniquely determine triangles, which are always convex. However, configurations of four points fall into two distinct classes. In particular, the convex hull of the four points may contain all four points, as in Figure 8.1(a), in which case a convex quadrilateral is uniquely determined. Alternatively, as in Figure 8.1(b), it may be the case that the convex hull contains only three of the points, with the fourth point being interior to the triangle formed by the other three points.

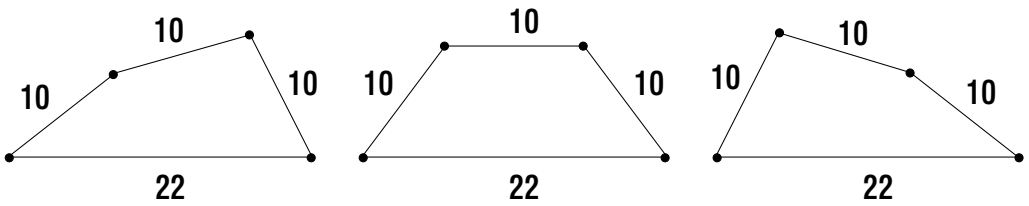


Figure 8.2: Non-rigidity

In this case, it is not possible to define a traditional quadrilateral unambiguously, as there are three equally valid possibilities, shown in Figure 8.1(d), (e), and (f). Furthermore, regardless of which of the three possibilities is selected, the resulting quadrilateral will fail to be convex.

In all cases, a set of six inter-point line segments (as shown in Figure 8.1(c)) is determined. However, in the case where the fourth point is interior to the triangle, it is not possible to partition these six line segments into a set of four "sides" and a set of two "diagonals".

8.1.3 Non-Rigidity

A triangle is always rigid; its shape is unambiguously determined by three vertex angles, or alternatively by the mutual ratios of the three side lengths (which can be shown using the relation between angles and sides described by the Law of Sines). In contrast, a quadrilateral (or other higher-order polygon) is in general not rigid. Even if the side lengths are completely determined, there remain unconstrained degrees of freedom. Loosely, the quadrilateral can be said to "flop around". For example, Figure 8.2 shows that in general the set of four side lengths does not completely determine the shape of a quadrilateral.

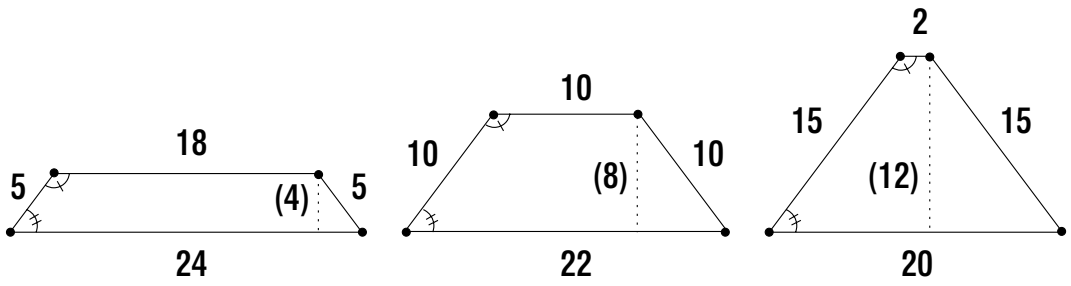


Figure 8.3: Non-isotropic Scaling

8.1.4 Non-Isotropic Scaling

Another interesting emergent property is that in many cases the shape of a quadrilateral is not completely determined by the set of vertex angles. In particular, whenever two (necessarily opposite) sides of a quadrilateral are parallel, the set of side lengths is not constrained by the vertex angles, but can vary anisotropically. Such behavior could be called "tromboning".

For example, Figure 8.3 shows three figures with identical vertex angles but with different shapes. The equivalence of the corresponding angles can be seen by observing that each figure consists of an inner rectangle flanked by a pair of congruent 3-4-5 right triangles (the dotted-line altitude delimits the boundary of one such triangle). Thus the doubly-slashed angle is $\arctan \frac{4}{3}$ or approximately 53.13°. Note that this example has been constructed so that the perimeters of the three figures are equal (at 52 units). While this will not generally be true, the example does demonstrate that there are cases in which the perimeter fails to distinguish between the varying shapes.

The effect of this analogy is similar to that of an ellipse, which can be defined by the length of its principal axis and whose shape depends on the positions of its

foci.

8.2 Size

Earlier definitions of size were based upon polygon perimeters. Although in the general case it is not possible to unambiguously map point quartets to quadrilaterals, the sum of the lengths of the six interpoint line segments is an acceptable proxy for the size of the arrangement. This measure easily generalizes to arrangements of more than four points.

8.3 Shape

In this section we analyze the concept of "shape" as it is applicable to a configuration of four points. We first examine the case of a traditional convex quadrilateral, which is followed by the extension of the analysis to the more general case.

As we have seen, the abstract shape of a traditional quadrilateral cannot (due to non-rigidity) be unambiguously characterized by a set of side lengths, nor (due to tromboning) can it always be unambiguously characterized by a set of vertex angles. Therefore, the values for a shape attribute must be derived from a combination of both side lengths and vertex angles.

To accommodate uniform scaling, the perimeter of the quadrilateral can be normalized (say to 1), with the normalization factor becoming one of the index attributes. Put another way, the "size" of the quadrilateral becomes one axis of

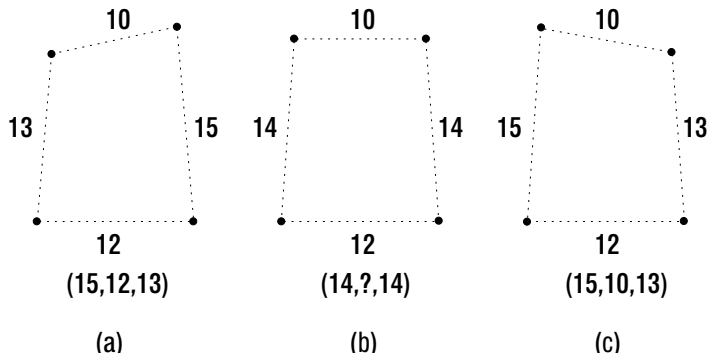


Figure 8.4: Discontinuity in side length characterization

the index space. After this normalization, any three side lengths unambiguously determine the length of the fourth side.

The method of organizing the set of side lengths also affects the continuity of the index (recall the definition of continuity in Section 5.2.1). Figure 8.4 shows the results of an attempt to characterize the side lengths starting with the longest side and proceeding clockwise (note that in spaces of more than 2 dimensions even the notion of "clockwise" breaks down). When the longest side is unambiguous (cases (a) and (c)), the ordering of the lengths is well defined, but at the seam (b) where two sides are nearly the same length, continuity from both directions cannot be obtained.

One way to avoid this discontinuity is to construct the index based on the sorted set of side lengths, irrespective of their relative position in the quadrilateral, as shown in Figure 8.5. While this method avoids the discontinuity described above, it does introduce the possibility for additional hash collisions. In particular, Figure 8.5 demonstrates that two very different shapes can generate the same sorted set of side lengths, and so would collide on this coordinate of a point-based index.

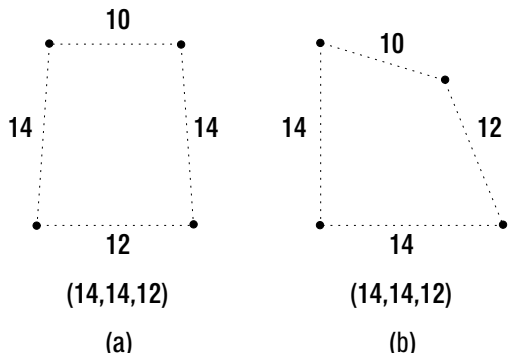


Figure 8.5: Ambiguity in side length histogram

One vertex angle is also required to complete the shape characterization. This can be the largest (or the smallest) of the vertex angles. Although choosing the largest angle, regardless of its geometric relationship to the sides, generates the same kind of ambiguity as does a histogram of the side lengths, it does assure continuity of the index function at seams where the geometric position of the largest angle suddenly shifts.

As discussed in Section 8.1.2, when the four points do not all lie on their mutual convex hull (as in Figure 8.1(b)), it is not possible to unambiguously define a traditional quadrilateral. But for any arbitrary configuration of four distinct points, the six interpoint distances (Figure 8.1(c)) are well defined (and nonzero), and thus could be used to construct an alternative shape component of an index. Shape changes that for a traditional quadrilateral would be described as "flopping" also change the lengths of the diagonals, thereby allowing a shape index to be defined based on lengths only, without requiring the consideration of vertex angles.

Figure 5.1 contains an analysis of the degrees of freedom that are involved in the characterization of the "shape" of arbitrary planar configurations of n points (in

two dimensions), and demonstrates that it is inherently $2n - 4$ dimensional. This can be seen by observing that each of the n points in the configuration introduces two degrees of freedom, in the form of x and y coordinates. Two degrees of freedom are subsumed by the absolute spatial position of the configuration as a whole (e.g., the x and y coordinates of the center of gravity), one degree of freedom is subsumed in uniform scaling, and one in rotation (e.g., orientation) in the plane. For example, the six initial degrees of freedom of a trio of points (e.g., a triangle), are reduced to two inherent shape dimensions. This makes sense, as the "shape" of a triangle is completely determined by two vertex angles, or (given a normalized perimeter) by two side lengths (rigidity was discussed in Section 8.1.3).

From this analysis, as the shape space for configurations of four points is inherently four-dimensional, the shape component of a point-based index could be based on any of a number of schemes such as the largest four of the six interpoint distances, the largest two and the smallest two, etc. Monte Carlo experiments could be performed to investigate the performance of these schemes under differing data distribution statistics.

Note that for a collinear configuration, the length of the longest segment (between the two extremal points) remains unchanged as the positions of the interior points vary. Thus, in order to fully utilize the shape variety of such configurations, the technique chosen to derive the shape attribute should ensure that segments other than the longest segment are included.

As we shall see in the next section, use of an "orientation" attribute can also contribute a component to the set of "shape" attributes.

8.4 Orientation

While rotation in a plane involves only a single degree of freedom, determining a reference direction for an arbitrary point quartet is not straightforward. The classical literature on the properties of quadrilaterals does not contain any useful analog to the concept of the "Euler line" of a triangle, as was discussed in Chapters 5 and 7. Moreover, point quartets exist that do not correspond to any classically studied quadrilateral. Nevertheless, it is always the case that the four points in a quartet can be paired in six distinct ways, thereby determining six interpoint line segments. In cases where the four points do form a classic convex quadrilateral, four of these lines are termed sides or edges, while the remaining two are termed diagonals. However, in all cases, six lines are determined. In the rest of this section we make some observations using the terminology of classic quadrilaterals, after which we present a representation of orientation that is based solely on the six interpoint lines.

Note that all of the methods described in this section for deriving a value for the orientation attribute are based on the concept of an *eccentricity vector*. Intuitively, our aim is to capture a measure of the deviation of a given configuration from a symmetric one. When the vector is nonzero, its direction component provides an absolute value for the "orientation" of the configuration.

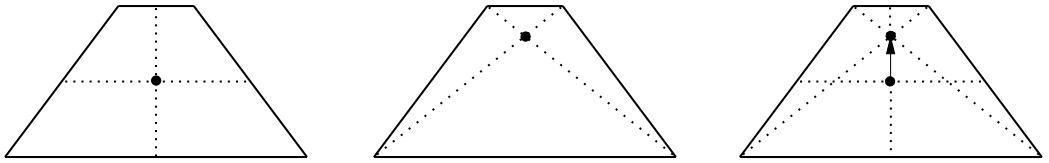


Figure 8.6: Bimedial and Diagonal Centers

8.4.1 The Intercenter Vector

In the classic quadrilateral, the two lines that connect the midpoints of opposite sides are called *bimedians* (Figure 8.6(a)). We use the term *bimedial center* to denote the intersection of these bimedians, and the term *diagonal center* to denote the intersection of the quadrilateral's diagonals (Figure 8.6(b)). When these two centers do not coincide, a vector directed from one center to the other (Figure 8.6(c)) provides a reference direction for deriving the value for the quadrilateral's orientation attribute. Because this vector is fixed to the frame of the quadrilateral, it rotates directly with it, and as the quadrilateral rotates through a full circle of 2π radians, the intercenter vector does the same. Thus the direction component of the intercenter vector will be referred to as a 2π measure of rotation.

However, a number of important quadrilaterals (such as the rectangle, the rhombus, and the rhomboid) possess a 2-fold rotational symmetry, and for such quadrilaterals the two centers coincide. In this case the direction component of the vector becomes indeterminate, and does not effectively characterize the orientation of the quadrilateral. Furthermore, this scheme is not useful for quartets that do not unambiguously determine a classic convex quadrilateral.

8.4.2 Pi-Space Characterization

The orientation of a classic quadrilateral can also be captured as a function of its two diagonals. The diagonals are line segments and we know that whenever we have a line segment in space, it can be parameterized in terms of its length and direction. The ambiguity associated with defining the direction of a line segment can be removed by mapping the length and direction of the line onto a space in which a rotation through 180° leaves the direction invariant. (This is equivalent to normalization by $360° \div 2$ as described in Section 5.3.5.) Mapping a line segment as a vector in such a π-*space* is equivalent to moving one endpoint to the origin and doubling the angle corresponding to its slope. The mathematics involved follow directly from the classical double-angle formulas. This can be seen with the aid of Figure 8.7. Substituting the double-angle formulas $\sin 2\alpha = 2\sin\alpha\cos\alpha$ and $\cos 2\alpha = \cos^2\alpha - \sin^2\alpha$ and simplifying yields:

$$x' = \frac{\Delta x^2 - \Delta y^2}{\sqrt{\Delta x^2 + \Delta y^2}}$$
$$y' = \frac{2\Delta x \Delta y}{\sqrt{\Delta x^2 + \Delta y^2}} \tag{8.1}$$

Note that reversal of the direction of the line segment (corresponding to the negation of both Δx and Δy) leaves Equation 8.1 unchanged, as the two negations cancel out in the numerator of the y' term and all other appearances of Δx and Δy are squared.

In this method, angles were multiplied by two in order to accommodate a two-way rotational symmetry. The general technique of multiplication of the angle

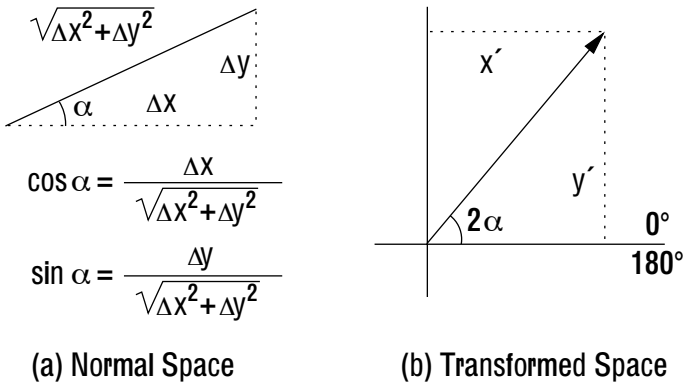


Figure 8.7: Mapping a line segment in π-space

by k to accommodate a k-way rotational symmetry is called *normalization*, and is discussed more extensively in Chapter 5.

In the case of a classic convex quadrilateral, we can try to capture its orientation by one parameter instead of two as in the case where the diagonals are considered independently. This parameter can be formed by taking the vector sum of the diagonals once they have been mapped into π-*space*, which serves as a reference direction for the rotation of the quadrilateral. The advantage of this approach is that although ambiguity exists in assigning a direction to each of the diagonals (segment (a) in Figure 8.8), both alternatives map to the same vector in π-space, and thus the process of mapping to π-space resolves the ambiguity in the assignment of direction. As the quadrilateral rotates through a half-circle of π radians, the π-space vector sum rotates through a full circle of 2π radians. In contrast with the 2π measure of rotation of Section 8.4.1 this will be referred to as a π measure of rotation.

Unfortunately, in cases where this vector sum is zero, this scheme will fail to yield a reference direction. The vector sum is zero only when the π-space vectors

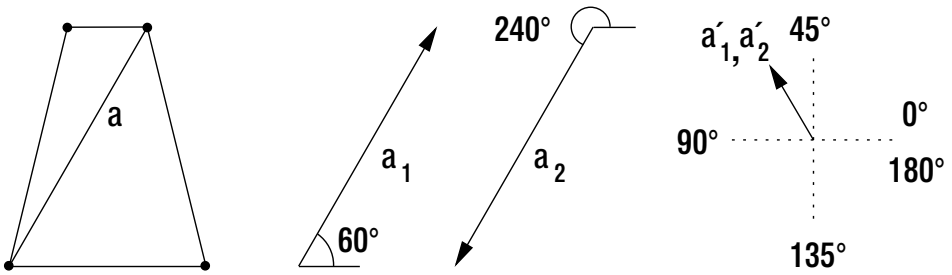


Figure 8.8: Diagonal in Pi-space

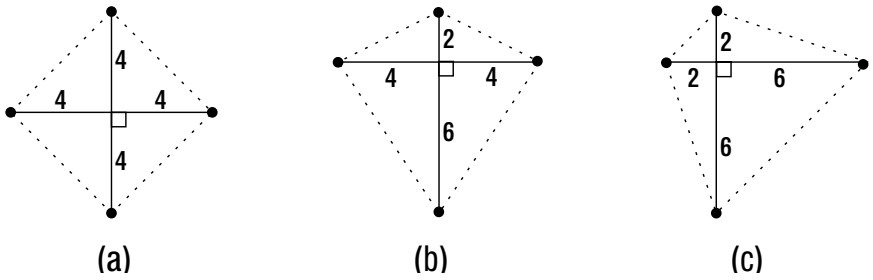


Figure 8.9: Kite-like figures

have equal length and opposite direction, i.e., the diagonals have equal length and are perpendicular to each other (as 90° in real space corresponds to 180° in π-space). This is the case not only for the square but also for a number of other kite-like figures. Figure 8.9 shows (a) a square with diagonals sharing a common midpoint; (b) a kite with the midpoint of one diagonal displaced along the other; and (c) a "bi-kite" with the midpoints of both diagonals displaced (note that (c) is actually a symmetrical bi-kite which is a special form of a trapezoid in which two equal diagonals intersect at right angles). In all of these cases, the π-space vector sum will be zero, and no reference direction will be determined. It is important to note also that this scheme is not applicable to point quartets that do not form a classic convex quadrilateral.

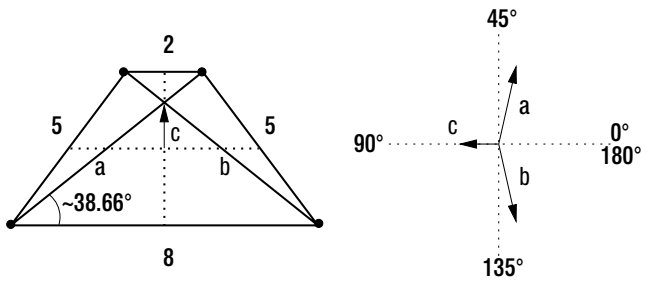


Figure 8.10: Vector sum of pi and 2 pi can be zero

8.4.3 Combined Intercenter Diagonal

The only quadrilateral for which both the 2π and π measures described in Sections 8.4.1 and 8.4.2, respectively, are both zero is the square (which is also the only quadrilateral possessing a 4-fold rotational symmetry). Given that the square represents the same kind of special case as the equilateral triangle presented for earlier work on point triples in Chapter 7, it is tempting to attempt to characterize orientation by combining the π and 2π measures into a single coordinate, as is the case, for example, when taking their vector sum. However, attempts to do so may result in the introduction of additional cases in which a reference direction cannot be determined.

This situation is demonstrated by the example in Figure 8.10 where the angle of diagonal (a) is $\arctan\frac{4}{5}$ or approximately 38.66°. In particular, the right half of the figure shows the result of combining the two measures by computing the vector sum in π space of the diagonal vectors (a) and (b) and the intercenter (or 2π) measure (c). No matter what relative weight is given to vector (c), cases will always occur in which the vector sum is zero (even though none of the component vectors of the sum is zero), and in such cases the direction of the vector sum again

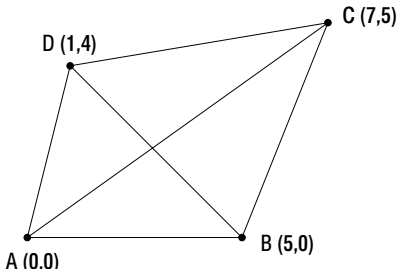
becomes indeterminate.

Alternatively, instead of attempting to capture orientation by combining the two measures (i.e., the 2π and π) into a single dimension, a point-based index can be constructed with two independent dimensions for orientation, one to represent each of these two rotation measures. As five dimensions are already being proposed (one for absolute scale and four for shape), this solution has the effect of expanding the proposed index space from six to seven dimensions, which is not expected to cause a significant performance degradation.

8.4.4 Interpoint Vector Sum

It is important to note that none of the schemes described in Sections 8.4.1–8.4.3 is applicable to cases in which the point quartet does not correspond to a classic convex quadrilateral. In order to accommodate such cases, the six interpoint lines can be mapped into π-space and summed, to yield a π-measure for the rotation of the quartet. As a concrete example, Figure 8.11 shows a quartet of points that actually does form a classic quadrilateral, with the six interpoint lines mapped into π-space, and the resulting sum computed using Equation 8.1.

The use of this approach to define a reference direction may fail when the quartet possesses a k-fold rotational symmetry. For example, consider Figure 8.12(a), where three points are arranged at the vertices of an equilateral triangle and a fourth point is located at the center. This figure possesses a 3-fold rotational symmetry. Figure 8.12(b) shows the result of mapping the six interpoint lines in π space and



Seg	Δx	Δy	x'	y'
AB	5.000	0.000	5.000	0.000
AC	7.000	5.000	2.790	8.137
AD	1.000	4.000	−3.368	1.940
BC	2.000	5.000	−3.900	3.714
BD	−4.000	4.000	0.000	−5.657
CD	−6.000	−1.000	5.754	1.973
		Vector Sum	6.006	10.107

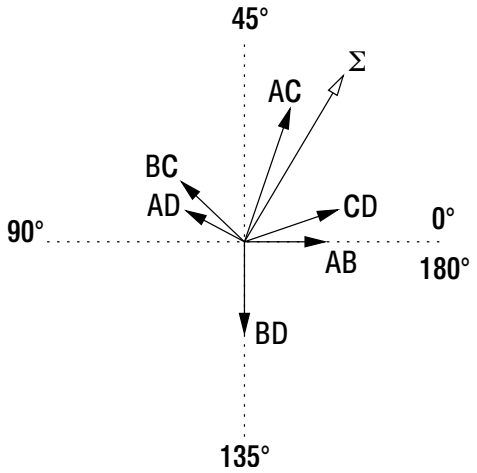


Figure 8.11: Point Quartet with Six Lines

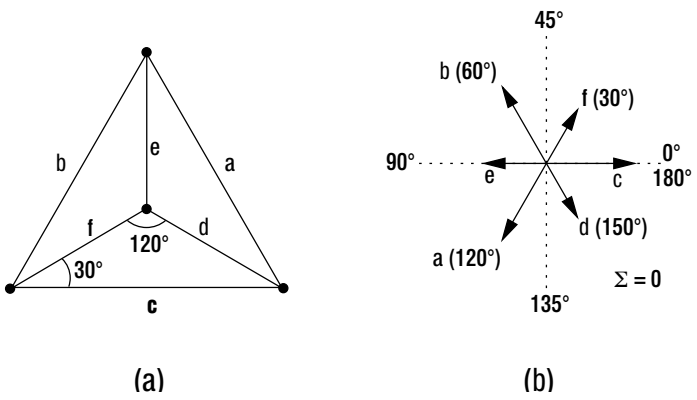


Figure 8.12: Three-fold rotational symmetry

we see that the vector sum in this space is zero, which means that in this case no reference direction is determined.

As another example, consider Figure 8.13(a) where four points are arranged at the corners of a square. This figure possesses a 4-fold rotational symmetry. In Figure 8.13(b) the six interpoint lines are mapped into π space, and again, the vector sum is zero (as a convention, dashed lines indicate vectors drawn off-origin for clarity). However, in the case of a figure with two-fold rotational symmetry (e.g., Figure 8.13(c)) the π space vector sum (d) is non-zero. Thus this scheme does allow the definition of reference direction for figures with a two-fold rotational symmetry.

8.4.5 Orientation Summary

In general, we recall from Figure 5.1 that the configuration space of point quartets is inherently four-dimensional, and that these methods map it to a two-dimensional (polar coordinate) space. Those shapes which map to the polar origin constitute the *kernel* of this mapping, and the Rank-nullity Theorem can be used to show that the dimensionality of this kernel is two. Thus the shapes which result in a

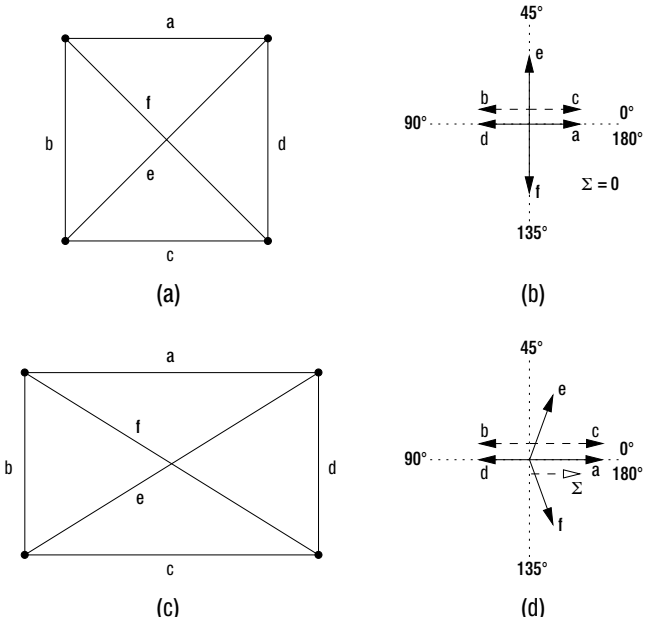


Figure 8.13: Four and two-fold rotational symmetry

zero eccentricity vector form some two-dimensional subspace of the four-dimensional shape space.

The choice of eccentricity vector used in any particular index should be influenced by the statistical properties of the expected data. Symmetric configurations have the potential of generating a zero eccentricity vector. We want to to make sure that whatever eccentricity vector method that we use generates a zero vector only for configurations (symmetric and non-symmetric) that are not frequently expected. The reason for doing so is to avoid the loss of orientational selectivity for frequently encountered configurations that is a consequence of mapping such configurations to the axis of a cylindrical index space. This is illustrated in the next section.

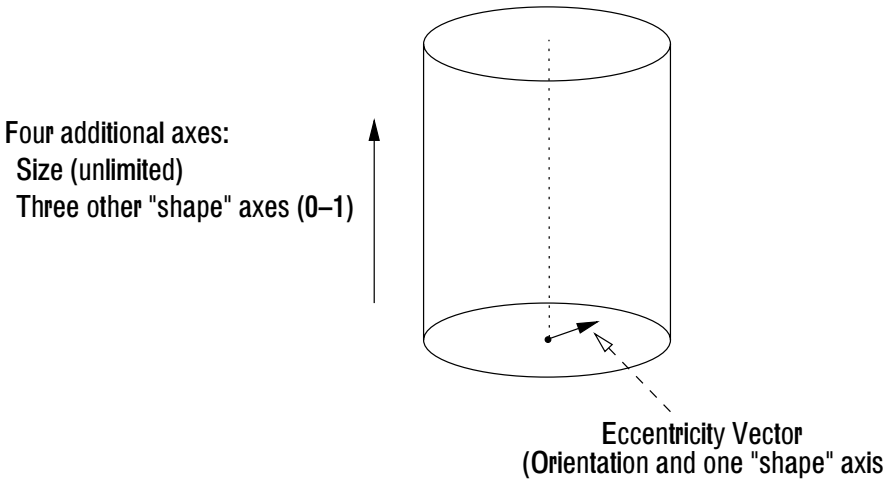


Figure 8.14: Integrated Index

8.5 Index Construction

After determining the shape and orientation methods to be used, the attribute axes must be integrated into an index space. As the orientation attribute is inherently circular, it is natural to use some form of polar coordinates, as in the index spaces described in Chapter 7, which are structured as the projection of a set of polar coordinates into a four-dimensional hypercylinder, with the polar coordinate origin projecting to the hypercylinder axis. This scheme can be extended quite naturally to configurations of four or more points, as shown in Figure 8.14.

For each indexed configuration, the value of the eccentricity vector determines both the angular and radial polar coordinates. The magnitude of the eccentricity vector is used as one component of a basis for the shape space. Therefore $2n - 5$ other basis components (to be selected as described in Section 8.3), in addition to one size component, become the linear axes of the hypercylindrical index space.

The drawback of this scheme is that all configurations for which the eccen-

tricity vector is zero will be mapped to points along the hypercylinder axis, and hence no information on the orientation of these configurations will be represented in the index. This includes not only the configurations with symmetries not accommodated by the normalization scheme chosen, but also a set of other configurations determined by the particular eccentricity vector method that is used.

8.6 Concluding Remarks

A point-based index for configurations of four points can be constructed with attributes axes for size, "shape", and orientation. Although orientation-independent search for point configurations can be supported by an index based only on size and shape, extending the index structure to support orientation-dependent search, by augmentation with a closed (e.g., circular) orientation axis, requires use of some algorithm to determine for each point configuration an absolute value for the orientation attribute, under which that configuration will be indexed.

Various methods were presented for deriving an "eccentricity vector" which captures the deviation of the configuration from a symmetrical one. Using the eccentricity vector's direction as the orientation attribute value and its magnitude as one component of the "shape" attribute value enables configurations to be mapped into a hypercylindrical index space.

However, for each of the methods described, there exists a set of pathological configurations which result in computation of a zero-length eccentricity vector. These configurations include rotationally-symmetric ones, such as the square and

the equilateral triangle plus center point, but for some methods include additional non-symmetric configurations. Such configurations map to the central axis of the cylindrical hyperspace, and their inherent orientational selectivity will not be used in the index.

For applications that expect to index large numbers of such figures, the technique of rotation attribute normalization described in Chapter 5 can be used, at the cost of a severe reduction in orientational selectivity for all indexed configurations.

Chapter 9

Concluding Remarks

Position-independent similarity search of an image database can be supported by a retrieval-by-content query system, of which MARCO is one example. The existential part of an iconic image database query (i.e., determining the images that contain particular icons, regardless of their spatial arrangement) can be satisfied by a traditional inverted-file index containing icon instances and the images in which they are present. The degree of improvement in real-world iconic image database access efficiency made possible by processing some of a query's spatio-orientational constraints directly in the index is an area open for future quantitative study.

We have shown now to construct an efficient index for configurations of point objects by deriving separable attribute values for size, orientation, and "shape" (that which remains after factoring out size and orientation). Such an index can efficiently support a variety of search modes, including both size-dependent and size-independent, and orientation-dependent and orientation-independent searches. The size and shape of a point arrangement is easily derived and indexed; however, characterizing the orientation of a configuration of points is sometimes complicated by the issue of rotational symmetry.

Orientation is not an issue for a point pair, once the inherent $360° \div 2$ symmetry is accommodated by normalization. All point pairs have the same shape, and size is

easily derived. A method of index pruning that greatly reduces the index size was presented.

Point triples introduce the concept of shape, and one ad-hoc solution for the shape of a point triple is to use the inherent geometric properties of their associated triangles, such as the Euler line. Solutions based on the "eccentricity vector" are also possible.

Several ad-hoc solutions were presented for collections of four points. In addition, the eccentricity vector approach provides a general solution to the problem of deriving a reference orientation for an arrangement. However, arrangements that possess a rotational symmetry generate a zero value for the eccentricity vector. While this does not prevent the index from operating correctly, it does impact efficiency, as configurations with zero eccentricity vectors map to the central axis of the hypercylindrical index space, and thus the orientation does not participate in generating selectivity in the index.

Normalization, or reducing the value of the orientation modulo $360° \div k$, is useful for point configurations in two-dimensional space, by enabling the "inter-point" eccentricity vector technique, and removing some rotational symmetries in databases where figures with those symmetries are expected to be encountered frequently. However, this technique reduces the selectivity of the index by a factor of k for all figures stored.

Extending the MARCO visual language to allow user control over the new search capabilities is easily accomplished by introducing new visual language elements. One area for future research is to evaluate such elements in a real-world

setting.

Bibliography

[1] Helmut Alt, Oswin Aichholzer, and Günter Rote. Matching shapes with a reference point. In *Proceedings of the Tenth Annual Symposium on Computational Geometry*, pages 85–92, Stony Brook, New York, 1994.

[2] W. G. Aref and H. Samet. Extending a DBMS with spatial operations. In O. Günther and H.-J. Schek, editors, *Advances in Spatial Databases—2nd Symposium, SSD'91*, vol. 525 of Springer-Verlag Lecture Notes in Computer Science, pages 299–318, Zurich, Switzerland, August 1991.

[3] D. Arkoumanis, M. Terrovitis, and E. Stamatogiannakis. Heuristic algorithms for similar configuration retrieval in spatial databases. In *Proceedings of the 2nd Hellenic Conference on Artificial Intelligence (SETN)*, pages 141–152, Thessalonica, Greece, 2002.

[4] S. Berretti, A. Del Bimbo, and E. Vicario. Managing the complexity of match in retrieval by spatial arrangement. In *10th International Conference on Image Analysis and Processing (ICIAP 1999)*, pages 1178–1183, Venice, Italy, September 1999. IEEE Computer Society.

[5] S. Berretti, A. Del Bimbo, and E. Vicario. Efficient matching and indexing of graph models in content-based retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(12):1089–1105, October 2001.

[6] P. B. Callahan and S. R. Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *Journal of the ACM*, 42(1):67–90, January 1995.

[7] S. K. Chang, Q. Y. Shi, and C. Y. Yan. Iconic indexing by 2-D strings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(3):413–428, May 1987.

[8] M. Cibelli, M. Nappi, and M. Tucci. Content-based access in image database by quantitative relationships. *Journal of Visual Languages and Computing*, 11(5):573–589, October 2000.

[9] H. S. M. Coxeter. *Introduction to Geometry.* John Wiley & Sons, New York, second edition, 1969.

[10] C. B. Cranston and H. Samet. Indexing planar point quartets via geometric attributes. Submitted to the 15th ACM International Symposium on Advances in Geographic Information Systems.

[11] C. B. Cranston and H. Samet. Orientation issues in position-independent indexing of planar point configurations. Submitted to the 24th IEEE International Conference on Data Engineering.

[12] C. B. Cranston and H. Samet. Efficient position-independent iconic search using an r-theta index. In *Proceedings of the 14th ACM International Symposium on Advances in Geographic Information Systems*, pages 27–34, Arlington, VA, November 2006.

[13] C. B. Cranston and H. Samet. Indexing point triples via triangle geometry. In *Proceedings of the 23rd IEEE International Conference on Data Engineering*, pages 936–945, Istanbul, Turkey, April 2007.

[14] A. Del Bimbo, E. Vicario, and D. Zingoni. A spatial logic for symbolic description of image contents. *Journal of Visual Languages and Computing*, 5(3):267–286, September 1994.

[15] M. J. Egenhofer. Query processing in spatial-query-by-sketch. *Journal of Visual Languages and Computing*, 8(4):403–424, August 1997.

[16] J. Enderle, M. Hampel, and T. Seidl. Joining interval data in relational databases. In *Proceedings of the ACM SIGMOD Conference*, pages 683–694, Paris, France, June 2004.

[17] C. Faloutsos and W. Rego. Tri-cell—a data structure for spatial objects. *Information Systems*, 14(2):131–139, 1989.

[18] M. Farshi and J. Gudmundsson. Experimental study of geometric t-spanners. In *Proceedings of the 13th Annual European Symposium on Algorithms*, volume 3669, pages 556–557, 2005.

[19] A. Folkers, H. Samet, and A. Soffer. Processing pictorial queries with multiple instances using isomorphic subgraphs. In *Proceedings of the Fifteenth International Conference on Pattern Recognition*, volume 4, pages 51–54, Barcelona, Spain, September 2000.

[20] R. K. Goyal and M. J. Egenhofer. Similarity of cardinal directions. In *Proceedings of the Seventh International Symposium on Spatial and Temporal Databases.*, volume 2121 of *Lecture Notes in Computer Science*, pages 36–55, Los Angeles, California, July 2001.

[21] V. Gudivada. $\Theta\Re$ string: A geometry-based representation for efficient and effective retrieval of images by spatial similarity. *IEEE Transactions of Knowledge and Data Engineering*, 10(3):504–512, May/June 1998.

[22] V. Gudivada and V. Raghavan. Design and evaluation of algorithms for image retrieval by spatial similarity. *ACM Transactions on Information Systems*, 13(2):115–144, April 1995.

[23] A. Guttman. R-trees: a dynamic index structure for spatial searching. In *Proceedings of the ACM SIGMOD Conference*, pages 47–57, Boston, June 1984.

[24] A. Henrich, H.-W. Six, and P. Widmayer. The LSD tree: spatial access to multidimensional point and non-point data. In P. M. G. Apers and G. Wiederhold, editors, *Proceedings of the 15th International Conference on Very Large Databases (VLDB)*, pages 45–53, Amsterdam, The Netherlands, August 1989.

[25] K. Hinrichs and J. Nievergelt. The grid file: a data structure designed to support proximity queries on spatial objects. In M. Nagl and J. Perl, editors, *Proceedings of WG'83, International Workshop on Graphtheoretic Concepts in Computer Science*, pages 100–113, Haus Ohrbeck (near Osnabrück), West Germany, 1983. Trauner Verlag.

[26] A. Holt. Spatial similarity and GIS: the grouping of spatial kinds. In P. A. Whigham, editor, *Eleventh Annual Colloquium of the Spatial Information Research Center (SIRC05)*, pages 241–250, Dunedin, New Zealand, December 1999. University of Otago.

[27] P. W. Huang and Y. R. Jean. Using 2D C^+-strings as spatial knowledge representation for image database systems. *The Journal of the Pattern Recognition Society*, 27(9):1249–1257, September 1994.

[28] U. Lauther. 4-dimensional binary search trees as a means to speed up associative searches in design rule verification of integrated circuits. *Journal of Design Automation and Fault-Tolerant Computing*, 2(3):241–147, July 1978.

[29] S. Y. Lee and F. J. Hsu. 2D C-string: a new spatial knowledge representation for image database systems. *Pattern Recognition*, 23(10):1077–1088, October 1990.

[30] D. Papadias, N. Karacapilidis, and D. Arkoumanis. Processing fuzzy spatial queries: A configuration similarity approach. *International Journal of Geographic Information Science*, 13(2):93–128, 1999.

[31] D. Papadias, M. Mantzourogiannis, and I. Ahmad. Fast retrieval of similar configurations. *IEEE Transactions on Multimedia*, 5(2):210–222, June 2003.

[32] D. Papadias and T. K. Sellis. A pictoral query-by-example language. *Journal of Visual Languages and Computing*, 6(1):53–72, March 1995.

[33] G. Petraglia, M. Sebillo, M. Tucci, and G. Tortora. A normalized index for image databases. In S. K. Chang, E. Jungert, and G. Tortora, editors, *Intelligent Image Database Systems*, pages 43–69. World Scientific Publishing, Singapore, 1996.

[34] J. B. Rosenberg. Geographical data structures compared: a study of data structures supporting region queries. *IEEE Transactions on Computer-Aided Design*, 4(1):53–67, January 1985.

[35] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, Reading, MA, 1990.

[36] H. Samet. Personal communication. (unpublished), 2000.

[37] H. Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan-Kaufmann, San Francisco, 2006.

[38] H. Samet and A. Soffer. A map acquisition, storage, indexing and retrieval system. In *International Conference on Document Analysis and Recognition*, volume 2, pages 992–996, Montréal, Canada, August 1995.

[39] H. Samet and A. Soffer. MARCO: MAp Retrieval by COntent. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):783–798, August 1996.

[40] B. Seeger and H.-P. Kriegel. Techniques for design and implementation of efficient spatial access methods. In F. Bachillon and D. J. DeWitt, editors, *Proceedings of the 14th International Conference on Very Large Databases (VLDB)*, pages 360–371, Los Angeles, August 1988.

[41] A. Silberschatz, H. F. Korth, and S. Sudarshan. *Database System Concepts*. McGraw-Hill, New York, third edition, 1996.

[42] A. P. Sistla, C. Yu, and R. Haddad. Reasoning about spatial relationships in picture retrieval systems. In J. Bocca, M. Jarke, and C. Zaniolo, editors, *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB)*, pages 570–581, Santiago, Chile, September 1994.

[43] S. Skiadopoulos, C. Giannoukos, P. Vassiliadis, T. K. Sellis, and M. Koubarakis. Computing and handling cardinal direction information. In *Proceedings of the 9th International Conference on Extending Database Technology (EDBT)*, volume 2992 of *Lecture Notes in Computer Science*, pages 329–347, Heraklion, Crete, Greece, March 2004.

[44] A. W. M. Smeulders and R. Jain, editors. *Proceedings of the First International Workshop on Image Databases and Multi Media Search (IDB-MMS'96)*, Amsterdam, The Netherlands, August 1996.

[45] A. Soffer and H. Samet. Handling multiple instances of symbols in pictorial queries by image similarity. In A. W. M. Smeulders and R. Jain, editors, *Proceedings of the 1st International Workshop on Image Databases and Multi-Media Search (IDB-MMS'96)*, pages 51–58, Amsterdam, The Netherlands, August 1996.

[46] A. Soffer and H. Samet. Pictorial queries by image similarity. In *Proceedings of the 13th International Conference on Pattern Recognition*, volume III, pages 114–119, Vienna, Austria, August 1996.

[47] A. Soffer and H. Samet. Retrieval by content in symbolic-image databases. In I. K. Sethi and R. Jain, editors, *Proceedings of the SPIE, Storage and Retrieval*

of Still Image and Video Databases IV, volume 2670, pages 144–155, San Jose, CA, January 1996.

[48] A. Soffer and H. Samet. Pictorial query specification for browsing through spatially referenced image databases. *Journal of Visual Languages and Computing*, 9(6):567–596, December 1998. Also an abbreviated version in *Proceedings of the Second International Conference on Visual Information Systems (VISUAL97)*, pages 117–124, San Diego, CA, December 1997.

[49] A. Soffer, H. Samet, and D. Zotkin. Pictorial query trees for query specification in image databases. In A. K. Jain, S. Venkathesh, and B. C. Lovell, editors, *Proceedings of the 14th International Conference on Pattern Recognition*, volume 1, pages 919–921, Brisbane, Australia, August 1998.

[50] J.-W. Song, K.-Y. Whang, Y.-K. Lee, M.-J. Lee, and S.-W. Kim. Spatial join processing using corner transformation. *IEEE Transactions on Knowledge and Data Engineering*, 11(4):688–695, July/August 1999.

[51] B. H. Weinberg. Predecessors of scientific indexing structures in the domain of religion. In *Proceedings of the 2002 Conference on the History and Heritage of Scientific and Technological Information Systems*, ASIS&T Monograph Series, pages 126–134, Medford, New Jersey, 2002. The American Society for Information Science and Technology and The Chemical Heritage Foundation, Information Today, Inc.