Matching **O**f **S**patial **T**emplates
Project Report Part 2

Exploiting spatial locality during automatic recognition of spatial templates

Steven Woods
DMR Groupe Inc.
Under the scientific authority of Stéphane Lapointe, Defence Scientist

March 27, 1995

Abstract

This document is delivered in fulfillment of a contract between the Defence Research Establishment Valcartier, DMR Groupe Inc., and Steven Woods. This document reports an investigation of the possible methods and approaches to exploiting spatial locality and spatial indexing in the process of identifying military deployment templates in situated sets of unit sightings. Two specific recognition methodologies developed in earlier DREV work are the intended application beds of this work.

Contents

List of Figures

Chapter 1

Introduction

When a particular military unit is deployed in the field, it is typically oriented as part of a larger military grouping such as a company. Companies in turn are deployed as part of battalions, and so on. Each of these individual unit deployments may be typically seen as belonging to some large-scale deployment plan. Intelligence analysts attempt to identify the sets of sighted military units which belong to a common higher level grouping, and further, to determine what task is being undertaken by each higher level grouping. Military doctrine provides specific information about the composition of higher level groupings for a particular activity or task with respect to unit types and their relative deployment positions and individual activities. This information may be extracted from doctrine and represented in spatial templates which describe a higher level grouping of units according to the relative position and activity of the constituent units. Information relating different units according to position or other attribution (such as orientation) may be regarded as constraints which define the range of acceptable matches for this template.

The determination by an intelligence analyst that a certain set of sighted units constitutes some higher level grouping and task (say a motor rifle regiment (MRR) in attack) corresponds to a mapping between a particular subset of sighted elements, and the general deployment template for an MRR in attack. Each unit in the MRR-attack template maps to an actual sighted unit, and further, the relational or attributional constraints must all be satisfied.

In earlier work methodologies and tools have been proposed for the representation and identification of these spatial templates in complex sets of sightings where a particular template instance may not be at all obvious. In this work we are *seeking to increase the heuristic effectiveness* of earlier approaches by exploiting the spatial structures inherent in these templates.

Instances of these templates are identified through a combination of problem reduction pre-processing and search. The representation chosen for both templates and situations has a direct affect on both of these. Essentially, a set of mappings from template parts to situation elements that is consistent with respect to template constraints or requirements constitutes a possible solution. Pre-processing attempts to eliminate entire ranges of not-possible solutions in advance of search, while search provides a mechanism with which to apply the constraints of the template systematically. Within this framework we would like to embody heuristics that reflect the way experts deal with problem complexity and reduce it to make problems more manageable. Typically these heuristics exploit some problem representation structures in the context of a particular pre-processing or search strategy.

One simple heuristic that allows us to make many inferences in everyday life is that objects that are spatially near one another are more likely or more typically related than two objects that are spatially distant. For instance, we observe that a computer's keyboard is typically near it's display monitor. Thus, if we observe a particular keyboard near a particular monitor a sensible inference is that the two are related, in this case as part of a *relation* we define as a single computer system. Contrariwise, if we observe a particular keyboard and also observe a monitor 20 feet away, any direct relationship between the two does not seem quite so likely. We might similarly have defined a computer spatial template in terms of the two primary objects monitor and keyboard, along with a spatial constraint that the two not be farther than 2 feet (reading distance) apart. This "nearness" heuristic is one simple encapsulation of the concept of spatial locality. The locality is exploited by *attempting* to relate only those objects that are **both** *near each other* and also of compatible types.

This paper describes an investigation into possible avenues in which the concept of spatial locality may be utilized to exploit knowledge about the spatial properties of spatial deployment templates. In particular, we wish to *reduce the overall computational cost* of the task of identifying template instances in situations which contain many extraneous or unrelated units, and units that are only partially defined. The spatial manipulation methods investigated are intended to be applicable to two currently separate but related representations of the spatial template matching problem.

The structure of this paper is as follows. In Chapter 2 we present a brief overview of the spatial template recognition problem , and discuss two representational methods which have been investigated at Defence Research Establishment Valcartier (DREV) by Defence Scientists Stéphane Lapointe and Consultant René Proulx[46, 59] and Steven Woods[77, 76]. Reference to the documents reporting this work is made frequently in this report, and it is expected that the reader is familiar with these documents. These two representations share many spatial properties which may potentially be exploited in order to improve the efficiency of recognition approaches. These properties are outlined further in Chapter 3. Chapter 4 summarizes how the hierarchical representations of our earlier work neatly connects with spatial locality application. Chapter 5 discusses in greater detail how spatial locality features embedded within situational information may be "pre-processed" in order to reduce the expense of extracting this information from scratch during search. In Chapter 5, section 5.1 describes how the situational representations of locality may be combined with abstracted template representations (embodying locality) during search. Chapter 6 summarizes the recommendations of this report in terms of desirable future work and applications, and gives a short list of acknowledgements. We conclude the report with a brief literature survey in

Chapter 7, followed by a bibliography.

Chapter 2

Spatial template recognition

In this chapter we first briefly overview the concept of spatial locality and the problem of spatial template recognition in general, including a brief discussion of the complexity of the spatial template recognition problem. Recent work in spatial template recognition (STR) at DREV has focused on two related representation schemes and approaches to the spatial template recognition problem. We describe each in turn. The first, *constraint-based recognition* is based on a body of work in artificial intelligence and search concerned with solving Constraint Satisfaction Problems (CSPs). The second, utilizing *fuzzy geometric relations*, is based on work in fuzzy set theory. Both approaches provide for flexible representation of spatial templates which arranges a set of "building blocks" of various attributional types according to inter-block spatial relationships (or constraints). Both representations explicitly provide for problem decomposition hierarchically, and map this representation to the solution methodology. In the next chapter (Chapter 3) we distill the common spatial features regarding locality as implied by each of the two STR represention al schemes to be crucial for efficient recognition.

2.1 What is Spatial Template Recognition ?

STR is the problem of finding an instance of some predefined configuration of objects in a given situation, or set of object instances. An example of a non-typically rigid spatial template in a two-dimensional domain would be a "map" of a typical molecule that describes the molecule in terms of the relative positions, types and orientations of the various atoms. An example of a rather discrete, exact situation might be a two-dimensional sample containing many instances of different types of molecules discernible only by individual sub-parts. The question formed by STR for this problem instance would be: "Where are the instances of this molecular template in this situation?". In this example both the template and the elements of the situation have relatively "rigid" design - the molecular model has little variability from instance to instance, and the situation has potentially little ambiguity with respect to both identity and position of situational elements.

A more complex domain for STR is intelligence analysis as described in detail in [77, 76, 46]. An intelligence analyst must draw on specific experiential and learned knowledge of enemy behaviour to recognize specific deployment patterns signifying organizational relationships of sighted enemy elements. One aspect of this complex task is recognizing the existence of enemy "doctrinal templates" which detail the patterns in which military units are commonly deployed for specific activities. These "templates" represent a flexible view of how the elements of a military grouping are commonly positioned. Since military units are often strictly structured and deployed according to fixed procedures, a fairly accurate "typical" spatial representation can be built of particular formations undertaking certain activities in the field, such as "Motor Rifle Regiment (MRR) in attack". The analyst must fuse spatially and temporally disjoint messages or sightings into a coherent picture of the entire situation, including higher level formations and their activity. The individual elements or sub-objects of the MRR spatial template may be envisioned as "slots" in the template, and the rules or restriction regarding deployment of these elements can be represented as constraints amongst the template slots. The current set of sightings in which one seeks template instances is referred to as a "situation" consisting of many object instances or "situation elements".

One major difficulty with such a task is that these templates are embedded within an extremely complex, dynamic and incomplete "view" of the enemy[1]. Intelligence messages from the field may contain vague or conflicting composition or location information, and inaccurate messages may have been modified by experts into inaccurate conclusions. Also, any template is simply an approximation of some expert-held "ideal" concept, related work on improving these approximations with experience may be found in [79]. Templates defined in terms of spatial relationships between components of specific types, sizes, activities and orientation approximate reality by allowing certain tolerances on the attributes or spatial constraints. Considering a template as a set of "stick-pins" of varying types and sizes interconnected by a set of rubber-bands with a minimum and maximum stretch capability, the process of template matching reduces to finding appropriate situation elements in which the stick pins may fit and where the bands between the pins allow this fit.

Matching of these templates to incompletely specified situational views can result in many partial solutions where only some subset of the pins could be fit to a situation element. In these cases which of many partial solutions is more promising may possibly be determined according to predefined criteria or measures. In addition, the knowledge of the precise structure of these templates themselves is only approximate, and the templates themselves may evolve over time. Consequently, any attempt to perform the recognition task algorithmically will find approximate solutions only, however, these may form a base-point for expert utilization.

[1]Other related work in pattern analysis such as [67], [25] and [75] models problems with a far greater "rigidity" in terms of the spatial template model.

2.2 Constraint-based spatial template recognition

A relatively obvious formulation of STR as a **Constraint Satisfaction Problem** (CSP) was outlined in [77] and described in detail in [76]. An STR is defined as follows: A situation consists of a set of elements each with a set of characteristics (such as color, type, etc.). A template consists of a set of variables, each possibly limited by a set of range or specific characteristic values. Additionally, each variable is possibly connected to any other variable (or possibly set of variables) by a constraint-arc or set of arcs which impose relative characteristics such as relative position, orientation, or possibly which insists upon pairwise variable characteristic assignments. A template instance is "found" in a situation if a 1:1 mapping exists from situation elements to variables such that no situation element maps to more than one variable, and based on this assignment, all constraints represented by constraint-arcs are satisfied. A decision problem based on this definition could be expressed as "Given a Template T = (set of Variables V + set of Variable-constraints VC + Arc-constraints AC), and a Situation S = (set of Situation elements), is there an instance of T in S ?". Problems specified in this fashion, where a solution is composed of a set of consistent element-to-slot assignments are well described in [44]. An example of a template constraint graph is shown in Figure 2.1 with five node slots where each slot is constrained by several attribute ranges, and 11 inter-slot spatial constraints on distance, position, and orientation.

2.2.1 CSP solution and terminology

CSP solution is typically accomplished through *constraint propagation* techniques which attempt to systematically reduce variable domain ranges through application of constraints between variables, or through *search-based* approaches which repeatedly select variables and instantiations and reduce remaining variable domains according to existing constraints. A thorough examination of the range of these techniques is presented in [54, 44]. During backtracking search, each variable instantiation is interpreted as extending the amount of a particular spatial template that has been recognized.

Constraint Propagation is the reduction of domain ranges locally or globally within the CSP problem graph. Existing algorithms include AC-3[47], AC-4[53], AC-5[70], and other variations[22, 10]. In [76] we experimentally examined the utility of some of these algorithms as well as partial consistency algorithms described in [54].

The efficiency of search algorithms is highly dependent on heuristics such as the order in which variables (spatial template parts) are selected for matching against a particular situation. *Variable Selection* is the determination of which component variable should be chosen next for instantiation consideration during search. The decision may be based on domain independent measures, such as the size of a variable's domain; on information specific to the problem instance such as spatial properties between variables; or on some combination of these types of information.

Another critical heuristic that can affect search performance is the order in which candi-

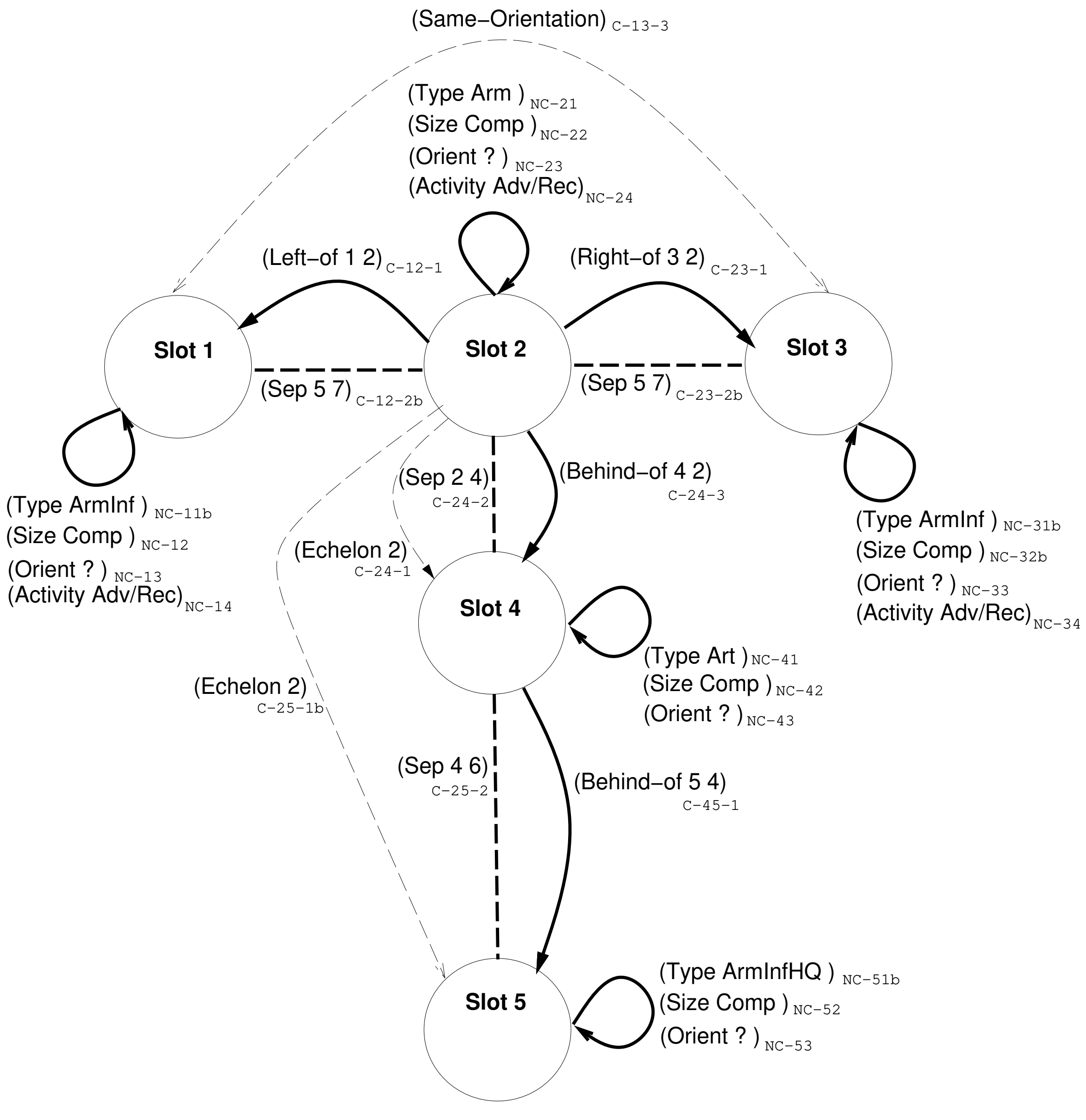

Figure 2.1: Whole CSP 5 variables, 11 arc constraints.

date situation elements or instances are selected for instantiation. This choice is referred to as *Domain Value Selection*. Typically this selection should be made so as to most effectively limit the remaining variable ranges.

During search it may be profitable to "take time out" from search and perform some *In-search Propagation* to reduce the size of remaining uninstantiated variable domains according to some constraint propagation algorithm. Problem characteristics such as variable domains that exceed some average or absolute bounds are potential signals that constraint propagation may be useful before continuing search. In [54] the advantages of exploiting various algorithms for achieving a limited degree of partial consistency amongst variable sets are examined.

When no candidate instances remain for an outstanding variable we have reached a "dead end" in search, and backtracking is required. *BackTrack point selection* is the determination, after it has become evident that no possible solution exists along a particular variable-instantiation path, of which instantiation to retract. Intelligent backtracking approaches such as BackJumping and BackMarking[2] make a limited attempt to determine the origin of the conflict that caused the failure, and to BackTrack as far up the search tree as possible to avoid a repeated failure of the same condition. As well, they may attempt to avoid repeated calls to check the same constraints.

Many search problems (such as ours) involve not just one solution, but consideration of many, if not all, possible solutions. *Solution Evaluation* heuristics determine whether or not a particular solution is satisfactory according to some domain criteria. In a cooperative, interactive strategy for recognition, an expert might function as a an ordering or selection heuristic to eliminate partial solutions that seem unpromising. Similarly, the number or importance of some set of *soft or preferential* constraints which may have been relaxed during search can serve as a measure of solution quality.

For the purposes of simplicity we have defined the STR-CSP representation at one level of abstraction only. For instance, all variables (features) to be bound to a particular domain-value (situation element) have the same "granularity". However, this clearly is not a necessary condition, and in fact much can be gained through viewing CSPs hierarchically, where a particular variable may refer to a set of lower level features. Both STR-CSP and a subsequent second representation of STR have been viewed in this fashion. We very briefly outline hierarchical STR-CSP in Section 4 and provide pointers to more detailed work in this and closely related area. The second STR approach is similarly outlined in Section 2.3.

2.2.2 Spatial locality and STR features

CSP solution involves both search and controlled constraint application algorithms. While much study has been focused on the construction of efficient algorithms [54, 76, 80, 81, 58, 72] for solution of problems in different real and toy domains represented as constraints, no

[2]These and other intelligent backtracking algorithms are described in detail by Nadel in[54].

"best" generalized approach has been discovered which matches problem properties to relevant solution strategies and heuristics. Consequently, the process of determination of best approach for a particular domain is typically arrived at through a combination of experimentation with different search strategies[3] and heuristics.

Creating a specific algorithm for solving constraint satisfaction problems involves merging the heuristic criteria discussed in the previous section. These criteria together determine how search is controlled, and also how effort is allocated attempting to simplify a particular problem instance in advance of search. The application of properties of spatial locality potentially fits into each of these criteria, and subsequently many possible algorithms can be envisaged.

In [76] STR is described with reference to application of a set of spatial constraints of varying types. The solution of a problem instance involves applications of literally thousands of these constraints in some order. We identify the following choice points or constraint applications as being of primary importance for exploitation of the feature of spatial locality. We shall elaborate on certain of these later in this paper when discussing application.

- **Relative spatial positionings** such as *in-front-of, behind-of, left-of, right-of, in-line-with*, and others, describe spatial properties existing between one object component and all others which satisfy the particular property or set of properties. For instance, given a particular object of type A at position (x,y) with a particular orientation, we may be interested in only objects of type B which are defined as "behind" object A. Here, a locality relation is intended to mean "near and behind object A".

- **Nearness** such as is the case where, given a particular object A at (x,y) we are interested in all objects within some range r of A. The determination of this "near" set efficiently is of prime importance and represents locality in its most commonly envisaged form. Similar questions such as "is B within r of A ?" are also important. It seems that in dealing with a particular object A, a sensible attribution of A in the perception of A by some external entity (such as an STR expert) is the objects "near" (say distance $<$ r) to A.

- **Relative proximity** such as would be the case where we wished to order a set of objects with regards to their relative proximity to some base object, A. Proximity strikes to the heart of what people *typically* mean when they discuss locality in spatial terms. This problem is related to constraint ordering, variable ordering, and also domain value ordering during solution of CSPs. We observe that choosing the closest object(s) may or not make sense in terms of any "ideal" match. For instance, description of an allowable range between template instances such as between 5 and 7 km may not suggest that instances near the median are any more "likely" or "suitable" as instances to fit a particular template. In contrast, elements just marginally **outside**

[3]See [76] for a comprehensive discussion of these strategies in this domain.

a specified range may be imagined to be more "likely" to satisfy the range constraint than another element further outside the range. In such an example an ordering of "nearest" N elements could be used to expand a search for a satisfying match. We note that an expansion according to *nearest neighbors* can be closely approximated as an expanding range of search. This expanding range may be envisaged spatially as ever-expanding circles of radius about a point, with neighbors in each successive circle as the next-nearest neighbor set.

- **Distance calculation** as a subpart of many other calculations, including those above. An efficient method to determine relative distance without excessive, repetitive calculation would be particularly valuable.

- **Intersecting Ranges** are used to determine if two objects known to be in particular areas (region polygons) are possibly at the same position. In essence, do two polygonal regions intersect.

- **Ring Membership** tests are needed in the case where a spatial template is defined including a constraint that says two objects may be between distance r1 and distance r2 apart, but not more or less. A set of objects up to r2 from some base point A is identified, and then those objects closer than r1 to A are removed from the set in an intersection operation.

The use of efficient representations and algorithms towards answering such spatial queries can dramatically reduce the expense of a larger approach to recognizing spatial template instances. In essence, the application of spatial constraints is accommodated through specialized handling of the sighted elements. Of course, independent of spatial constraints other constraint sets must be considered. Consequently, queries are constantly being formed with a conjunctive nature such as "give me all objects within distance d of object A of type t". An indexing system that was capable of *efficiently* retrieving all objects according to a set of pre-defined frequent query combinations, some of which were spatial in nature, would be of great benefit.

In essence, questions about relative spatial positioning are central to the task of spatial template recognition. Efficient solutions to such questions can dramatically affect recognition performance since these functions will be called repeatedly *during* search. Possibilities for improving performance include (1) pre-processing the spatial information in some way *in advance* of search thus resulting in "immediate" responses to anticipated questions, or (2) answering the questions as they are encountered during search and relying on the existence of efficient algorithms for each question type. In fact, (1) and (2) are typically merged in online graphical information systems through the use of some type of spatial indexing strategy in which the spatial information is either pre-compiled into an index (if the information is statically known in advance), or is incrementally added to an index (if the information arrives intermittently). Many different spatial indexing representations exist, although each attempts to maintain a structure of spatially positioned information which readily supports the quick solution of anticipated spatial queries (such as range-queries, etc.).

Ideally, we require an indexing system capable of answering all of our identified important queries efficiently. Unfortunately, we are also constrained by space and algorithm complexity issues. One strategy is invariably best for a particular query type, and less efficient for another. Tradeoffs must be made in selection of an index strategy. Consequently, in order to best exploit the spatial locality properties in the spatial template recognition domain, we identify that the following are the most critical:

- *Ring Membership* as described earlier. Spatial indices are typically good at returning elements from a particular spatial area such as a particular rectangular area. A circle is frequently not so readily supported. However, it is easy to imagine circle-approximation by sets of rectangles as serving the purpose of overall domain reduction quite adequately. We further discuss intersection algorithms and structures later in the paper in order to give an example of efficient object-set intersection.

- *All objects within range r of object A or point (x,y).* Given a typical object density it is easy to imagine a k-nearest neighbor approximation algorithm from such a query. Thus, the spatial index that supports a range query efficiently supports a k-nearest neighbor approximation algorithm in the same order of complexity. If one wishes to sort objects with respect to relative distance from object A or point (x,y), it is easy to approximate this with use of the range and ring membership functions iteratively, slicing the r distance from A and grouping those elements in each slice. In this way we can provide a partial ordering of spatial domains for "most likely" assignment first and subsequent search performance improvement. Similarly, for "soft" or preferential constraints in which one might prefer a nearer object over a more distance one, this can result in "better" solutions being identified first.

  During search, range queries can be directly exploited as a method of reduction of variable domain ranges using the concept of spatial locality as embedded in spatial template representations. Since such domain reductions could conceivably involve a very high number of in-range checks, the usefulness of such a query hinges on the near-instantaneous quality of the computation.

In this section we briefly overviewed one approach to spatial template recognition (constraint-based STR) that makes use of both known and new technology in constraint satisfaction. In addition, we introduced several potential areas in which the combination of efficient spatial query functionality and the concept of spatial locality as embedded in both spatial templates, and situational data could significantly improve the efficiency of these search and propagation algorithms. In the next section we briefly describe another approach to template recognition, and subsequently identify the commonalities of the two approaches with regards to spatial constraint usage and spatial locality application.

2.3 Fuzzy Geometric Relations for STR

In [46], spatial templates are defined as "a model of how objects are approximately positioned and oriented relatively to other objects [in the model]. These relative locations and orientations are specified through constraint on object attributes". The Fuzzy Geometric Relation approach to STR (STR-FGR) shares an important representational base in spatial constraints with the STR-CSP approach. STR-FGR defines a *spatial template* in three parts: (1) a set of k attributes A, with a range or set of allowable values for each attribute, $R(A_i)$, (2) a set of l spatial objects characterized by A, and (3), a set of m constraints (or relations) C including fuzzy constraints[35], over subsets of O. A template is a class where the constraint set is satisfiable through a correspondence between template objects and some subset of actual situation elements or objects. In a satisfying instance, all non-fuzzy constraints are satisfied, and any fuzzy constraint is at least partially satisifed, in that it has a non-zero satisfying value.

We see a very close correspondence between template representations in STR-CSP and STR-FGR. STR-FGR adds an additional structure or building block for spatial templates which is closely related to the abstracted STR-CSP as described in [77], and the embedded decomposition shown in [76][p.91]. This new building block is a *Fuzzy Geometric Relation* (FGR). An FGR is essentially a meta-template that specifies some grouping of template objects and constraints. A "complete" template may now be specified by defining several FGRs, each of which contains "base-level" FGR objects and constraints, defining any constraints between FGRs as required, and additionally adding more "base-level" template objects and constraints amongst FGRs and objects as required. An FGR is treated just like any other template object in that it may belong to relations with other FGRs, or other template objects. This representation thus defines a hierarchical structure of template objects in which some objects belong to "higher-level" groupings known as FGRs which, along with simple objects, are components of the template.

The matching process for STR-FGR involves matching templates to a situation in a "bottom-up"[4] fashion in which all FGRs with "base-level" template object components are matched first. Following identification of root or base FGR matches, an attempt is made to "combine" (essentially a constraint-addition process over the possible combinations of FGR instances) and identify subsequent "higher-level" matches that would constitute a *sufficient* match of the entire template. Exact details of the matching process are described in [59]. Since this methodology utilizes "fuzzy" measures of "fit", a combination of fuzzy fit values to obtain a measure for a set of fuzzy component matches is obtained. The **search** algorithm that lies at the heart of this matching process is the intelligent backtracking algorithm

[4] A related work which performs a similar bottom-up mapping between a hierarchically specified model and an input situation is presented in [81, 82]. This work attempts to exploit both bottom-up template matches and top-down search control during a specific type of plan recognition that exhibits many similarities to the STR problem.

embedded in Quintus Prolog[5].

2.3.1 Spatial locality and FGR features

STR-FGR makes use of the identical set of spatial relations required by STR-CSP for "basic" template elements. In addition, STR-FGR similarly constitutes relations for "grouped element" FGR relationships. Just as STR-CSP can exploit conceptions of spatially restricted element instantiations for a particular spatial template to highly localize the search for successive elements based on previously hypothesized elements, so to can STR-FGR localize search for individual elements and as well, for FGRs based on existing instantiations of other FGR(s).

Both approaches attempt to create a mapping between a spatial template and a set of situation elements. Both approaches may subdivide the spatial template into "parts" (template objects, sets of template objects, or FGRs which are sets of template objects connected in a particular fashion), match some parts to the situation, and restrict the range of the search for subsequent parts according to relations with the matched part. In essence, the spatial questions outstanding are identical between the two approaches, and suggest a similar approach for dealing with both templates (in either representional paradigm), and situational elements.

In the next chapter we describe how the shared properties of the STR-CSP and STR-FGR recognition approaches can be implemented efficiently so as to improve the performance of either approach. The key factors that allow this mutual use of a set of algorithms are: (1) a highly similar representation paradigm for both template and situation elements that will allow sharing of a spatial object index, and (2) algorithms for range returns and spatial manipulations that are independent of a particular solution approach. We note that the input situational information is identical in either case.

[5] In [78] a detailed discussion is given regarding the drawbacks of using a backtracking approach for template recognition and combination for CSPs which is *decoupled* from the problem representation as is the case with Prolog.

Chapter 3

Spatial locality in recognition

In this chapter we examine the concept of spatial locality more closely in terms separated from either of our primary template recognition schemes. We wish to work towards developing a methodology for exploiting the spatial locality in various ways so as to make either solution method more efficient.

A question of interest arises in defining what is meant by spatial locality in spatial template recognition. We are attempting to locate template instantiations in spatial situations. In this context, is locality a property of *templates*, of *situations*, or of the relationship between templates and situations ? We consider each possibility as follows.

- We might say that *locality is a property of templates alone.* In a template, nearer objects are typically related more tightly or more directly than distant objects. In fact we typically group objects hierarchically based on this spatial clustering as discussed in both abstracted STR-CSP and STR-FGR. Distant objects are typically not tightly related, so if we are attempting to identify a template instance part by part, it would seem to be best to focus on one group of tightly-related (local) template objects first, before considering loosely-related (non-local) template objects. Thus, spatial locality within a template is used to define a locus of search, or possibly an ordering of search for template objects. In essence, we are defining a bottom-up search in which constituent "sub-templates" (or FGRs) are identified and thus restrict the focus of search for subsequent template parts. Spatial locality is used then as a justification for decomposition of template and for composition of matching.

- On the other hand, we might say that *locality is a property of situations alone.* The situation objects we view are know to all share a common relationship, however, this relationship may be very high level. We are more interested in easier to identify local relationships evidenced by certain clusters of situational objects. Thus, we assume that (typically) some situation object is more likely to share a tight relation with another object "nearby" spatially.

- We defined locality for templates in terms of a qualifier between elements named "near". It is clear that "near" is a property specific to a particular template instance and may have widely varying definitions over a set of templates. If we wish to localize the nature of recognition to one part of a template then this template part (feature, FGR) is mapped to a situation in the same way as the entire template. Once one (or more) parts of some template (or partial template) is mapped to a situation (uniquely or multiply), the locality of a particular template as defined in terms of the mapped template feature may be used to restrict the range of search for subsequent template features within the situation. The situational features spatially local to this "starting point" may be extracted according to the template locality definition. The specific template locality definition is mapped into a situation and the focus (of search) is restricted to situation elements according to the template locality property about the "starting point". Thus, locality is *defined* in a **template**, and *applied* in a **situation**. The use of locality is of interest only in the interaction between these two. We need then to consider both definition and application as separate issues, keeping the interaction as our primary concern.

We have seen that locality has a definitional and an applicational nature. We shall examine each in turn in the subsequent chapters 4 and 5. In the remainder of this chapter we provide a high level example of the use of locality in identifying template instances. From this example we motivate two simple ways of extending template locality properties into situations. We identify several specific spatial functions entailed by the locality realization functions which must be present in any recognition system.

3.0.2 An example of locality use

In a spatial setting, macro-objects may be seen as a related set of sub-objects, or parts. Each sub-object has spatial location, orientation, and other properties such as size, orientation, color, etc. In the problem of spatial template recognition, we are given a set specification for all elements that may be classified as macro-objects. This specification or template consists of a set of sub-objects, each constrained in terms of allowable attribute values, and a set of constraints which constrain the sub-objects according to related attribute values, in particular, relative spatial location and orientation.

Consider the following simple example. Suppose that the sighting of a tank and an infantry company advancing in the same direction, where the inter-company distance is less than 10 km, is used by intelligence analysts as an indication of a possible tank battalion in advance. We may define a spatial template for the macro-object *tank battalion in advance* which contains two sub-objects, object A and object B. Object A is constrained to be of type tank, and of size company, while object B is constrained to by of type infantry and of size company. In addition, we add a constraint between objects A and B that they be not further apart than 10km. Imagine that the analyst is aware that infantry companies are much more common in the field than tank companies. Given a scenario in which all sightings of military units are drawn on a two dimensional map display, a typical method for locating instances of this template might be to identify instance of the rarer tank-type company, and then for each instance, attempt to locate infantry company instances of appropriate orientation within 10 km. The analyst is essentially defining a spatially restricted range of search for each candidate template instance.

For more complex templates that contain many more objects and constraints, locality can be defined in terms of a certain "key" object or set of objects, or perhaps in terms of some measure derived from the entire defining object set, such as "maximum template radius". In the case of a single "key" or trigger object, we proceed much as in the simple example described, identifying instances of the key and searching for subsequent siblings within the specified range of the key. In the case of multiple key objects, all key instances are first identified, and then the range for additional siblings is reduced according to the particular key combinations. It is easy to imagine a progressive reduction of range according to a pre-determined order of object "importance" and a corresponding range application for each subsequent object instance selection combined with the previous object instance identifications. In a similar fashion, a single range constraint (such as maximum radius) may be applied after a single object instance is determined, after each object instance identification, or perhaps after only certain object instance identification.

In the following brief sections, we outline examples of each of these different applications of locality during template recognition. We use as an example a spatial template with three sub-objects detailed in Figure 3.1. In this template there are 3 sub-objects, each of a different type (depicted as shading), each with an orientation labeled via an arrow. There are constraints on the separation and relative position between object 1 and object 2, and also between object 2 and object 3.

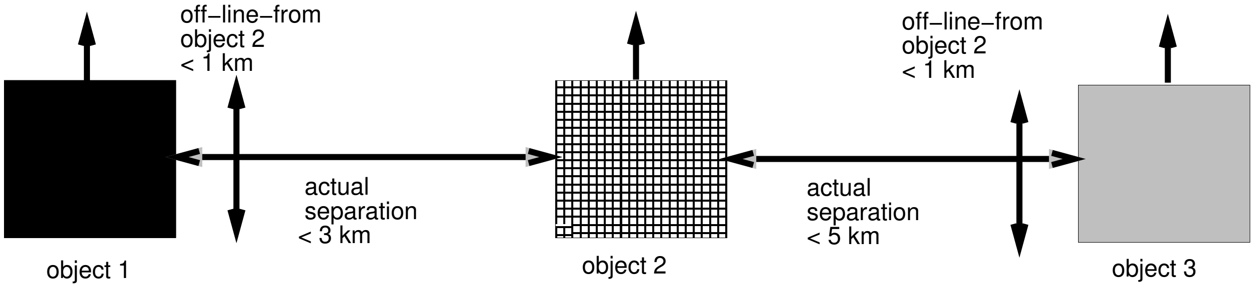

Figure 3.1: A simple template example.

In choosing to exploit properties of spatial relations to help us identify spatial templates, we have made the assumption multiple-parted objects located in space tend to be "localized" rather than "distributed". Spatial locality means that when attempting to identify a whole object from the recognition of the existence of its parts, that it should be possible to restrict the area of "interest" in the search around the area in which some subset of sub-objects have been identified.

In this work we intend to outline methods of exploiting this locality in a fashion that is specific to the problem of template recognition (outlined subsequent sections), but which are independent of either the FGR template identification method of Lapointe and Proulx[46, 59] or the constraint reductionistic approach of Woods[77, 76]. Essentially, the application of spatial locality may be seen as either a method for preprocessing the spatial information so as to conveniently reduce the size of the set of candidate instances that may be part of a particular whole object.

3.1 Single key locality reduction

For the purpose of this example, we consider that template sub-object 2 is the single key object, and consequently we identify instances of this sub-object first. In our example situation shown in Figure 3.2, there are three instances of sub-object 2, labeled 10, 14, and 5. We consider 14 in this example. Our example template has three sub-objects. Given an instance of one sub-object we now wish to limit the spatial range of our search for candidate matches for the remaining template sub-object slots. Some sub-object 1 shaded black may be at most 3 km distant from the instance of sub-object 2, instance 14. Simlarly, some sub-object 3 shaded light may be at most 5 km distance from instance 14. Given a function that could return all instances within a given range of some location[1] it would be possible, with two applications of the range function, to determine a localized range of instance candidates for template sub-object 1 and 3. Note that a *single* application of the range function for the *larger* (less restrictive) of the two inter-object spatial constraints would result in a localized range for each of object 1 and object 2, however the range of object 1 would slightly exceed that implied by the actual constraint. Figure 3.2 shows the range of 5 km around instance 14. Instances outside the range would be excluded by the range function, while instances inside 5 km would be returned as the potential instance candidates for sub-objects 1 and 3. In this example, 11 instances are excluded, however, a more typical example would have the entire range of approximately 100 km, with as many as several hundred instances, and consequently the reduction would be proportionately higher. This reduction is an example of achieving *spatial locality* **with respect to location** of a single object instance.

Figure 3.3 shows the "localized" range for any subsequent instances matches for this template application. The second spatial constraint detailed in our example is "off-line" distance. If one sub-object instance has a particular location and orientation, then "off-line" distance is the distance another sub-object instance is from a line perpendicular to the angle of orientation of the first object. In Figure 3.3 an off-line distance of 1 km is shown with respect to object 2 (both to the left of and to the right of, object 2). This reduction is

[1] We identify an interest in a function to return all instances within a defined radius of a given point.

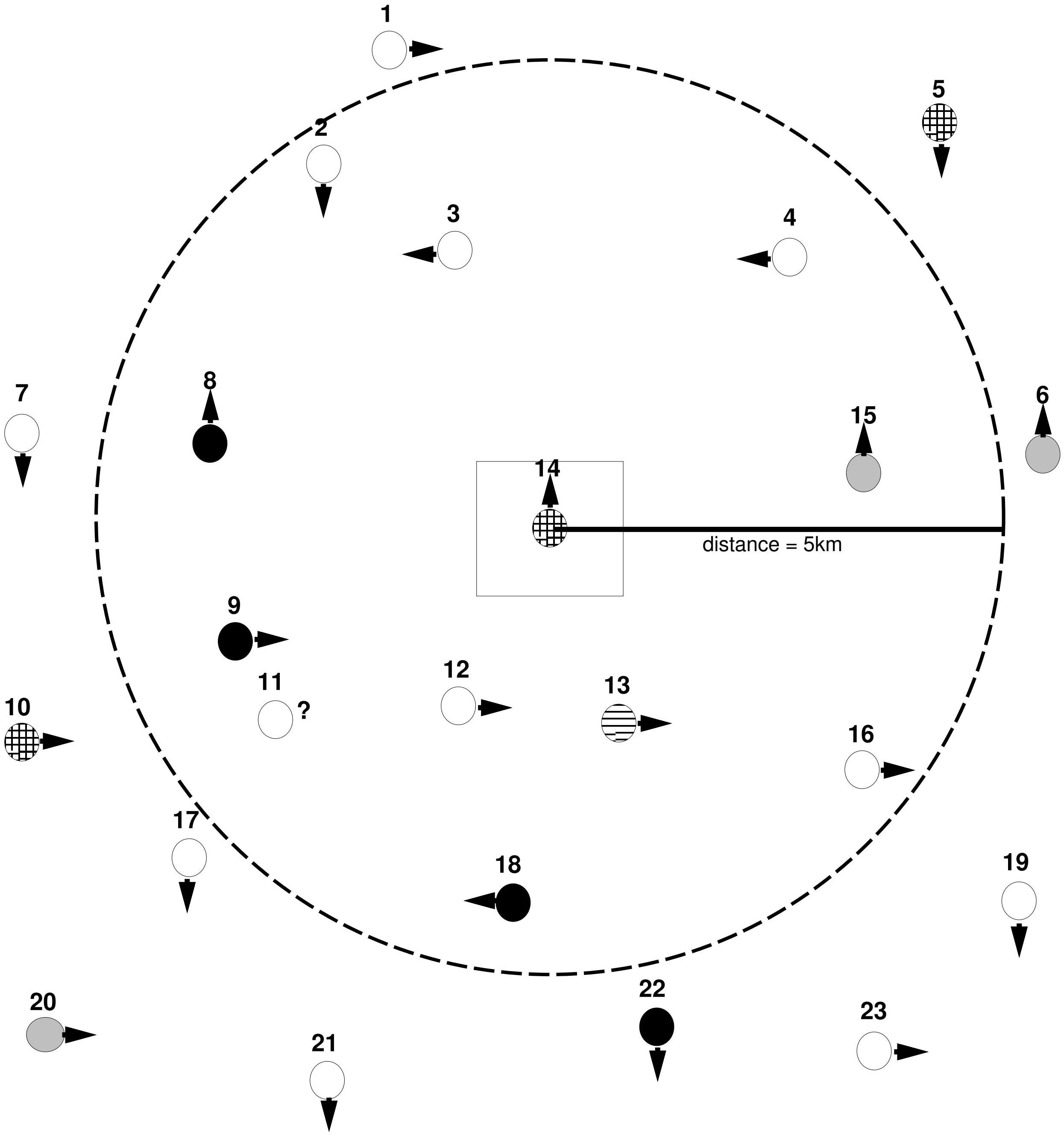

Figure 3.2: A locality of 5 km around instance 14.

an example of achieving *spatial locality* **with respect to orientation** to a single object instance.

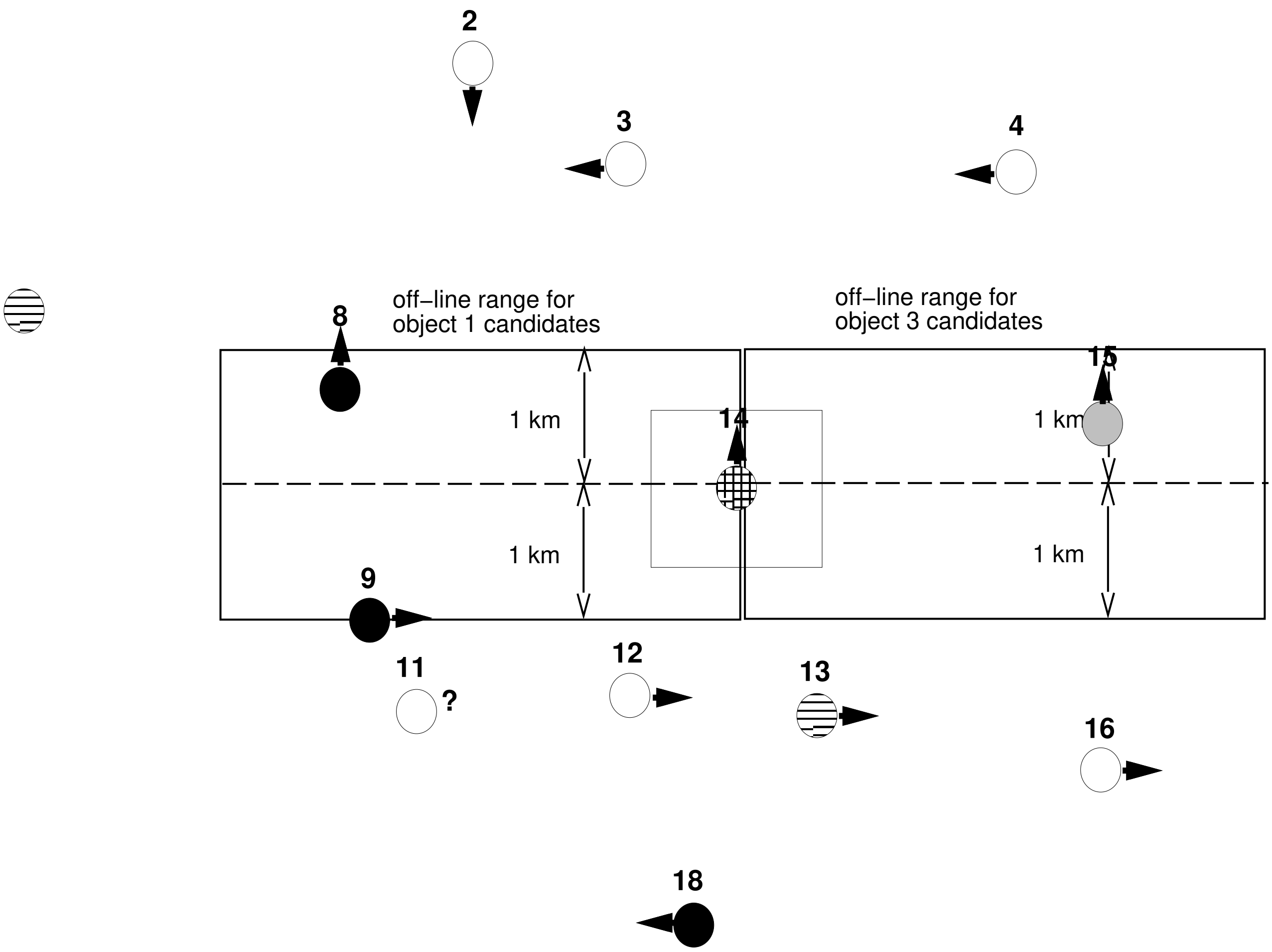

Figure 3.3: Off-line range of 1 km for instance 14.

Single-key matching has the advantage of simplicity. A single application of a range function results in a one-time reduction of the number of candidates for each template sub-object. Using more accurate reductions for the range of search for different subsequent instances (i.e. 3 km for object 1 with respect to object 2 and 5 km for object 3 with respect to object 2) requires more range function applications and subsequently more work. The determination of exactly how to formulate the spatial reduction would appear to be best determined through experimentation in a particular domain and will depend on how many potential instance candidates can be removed with each application. Single-key matching for orientation *following* distance reduction may or may not be advantageous, once again according to the relative cost vs restrictiveness of the constraint in a particular domain, or even particular problem instance.

The primary disadvantage of single-key locality applications are that reduction with a single key may entail a substantial cost with little payback in terms of candidate reduction. In subsequent sections of this paper we will examine the cost of performing these locality operations.

The determination of a particular range of "locality" according to a single instance fits well with the intuitive understanding of "local". However, it is just as sensible to talk about locality in terms of "local to a set or configuration of instances", consequently we now examine locality determination in terms of multiple instances.

3.2 Multiple key locality reduction

Multiple-key matching differs from single-key in that a particular application is based on two or more identified instances combined with a particular spatial constraint. For instance, in our example of Figure 3.1, given instances of object 1 and also of object 3, we might define a multiple key spatial locality as shown in Figure 3.4. This example might be seen as an intersection[2] of the result of two single-key reductions, or possibly as the definition of an area (the range overlap) which is then searched for object instances[3].

Combining spatial relations in advance of attempting to reduce the search range can be highly beneficial. For example, consider the case of combining the distance locality and the orientation locality for these multiple keys. The combination of the 3 km range from object 1 instance and the 5 km range from object 3 instance with the off-line range of 1 km for each results in a pair of rectangles localizing the search for an object 2 instance. The position of these rectangles is shown in Figure 3.5. In fact, the determination of the exact location of each rectangle is trivial, and subsequently the resulting candidate set could be determined either by two operations to determine instances in each rectangle combined with an intersection operation, or more profitably, by computing the overlapping rectangle directly from coordinates and then identifying the object instances within this rectangle[4].

A different type of multiple key match can arise when a constraint is defined over two or more instances rather than combining constraints. For example, a particular constraint might be that given an instance of object 1 and an instance of object 3, an object 2 instance should be found within a particular range of the *midpoint* of instance 1 and instance 3.

We see from these examples that the search for a particular instance given one or more instances and some spatial constraints can be very restrictive and result in a minimal number of candidates. The challenge will be to implement this locality determination efficiently. In

[2]We identify an interest in a function that will efficiently return set-intersection.

[3]We identify an interest in functions to (1) compute a polygonal approximation to a range (circle) intersection, and (2) return the set of instances in such a region.

[4]We identify an interest in a function to return object instances within an arbitrary rectangle.

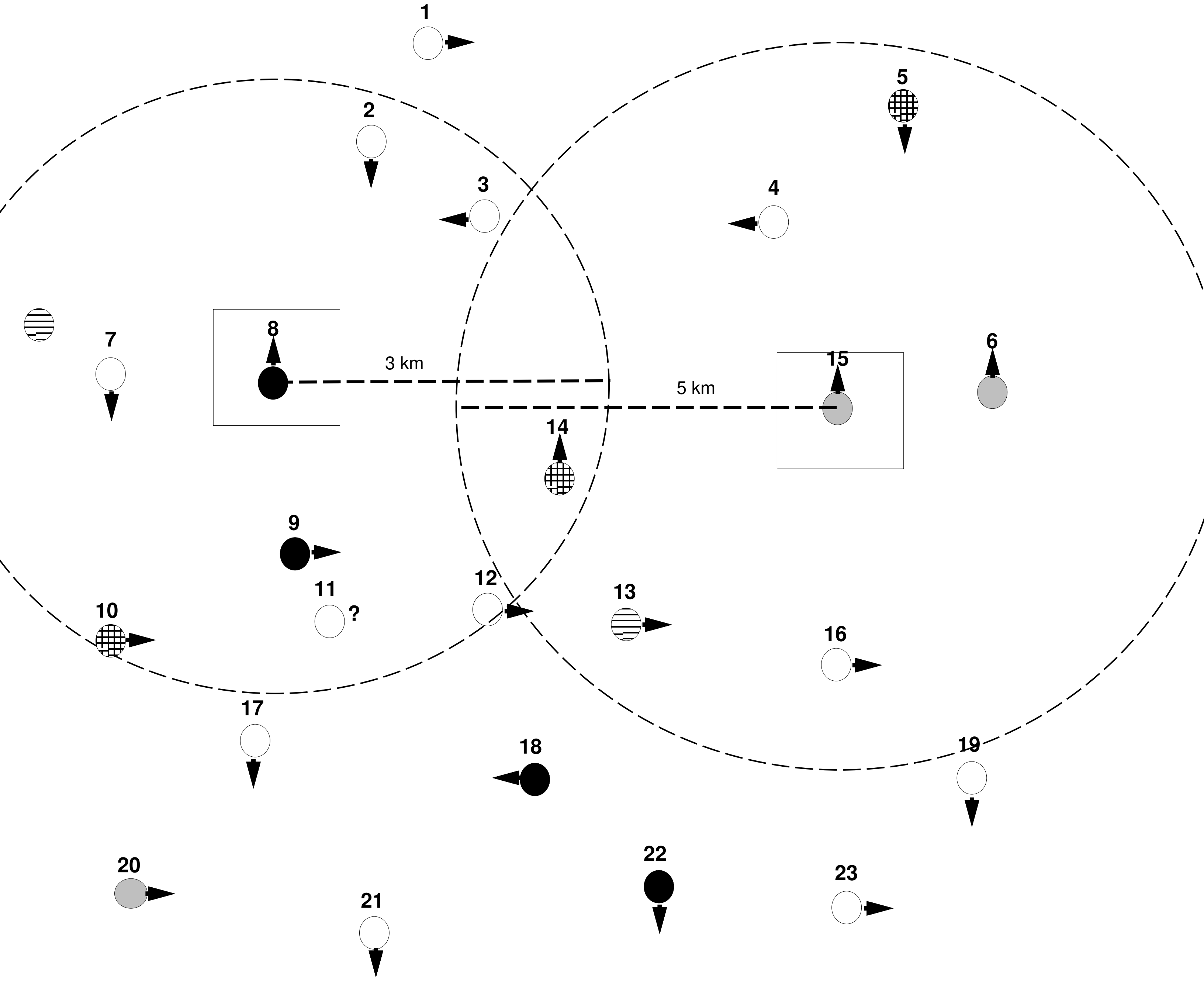

Figure 3.4: Overlapping radii for instances 8 and 15.

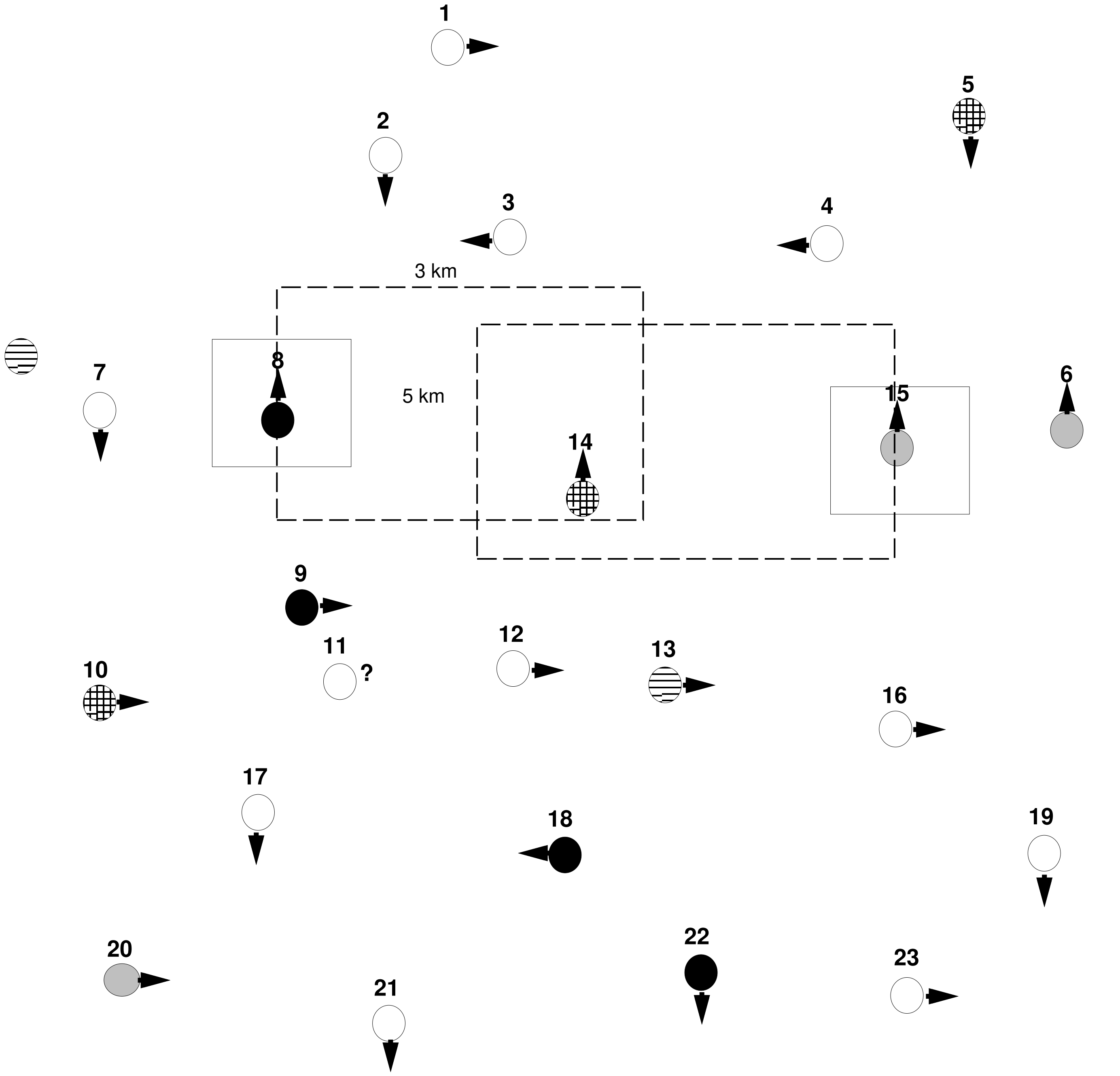

Figure 3.5: Combining radius and off-line range constraints.

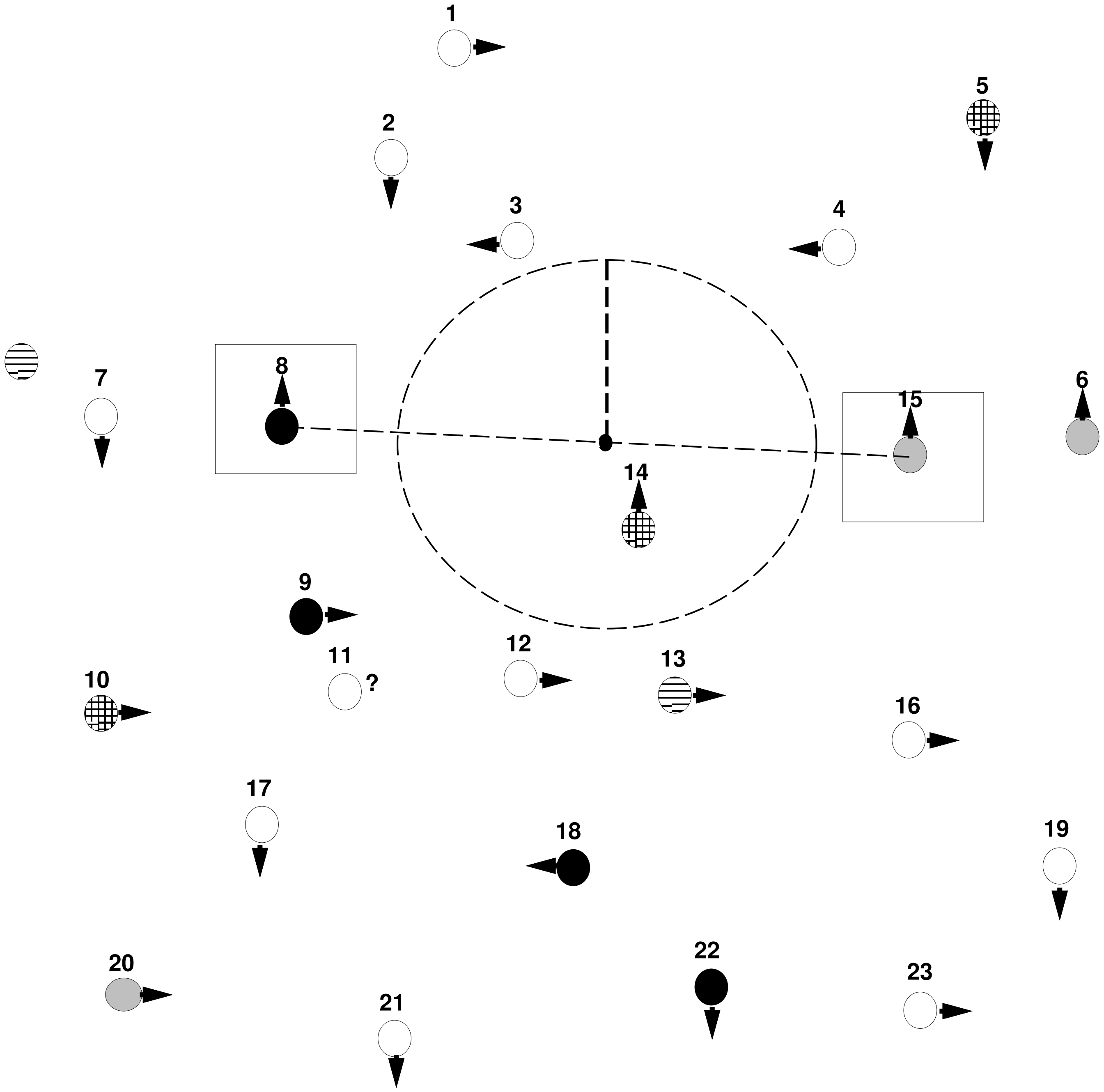

Figure 3.6: Midpoint locality for instances 8 and 15.

a later section we examine each of the identified functions individually.

3.3 Spatial operations in locality use

In order to exploit these concepts of spatial relations is is necessary to first identify what exactly we are given as input, and also describe the algorithms that will be able to return instances sets according to our spatial reasoning needs.

3.3.1 Situational input

The input format is simple. We are provided with spatially positioned "situation elements" which have some number of attributes, any of which may be *known precisely*, *known in a range*, or *unknown*. Spatial location may be viewed as such an attribute, and consequently the location may only be known within some (arbitrary ?) area or region rather than a precise point. For the purposes of this work we have assumed a known spatial location at a point rather than in an area.

3.3.2 Situational outputs

We have identified several spatial functions crucial to the task of restricting the locality of template object instance searches. This set, described further in section 5.2 includes:

- Point and radius instance return.
- Instance set intersection.
- Polygonal approximation (or exact) to area of two circle intersections.
- Two circle intersection instance return.
- Arbitrary rectangle instance return.

Chapter 4

Locality representation : hierarchy in templates

In this chapter we bring together earlier references in this paper to potential hierarchical template representations, portions of our earlier work on hierarchical templates and solution methodologies with locality exploitation.

The STR-FGR approach[46] embeds hierarchy as FGR template components. The FGR template elements abstracted as potential separate template parts are expert defined. It is important to note the difference between what in [77] is referred to as "partition" and "abstraction". Partition describes the breaking of a single-level representation into different parts, each *at the same level*. In essence, each *set* forms a sub-template and the *inter-sub-template* constraints serve to re-combine the sets into a whole larger template. Abstraction involves rather the removal of detail throughout a template thus leaving the *essential details at one level*, and the refinement method from more-abstract to more-specific is the (possibly systematic) re-addition of the details. As alluded to in [77], the combination of "partition" and "abstraction" suggests a near-boundless opportunity for problem decomposition in spatial template definition. FGRs are most similar to this concept of partition, with the important extension that FGRs are "typical" building blocks for many templates. Similarly, a template may be "abstracted" using FGRs at a particular level of detail where one imagines a single-level template with circle drawn around sections that will represent FGRs. These circles, if viewed as a single entity at the level at which they are drawn rather than the complex set which they encompass, may be seen as explicit identification of an abstraction of one *part* of a template.

The STR-CSP approach[77, 76] has multiple levels of detail implied through viewing the problem as a Partial CSP[21, 84]. A Partial CSP defines several problem layers in which certain constraints are relaxed between layers as defined by a domain expert. These constraints may be specifically retained as a description of representation change [29, 30, 31] and applied systematically in order to reduce the set of possible solutions. The key to such an approach is that an abstract (relaxed) template allows many solutions while requiring fewer constraint checks. The addition of subsequent constraints (or detail) limits the set of solutions. In [76], this iterative refinement approach is integrated into an algorithm for expert interaction. Other work [48] specifically describes an algorithm for simplifying problems by exploiting the hierarchical nature of a restricted set of problems.

So then partition and hierarchy have been shown to be useful tools in both representing and solving template problems. How then is this decomposition of problems related to spatial locality ?

4.1 Combining decomposition and locality

As stated earlier, locality restricts the range of search based on partial solutions. To do this effectively and efficiently requires a set of tools and structures that allow speedy responses to spatial queries. Decomposition fits well with expert conceptions of templates. Important problem parts are "keys" and frequently serve as indicators of the possible existence of a template in situation. For instance, experts in other template-recognition domains frequently locate starting points for "search" according to quick scans of a source for easily applicable keys learned through experience. Consider how you might search a text for reference to a particular "abstract" concept, and how obviously related words cause you to pause in your search and more closely examine the near-by (spatially local) text for additional evidence or confirmation that the reference is pertinent. In [80] these types of keys are exploited in some detail for very large program understanding problems.

Decomposition by its nature represents an expert's vision of problem locality, including but not necessarily limited to spatial locality. One imagines decomposing familiar concepts such as a bed as partitions. For example a bed consists of the partitions "frame", "headboard","mattress", "bed covers", and perhaps, "pillows". Perhaps it may abstract easily as well. Consider that we have all seen many *kinds* of beds - bunk beds, water beds, feather beds, canopy beds, and the like. The details of any one of these kinds of beds are certainly not shared (where is the water bag in a feather bed ?), however the "key" or primary features emerge easily - for example, "mattress", "bed frame". These key features refine to the more specific through representation changes such as addition of constraints. For example, an instance of the abstract concept "bed" may be refined so that the bed's mattress is constrained to contain water, thus giving the more specific concept "water-bed". Similarly, the bed's mattress may be constrained to contain feathers, thus giving the concept "feather-bed".

If one is attempting to piece together a single bed "image" from a perceived set of many uncategorized, possibly unrelated, objects in a room, beds provide a very good example of inter-partition spatial locality, as well as abstracted spatial locality. In the first case, the "bedness" of a particular instance requires that a "mattress" and a "frame" connected through a single unique bed relation be necessarily spatially close. In fact, if one were to identify a pair of mattresses and a pair of bed frames in a particular scene (such as viewing a double hotel room, for instance) it would make good sense to try first to relate the pairs of mattresses and frames nearest each other. In the second (abstracted) case a similar argument holds. If a mattress were a "key" indicator of the presence of a bed, we restrict the search for other same-bed relation objects (frame, pillows, etc.) local to the identified mattress instance.

None of this description differs significantly from our earlier spatial examples, however, we now have a very natural way of encoding spatial locality in a template representation containing both partition and abstraction. Abstraction encodes the "key" features of a model. Hierarchies are identified either by and expert, or automatically from a single-level description as described by Knoblock[38, 39, 36, 40, 37]. Similarly, locality features are expert defined or extracted from the template according to some, possibly domain-specific, approach. Given a partial mapping to a template partition or abstraction, locality restricts the range of additional related partition instances or template detail extensions.

We have briefly discussed some of the issues of problem representation in the STR domain, in particular, problem decomposition. The connection between problem solution and decomposition were seen to provide for the expression of partial solutions, allow for the interactive solution of these problems, and also could be used (as in the case for partial or relaxed CSPs) to provide "good" or partial solutions quickly when "best" or complete solutions cannot be found in reasonable time. Greater detail may be found in [77], where constraint representations were discussed as *single-level* and *embedded* partions [pp.90-91], as abstractions [pp.91-96], as both partitions and abstractions [p.96], and as partitions with *spatial cohesion* [p.98-99]. Problem partitions and abstract representations can be tied to locality algorithmic heuristics in at least the following ways:

- *Partition instance combination* - identification of a particular partition instance can be used to limit search for other subsequent replaced partition instances locally. Constraint on inter-partition combination may directly specify where to *expect* subsequent partitions.

- *Abstraction expansion restriction* - once an abstract solution is proposed based on certain "key" template portions, the subsequent portions can be searched for in a very directed way according to the *representation change* specifications regarding spatial constraints. Some subset of the "added" template portions will be connected to the abstract portions via spatial constraints and these can be used to directly guide search for candidates.

In earlier work with a simplistic spatial locality application referred to as "spatial cohesion"[77, 76] our experimental template recognizer (MOTORS) included the ability to focus on a certain pre-determined range of candidates given a particular sub-object instantiation. This "focus" was implemented by a filtering of remaining template objects domains or candidates. In this way, all candidates outside a pre-specified template range are discarded without search. While this approach was "sharp" in the sense that "unlikely" solutions just outside the range of specification in a template are missed entirely, the configuration of these ranges is actually encoded as part of then domain template knowledge rather than a specific heuristic about all templates. Relaxation techniques can iteratively increase this range during some expanding search algorithm so that such solutions are not entirely missed, only delayed. The application of this "spatial cohesion" heuristic displayed a dramatic improvement by reducing the number of constraint checks required and the number of search states encountered during search.

It is important to note that while reducing the number of constraint checks, the locality enforcement itself is computationally expensive. If the combined work of applying some spatial check against a large set of candidates is not profitable, that is, if few removals are made for a large computational cost, then an overall loss in performance may be incurred.

4.2 Using locality

The way we have conceptualized the use of locality in this paper is certainly not the only possible interpretation. For example, why don't we view locality more in terms of each particular object - each object could potentially "know" who it's n, $n+1$, etc. neighbors are, and this information could then be used when fitting sets of objects together as template instances. Approaches to solving the k-nearest neighbor problem have been studied in earlier graph theoretic work, such as [23, 1].

We have divided locality discussion into two areas: (1) representation in templates, and (2) application of templates into situations. We interpret preprocessing as the process of maintaining a particular type of spatial structure that is amenable to quick spatial queries of the type(s) needed to apply spatial functions as part of the process of limiting search by locality. K-nearest neighbor information can thus be efficiently extracted from an index through combination of range queries. The limiting factor identified in our particular problem is the **dynamic** nature of the situation information - i.e. elements are continually arriving, and consequently the situation is changing continuously in structure.

K-nearest neighbor approaches are typically applied to problems that embody static graphs in which tables of neighbor relations may be compiled in advance for subsequent questions about the same graph. A continuously updated index can approximate K-neighbor behaviour with k-range, k+1-range, K+2-range queries in ever increasing circles of radius about a point or element.

In this work we have focused on empirical improvement of the heuristic adequacy of solutions to STR. In other recent work[80] in another domain, we have presented empirical results that strongly suggest that a well-constrained CSP template representation can be identified in large complicated situations using known heuristic CSP solution algorithms (combining pre-processing and search) in quadratic time complexity w.r.t. problem size. In fact, if the problem complexity is viewed as a quadratic function $ax^2 + bx + c$, the constants a and b are *very* small, thus resulting in solutions for problems of significant size in near-real time. These results are a direct result of the "tightness" of spatial constraints in a template.

While we have as yet no theoretical results in that domain (or STR for that matter), our future work will include an interest in isolating the types (and degree) of constrainedness that will allow us to identify complexity bounds for certain classes of problem instances.

In moving towards solutions that offer heuristic adequacy with regards to our need for quick, expert-interactive solutions, the inclusion of a spatial index that allows for *very quick* application of spatial range checks will impact the overall completion time signifcantly. Considering the use of these constraint during search or during a constraint propagation-type of algorithm, we are talking about a method of reducing an entire (domain) set of K elements to some set of $T << K$ elements while requiring only 1 operation of small cost C[1]. Since we may eliminate *many* candidates with a single operation, the search space is pruned significantly at a very early stage, and the subsequent improvement in performance could easily be imagined to exceed that of a constant factor that one would expect from a simple "speeding-up" of a particular functionality.

As a simple example, consider a situation of matching template instances in a situation of 1000 elements to a template with 5 slots (features) each of a distinct type. If each of the 1000 elements were evenly distributed as 1 of 4 possible types then each slot would have an initial assignment (domain) set of 250 elements. One of the the slots is selected, and a template "extent"[2] constraint is applied, imposing a maximal inter-template-element distance for all variable-element selections. If this application is made to each of the 250 elements as an *index lookup function* of very low cost relative to the area included and the rate of success, the subsequent allowable (spatial) range of the related slot is dramatically reduced. It would be possible, if the range restriction were 10% of the total area and given a uniform element distribution, to reduce the related slot range to 25 elements. If we repeat this for each of the 25 elements with respect to a third slot, would result in about 3 candidates for that slot. The search space is severly limited in depth by the restrictiveness of overlapping spatial constraints. Further, the branching factor decreases dramatically with each subsequent level. This entire interpretation of the behaviour of search *hinges* on cheap application of spatial range-type queries. Thus, the extraction of locality information from a spatial situation according to template extent information (note that extent is merely one possible extracted constraint) should be of extreme *practical* significance in both the STR and related domains.

It is crucial to note that the overhead that is necessary in order to obtain such *quick response to spatial queries* is incurred in the up-front (pre) processing of cataloging spatial information as it arrives. The cost of spatial structure maintenance and overhead is distributed in time. The spatial locality knowledge encoded in specific indexing structures cost some memory, but presents this knowledge in a way that can be easily and cheaply retrieved at processing or search time. We have thus exploited spatial locality by *compiling* spatial knowledge about situations, *extracting* spatial template characteristics from templates (say

[1]The effectiveness of such an operation depends on the object density and the restrictiveness of the constraint or range limitation.

[2]Note that *extent* might be defined as the maximal inter-object distance in a template, thus no two template object instances may be further apart than the extent value.

features such as extent), and applying the two in highly repetitive search situations (common to both STR approaches). The end achievement of this compilation is a projected dramatic performance increases in recognition over strategies in which spatial knowledge is manipulated "as needed" during search without the benefit of specially constructed structures.

4.3 Feature extraction

The process of *feature extraction* of "key" or "important" template aspects is integral to creating abstracted template representations that correspond to expert perceptions of *partial* matches, and of increasing confidence is a particular match according to the satisfaction of additional problem constraints.

We are interested in both feature extraction from templates as part of the process of abstraction construction, and also from situations in the case of specific signifiers or (rare) elements that may typically alert experts to certain important situations. We examine each of these briefly in turn.

4.3.1 Template features

Feature extraction from template is essentially a reduction of the template object (variable or FGR) and structural glue (constraints or relationships) so as to identify "crucial" relationships that may serve as first-cut indicators of a possible template. Many useful spatial features are explicitly encoded in templates[3] (object sets restricted by relative distance, orientation, alignment about axes, etc.) and may be simply "read" from the representation directly. In addition, other features may be extracted via simple rules or geometric reasoning automatically if desired. The process of extraction is beyond the scope of this paper, however, a simple example might be the case in which a template has three slots/variables A, B, C such that there exist certain spatial and orientational constraints between A and B, and also between B and C. It is reasonable to assume that either *during search* or during representation extension, one might wish to determine some set of the implicit A and C spatial relations. Two simple examples of such extractable spatial features might include:

- *Extent*, as discussed earlier in this paper, is one "derived" spatial property of a template that can be effectively exploited in cases where a template extends over a range signifantly less than that of the entire situation. In fact, "extent" might be seen as the primary constraint of interest in applying spatial locality during search. As described earlier, extent can be local within a template (simple maximal inter variable/slot range) and thus read directly from the representation, or global to a template, and either computed in advance of any use of a template, or even specified directly in the representation (as is the case in earlier experiments discussed in [77]).

[3]For a detailed discussion of these the reader is referred to the baseline works of [77] and [46].

- *Multi-element alignment*, where a sub-set of element instances are specified to appear such that element A and element B are "in-line" with respect to a particular orientation of A and position of B; and, element B and element C are constrained similarly with regards to B's orientation and C's position. In such a case one may wish to (say during search) identify instances of C given only a position of A and no determined B positions. In such a case one might either ignore the A and C spatial relations implied from the transitivity of these geometric relations, or, compute the relative A to C restrictions. Such a computation could (and likely should) be made in advance of any search as a template extension (perhaps automatic), however, if such a computation were indicated according to some heuristic desire during search, this information might be *added* to a template representation for future use (if applicable).

4.3.2 Situational features

In this section we discuss in greater detail how certain spatial defined features embedded within situational information may be "pre-processed" in order to reduce the expense of extracting this information from scratch during search. We attempt to address the question of how spatial locality is specifically represented via preprocessing of situational information. We have previously defined this knowledge as implicitly embedded in spatial indices which support rapid access of typical queries such as range checking, however, how else might we encode higher level knowledge of varying types ? We examine each of these encodings in turn.

4.3.3 Spatial indices as feature encoding

There are two obvious possibilities for maintaining the spatial data in an index structure. Either the data is maintained consistently within some structure (preprocessing in which the cost is distributed over time), or it is processed all at once when a spatial query or template look-up is requested (pay all of the processing cost at query time). We know then that the primary issues we must address (and which we do address, later in this paper) are:

- In spatial indexing with continuous update of our indexing structure as elements arrive: what is the cost of insertions and deletions ?
- In spatial indexing with one-time index construction before search, what is the cost in time and space of building such a structure ?

In contrasting these options we wish to determine the relative cost/benefit of "distributing" the work of processing of a new situation element arrival ? It appears obvious that in such a system where processing time is of the essence when a query is asked that moving into arrival processing what can be moved is an obvious time saver, given that the space is not a problem. It would seem preferable to dynamically allocate index space if possible (without horrendous balancing overhead) if possible.

Spatial indices *encode* locality in that spatially "near" elements are stored nearby (in terms of ease of accessibility) in specialized structures. This encoding supports the identification of range searches in time of order logarithmic to the size of the overall sample, often plus a constant factor relating to the success rate of the search.

4.3.4 Situational feature identification

One might extract "certain" template feature knowledge such as distances of related **pairs** of template slots of particular attribution, and attempt to identify these "complex objects" or "key-combination" instances in a particular situation. Each satisfying instance might be added to the situation as an *annotation*, essentially adding a piece of high(er) level structure to the base-level situational information. These macro-instances could consequently be built into even higher level structures. As an example, a new feature X is added to a situation indicating the presence of an "A" instance within some distance range (say d) of a "B" instance. This new feature X could then be exploited during search by avoiding a range check for a B within d of any A, and also by specifying various templates, including the X type as an integral template aspect.

The main advantage of this vision of bottom-up processing of situations in advance of particular template matching is:

- This approach distributes "costs" in time similar to general spatial index construction. Each new situational X instance insertion requires a specific amount of search. If this is done in-advance of search and annotated on the situation, this search will occur only once, and subsequent searches will not repeat this matching. If the occurrence of such a relation is rare in general (as one might expect of a "key" feature type) a great deal of work will potentially be expended searching for instances, however, the few successes indicate "key" positions for search starting points, thus reducing the breadth of the crucial **top** of a search tree.

There are several problems with this situational annotation vision:

- It presupposes that the type(s) of search are known at situational element insertion/acquisition time so that the maintenance of the "special" features may be undertaken. Clearly this will not be the case in general (see the third item, below).

- To be especially effective it would be necessary to analyze *many* templates in order to identify the crucial/key relations that should be "watched for" during insertion and "highlighted" with situational annotation to aid future search. I believe that this is an exceptionally promising area for future research of a slightly different nature, i.e. focusing on recognizing instances of *many* template types rather than instances of one type.

- Situational processing or annotation of this type *at the time of, but in advance of* search does not seem different from locality application during search, except when one considers multiple template types where it will be possible to *avoid repetitive* application in the same spatial constraints in searches for instances of templates of different types. Once again - a ripe field for future extensions, but not for single template instance identification.

4.4 Overview

In this paper we have attempted to clarify the close connection between spatial feature extraction from template representations and some earlier work done in exploiting problem decompositions and abstractions, one example of which is the use of FGR objects in STR-FGR representations, and another is the PCSP[21] strategy supported in STR-CSP. In addition, we believe that there is room for a great deal of extension of our ideas to include the work of Robert Holte [29, 30, 31] in explicitly representing and applying *representation change*, and also that of Craig Knoblock[38, 39, 36, 40, 37] in automatic template feature extraction as part of the process of automatically generating abstraction hierarchies for problem solving. It is not only templates which may be abstracted, but also situations and the mapping process. These areas provide good clues for future extension of the heuristics for controlling the recognition process.

In the next chapter we discuss the actual methods of efficiently extending the locality represented in a template specification to the situation, both for partition and for abstraction.

Chapter 5

Exploiting locality

5.1 Locality and search

This section describes how the situational representations of locality may be combined with abstracted template representations embodying locality during search.

5.1.1 A simple constraint ordering for locality

It is possible to design an entire algorithm based on the concept of spatial locality as embodied in the template representation itself. We explain this algorithm as follows. A template consists of a set of object components and a set of constraints among these components. Locating a solution involves finding all (or some) matches to a situation that satisfy these constraints. Now, many of these constraints are restrictive with respect to spatial relations such as distance. In general it is difficult to determine which constraints are the most restrictive. An excellent heuristic during CSP search is to apply a known computationally cheap and highly restrictive constraint so as to dramatically reduce the remaining candidate set of a particular component. Since "short" distance constraints (i.e. up to r distance from a point) return a set of corresponding objects that will be smaller than a "long" constraint (i.e. up to 2r distant from that point), a smart approach would be to apply short constraints before long ones. To solve a CSP then, we order the constraint set by shortest distance first. Next, we identify an unsolved variable or component from the shortest constraint and instantiate it. Apply spatial locality by discarding all candidates outside the constraint range of the instantiated object, and repeat for the next shortest constraint. In this way we localize our search immediately. A similar approach can be taken for propagation techniques in which only very short distance constraint might be propagated in an attempt to dramatically reduce the problem size before search. Once again, such a method would benefit from a spatial index which would allow us to identify candidates without explicitly checking all candidates for range.

5.1.2 Spatial reasoning and search

Both STR-FGR and STR-CSP use *search* in the production of potential template matches. Spatial functions are called to restrict domain ranges during search. Questions about which the order in which spatial functions are applied, or the frequency of their use with respect to pre-search problem reduction, or during-search application have not been examined at this time. Other similar systems sometimes attempt to avoid repetitive function call evaluations, however a storage system is then required for applied functions, and this is one potential area of possible future interest. However, caching repetitive function call results can add an unecessary level of complexity to function calls, and also potentially cause storage problems. In addition, it is not necessarily obvious which function call results should or should not be cached.

5.1.3 Experimentation

This paper does not describe any implementation of the methods described. We would, in the future, like to empirically demonstrate the effectiveness of applying spatial locality to a search problem of this type, and to provide subsequent analysis which can lead to a positive recommendation for inclusion of template recognition in the demonstrator project know as MOST. In moving in this direction, the primary goal of this work has been to justify and describe how spatial locality can be readily applied to template recognition through use of spatial indexing.

We observe immediately that a spatial index system is clearly going to be an improvement over no such system, the primary questions are : (1) Is the additional work worth the effort in terms of response time gain ? and (2) Is the resulting performance sufficient to aid intelligence analysts ?

5.2 Exploiting Spatial Indexing to enforce spatial locality

In this section we examine alternative algorithms for spatial indexing, outlining what possible mechanisms exist to allow us to accomplish our set of goal spatial functions efficiently.

Point and radius instance return

Return all situation elements within a given range d of a point (a,b).

In [62] we are given an extensive overview of structures and how they relate to providing solutions to these type of *range queries*. We refer the reader to his evaluations and descriptions of the following structures:

- *Sequential Lists* are the simplest approach to storing point data, and are inefficient for almost all queries !

- *Inverted Lists* are a simple improvement of sequential lists.

- *Fixed Grid Cells* divides the space into conveniently sized buckets..

- *Point QuadTrees* are based on grids and binary search trees. An optimized point quadtree requires that all data points are known in advance. An alternative is to build an optimized tree as the data points are inserted, and then spend some work rebalancing when the tree becomes "badly" balanced. The cost of an average insertion is $O(log_4 n)$ where n is the size of the tree, and the amount of work required to build a tree is equal to the total path length, shown to be proportional to $nlog_4 n$. Point quadtrees are suited to "proximity" searches such as range queries. The worst case cost of 2d range searching is $O(2 * n^{1/2})$.

- *K-D Trees*, an improvement over the point-quadtree which reduces the branching factor of each node as well as the storage requirements. The average cost of inserting to a K-D tree is $O(log_2 n)$, and similarly, the cost of producing an entire tree is $O(n * log_2 n)$. Rebalancing algorithms similar but more complicated than those of point quadtrees may be used for continued insertion. Range queries for a tree with n points may be completed with a worst case cost of $O(2 * n^{1/2})$. It is important to note that *unlike the point quadtree*, a complete binary k-d tree may *always* be constructed. Point quadtrees are an inherently *parallel* structure allowing key comparisons to be performed in parallel for some larger number of keys (for instance if we consider a tree that is partitioned according to not only location, but also for some number of attributes), however this cannot be done in a k-d tree. [62] concludes that k-d trees are superior *serial* structures, while point quadtrees are superior *parallel* structures. It is important to note that in cases where an *approximate* answer is sufficient for identifying elements within a particular polygonal or rectangular region, this region can be approximated (or at least subsumed with a "small" excess) by a point and range search.

- *Range Trees* are asymptotically faster at range searching than point quadtrees or k-d trees, however, an additional price is incurred in the form of additional storage costs. For n points, a range search takes place in $O(log_2^2 n + F)$ time and $O(nlog_2 n)$ space, where F is the number of located points in the range.

- *Priority Search Trees*[49] support time requirements for semi-infinite 2-d range searching of $O(log_2 n + F)$[17] with space requirements of only ($O(n)$). However, in a priority search tree the "cheaper" range searching is obtained for "semi-infinite" queries only, that is, where one of the x or y coordinates range over a finite range and the other ranges over an infinite range. To get results in which both coordinates range over a finite range (as we must), a two step process is required: (1) a semi-inifite query is made on one coordinate, and (2), all points with the second coordinate exceeding a particular range value are discarded[62][p.84].

- Range priority search trees[17] were designed as an extension of pririty trees specifically for range searching, and consequently can perform 2-d finite range queries in time $O(log_2 n + F)$ in one step.

- *Region-based QuadTree*, is a hierarchical data structure (actually a *trie*) based on recursive decomposition that can be adapted to handle point data. A (MX) Quadtree has a worst case cost of search for all point in a rectangle with a certain orientation is $O(F + 2^n)$ where n is the maximum depth of the tree and F is the number of points found. A (PR) Quadtree is an extension of the (MR) which allows for an arbitrary (minimum) separation of points.

Instance set intersection

Computing the intersection of two (large) sets can be expensive. For instance a naive implementation might simply take the larger set and subtract each member of the smaller set. If the larger set is size n and the smaller size m that equates to m deletions each costing on average $n/2$. A simple but powerful structure that implements sets as pointers to objects, where the objects "know" to which sets they belong can provide for a simple intersection algorithm that is of the order of the size of the larger set m. Figure 5.1 outlines a structure for two lists of pointers to objects.

In this example structure, the objects "know" to which lists they belong and consequently there is no "lookup" required to remove an object in List 1 from List 2. In this type of algorithm, the smaller set is copied (at a cost of order m), and then each of the members of the larger set is sent a message to *remove itself* from the copied set. This removal is an $O(1)$ cost function since each object "knows" its position in the copied list directly. Finally, each member of the resultant set is sent a message to remove itself from the smaller set. This final set contains the intersection.

Two circle intersection instance return

> Given two circles we wish to return those elements that fall in the circle intersection. This is a combination of the previous two functions.

Using the approaches described for *point and radius instance return* and *instance set intersection* we may easily obtain a set of objects for each circle (point and radius) and then compute the intersection.

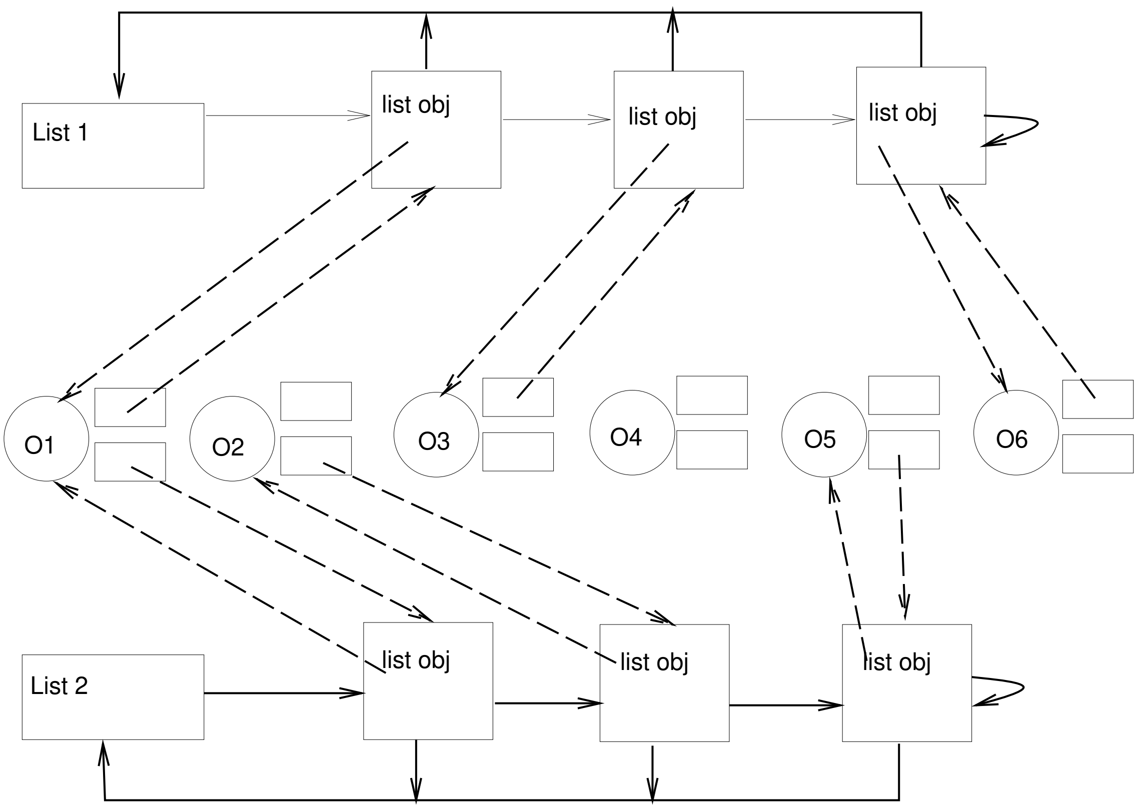

Figure 5.1: Simple list structure for computing intersection.

Arbitrary rectangle instance return

Given a rectangle defined by two opposing corner points, return all spatial objects within this area in a set.

5.3 Notes on locality and constraint-propagation

- An easy analogy exists between spatial locality application and the constraint satisfaction and search paradigm, where pre-processing algorithms such as various N-Consistency algorithms may be applied to a problem to reduce the combinatoric size of the problem in advance of search, OR partial consistency algorithms may be applied **during** search, resulting in search algorithms that closely resemble Forward Checking or Partial Lookahead which are well known to outperform standard heuristic backtracking. However, in these search algorithms the precise correlation(s) between problem characteristics, problem representation(s) and search strategy remain unknown.

- Any preprocessing algorithms may be seen as application of a certain well defined set of constraint knowledge that quite possibly would have been applied during standard search in any case, and consequently it remains to be shown that this initial work actually reduces the subsequent search or is merely a re-distribution, and under what conditions this improvement is realized. The possibility also exists that a tradeoff of space (in terms of indices, for example) may be made for search time, and this may be quite desirable when timely response is imperative and resources such as memory are not scarce.

- Preprocessing algorithms may be seen as special cases of constraint propagation - with spatial constraints. The use of a specific data structure can reduce the complexity of these checks. However in general we are left with a problem common to other database query applications ... we may wish/need to formulate complicated queries involving logical AND and OR over some number of k attributes, only one of which is relative spatial location.

- If we have a set of domain elements for some VariableK and we know that no domain element of VariablesI ... VariableJ may be outside d-distant from an instance for VariableK then it is clearly possible to apply a consistency algorithm to reduce the domain ranges of VariablesI ... VariableJ based on some intersection of the ranges of the domain of VariableK. However, is it better to pre-compute some area, perhaps by merging these element ranges into one large range and then comparing with those of Variables I..J ? A standard revise-algorithm revise(rangeVariableK,rangeVariableI) would take each element i of the domain of VariableI and see if one of the elements of VariableK is compatible/consistent with i. Any i which is found to have no consistent partner in rangeVariableK is removed from rangeVariableI. This is an expensive process. If VariableK has m elements in its current domain, and VariableI has n elements, then just discovering that all values of VariableI are consistent with those of VariableK requires m*n constraint checks.

- The preceding discussion raises the excellent point that it may be a mistake to preassign variable domain ranges in the typical CSP fashion, but rather one might assign ranges only after solving some subset of the original problem. In this way, a range is selected according to some spatial range identified in conjunction with previous selections. Certainly if one were able to (abstractly) define the crucial parts of the template versus the subsidiary parts, the subsidiary parts could be left for range selection in accordance with earlier instantiations.

Chapter 6

Conclusion

6.1 Recommendations

In this chapter we present our overall recommendations with respect to adoption and integration of measures to exploit spatial locality in the STR-CSP and/or STR-FGR approaches to template recognition. We expect that future work will take the form of extended algorithm implementation and demonstration of the heuristic adequacy of the approaches outlined.

It is recommended that a simple spatial indexing methodology be implemented according to the algorithms outlined by Samet, so as to provide for spatial indexing as range functions which can then be incorporated into the functionality of either the STR-FGR or STR-CSP algorithms. Assuming that the primary operation of interest will be the point-range requirement it appears as if a Range Priority Search Tree is the best selection for a stand-alone spatial index. A Range Priority Tree gives asymptotically better performance on range queries $(O(log_2 n+F))$[17, 49] compared to K-D trees $(O(2*n^{1/2}))$ at the expense of additional storage $(O(nlog_2 n))$. This index will be maintained continuously as elements are added to the situation so that at *query* time (i.e. a request is made to template-match), the structure is constructed and may be immediately utilized. This spatial index structure exhibits insertion behaviour that is logarithmic[62][pp.172] with respect to the current structure size.

The algorithms described in [62] and others which manipulate these tree structures are currently considered as "stand-alone", apart from any larger system or database in order that we may empirically demonstrate the viability (with respect to increasing heuristic adequacy) of this approach. An existing testbed exists for the STR-CSP demonstrator (MOTORS), and the improved spatial functionality might be readily tested in this application. This subsequently enhanced implementation may be best viewed as a prototype of the possible functionality available to a TIF-embedded version. Any implemented indexing algorithm(s) can be easily ported and applied for use in the standalone STR-FGR implementation.

Commercial packages providing spatial indexing functionality are certainly one possible option to consider as an alternative to a specific data index structure implementation for inclusion in the larger Tactical Information Fusion (TIF)[13, 14, 32, 24] demonstrator in place of an ad-hoc implementation. However, one would imagine that this type of investment in technology might be excessive at this stage. Prior to investment in time implementing specialized indexing structures, the internal data structures currently in use for TIF should be examined to determine if they may be readily extended or replaced with new structures supporting spatial indexing without adversely affecting the rest of TIF.

6.2 Acknowledgments

This research has been carried out as part of a contractual agreement between DMR Groupe Inc. (Quebec City, Quebec), Steven Woods, and the Department of National Defence, Defence Research Establishment Valcartier (Quebec City, Quebec). This work has been carried out under the scientific authority of Stéphane Lapointe, Defence Scientist.

Chapter 7

Selected annotations

This chapter provides a listing of some of the primary sources consulted during the course of our preliminary literature survey into spatial databases and spatial reasoning. A brief description is included for selected entries.

Primary sources for spatial algorithms adopted

The books by Hanan Samet are probably the best source for material on hierarchical approaches, and several experts[1] contend that they remain comprehensive and current.

- Hanan Samet. *The Design and Analysis of Spatial Data Structures*[62]. Presents in depth discussion of various structures for representing spatial information, including algorithms and complexity discussion. The highlight of this book is an *extensive* bibliography.
- H. Samet. The quadtree and related hierarchical data structures[61]. A specific discussion of quadtrees and how they relate to other spatial data structures. Most of this material is contained Samet's book.
- Beng Chin Ooi. *The Design and Analysis of Spatial Data Structures*[56]. An *excellent* overview of two-dimensional GIS spatial data handling, including extensive discussion of spatial indexing structures and query optimization.

Nearest neighbor algorithms

- J. Abel and J.L. Smith. A simple approach to the nearest neighbor problem[1]. Abel and Smith outline one novel method of solving the nearest neighbor problem. In

[1]Personal correspondence with the Centre for Spatial Information Systems, Canberra Australia. Thanks to Dr. Dave Abel, Dr. John Smith, Scott Milton and Peter Milne for helpful discussion.

addition, they summarize work on this problem which predates 1984.

- Michael R. Garey and David S Johnson. *Computers and Intractability : A guide to the theory of NP-Completeness*[23]. This book is perhaps the best source of proofs of NP-completeness, as well as an excellent guide to dealing with NP-completeness via approximation algorithms.

- Cormen et al [11] provide another source that is more up to date in some ways and provides more examples of approximation methods such as diagonalization.

Indexing and spatial databases

- K.K. Al-Taha, R.T. Snodgrass, and M.D. Soo. *Bibliography on spatiotemporal databases*[5]. Good overview of the recent literature in spatial database systems.

- C. Kolovoson, M. Neimat, and S. Potomyanos. *Interoperability of spatial and attribute data managers : A case study*[41]. A case study of integrating a commercial relational storage manager and a commercial spatial data manager. This work presents the challenge of interoperating such data managers and exposes the problems that must be solved to make such interoperability feaible, painless, transparent, and efficient. Gives examples that emphasize the need for a comprehensive cost-based query optimization strategy.

- C. Kolovson and M. Stonebraker. *Segment indexes: Dynamic indexing techniques for multi-dimensional interval data*[43]. The authors propose new indexing techniques for interval data in K >= 1 dimension(s) consisting of a set of extensions to a class of database indexing structures. These techniques are useful for improving index search performance for spatial data composed of multi-dimensional intervals that have non-uniform length distributions. Interval data collections having non-uniform length distributions are likely to occur in practice, and may be typical of historical data collections in which tuples represent intervals in the time dimensions. The authors present these indexing techniques, illustrate how they may be applied to the R-Tree index, and provide the results of performance experiments.

- Additional references in this area include [42, 64, 63, 51, 83, 74, 52].

Spatial locality

- Z. Pylyshyn. *The role of location indexes in spatial perception: A sketch of the FINST spatial-index model*[60]. A detailed discussion from the point of view of vision of the value of separating *attribute* information and *spatial locational* information in representing and reasoning about objects in space. Strong motivation for spatially-based indexing as part of recognizing macro-concept instances composed of many spatially located sub-objects.

Spatial model representation and recognition

- David M. Chelberg and An doo Yang. *Improving the results of object recognition and localization using probabilistic methods*[8]. This work is derivative of vision systems, however, it is included for its use of localization techniques. Due to the fact that there are always some noise, errors, and ambiguities resulting from image formation and segmentation, measurements obtained from images are never perfect and are likely to affect the results of object recognition and localization. Therefore, most object recognition systems use many parameters, such as error margins, for the matching process. However, there is currently no general method to choose appropriate values for these parameters. The reliability and robustness of the systems are thus reduced. Moreover, errors and ambiguities commonly exist at each step of recognition and localization processes. Without taking these uncertainties into account, the computed quantities may not be accurate. We are investigating a probabilistic methodology which will give a comprehensive basis for setting parameter values and also account for uncertainties so better measurements can be obtained.

- J. Yi and D.M. Chelberg. *Rapid object recognition from a large model database*[85]. The authors attack the difficult problem of rapid object recognition from a large model data base. The research focuses on efficient indexing. A decision-theoretic approach using a Bayesian framework is employed to achieve efficient indexing of model objects. A decision-theoretic measure of the discriminatory power of a feature for a model object is defined in terms of posterior probability. Domain-specific knowledge is compiled off-line from CAD model data in order to estimate posterior probabilities that define the discriminatory power of features for model objects. Detectability of a feature defined as a function of the feature itself, viewpoint, sensor characteristics, and the feature detection algorithm(s) are also considered in the computation of discriminatory power. In order to speed up the indexing or selection of the correct objects, we generate and verify the object hypotheses for the features detected in the scene in the order of the discriminatory power of these features for model objects.

- A discussion of the solution of constraint satisfaction problems for spatial planning is presented by duVerdier in [15].

- Additional references in the area of spatial models and representation include [33, 73, 4, 12, 66, 65, 45, 28, 26, 6, 7, 68, 69, 27, 3, 2].

Spatial reasoning

- A. U. Frank, I. Campari, and U. Formentini (eds.). *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*[19]. A collection of papers from the first international conference dedicated to spatial and temporal reasoning in geographic space. Including several excellent overview papers such as "Do people understand spatial concepts: The case of first-order primitives" by Reginald Golledge. This collection covers a range of spatial reasoning from defining spatial concepts to discussion of hierachical aspects of spatially encoded information.
- D. McDermott. *Spatial reasoning*[50]. An excellent high level overview of the field of spatial reasoning from the perspective of artificial intelligence research. Many pointers to other related work in spatial databases and reasoning.
- T. Nishida, A. Yamada, and S. Doshita. *Figuring out most plausible interpretation from spatial constraints*[55].
- Additional references in this area include [9, 18, 16, 57].

Background for spatial problems

The following references cover a broad spectrum of papers and discussions involving GIS systems, methodologies, and theory: [20, 34, 71].

Bibliography

[1] J. Abel and J.L. Smith. A simple approach to the nearest neighbor problem. *Australian Computer Journal*, 16:140–146, 1984.

[2] G. Adorni, A. Boccalatte, and M. DiManzo. Object representation and spatial knowledge: an insight into the problem of men-robots communication. In *Proc. of the 7th Conf. of the Canadian Man-Computer Com. Soc*, pages 37–45, Waterloo, Ontario, 1981.

[3] G. Adorni and M. Di Manzo. Top-down approaches to scene interpretation. In *Proc. CIL 83*, pages 28–25, 1983.

[4] J. K. Aggarwal, L. S. Davis, W. N. Martin, and J. W. Roach. Survey: Representation methods for three-dimensional objects. In L. H. Kanal and A. Rosenfeld, editors, *Progress in Pattern Recognition*, pages 371–391. North-Holland, Amsterdam, 1981.

[5] K.K. Al-Taha, R.T. Snodgrass, and M.D. Soo. Bibliography on spatiotemporal databases. *International Journal of Geographical Information Systems*, 8(1):95–103, January-February 1994.

[6] E. Bialystok and D. R. Olson. Spatial categories: The perception and conceptualization of spatial relations. In S. Harnad, editor, *Categorial Perception: The Groundwork of Cognition.* Cambridge University Press, Cambridge, 1987.

[7] I. Biederman, R. J. Mezzanotte, and J. C. Rabinowitz. Scene perception: Detecting and judging objects undergoing relational violations. *Cognitive Psychology*, 14:143–177, 1982.

[8] David M. Chelberg and An doo Yang. Improving the results of object recognition and localization using probabilistic methods. work reported online at http://dynamo.ecn.purdue.edu/Researchsummary/Section5/sec5_30.html, 1994.

[9] S. Chen, editor. *Advances in Spatial Reasoning*, volume 1. Ablex Publishing Corporation, Norwood, NJ, 1990.

[10] Martin C. Cooper. An optimal k-consistency algorithm. *Artificial Intelligence*, 41:89–95, 1989.

[11] Thomas H. Cormen, Charles E. Leiserson, and L. Rivest, Ronald. *Intoduction to Algorithms.* McGraw-Hill Book Company, 1992.

[12] Ernest Davis. *Representations of commonsense knowledge.* Morgan Kaufmann, San Mateo, CA, 1990.

[13] L. Des Groseilliers. Tactical information fusion prototype: Basic intelligence. Memorandum 3093/92, Defense Research Establishment Valcartier, Valcartier, Quebec, February 1992. Unclassified.

[14] L. Des Groseilliers, J.-C. Pare, and D. Soucy. Tactical information fusion prototype: Enemy order of battle database. Technical Report 3066/91, Defence Research Establishment Valcartier, June 1991. Unclassified.

[15] F. du Verdier. Solving geometric constraint satisfaction problems for spatial planning. In *Proceedings of the 13th IJCAI*, pages 1564–1569, 1993.

[16] S. Dutta. *Topological Constraints: A Representational Framework for Approximate Spatial and Temporal Reasoning*, volume 525 of *Lecture Notes in Computer Science*, pages 161–180. Springer-Verlag, New York, 1991.

[17] Herbert Edelsbrunner. A note on dynamic range searching. *Bulletin of the EATCS*, 10:34–4–, October 1981.

[18] Herbert Edelsbrunner. *Algorithms in combinational geometry*, volume 10 of *EATCS Monographs on theoretical computer science.* Springer-Verlag, 1987.

[19] A. U. Frank, I. Campari, and U. Formentini (eds.). *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, volume 639 of *Lecture Notes in Computer Science.* Springer-Verlag, sep 1992.

[20] Andrew U. Frank and Irene Campari, editors. *Spatial information theory : a theoretical basis for GIS : European conference, COSIT'93.* Springer-Verlag, 1993.

[21] E. Freuder and J. Wallace. Partial constraint satisfaction. *Artificial Intelligence*, 58:21–70, December 1992.

[22] E.C. Freuder. A sufficient condition of backtrack-free search. *Journal of the ACM*, 29(1):23–32, 1982.

[23] Michael R. Garey and David S Johnson. *Computers and Intractability : A guide to the theory of NP-Completeness.* W. H. Freeman and Company, Bell Laboratories, Murray Hill, New Jersey, 1979.

[24] M. Gauvin, L. Des Groseilliers, and D. Roy. Tactical information fusion prototype: Research project definition. Memorandum 3068/91, Defense Research Establishment Valcartier, Valcartier, Quebec, May 1991. Unclassified.

[25] D. Ghahraman, A. Wong, and A. Tung. Graph optimal monomorphism algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-10(4):181–188, April 1980.

[26] R. Gillespie and W. A. Davis. Tree data structures for graphics and image processing. In *Proceedings of the 7th Conference of the Canadian Man-Computer Communications Society (CMCCS '81 / ACCHO '81)*, pages 155–162, jun 1981.

[27] W. Eric L. Grimson and Tomás Lozano-Pérez. Model-Based Recognition and Localization from Sparse Range or Tactile Data. *International Journal of Robotics Research*, 3(3):3–35, Fall 1984.

[28] D. D. Hoffman and W. A. Richards. Parts-of recognition. *Cognition*, 18:65–96, 1984.

[29] Robert Holte, Robert Zimmer, and Alan MacDonald. When does changing representation improve problem-solving performance ? In *Proceedings of the Workshop on Change or Representation and Problem Reformulation*, April 1992. NASA Ames Technical Report FIA-92-06.

[30] Robert Holte, Robert Zimmer, and Alan MacDonald. A study of the representation-dependency of abstraction techniques. In *Proceedings of the 1993 Workshop on Knowledge Compilation and Speedup Learning*, 1993.

[31] Robert C. Holte. Automatic change of representation. personal correspondence, December 1993.

[32] DMR Group Inc. Knowledge representation for tactical information fusion. Contract W7701-1-0659/01-XSK, Defense Research Establishment Valcartier, May 1992. Unclassified.

[33] A. Kak, editor. *Special Issue AI Magazine on Object Recognition*, volume 9. AAAI, 1988.

[34] Avi Kak and Su shing Chen, editors. *Spatial reasoning and multi-sensor fusion : proceedings of the 1987 workshop.* American Association of Artificial Intelligence, 1987.

[35] A. Kaufmann. *Introduction to the theory of fuzzy sets*, volume 1. Academic Press Inc., 1975.

[36] Craig Knoblock. Learning abstraction hierarchies for problem solving. In *Proceedings of Eighth National Conference on Artificial Intelligence*, pages 923–928, Boston, MA, 1990.

[37] Craig Knoblock. Search reduction in hierarchical problem solving. In *Proceedings of the 9th AAAI*, volume 2, pages 686–691, 1991.

[38] Craig A. Knoblock. Learning hierarchies of abstraction spaces. In *Proceedings of the Sixth International Workshop on Machine Learning*, pages 241–245, Los Altos, CA, 1989. Morgan Kaufmann.

[39] Craig A. Knoblock. A theory of abstraction for hierarchical planning. In Paul Benjamin, editor, *Proceedings of the Workshop on Change of Representation and Inductive Bias*, Boston, MA, 1989. Kluwer.

[40] Craig A. Knoblock. *Automatically Generating Abstractions for Problem Solving*. PhD thesis, School of Computer Science, Carnegie Mellon University, 1991.

[41] C. Kolovoson, M. Neimat, and S. Potomyanos. Interoperability of spatial and attribute data managers : A case study. In Dave Abel and Beng Chin Ooi, editors, *Proceedings of the 3rd International Conference on Large Spatial Databases*, Advances in Spatial Databases, 3rd edition, pages 239–263. Springer Verlag, June 1993. Lecture Notes in Computer Science.

[42] C. Kolovson. *Indexing Techniques for Multi-Dimensional Spatial Data and Historical Data in Database Management Systems*. PhD thesis, University of California, Berkeley, nov 1990.

[43] C. Kolovson and M. Stonebraker. Segment indexes: Dynamic indexing techniques for multi-dimensional interval data. In J. Clifford and R. King, editors, *Proceedings of ACM SIGMOD 1991*, pages 138–147, jun 1991.

[44] Vipin Kumar. Algorithms for constraint-satisfaction problems. *AI Magazine*, pages 32–44, Spring 1992.

[45] Y. Lamdan and H. J. Wolfson. Geometric Hashing: A General and Efficient Model-Based Recognition Sceme. In *Proceedings of the Second International Conference on Computer Vision*, pages 238–249, 1988.

[46] Stéphane Lapointe and René Proulx. Fuzzy geometric relations to represent hierarchical spatial information. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, 1994. to appear.

[47] A.K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8:99–118, 1977.

[48] Alan Mackworth, Jan Mulder, and William Havens. Hierarchial arc consistency: exploiting structured domains in constraint satisfaction problems. *Computational Intelligence*, 1:188–126, 1985.

[49] E.M. McCreight. Priority search trees. *SIAM Journal on Computing*, 14(2):257–276, May 1985.

[50] D. McDermott. Spatial reasoning. In S. C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 863–870. Wiley, New York, 1987.

[51] D. M. McKeown. Maps: The organization of a spatial database system using imagery, terrain, and map data. Technical Report CMU-CS-83-136, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, 1983.

[52] Peter Milne, Scott Milton, and John L. Smith. Geographic object-oriented databases, a case study. *International Journal of Geographic Information Systems*, in print.

[53] R. Mohr and T.C. Henderson. Arc and path consistency revisited. *Artificial Intelligence*, 28:225–233, 1986.

[54] Bernard A. Nadel. Constraint satisfaction algorithms. *Computational Intelligence*, 5:188–224, 1989.

[55] T. Nishida, A. Yamada, and S. Doshita. Figuring out most plausible interpretation from spatial constraints. In *Proc. of the 1987 Workshop on Spatial Reasoning and Multi-Sensor Fusion*, pages 158–167, St. Charles, IL, 1987.

[56] Beng Chin Ooi. In G. Goos and J. Hartmanis, editors, *Efficient Query Processing in Geographic Information Systems*, number 471 in Lecture Notes in Computer Science, pages 239–263. Springer-Verlag, June 1990.

[57] B. T. Perricone. Spatial Representation and Reasoning - A Critical Survey and Proposal. Technical Report TR-1275, University of Maryland, 1983.

[58] Patrick Prosser. Hybrid algorithms for the constraint satisfaction problem. *Computational Intelligence*, 9(3):268–299, 1993.

[59] René Proulx (APG Inc.). Fuzzy representation and matching of spatial templates. DREV Technical Report W7701-3-0847/01-XSK, Defence Research Establishment Valcartier, 1994.

[60] Z. Pylyshyn. The role of location indexes in spatial perception: A sketch of the FINST spatial-index model. *Cognition*, 32:65–97, 1989.

[61] H. Samet. The quadtree and related hierarchical data structures. *Computing Surveys*, 16(#2):187–260, 1984.

[62] Hanan Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, 1990.

[63] R. Schneider and H.-P. Kriegel. Indexing the spatiotemporal monitoring of a polygonal object. In *Proceedings of the 5th International Symposium on Spatial Data Handling*, volume 1, pages 200–209, Charleston, SC, 1992.

[64] Clifford Shaffer and Patrick Brown. Paged pointer-based quadtrees. In Dave Abel and Beng Chin Ooi, editors, *Proceedings of the 3rd International Conference on Large Spatial Databases*, Advances in Spatial Databases, 3rd edition, pages 89–104. Springer Verlag, June 1993.

[65] L. G. Shapiro, J. D. Moriarty, R. M. Haralick, and P. G. Mulgaonkar. Matching three-dimensional objects using a relational paradigm. *Pattern Recognition*, 17(4):385–405, 1984.

[66] T. M. Silberberg. A Survey on Object Representation and Recognition Techniques. Technical Report CAR-TR-107, University of Maryland, Center for Automation Research, 1985.

[67] D. Stacey and A. Wong. A largest common subgraph isomorphism algorithm for attributed graphs. University of Waterloo, Systems Design Engineering, June 1988.

[68] A. Stevens and P. Coupe. Distortions in judged spatial relations. *Cognitive Psychology*, 10:422–437, 1978.

[69] George C. Stockman. Object Recognition and Localization via Pose Clustering. *Computer Vision, Graphics, and Image Processing*, 40:361–387, 1987.

[70] P Van Hentenryck, Y. Deville, and C-M. Teng. A generic arc-consistency algorithm and its specializations. *Artificial Intelligence*, 57:291–321, 1992.

[71] Petrus Johannes Maria van Oosterom. *Reactive data structures for geographic information systems*. Oxford University Press, 1993.

[72] Paul van Run. Domain independent heuristics in hybrid algorithms for CSPs. Master's thesis, University of Waterloo, 1994.

[73] B. C. Vemuri and J. K. Aggarwal. Resolving the Orientation and Identity of an Object From Range Data. In *Proc. 1987 Workshop on Spatial Reasoning and Multisensor Fusion*, pages 178–187, October 1987.

[74] Graham Williams and Steven Woods. Representing expectations in spatial information systems: A case study. In Dave Abel and Beng Chin Ooi, editors, *Proceedings of the 3rd International Conference on Large Spatial Databases*, volume 692 of *Advances in Spatial Databases, 3rd edition*, pages 465–476. Springer Verlag, June 1993. Lecture Notes in Computer Science.

[75] A. Wong, M. You, and S. Chan. An algorithm for graph optimal monomorphism. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(3):628–636, May/June 1990.

[76] Steven Woods. Interactive recognition of spatially defined model deployment templates. Research memorandum, Combat Intelligence Automation Group, Defence Research Establishment Valcartier, Courcelette, Quebec CANADA, August 1993. unclassified.

[77] Steven Woods. A method of interactive recognition of spatially defined model deployment templates using abstraction. In H. Merklinger, M. Farooq, P. Roberge, and R. Grodski, J.J.and Dobson, editors, *Proceedings of the Knowledge -Based Systems and Robotics Workshop*, pages 665–675. Department of National Defence, Government of Canada, November 1993.

[78] Steven Woods. Addressing spatial template recognition as a csp within the context of clp. advance copy, technical report, 1994.

[79] Steven Woods. Learning refinements to spatial template models based on false-positive and false-negative examples. advance copy, technical report, April 1994.

[80] Steven Woods. A constraint-based approach to program plan recognition in software reverse engineering. Ph.D. Thesis Proposal, University of Waterloo, February 1995.

[81] Steven Woods and Qiang Yang. A constraint-based approach to program plan recognition. submitted for publication, 1995.

[82] Steven Woods and Qiang Yang. Program understanding as constraint satisfaction. In *Proceedings of the IEEE Seventh International Workshop on Computer-Aided Software Engineering (CASE-95)*. IEEE Computer Society Press, July 1995. to appear.

[83] X. Xu, J. Han, and W. Lu. Rt-tree: An improved r-tree index structure for spatiotemporal databases. In *Proceedings of the 4th International Symposium on Spatial Data Handling*, 1990.

[84] Qiang Yang and Philip Fong. Solving partial constraint satisfaction problems using local search and abstraction. Technical Report cs-92-50, University of Waterloo, 1992.

[85] J. Yi and D.M. Chelberg. Rapid object recognition from a large model database. In *Proceedings of the 2nd IEEE Workshop on CAD-based Vision*, 1994.