On the utility of explicit application of spatial locality features in the automatic recognition of spatial templates

Steven Woods, Stephane Lapointe

January 19, 1995

Abstract

This document is intended as a background report on exploiting the property of spatial locality and spatial indexing in the process of identifying military deployment templates in situated sets of unit sightings. Two specific methodolgies have been examined as part of earlier phases of this work, and we identify several key functions as common to the requirements of these methods.

Contents

List of Figures

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 battallions, 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 acceptabile 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 extend these earlier approaches by exploiting the spatial nature these templates.

A 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 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 *potentially relatable.*

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 depoloyment templates. In particular, we wish to reduce the complexity of the task of identifying template instances in situations which contain many extraneous or unrelated units, and units that are only partially defined. The methods investigated are intended to be applicable to two currently separate visions of the solution spatial template matching problem. We present several applicable spatial processing techniques and algorithms, and describe how these fit into the task of recognizing spatial templates. Future work will involve specific implementation and experimentation to demonstrate the feasibility and advisability of applying these algorithms to exploit the property of spatial locality.

Please note that this paper is a DRAFT only, and is intended as a basis for discussion and evaluation of alternatives, *not* as a publishable work or technical report at this time.

Chapter 1

Spatial template recognition

In this chapter we will briefly overview the concept of spatial locality and the problem of spatial template recognition in general. In addition, we will describe and cite the two approaches investigated in earlier work related to this same project and identify their common interest in exploiting properties of spatial locality.

What is Spatial Template Recognition (STR) ?

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 (Woods 1993*b*, Woods 1993*a*, Lapointe & Proulx 1994, Lapointe & Prouix 1994). 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 (Woods 1994*c*). 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 inter-connected 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

[1]Other related work in pattern analysis such as Stacey & Wong (1988), Ghahraman, Wong & Tung (1980) and Wong, You & Chan (1990) models problems with a far greater "rigidity" in terms of the spatial template model.

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.

Recent work at Defence Research Establishment Valcartier (DREV) has focused on two related representation schemes and approaches to the spatial template recognition problem. 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, 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. Both representations explicitly provide for problem decomposition hierarchically, and map this representation to the solution methodology. We will provide a brief overview of each of the methods in the next two subsections, and in a third subsection we shall distill the common spatial relations that can be exploited.

1.1 Constraint-based spatial template recognition

A relatively obvious formulation of STR as a **Constraint Satisfaction Problem** (CSP) was outlined in Woods (1993*b*) and described in detail in Woods (1993*a*). 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 (Kumar 1992). An example of a template constraint graph is shown in Figure 1.1 with five node slots where each slot is nnconstrained by several attribute ranges, and 11 inter-slot spatial constraints on distance, position, and orientation.

1.1.1 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 (Nadel 1989, Kumar 1992). 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(Mackworth 1977), AC-4(Mohr & Henderson 1986), AC-5(Van Hentenryck, Deville & Teng 1992), and other variations(Freuder 1982, Cooper 1989). In (Woods 1993*a*) we experimentall examined the utility of some of these algorithms as well as partial consistency algorithms described in (Nadel 1989).

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

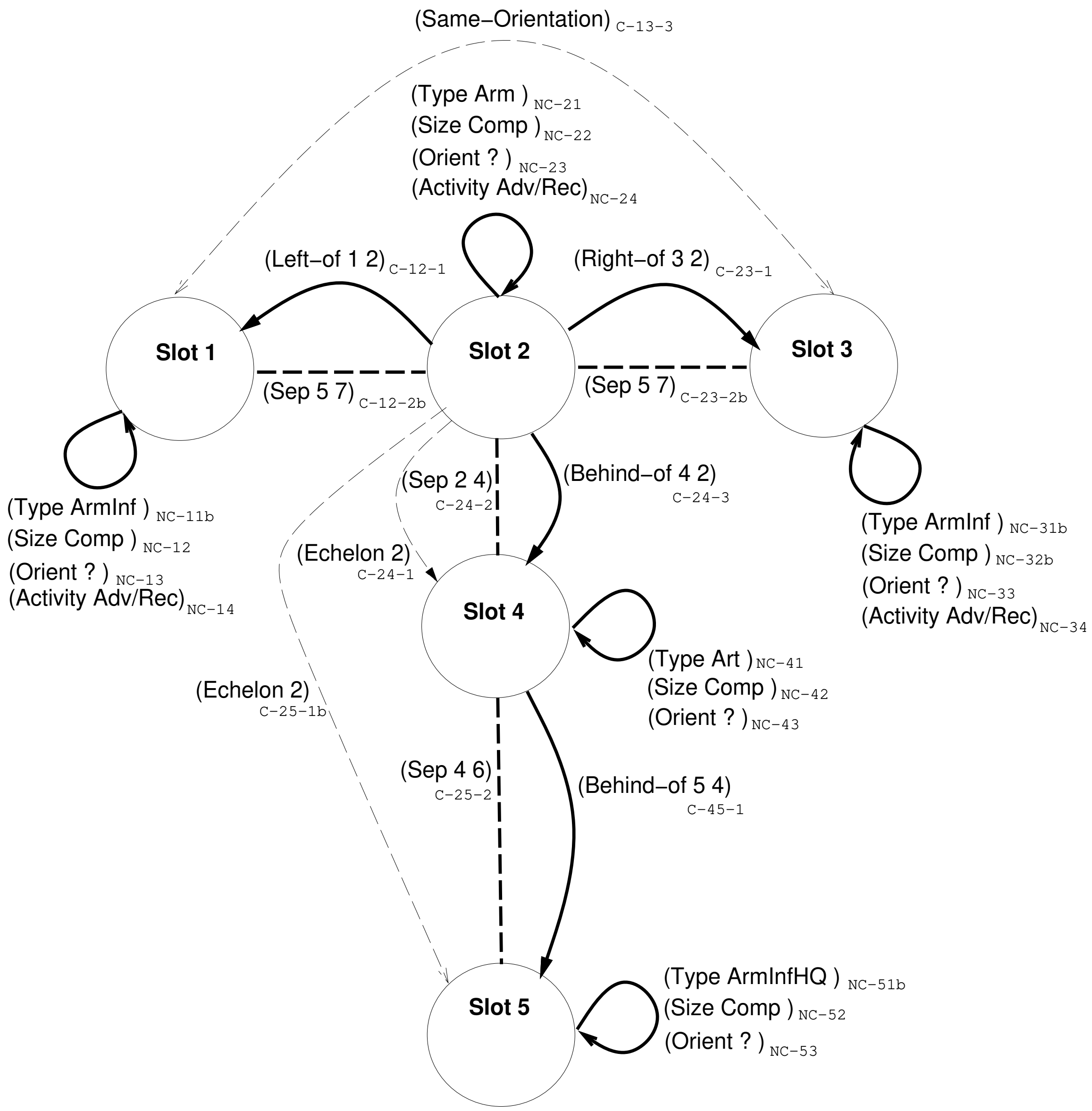

Figure 1.1: Whole CSP 5 variables, 11 arc constraints.

in which candidate 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, that is, to be the most context limiting.

During search is 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 (Nadel 1989) 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] These algorithms 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.

Many search problems (such as ours) involve not just one solution, but consideration of many, if not all, possible solutions. *Solution Evalution* heuristics determine whether or not a particular solution n is satisfactory according to some domain criteria. In a cooperative interactive approach to program understanding, it is at this point that an expert might interact and evaluate a particular solution for adequacy. Similarly, if there existed particular measures of adequacy or *soft or preferential* constraints that may have been relaxed during search, such a measure could be applied.

[2]These and other intelligent backtracking algorithms are described in detail by Nadel in(Nadel 1989).

1.1.2 Applicability of spatial locality to problem features

CSP solution involves both search and controlled constraint application algorithms. While much study has been focused on the construction of efficient algorithms (Nadel 1989, Woods 1993*a*, Woods 1994*b*, Woods & Yang 1994, Prosser 1993, van Run 1994) for solution of problems in different real and toy domains, no "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 (Woods 1993*a*) 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 literaly 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, locality may defined in terms of "near and behind object A".

- **Nearness** such as is the case where, given a particular object A at (x,y)

[3] See (Woods 1993*a*) for a comprehensive discussion of these strategies in this domain.

we are interested in all objects within some range r of A. The determination of this "near" set efficiently is of prime importance. Similar questions such as "is B within r of A ?" are also important.

- **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** 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. Examples of

  It should be noted that such an expansion can be closely approximated with an expanding range of search, which one may envisage spatially as ever-expanding circles of radius about a point.

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

- **Intersecting Ranges** such as determining if two objects known to be in particular areas (polygons) are in fact possibly at the same position.

- **Ring Membership** such as is indicated when 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. Consequently it is necessary to identify a set of objects up to r2 from some base point A and then remove (intersection operation) those objects closer than r1 to A.

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 contraints is accomodated through specialized handling of the sighted elements. Of course, independently of spatial constraints we have other constraint sets to consider, and consequently queries are constantly being formed with a conjunctive nature such as "give me all objects within 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.

Ideally, we require an indexing system capable of answering all of these queries efficiently. Unfortunately we are also constrained by space and algorithm complexity issues, and so tradeoffs must be made. Consequently, in order to best exploit spatial locality, we identify that the following are the most critical:

- *All objects within range r of object A or point (x,y).* This feature can be exploited through direct reduction of variable domain ranges. In order for such a query to be useful, the reduction of a set of elements needs to be done in near-instantaneous, and certainly constant time. For example, given a set of candidate elements S, a single operation should reduce this set to S2 in which all elements not satisfying the condition have been removed.

- *Ring Membership* as described.

- As described earlier, we wish to sorted objects with respect to distance from object A or point (x,y). We may easily approximate this with use of the previous 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, this can result in better solutions being identified first.

1.1.3 Earlier related work

In (Woods 1993*a*) the issues of problem representation were examined in some detail, in particular, problem decomposition. The connection between problem solution and decomposition were examined so as to provide for the expression of partial solutions, allow for the interactive solution of these problems, and to provide "good" solutions quickly when "best" solutions cannot be found in reasonable time. 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 replated 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.

At least two additional aspects of our past work are interesting with regards to spatial locality application. First, in (Woods 1993*b*, Woods 1993*a*) our experiments were extended to include the ability to focus on a certain pre-determined range of candidates given a particular sub-object instantiation. In this way, all candidates outside this range are not considered. While this approach was "sharp" in the sense that "unlikely" solutions 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 must be noted that such an approach can actually increase the amount of work in certain situations since a distance check is explicitly done against an entire set of candidates. If this work is not profitable in that few removals are made for a large computational cost then we encounter an overall loss in performance.

We need then, a better method of checking range, as indicated in the previous subsection. Second, 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 (ie up to r distance from a point) return a set of corresponding objects that will be smaller than a "long" constraint (ie 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 greatly from a spatial index which would allow us to identify candidates without explicitly checking all candidates for range.

In this section we briefly overviewed one approach to spatial template recognition that makes use of both known and new technology in constraint satisfaction. In addition, we introduced several potential areas in which efficient spatial functionality and concepts of spatial locality could improve the efficiency of these search and propagation algorithms. In the next section we briefly describe another approach to template recognition, and we will point out the commonalities of the two approaches with regards to spatial constraint usage and spatial locality application.

1.2 Fuzzy Geometric Relations (FGRs) in spatial template recognition

In (Lapointe & Prouix 1994), 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 STR-FGR approach 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(Kaufmann n.d.), over subsets of O. A template is a class where the constraint set is satisfiable through a correspondence betwween template objects and some subset of actual situation elements or objects. In a satisifying instance, all non-fuzzy constraints are satisfied, and any fuzzy constraint is at least partially satisifed, in has a non-zero satisfying measure or correspondence.

We see a very close correspondence between template representations as for STR-CSP and STR-FGR. STR-FGR now adds an additional structure or building block for spatial templates which is closely related to the abstracted STR-CSP as described in (Woods 1993*b*), and the embedded decomposition shown in (Woods 1993*a*)[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 in turn are components of the template, while other template objects are simply 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 Proulx (1994). 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 embedded in Quintus Prolog[5].

STR-FGR makes use of the identical set of spatial relations required by STR-CSP. In addition, STR-FGR similarly constituted relations for FGR relationships. Just as STR-CSP can exploit a spatial indexing system to highly localize the search for FGRs based on a current instantiation of an FGR to a situation element set, so can STR-FGR. Both approaches 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 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 indepenent of a particular solution approach.

---

[4]A related work which performs a similar bottom-up mapping between a hierarchically specified model and an input situation is presented in (Woods & Yang 1995*a*, Woods & Yang 1995*b*). 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.

[5]In Woods (1994*a*) 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 2

Spatial Locality

In this chapter we are going to 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.

In a spatial setting, macro-objects may be seen as a related set of sub-objects, or parts. Each sub-objects has spatial location, orientation, and other properties such as size, orientation, color, etc. In the problem of spatial template reccognition, 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 2 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 an infantry company instance of appropriate orientation within 10 km. The analyst is essentially defining a spatially restricted range of search for each candidate template instance, and is exploting the property of *spatial locality.*

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 instances.

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 2.1. In this template there are 3 sub-objects, each of a different type (depicted as shading), each with an orientation labelled 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.

In choosing to exploit properties of spatial relations to help us identify spatial tempaltes, 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 in

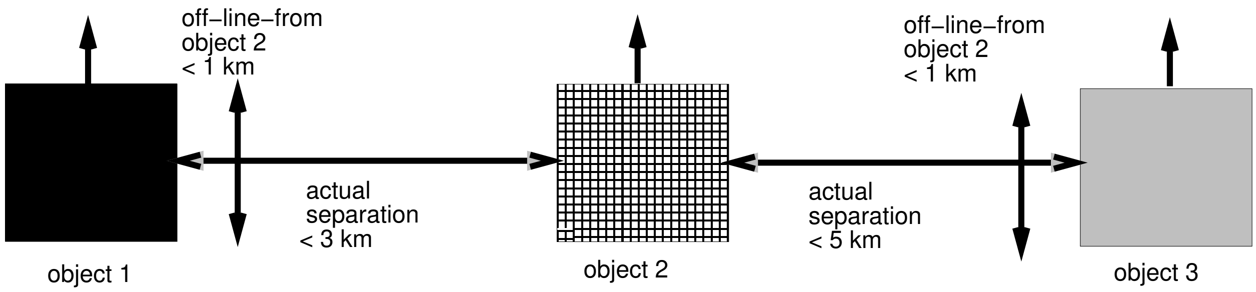

Figure 2.1: A simple template example.

the next Subsection), but which are independent of either the FGR template identification method of Lapointe(93,94) or the constraint reductionistic approach of Woods(93,94). 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.

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 2.2, there are three instances of sub-object 2, labelled 10, 14, and 5. We consider 14 in this example. Our example template has three sub-objects. Given an instance of one sub-objects 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

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

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 2.2 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 2.3 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 2.3 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 an example of achieving *spatial locality* **with respect to orientation** to a single object instance.

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.

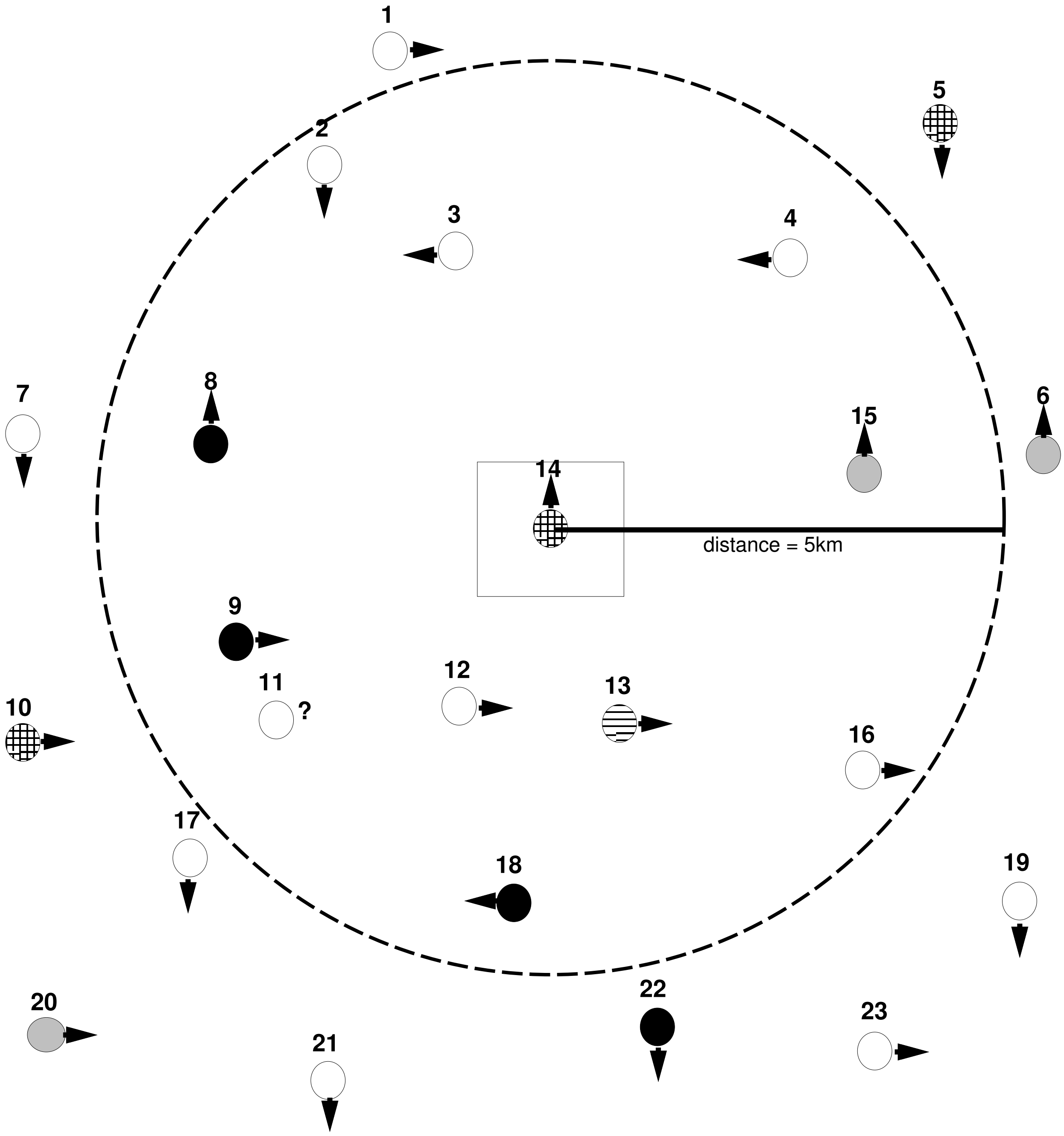

Figure 2.2: A locality of 5 km around instance 14.

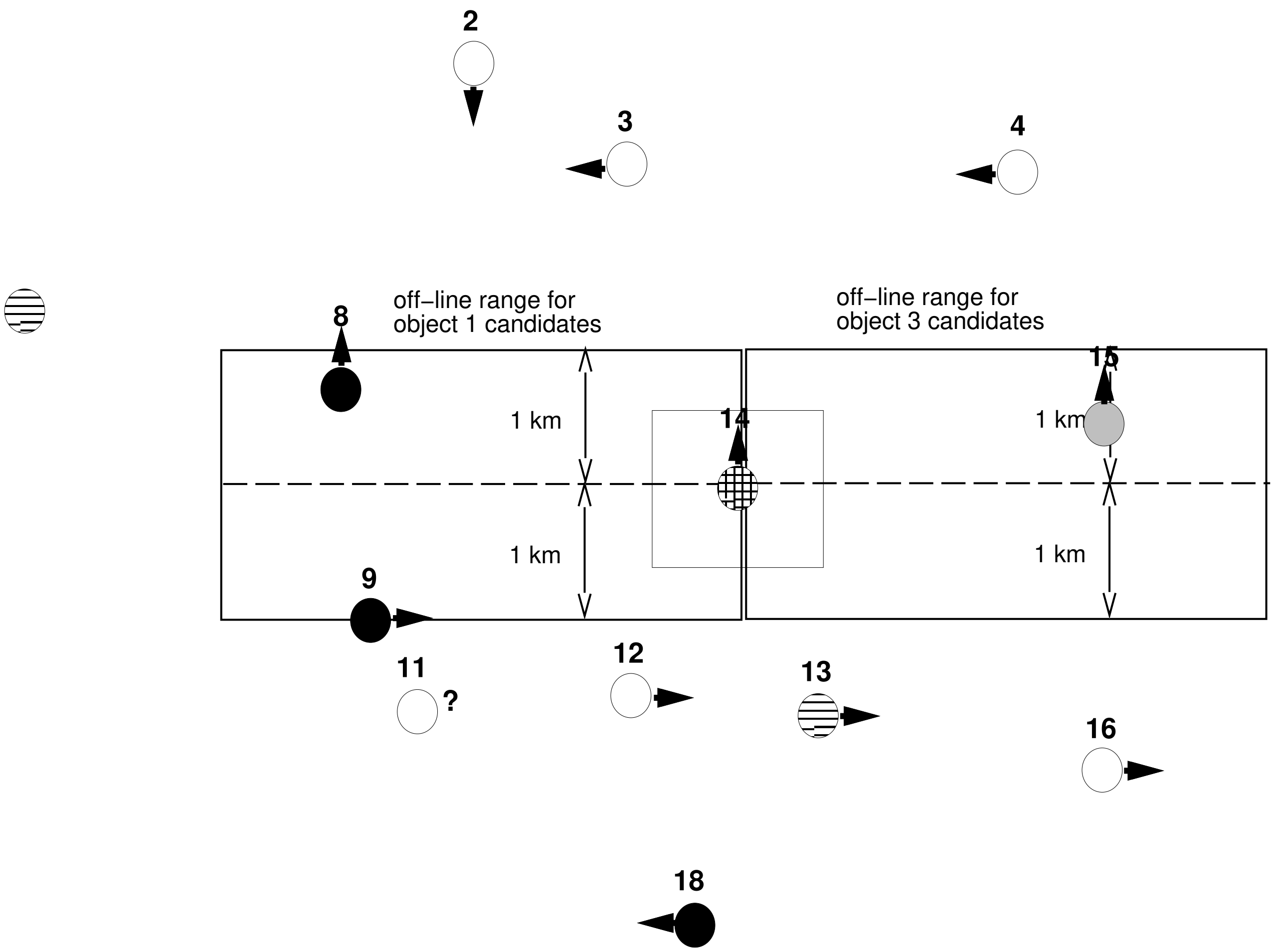

Figure 2.3: Off-line range of 1 km for instance 14.

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.

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 2.1, given instances of object 1 and also of object 3, we might define a multiple key spatial locality as shown in Figure 2.4. 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 2.5. 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, my 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

[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.

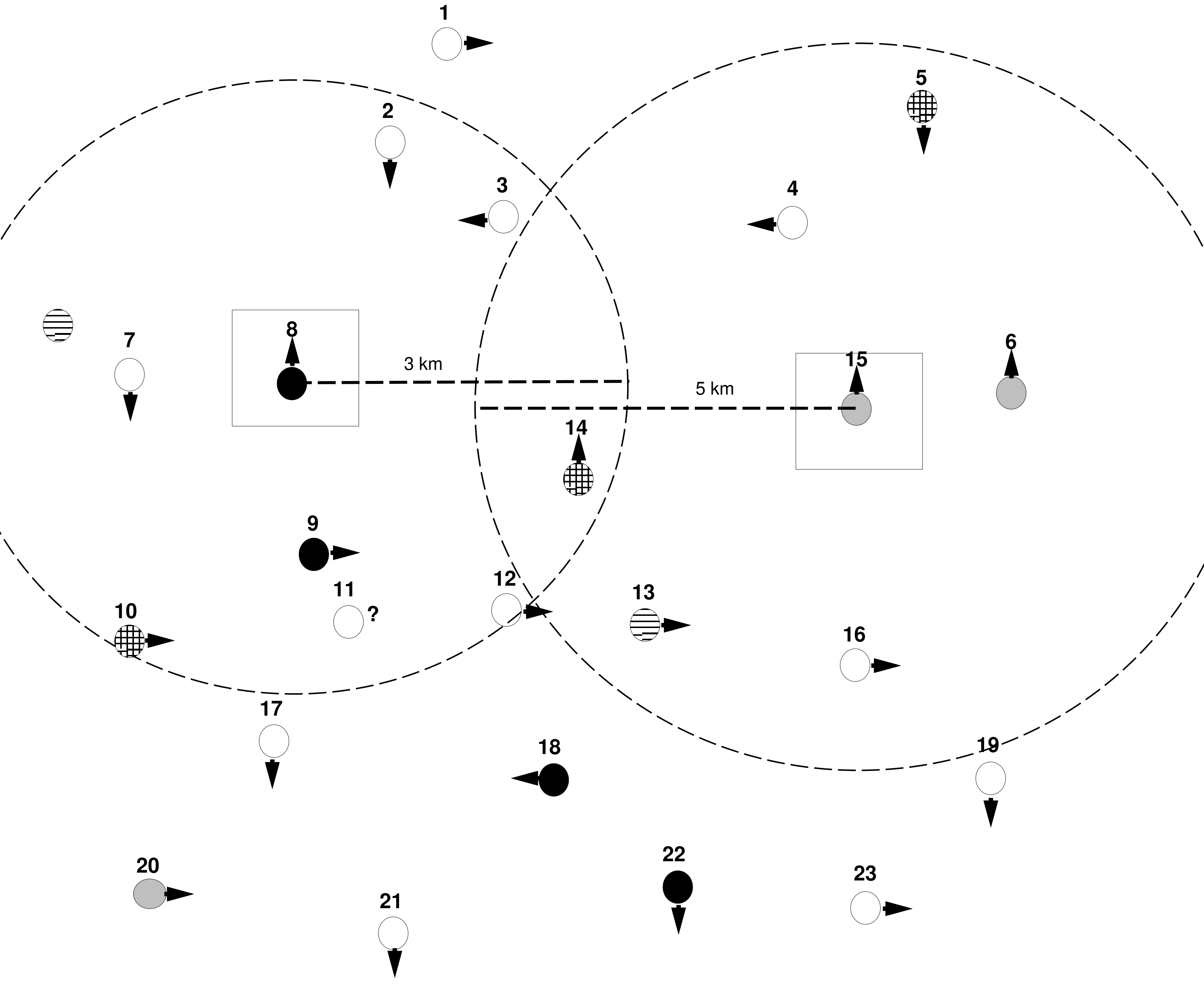

Figure 2.4: Overlapping radii for instances 8 and 15.

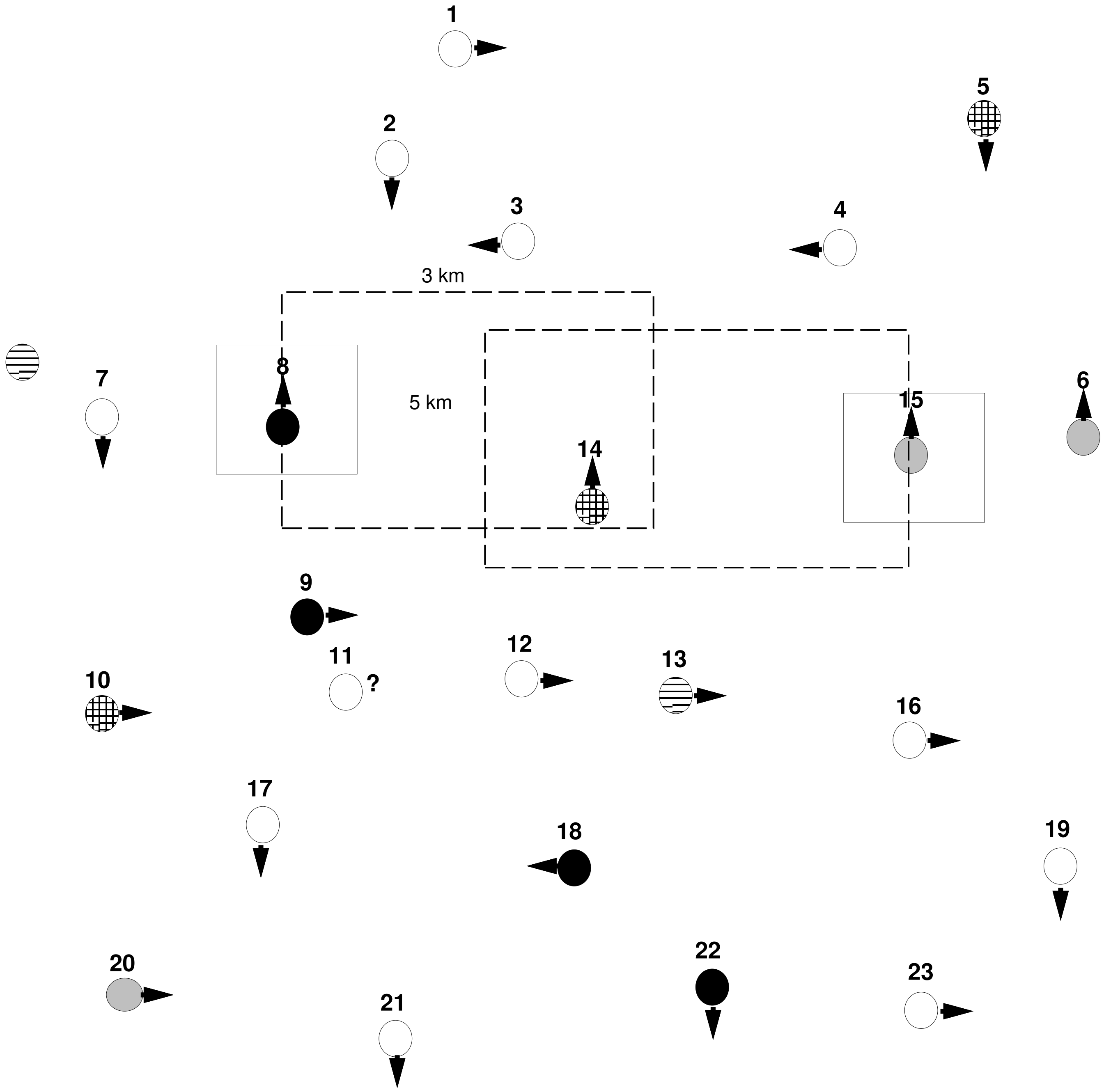

Figure 2.5: Combining radius and off-line range constraints.

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

2.1 Necessary spatial operations

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 some of the needs we have identified.

2.1.1 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*. It is also Spatial location may be viewed as such an attribute, and consequently the location may only be known with some (arbitrary ?) area or region rather than a precise point. For the purposes of this work we have assumed a known spatial location.

2.1.2 Output

We have identified several spatial functions crucial to quick application of algorithms to restrict the locality of template object instance searches. These have included:

- 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.

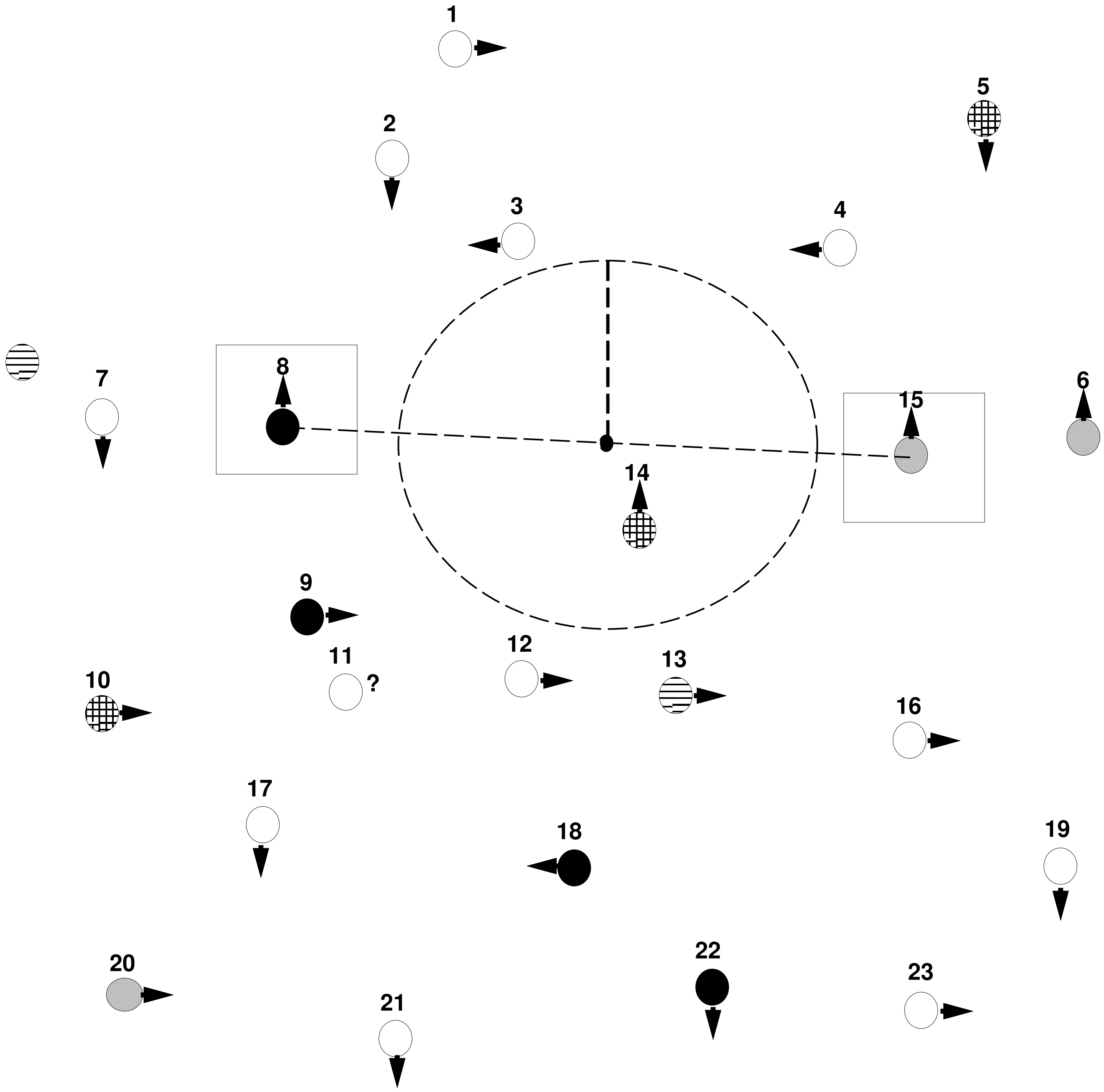

Figure 2.6: Midpoint locality for instances 8 and 15.

Chapter 3

Spatial Indexing Methodologies

In this chapter we will summarize in some detail what is meant by spatial locality in terms of geometric configurations in two dimensions, and explain how this locality can be accessed efficiently through known forms of spatial indices.

3.1 Preprocessing spatial data

In this section we wish to outline the important efficiency issues that arise from utilizing an additional structure as a spatial index. We examine how we may manipulate raw spatial situation information into a format suitable for efficient spatial indexing and operation application.

There are two obvious possibilities for maintaining the spatial data in an index structure. Either the data is maintained consistenctly within this 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 requeste (pay all of the processing cost at query time).

- Spatial indexing with continuous update. What is the cost of insertions and deletions ?
- Spatial indexing with one-time index construction. 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 horendous balancing overhead) if possible.

3.2 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. Perhaps for the more expensive (?) function applications this may be a consideration.

3.3 Experimentation

In this section we intend to provide actual numerical examples of situations demonstrating the benefit or lack of same of applying spatial locality, and preliminary analysis of results leading to either a positive conclusion and recommendation about how to proceed in terms of implementation, or a negative conclusion and explanation. The primary goal of this work is to determine if spatial locality can be readily applied to template recognition to noticeably reduce the cost of template instance identification. Additionally, we must show how this can be achieve, or else explain why it cannot.

We observe immediately that a spatial index system is clearly going to be an improvement over no such system, the question is : Is the additional work worth the effort in terms of response time gain?

Chapter 4

Exploiting Spatial Indexing to enforce spatial locality

In this chapter 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 (Samet 1990) 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. A 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 not that *unlike the point quadtree*, a complete binary k-d tree may *always* be constructed. It is important to note that 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. Samet (1990) 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 that 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. Extensions exists reducing the time requirements to $O(log_2 n + F)$ with the same space requirements using a *priority search tree* structure.

- *Region-based QuadTree*, is a hierarchical data structure (actuall 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 iss $O(F + 2^n)$ 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 4.1 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.

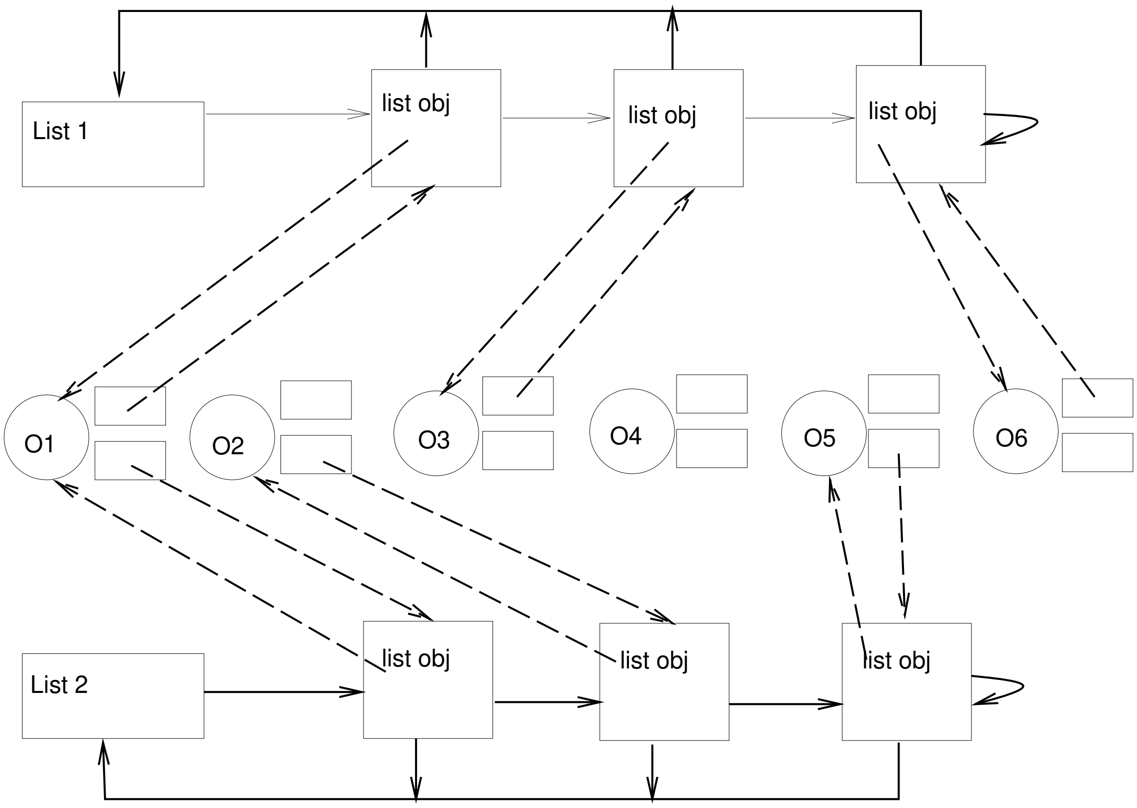

Figure 4.1: Simple list structure for computing intersection.

Arbitrary rectangle instance return

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

4.1 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.

- It is crucial to note that 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.

- Actually this is only a special case 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 ivolving 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 element from 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 size of Variables i..j based on some intersection of the ranges of the domain of VariableK. 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 clearly 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.

Chapter 5

Primary literature listing

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.

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] claim that they remain pretty much comprehensive.

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

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

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

[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.

Nearest neighbor algorithms

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

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, p.129.

Indexing and spatial databases

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.

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.

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.

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

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.

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.

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.

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.

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*, Advances in Spatial Databases, 3rd edition, pages 465–476. Springer Verlag, June 1993. Lecture Notes in Computer Science.

Spatial locality

R. G. M. Morris. Spatial localization does not require the presence of local cues. *Learning and Motivation*, 12:239–260, 1981.

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

Spatial model representation and recognition

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

David M. Chelberg and An doo Yang. Improving the results of object recognition and localization using probabilistic methods. work reported online at WWW site, 1994.

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.

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

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.

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.

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

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

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.

Y. Lamdan and H. J. Wolfson. Geometric Hashing: A General and Efficient Model-Based Recognition Scheme. In *Proc. Second International Conference on Computer Vision*, pages 238–249, 1988.

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

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.

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.

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

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

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

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.

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

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.

Spatial reasoning

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

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

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.

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, Sept. 1992.

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.

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

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

Background for spatial problems

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

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

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

Recommendations

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 K-D tree is the best selection for a stand-alone spatial index. Ideally, this index is maintained continuously as elements are added to the situation so that at *query* time (ie a request is made to template-match), the structure is constructed and may be immediately utilized.

These algorithms are currently "stand-alone" apart from any larger system or database for proof-of-concept purposes. Since the author is familiar with STR-CSP, the improved spatial functions can be tested in this application with little difficulty. This combined implementation may be best viewed as a prototype of the possible functinality 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.

Commerical packages providing this functionality are certainly one possible option to consider as an alternative to a specific data index structure implementation for the matching funtion which (may) become part of the larger TIF system. However, one would imagine that this type of investment in technology might be excessive. Investigation of the actual GIS data structures in TIF might provide insight into the power already provided, and make redundant data structure construction for a particular function such as template matching.

Acknowledgments

This research has been carried out as part of a contractual agreement between DMR, Steven Woods, and the Defence Research Establishment Valcartier, Valcartier, Quebec City, Quebec, under the auspices of Stephane Lapointe, Defence Scientist.

Note that an understanding has been reached between the parties involved that some of the research presented in this document may overlap with the production of a Ph.D. thesis by Steven Woods, and that rights to publication

are reserved by Steven Woods for that purpose in such areas. These areas are specifically those which describe, implement, or relate to unique search algorithms or heuristics which are independent of the domain of application of concern in this paper.

Bibliography

Cooper, M. C. (1989), 'An optimal k-consistency algorithm', *Artificial Intelligence* **41**, 89–95.

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

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

Kaufmann, A. (n.d.), *Introduction to the theory of fuzzy sets*, Vol. 1, Academic Press Inc.

Kumar, V. (1992), 'Algorithms for constraint-satisfaction problems', *AI Magazine* pp. 32–44.

Lapointe, S. & Prouix, R. (1994), Fuzzy geometric relations to represent hierarchical spatial information, *in* 'Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence'. to appear.

Lapointe, S. & Proulx, R. (1994), Statistical identification of templates in message data, Contract Report W7701-3-0847/01-XSK, APG inc. (1994). Combat Intelligence Automation Group, Defence Research Establishment Valcartier.

Mackworth, A. (1977), 'Consistency in networks of relations', *Artificial Intelligence* **8**, 99–118.

Mohr, R. & Henderson, T. (1986), 'Arc and path consistency revisited', *Artificial Intelligence* **28**, 225–233.

Nadel, B. A. (1989), 'Constraint satisfaction algorithms', *Computational Intelligence* **5**, 188–224.

Prosser, P. (1993), 'Hybrid algorithms for the constraint satisfaction problem', *Computational Intelligence* **9**(3), 268–299.

Proulx, R. (1994), Fuzzy representation and matching of spatial templates, DREV Technical Report W7701-3-0847/01-XSK, Defence Research Establishment Valcartier.

Samet, H. (1990), *The Design and Analysis of Spatial Data Structures*, Addison-Wesley.

Stacey, D. & Wong, A. (1988), A largest common subgraph isomorphism algorithm for attributed graphs, University of Waterloo, Systems Design Engineering.

Van Hentenryck, P., Deville, Y. & Teng, C.-M. (1992), 'A generic arc-consistency algorithm and its specializations', *Artificial Intelligence* **57**, 291–321.

van Run, P. (1994), Domain independent heuristics in hybrid algorithms for CSPs, Master's thesis, University of Waterloo.

Wong, A., You, M. & Chan, S. (1990), 'An algorithm for graph optimal monomorphism', *IEEE Transactions on Systems, Man, and Cybernetics* **20**(3), 628–636.

Woods, S. (1993*a*), Interactive recognition of spatially defined model deployment templates, Research memorandum, Combat Intelligence Automation Group, Defence Research Establishment Valcartier, Courcelette, Quebec CANADA. unclassified.

Woods, S. (1993*b*), A method of interactive recognition of spatially defined model deployment templates using abstraction, *in* 'Proceedings of the 1993 DND Workshop on Advanced Technologies in Knowledge Based Systems and Robotics'.

Woods, S. (1994*a*), Addressing spatial template recognition as a csp within the context of clp, advance copy, technical report.

Woods, S. (1994*b*), An integrated model of program understanding and program template recognition as as constraint satisfaction, University of Waterloo PhD thesis proposal, under revision.

Woods, S. (1994*c*), Learning refinements to spatial template models based on false-positive and false-negative examples, advance copy, technical report.

Woods, S. & Yang, Q. (1994), Program understanding as constraint satisfaction, Submitted for publication.

Woods, S. & Yang, Q. (1995*a*), A constraint-based approach to program plan recognition, submitted for publication.

Woods, S. & Yang, Q. (1995*b*), Program understanding as constraint satisfaction, submitted for publication.