AN EVOLUTIONARY FRAMEWORK FOR MATCHING GEOSPATIAL OBJECT CONFIGURATIONS

A Thesis presented to the Faculty of the Graduate School

University of Missouri

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

---

by

ANDREW R. BUCK

Dr. James Keller, Thesis Supervisor

MAY 2012

The undersigned, appointed by the dean of the Graduate School, have examined the thesis entitled

AN EVOLUTIONARY FRAMEWORK FOR MATCHING

GEOSPATIAL OBJECT CONFIGURATIONS

Presented by Andrew R. Buck

A candidate for the degree of Master of Science

And hereby certify that in their opinion it is worthy of acceptance.

Dr. James Keller, Ph.D.

Dr. Marjorie Skubic, Ph.D.

Dr. Mihail Popescu, Ph.D.

ACKNOWLEDGEMENTS

I would like to thank Dr. James Keller for introducing me to this project. His outstanding support, guidance, and patience have helped me become a successful graduate student. I am thankful for the help and assistance of Dr. Marjorie Skubic, who provided the initial project assignment from which this work grew. I am also grateful to Dr. Mihail Popescu for serving on my committee and providing insight to various problems. Finally, Dr. Ozy Sjahputera deserves credit for helping me understand the technical aspects of the histograms of forces, and for explaining the underlying code.

This work was funded by the U.S. National Geospatial-Intelligence Agency NURI grant HM 1582-08-1-0020.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

AN EVOLUTIONARY FRAMEWORK FOR MATCHING GEOSPATIAL OBJECT CONFIGURATIONS

Andrew R. Buck

Dr. James Keller, Thesis Supervisor

ABSTRACT

This thesis presents a framework for modeling and comparing the spatial configuration of sets containing two-dimensional geospatial objects. This situation can arise in the conflation of a hand or machine drafted map to a satellite image, or in the correspondence problem of matching two images taken under different viewing conditions. We focus here on the specific problem of matching a sketched map containing several 2D objects to actual satellite imagery. Spatial relationships between objects are captured by the histograms of forces and used to construct an attributed relational graph representation of the scene. Scene matching is performed with an evolutionary algorithm, combined with a local-search heuristic. Four problem-specific mutation operators are developed and tested experimentally.

1. INTRODUCTION

1.1 Problem Statement

Geospatial intelligence is a growing field which seeks to describe and analyze spatial information about the earth. Objects and landmarks that appear in geospatial images can be related to each other by their spatial relationships. Several techniques have been developed for modeling spatial relationships which make it easy to describe object configurations using natural language. A statement such as "The building is to the right of the parking lot" conveys a spatial relationship between two objects using qualitative descriptors. The ambiguous nature of statements such as these calls for the use of fuzzy methods, such as the histograms of forces [Matsakis & Wendling, 1999], to describe spatial relations. Using the histograms of forces, a set of linguistic descriptions can be obtained from a fuzzy rule system [Matsakis, et al., 2001] and then used to construct a sketch depicting a set of objects and their spatial configuration [Sledge & Keller, 2009].

For the purposes of this work, a sketch is a simple representation of the spatial configuration of a group of geospatial objects. Apart from a small collection of labels such as "building" and "parking lot", the only defining features of these sketches are the shapes, sizes, and spatial relationships between the objects. Given a sketch and a geospatial database containing a large number of reference objects, our goal is to find the set of objects from the reference database which most closely matches the spatial configuration of the sketch. This is accomplished with an evolutionary algorithm and several problem-specific local search mutation operators. Although designed for sketch

matching, this method can be extended to any situation in which we must locate a specific arrangement within a large search area.

1.2 Overview

The focus of this work is the development of an evolutionary framework for matching geospatial object configurations. We begin by reviewing the theory of spatial relationships and how they can be modeled quantitatively. The histograms of forces are reviewed and the concept of a main direction is defined. We then discuss how spatial relationships can be used for scene matching and describe how sketches and the geospatial reference database can be created. The concept of scene matching is cast as a subgraph isomorphism problem and we describe how spatial configurations can be represented as attributed relational graphs. A brief history of graph matching is given and the VF2 subgraph isomorphism algorithm is described in detail. The overall matching algorithm is then presented in terms of an evolutionary framework. By using a local search operator to improve candidate solutions, the framework becomes a type of memetic algorithm.

The design of the matching algorithm begins by defining a similarity measure between two sets of objects based on their spatial configuration. This is accomplished by using attributed relational graphs in which each object is represented as a node of the graph, and the force histogram relationships between objects are stored as edge attributes. A rotation-invariant similarity measure is defined which is used to compute the fitness value of each candidate solution. The evolutionary algorithm creates a population of

individual object sets in the reference database which could each potentially match the spatial configuration of the sketch. Each candidate solution is improved through one of four possible local search mutation methods. These methods are described in detail and are tested in the experiments section.

The first set of experiments compares the different mutation methods. This is done for both resubstitution sketches which come directly from the reference database, and simplified sketches which have undergone some simplification and rotation. The second set of experiments investigates the impact of sketch size on matching performance. Sketches containing between 4 and 12 objects are matched onto the reference database using one of the two leading mutation methods. The results show that the evolutionary framework can successfully locate high quality matches of a sketch, and that some mutation operators are better suited for different problem sizes.

2 LITERATURE REVIEW

2.1 Spatial Relationships

The concept of space is so intrinsically fundamental to human nature that we often accept its existence as part of everyday life without a second thought. Yet the psychological aspect of how we perceive space has taken on new meaning as we seek to give machines more human-like capabilities. As humans, we can easily identify distinct objects in a scene and understand how they are related spatially. For computers, this is not such a simple task, and object recognition has been one of the key pillars of computer vision since the origin of the field. By defining regions of an image as distinct objects, a computer can begin to perform qualitative reasoning about the image and the objects it contains. The spatial organization of objects in an image is an important high-level feature that can be used to represent scenes and provide a way to compare and communicate scene content.

Some of the earliest work with spatial relations in regard to computer vision is credited to Winston, who in [Winston, 1975] developed a machine algorithm for recognizing the spatial relationships between simple 3D block models, represented as line drawings. In his work, object relationships were deduced from a set of crisp rules and were combined in a graph structure to create an entire scene description. In [Freeman, 1975], Freeman studied the essential spatial relationships required to describe a scene. He proposed 13 primitive spatial relations: 1) LEFT OF, 2) RIGHT OF, 3) ABOVE, 4) BELOW, 5) BEHIND, 6) IN FRONT OF, 7) BESIDE, 8) NEAR, 9) FAR, 10) TOUCHING, 11) BETWEEN, 12) INSIDE, and 13) OUTSIDE. Freeman also noted that

these relations can be difficult to define and are context sensitive. He was among the first to suggest the use of fuzzy sets to capture the inherent uncertainty in their meaning.

Rosenfeld developed several methods for evaluating the spatial properties of image objects as fuzzy sets [Rosenfeld, 1979], [Rosenfeld, 1983], [Rosenfeld, 1984]. He defined terms such as the connectedness, adjacency, and surroundedness of objects using fuzzy relations. These efforts were further generalized in [Dubois & Jaulent, 1987]. Keller and Szatendera [Keller & Sztandera, 1990] used fuzzy sets to evaluate the relative position of objects by comparing their projections onto the principle axes.

Geographic Information Systems (GIS) have played a key role in developing the theory of spatial relations. In [Egenhofer & Franzosa, 1991], Egenhofer studied the topological relationships between 2D objects and defined the 9-intersection model for spatial relations. This model creates a 3x3 matrix which represents the intersecting interior, exterior, and boundary regions between two objects. In [Mark & Egenhofer, 1994], human test subjects were presented with various configurations of a road intersecting a park and asked to divide the examples into similar groups. The results show that the 9-intersection model provides a sound basis for representing line-region topological spatial relationships.

Although topology can often characterize the spatial relationships of intersecting objects, direction and distance are usually more descriptive measures for non-intersecting objects. As a simplification of the work in [Dubois & Jaulent, 1987], Krishnapuram et al. [Krishnapuram, et al., 1993] developed an aggregation method for assessing the spatial properties and relationships of fuzzy image regions. These are viewed as possibility

distributions defined over the α-cut sets of the fuzzy regions. Using this method, they are able to mathematically define the primitive spatial relations defined in [Freeman, 1975].

A histogram-based approach is presented in [Miyajima & Ralescu, 1994], which models the relative angles between objects in raster images. Given two objects A and B, Miyajima and Ralescu consider all pairs of points (a, b) where a is a point in A and b is a point in B. The angle defined by the pair (a, b) is recorded on a histogram of angles, which can be compared to predefined fuzzy sets of the primitive directions using a compatibility method. A comparison of the various fuzzy methods for generating fuzzy spatial relationships was made in [Keller & Wang, 1995].

2.2 Histograms of Forces

As a generalization of the histograms of angles, the histograms of forces [Matsakis & Wendling, 1999] provides a solid mathematical foundation for evaluating the spatial relationship between a pair of 2D objects. This method can process both raster and vector data, and also has the capability to evaluate fuzzy objects. The framework allows for the computation of multiple spatial representations, such as the histograms of constant and gravitational forces. The histogram of constant forces provides a global perspective and is similar to the histogram of angles, whereas the histogram of gravitational forces places more emphasis on regions that are close to one another.

2.2.1 Definition of the F-Histogram

The spatial relationship between a pair of two-dimensional objects A and B can be represented by the forces acting between them. For every direction θ, the sum of elementary forces acting between A and B in direction θ are computed (Figure 2.1). These forces are aggregated into the F-histogram $F_r^{AB}(\theta)$, which maps $\mathbb{R} \to \mathbb{R}^+$ and represents the degree of support for the proposition, "A is in direction θ of B." Provided that A and B are both non-empty regions and θ is evaluated on a fine enough scale, F_r^{AB} will have at least one element greater than zero. The magnitude of the individual forces are calculated as an inverse ratio of d^r, where d represents the distance between the points of A and B, and r provides a way of capturing different information. When $r = 0$, we obtain the histogram of constant forces (F_0), which provides a global perspective, independent of the distance between A and B. When $r = 2$, we obtain the histogram of gravitational forces (F_2), which gives a local view, more sensitive to nearby points, but independent of global scale.

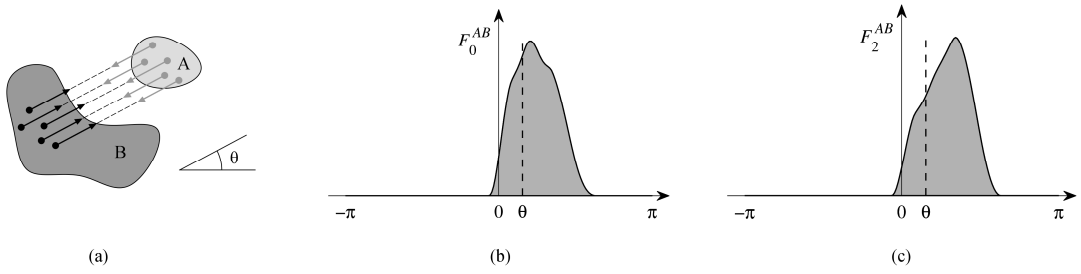


Figure 2.1 Calculation of the histograms of forces. (a) A force histogram F_r^{AB} is the scalar resultant of elementary forces exerted by the points of A on those of B. Each one pulls B in direction θ. (b) The histogram of constant forces $(r = 0)$ is one representation of the spatial relationship between A and B providing a global perspective. (c) The histogram of gravitational forces $(r = 2)$ is another possible representation, which is more sensitive to nearby points.

2.2.2 Main Direction

In many instances, it is useful to reduce the spatial relationship between two objects to a single scalar direction φ^{AB}, called the *main direction*. This can be accomplished in a variety of ways, such as measuring the angle between the centroids or bounding boxes. However, for complex shapes this can produce results which are inconsistent with human intuition. In [Matsakis, et al., 2001], a method is presented which uses both the F_0 and F_2 histograms to evaluate the degree of support for the proposition, "A is in direction θ of B." This can be used to create an accurate linguistic description, or to find the angle which maximizes the degree of support. The use of both constant and gravitational force histograms is especially important in cases where they would by themselves indicate different primary directions such as in (Figure 2.2).

For each angle θ, the forces of F_r^{AB} are categorized as effective, contradictory, or compensatory. Effective forces are those which support the proposition, "A is in direction θ of B," and contradictory forces are those which oppose it. Some effective forces may be relabeled as compensatory forces to help balance the contradictory forces. From these sets of forces, four values are computed, $a_0^{AB}(\theta)$, $a_2^{AB}(\theta)$, $b_0^{AB}(\theta)$, and $b_2^{AB}(\theta)$. Here, a_r^{AB} represents the calculated degree of truth according to the F-histogram F_r^{AB}, and b_r^{AB} represents the percentage of forces which are effective. Details of this computation can be found in [Matsakis, et al., 2001]. By evaluating all directions, the main direction histogram is defined as

$$\Phi^{AB}(\theta) = \max\{a_0^{AB}(\theta), \min\{a_2^{AB}(\theta), b_0^{AB}(\theta)\}\}. \tag{2.1}$$

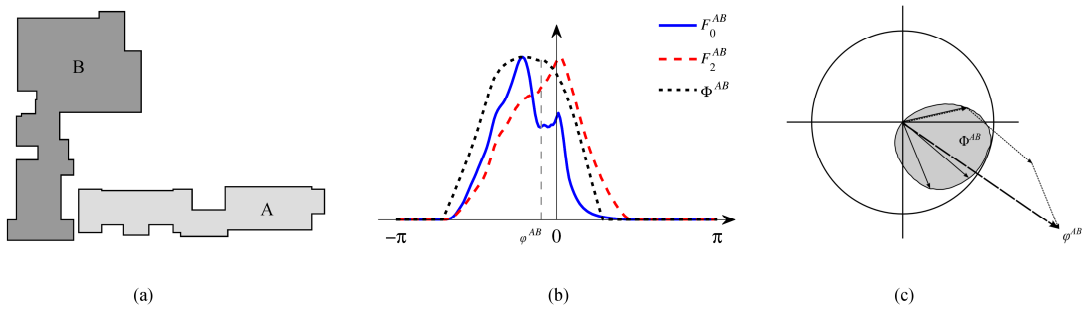


Figure 2.2 Calculation of the main direction. (a) A pair of objects for which the constant and gravitational force histograms indicate different primary directions. (b) The F_0^{AB} and F_2^{AB} histograms can be combined into the main direction histogram, Φ^{AB}. The centroid of this histogram gives the scalar main direction φ^{AB} which is a compromise between the primary directions of the two F-histograms. (c) φ^{AB} is computed using polar vector summation where each angle of Φ^{AB} is treated as a vector. By summing all of the vectors and computing the resultant angle, we avoid the problem of the periodic boundary.

Skubic et al. define the main direction as the direction θ for which $\Phi^{AB}(\theta)$ is maximum [Skubic, et al., 2004]. A more robust approach is to use the centroid of $\Phi^{AB}(\theta)$, which is the method used in the remainder of this work. Because Φ^{AB} is a periodic function, polar vector summation must be used (Figure 2.2c) when computing the centroid to ensure that all directions are treated equally [Fisher, 1993]. This is especially true for cases in which A surrounds B or vice versa for which there is no suitable 2π range of Φ^{AB} that could serve as a linear mapping. The main direction is defined as

$$\varphi^{AB} = \operatorname{atan2}\left(\sum_{\theta\in[0,2\pi]} \sin\left(\Phi^{AB}(\theta)\right), \sum_{\theta\in[0,2\pi]} \cos\left(\Phi^{AB}(\theta)\right)\right) \tag{2.2}$$

where $\operatorname{atan2}(y,x): \mathbb{R}\times\mathbb{R} \rightarrow [0,2\pi)$ is the two-argument variation of the arctangent function. Although the main direction could be computed from just the centroid of either the F_0 or F_2 histogram, using a common value reinforces the fact that the two histograms

are a pair with a single reference axis. This gives a unified framework that is consistent with the natural human interpretation of spatial relationships.

2.3 Text to Sketch

One of the principle motivations for studying spatial relationships is to simplify the human-computer communication barrier for describing and representing scenes. Linguistic interpretation of scene content allows both humans and computers to interact using the same language. Fuzzy methodologies are particularly useful in this context for their ability to model the inherent uncertainty in a linguistic spatial relationship. In [Keller, et al., 1999] and [Keller & Wang, 2000], a fuzzy rule base is used to generate linguistic descriptions of scenes using LADAR imagery. This method is refined in [Matsakis, et al., 2001], which presents a robust algorithm for creating linguistic descriptions from force histograms. This method is used in [Skubic, et al., 2004] to interpret hand-drawn sketches and provide a navigable route for a robot through a scene using a set of linguistic rules.

Sketches are a useful way to communicate spatial content. Although a sketch contains quantitative information, it is often drawn to represent only qualitative spatial relationships. The task of building a quantitative sketch from a qualitative linguistic description is called "Text to Sketch" (T_2S) [Sledge & Keller, 2009]. The inverse of this problem is solved using the histograms of forces and the fuzzy rule method in [Matsakis, et al., 2001]. The resulting linguistic description can be used to verify the quality of the sketch created by the T_2S system. To construct a sketch, individual descriptions are

modeled as fuzzy region templates [Bloch, 1999]. These images represent the degree to which a region matches a particular linguistic description. The objects described can then be placed at the most probable locations and the resulting image tested for linguistic similarity to the original description. This iterative process allows very simple sketches to be created using only linguistic spatial information.

To create more complicated sketches involving actor movement or inter-object relationships, some additional natural language processing is required. Logical form graphs can be parsed directly from sentences to create deep semantic representations [Allen, et al., 2008]. These graphs contain all of the relations between words and the underlying objects they describe. Through additional processing, logical form graphs can be used to infer information about the actor and objects in the scene, as well as their relationships. This would allow for more complex sketch building techniques.

2.4 Scene Matching

A scene can be defined as a certain configuration of objects or image features. Often we are presented with two views of the same scene and wish to identify the correspondence between the two, or we may want to find an instance of one scene within a larger image. Scene matching is a high-level computer vision task which seeks to find corresponding regions in multiple images which represent the same scene.

The core requirement of scene matching is image registration, where scene elements of one image are assigned to those of another image. Several examples of

image registration algorithms are provided in [Brown, 1992]. In general, they fit into one of the following categories:

1. Techniques which use pixel values directly. In [Svedlow, et al., 1978], a cross-correlation measure is used for registering multiple views of a common scene in Landsat satellite images. While correlation methods are typically used for this class of problem, other similarity measures such as the sequential similarity detection algorithm [Barnea & Silverman, 1972] can be employed which improve the efficiency.
2. Techniques which use the frequency domain. As shown in [De Castro & Morandi, 1987] and [Reddy & Chatterji, 1996], the FFT of an image can be used for registration, and can be made invariant to affine transformations.
3. Techniques which use low-level image features, such as edges or keypoints. Edges can be used for registration as in [Wong, 1978], where an edge extraction algorithm and a sequential, hierarchical search method seek to maximize the cross-correlation of features. In [Lowe, 2004] the SIFT keypoint detector is used to find scene elements which can be identified across multiple viewing scales and orientations, providing a robust algorithm for scene matching at the image level.
4. Techniques which use high-level image features such as segmented objects and the relationships between objects. This work fits into this fourth type, and focuses on spatial relationships for scene matching.

In [Sjahputera, et al., 2000] the histograms of forces are used to evaluate the similarity between sets of linguistic spatial descriptions, such as those generated in

[Keller, et al., 1999] and [Keller & Wang, 2000] using LADAR imagery. Later in [Marjamaa, et al., 2001] and [Sjahputera, et al., 2003] scenes were compared directly using the set of all force histograms between objects as a scene descriptor. The force histograms were compared in order to recover the estimated sensor pose parameters between the two scenes. The effects of affine transformations on the histograms of forces were studied in [Matsakis, et al., 2004] and an affine-invariant force histogram representation was proposed. In [Sjahputera, 2004] normalized force histograms were used to generate scene descriptors representing all the spatial relationships between objects in a scene. A nearest neighbor (NN) method and a fuzzy sequential nearest neighbor (FSNN) method were introduced to build a correspondence map between two scene descriptors. This resulted in a one-to-one mapping between the objects in the two different scenes. A particle swarm optimization (PSO) algorithm was used in [Sjahputera & Keller, 2005a] for finding the best correspondence map between two scene descriptors and a possibilistic c-means (PCM) algorithm was used in [Sjahputera & Keller, 2005b] and [Sjahputera & Keller, 2007].

Scene matching techniques have also been used for robot path planning. In [Skubic, et al., 2003] a sketch interface is demonstrated using a PDA that can provide a robot with a hand-drawn map of object locations and a desired path. In [Skubic, et al., 2004] the histograms of forces are used to generate linguistic descriptions of relative object locations within a sketch. These are compared to a user-defined set of linguistic rules describing the desired robot path. In [Parekh, et al., 2007] and [Parekh, 2007], Parekh used the histograms of forces to build object correspondence maps between a

sketch and the observed environment. He also proposed a novel evolutionary algorithm for scene matching (EASM) and compared it to the FSNN method.

The geospatial community has been interested in scene matching for the task of locating a group of objects within a large geospatial database by means of a sketch. In [Bruns & Egenhofer, 1996], the spatial similarity of two scenes is evaluated in terms of topological, directional, and metrical properties. Using crisp definitions of these properties, scene similarity is computed by counting the number of gradual changes required to transform one scene into another. A spatial query language is used in [Egenhofer, 1997] to find scenes that match a user input sketch. Here, Egenhofer outlines a constraint relaxation method that emphasizes cognitively important criteria while suppressing aspects of lesser importance. In [Nedas & Egenhofer, 2008] the crisp spatial properties defined above are used to construct a graph-based representation of a scene. Scene matching is then cast as a constraint satisfaction problem, which has the ability to find a small sketch in a large scene using a constrained subgraph isomorphism. Graph-based matching methods are discussed in greater detail in Section 2.6.

2.5 Image Segmentation

The use of spatial relationships for scene matching with a geographic information system requires a certain amount of preprocessing on the geospatial database. A raw satellite image contains only pixel-level information. In order to compute spatial relationships between objects in the image, they must first be located. Segmenting buildings and objects in satellite imagery is a difficult task and requires the use of some

high-level features, such as texture. In [Shackelford & Davis, 2003a] a fuzzy classifier is proposed for segmenting high-resolution multispectral data over urban areas. This method uses both spectral and spatial properties to determine the appropriate fuzzy class for each pixel in an image. An expanded version of the classifier is given in [Shackelford & Davis, 2003b] which further classifies segmented regions using their structural and spatial properties.

An alternative approach to image segmentation is the energy minimization of contours which define segment boundaries. In [Chan & Vese, 2001] an active contour model is used with the level set method to perform segmentation. This is integrated with the use of shape priors in [Cremers, et al., 2006] and [Riklin-Raviv, et al., 2007] to locate objects seen from different viewing angles and with partial occlusions. The use of shape priors for locating objects in a satellite image is proposed in [Sledge, et al., 2011]. By using vector building outlines taken from a GIS database as shape priors, the level set method can produce crisp object boundaries and a reasonably accurate segmentation. In this thesis, exact object extraction for the reference data, generated either by hand or from a geospatial database, was used so that the effects of the matching process could be studied directly.

2.6 Graph-Based Methods

Graphs provide a powerful analytic and modeling tool used in many fields, including computer vision. They can be used to model a collection of objects and their individual relationships, making them very useful for representing spatial configurations.

By casting the scene matching problem as a graph search problem, we can make use of the many existing graph matching techniques.

2.6.1 Graph Definitions

The standard definition of a graph is an ordered pair $G = (V, E)$, where V is the vertex set and E is the edge set. A node $v_i \in V$ represents some structural entity which is related to other nodes by an edge $e_k = (v_i, v_j) \in E$. If each edge is an ordered pair, then the graph is a *directed graph*, implying that the relationships between nodes are not necessarily symmetric. A *complete graph* has an edge relationship between every pair of possible nodes such that $E = \{(v_i, v_j) \in V \times V\}$. The types of graphs used in the remainder of this work are all directed graphs, although not all are complete.

Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic to each other if there exists a bijective mapping $M: V_1 \rightarrow V_2$ which preserves the edge structure of the two graphs. Formally, M is an *isomorphism* if and only if every edge $(v_i, v_j) \in E_1$ has a unique corresponding edge $\big(M(v_i), M(v_j)\big) \in E_2$. A graph $G_1' = (V_1', E_1')$ is a *subgraph* of G_1 if $V_1' \subseteq V_1$ and $E_1' \subseteq E_1$. If G_2 is isomorphic to G_1', then the mapping $M: V_1' \rightarrow V_2$ is a *subgraph isomorphism* between the two graphs. Our goal for scene matching is to find a subgraph of G_1 (the reference database) that is isomorphic to G_2 (the sketch).

2.6.2 Attributed Relational Graphs

The usefulness of graph matching as a tool for scene correspondence can be greatly enhanced by adding attributes to the nodes and edges. Such graphs are known as attributed relational graphs (ARGs) and are a modest, but powerful extension to the standard graph definition. Work done in [Tsai & Fu, 1979] shows how modeling a pattern as a graph of primitives and relationships, each with a set of attributes, can be used to find matching patterns. A standard graph isomorphism compares only the syntactic aspect of graphs, and is intolerant of structural differences. However, by comparing attributes and allowing for small errors in the graph structure, one can ease the constraints of the search, which is very useful in computer vision.

An ARG is formally defined as a 4-tuple $G = (V, E, A_V, A_E)$, where V and E follow the same definitions as before, A_V is a set containing a unary attribute a_i for each node $v_i \in V$, and A_E is a set containing a binary attribute a_{ij} for each edge $(v_i, v_j) \in E$. The attributes a_i and a_{ij} can both be further represented as vectors containing multiple attributes for each node and edge. Two ARGs are considered isomorphic only if their structural graphs are isomorphic and the associated attributes between nodes and edges are compatible. Depending on the problem, numeric or semantic attributes can be compared directly or with some window of tolerance. Alternatively, more complex attributes may require compatibility functions to determine if two node or edge attributes are congruent. The process of finding a match to a pattern ARG will be discussed in the next section.

2.6.3 Graph Matching

Graph matching in general is an area of active research and many techniques have been developed. In [Conte, et al., 2004], an attempt is made to classify the many different approaches into the broad categories of exact and inexact matching methods. Exact matching methods must define a one-to-one correspondence between the nodes and edges of two graphs, whereas an inexact method may alter the graph structure or attributes in order to find a best match. Some methods can only handle graph isomorphisms or unlabeled graphs, while others can handle subgraph isomorphisms and fully labeled ARGs.

One of the earliest and most influential algorithms in this field is due to Ullmann [Ullmann, 1976], which can find both graph and subgraph isomorphisms. The technique, like many of the others that follow, implements a depth-first tree search with branch and bound. These methods work by recursively finding all pairs of compatible nodes between graphs, and adding additional compatible nodes to these subgraphs one at a time until a complete match is found. Ullmann's method includes a refinement procedure which seeks to remove incompatible node pairs as early as possible, thereby reducing the search time of the algorithm.

By working in terms of node and edge compatibility, tree search methods are easily extendable to ARG matching. In [Shapiro & Haralick, 1981] an object is represented by a structural description, which is stored as an ARG. A method is developed for evaluating ARG similarity by comparing the attributes of the two ARGs. The tree search method is further refined by including look-ahead rules to ensure that only good paths are followed in the search.

2.6.4 VF2 Algorithm

The VF algorithm [Cordella, et al., 1998], [Cordella, et al., 1999], is a more recent graph matching algorithm, capable of finding both isomorphisms and subgraph isomorphisms, although it is particularly suited for finding a subgraph isomorphism between a small sample graph and a large reference graph. An updated version was later introduced as the VF2 algorithm [Cordella, et al., 2004]. Given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, the algorithm seeks to find a mapping $M = \{(n, m) \in V_1 \times V_2 | n \text{ is mapped onto } m\}$ to represent either an isomorphism between G_1 and G_2, or a subgraph isomorphism between a subgraph of G_1 and G_2. This is accomplished using a State Space Representation (SSR) in which each state contains a partial mapping $M(s)$ which is a subset of the complete mapping function M. $M(s)$ uniquely defines the intermediate subgraphs $G_1(s)$ and $G_2(s)$, which contain only the nodes included in $M(s)$ and the induced edges. The sets $M_1(s)$ and $M_2(s)$ denote the projection of $M(s)$ onto V_1 and V_2 respectively. An example of a pair of graphs and their complete and partial mapping functions is given in Figure 2.3.

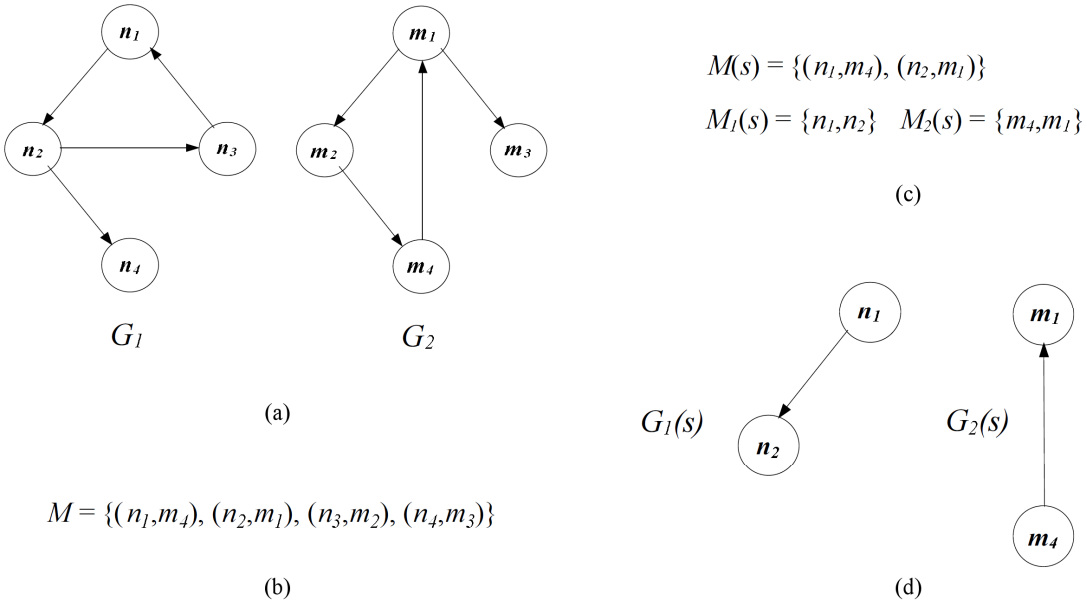


Figure 2.3 Example of the partial mapping function used in the VF algorithm [Cordella, et al., 1999]. (a) Two graphs G_1 and G_2. (b) The only possible mapping M. (c) A partial mapping solution $M(s)$ and the corresponding partial node sets $M_1(s)$ and $M_2(s)$. (d) The corresponding subgraphs $G_1(s)$ and $G_2(s)$.

The transition from state s to a new state s' involves the addition of a node pair (n, m) to the mapping function, resulting in new intermediate subgraphs. Typically, only a small number of node pairs can be added to $M(s)$ while maintaining consistency with the desired morphism type. Any node pair which would prevent $M(s)$ from growing into a completely defined mapping function represents an unfruitful path and the resulting branches of the SSR can be effectively pruned from the search space.

The basis of the VF2 algorithm is a set of feasibility rules for evaluating whether a node pair (n, m) can be safely added to a partial mapping function $M(s)$. A feasibility function is introduced

$$F(s, n, m) = F_{\text{syn}}(s, n, m) \wedge F_{\text{sem}}(s, n, m) \tag{2.3}$$

where $F_{\text{syn}}(s,n,m)$ is the syntactic feasibility, which depends on the structure of the graphs, and $F_{\text{sem}}(s,n,m)$ is the semantic feasibility, which depends on the attributes. A high-level outline of the entire matching algorithm is given in Figure 2.4. The initial input to the recursive matching function is the empty set $M(s_0) = \emptyset$, containing no matching elements. For each state that is evaluated, a set of candidate pairs $P(s)$ is generated and evaluated using the above feasibility function. The pairs which are considered feasible are added to the map, and the function is called again recursively. The final output of the algorithm occurs each time a complete map is generated, at which point the algorithm can either halt, or continue to search for all possible mappings.

VF2 Graph Matching Algorithm

Procedure: Match(G_1, G_2, s)
Input: Graphs G_1 and G_2
Intermediate state s; the initial state s_0 has $M(s_0) = \emptyset$

If $M(s)$ covers all the nodes of G_2 **Then**
    Output: $M(s)$
Else
    Compute the set $P(s)$ of the pairs candidate for inclusion in $M(s)$
    For Each $p = (n, m)$ in $P(s)$
        If $F(s, n, m)$ == TRUE **Then**
            Create a new state s' by adding p to $M(s)$
            Call Match(G_1, G_2, s')
        End If
    End For
End If

Figure 2.4 High-level outline of the VF2 graph matching algorithm [Cordella, et al., 2004].

To create the set of candidate pairs $P(s)$ we look at all of the adjacent nodes to $G_1(s)$ and $G_2(s)$. The sets $T_1^{\text{out}}(s)$, $T_1^{\text{in}}(s)$, $T_2^{\text{out}}(s)$, and $T_2^{\text{in}}(s)$ are defined as the outgoing or incoming nodes of $G_1(s)$ and $G_2(s)$ respectively, shown in Figure 2.5. The

set $P(s)$ consists of all pairs (n, m) in which $n \in T_1^{\text{out}}(s)$ and $m \in T_2^{\text{out}}(s)$, provided that neither of these sets are empty. If either of these sets are empty, then $T_1^{\text{in}}(s)$ or $T_2^{\text{in}}(s)$ is used instead. For disconnected graphs, these later sets may also be empty, in which case the sets $V_1 - M_1(s)$ and $V_2 - M_2(s)$ are used instead.

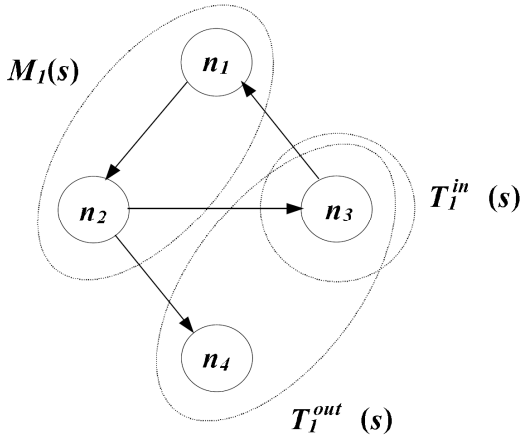


Figure 2.5 The sets $T_1^{\text{out}}(s)$ and $T_1^{\text{in}}(s)$ relative to $M_1(s) = \{n_1, n_2\}$ [Cordella, et al., 1999].

The syntactic feasibility rule is actually composed of five rules,

$$F_{\text{syn}}(s, n, m) = R_{\text{pred}} \wedge R_{\text{succ}} \wedge R_{\text{in}} \wedge R_{\text{out}} \wedge R_{\text{new}}, \tag{2.4}$$

which are explained in Table 2.1. The first two are necessary and sufficient conditions to ensure acceptable solutions, while the remaining three serve to prune the search space. R_{in} and R_{out} are 1-look-ahead rules and R_{new} is a 2-look-ahead rule. In defining the rules, some additional notation is used. $\text{Pred}(G, n)$ and $\text{Succ}(G, n)$ respectively denote the predecessor and successor nodes of node n in graph G. Additionally, $T_1(s) = T_1^{\text{in}}(s) \cup T_1^{\text{out}}(s)$ and $\tilde{V}_1 = V_1 - M_1(s) - T_1(s)$ with similar expressions for $T_2(s)$ and $\tilde{V}_2$. The syntactic feasibility rules are formally defined in [Cordella, et al., 2004] as

$$R_{\text{pred}}(s,n,m)$$
$$\Leftrightarrow \left(\forall n' \in M_1(s) \cap \text{Pred}(G_1,n) \exists m' \in \text{Pred}(G_2,m) \middle| (n',m') \in M(s)\right) \tag{2.5}$$
$$\wedge \left(\forall m' \in M_2(s) \cap \text{Pred}(G_2,m) \exists n' \in \text{Pred}(G_1,n) \middle| (n',m') \in M(s)\right),$$

$$R_{\text{succ}}(s,n,m)$$
$$\Leftrightarrow \left(\forall n' \in M_1(s) \cap \text{Succ}(G_1,n) \exists m' \in \text{Succ}(G_2,m) \middle| (n',m') \in M(s)\right) \tag{2.6}$$
$$\wedge \left(\forall m' \in M_2(s) \cap \text{Succ}(G_2,m) \exists n' \in \text{Succ}(G_1,n) \middle| (n',m') \in M(s)\right),$$

$$R_{\text{in}}(s,n,m)$$
$$\Leftrightarrow \left(\text{Card}\left(\text{Succ}(G_1,n) \cap T_1^{\text{in}}(s)\right) \geq \text{Card}\left(\text{Succ}(G_2,m) \cap T_2^{\text{in}}(s)\right)\right) \tag{2.7}$$
$$\wedge \left(\text{Card}\left(\text{Pred}(G_1,n) \cap T_1^{\text{in}}(s)\right) \geq \text{Card}\left(\text{Pred}(G_2,m) \cap T_2^{\text{in}}(s)\right)\right),$$

$$R_{\text{out}}(s,n,m)$$
$$\Leftrightarrow \left(\text{Card}\left(\text{Succ}(G_1,n) \cap T_1^{\text{out}}(s)\right) \geq \text{Card}\left(\text{Succ}(G_2,m) \cap T_2^{\text{out}}(s)\right)\right) \tag{2.8}$$
$$\wedge \left(\text{Card}\left(\text{Pred}(G_1,n) \cap T_1^{\text{out}}(s)\right) \geq \text{Card}\left(\text{Pred}(G_2,m) \cap T_2^{\text{out}}(s)\right)\right),$$

$$R_{\text{new}}(s,n,m)$$
$$\Leftrightarrow \text{Card}\left(\tilde{V}_1(s) \cap \text{Pred}(G_1,n)\right) \geq \text{Card}\left(\tilde{V}_2(s) \cap \text{Pred}(G_2,n)\right) \tag{2.9}$$
$$\wedge \text{Card}\left(\tilde{V}_1(s) \cap \text{Succ}(G_1,n)\right) \geq \text{Card}\left(\tilde{V}_2(s) \cap \text{Succ}(G_2,n)\right).$$

Table 2.1 Syntactic Feasibility Rules for Subgraph Isomorphism

Look-Ahead	Rule	Condition
0	R_{pred}	Iff for each predecessor n' of n in the partial mapping, the corresponding node m' is a predecessor of m, and vice versa.
	R_{succ}	Iff for each successor n' of n in the partial mapping, the corresponding node m' is a successor of m, and vice versa.
1	R_{in}	Iff the number of predecessors (successors) of n that are in $T_1^{\text{in}}(s)$ is greater than or equal to the number of predecessors (successors) of m that are in $T_2^{\text{in}}(s)$.
	R_{out}	Iff the number of predecessors (successors) of n that are in $T_1^{\text{out}}(s)$ is greater than or equal to the number of predecessors (successors) of m that are in $T_2^{\text{out}}(s)$.
2	R_{new}	Iff the number of predecessors (successors) of n that are neither in $M_1(s)$ nor in $T_1(s)$ (new models) is greater than or equal to the number of predecessors (successors) of m that are neither in $M_2(s)$ nor in $T_2(s)$.

The semantic feasibility rule will depend on the specific attributes of the graphs, but can be represented formally in terms of compatibility relations. Let $n \approx m$ represent a compatible pair of nodes, and $(n, n') \approx (m, m')$ represent a compatible pair of edges. The formal definition is then

$$\begin{aligned} &F_{\text{sem}}(s, n, m) \Leftrightarrow n \approx m \\ &\wedge \forall (n', m') \in M(s), (n, n') \in E_1 \Rightarrow (n, n') \approx (m, m') \\ &\wedge \forall (n', m') \in M(s), (n', n) \in E_1 \Rightarrow (n', n) \approx (m', m). \end{aligned} \tag{2.10}$$

The complexity of the VF and VF2 algorithms depend on the specific graphs being matched, but can be represented in terms of the best and worst case scenarios. In the best case, the VF algorithm has both a time and spatial complexity of $\Theta(N^2)$, where N is the number of nodes in the largest graph. In the worst case, the time complexity increases to $\Theta(N!\,N)$. The VF2 algorithm uses a common shared memory location during the tree search, reducing the memory requirement to just $\Theta(N)$ in all cases. This final property allows the VF2 algorithm to search for subgraph isomorphisms within very large reference graphs containing thousands of nodes. A summary of the complexities of the VF2, VF, and Ullmann's algorithm is given in Table 2.2.

Table 2.2 Spatial and Time Complexity of Different Graph Matching Algorithms

	VF2		**VF**		**Ullmann's Algorithm**	
Complexity	**Best Case**	**Worst Case**	**Best Case**	**Worst Case**	**Best Case**	**Worst Case**
Time	$\Theta(N^2)$	$\Theta(N!\,N)$	$\Theta(N^2)$	$\Theta(N!\,N)$	$\Theta(N^3)$	$\Theta(N!\,N^3)$
Spatial (memory)	$\Theta(N)$	$\Theta(N)$	$\Theta(N^2)$	$\Theta(N^2)$	$\Theta(N^3)$	$\Theta(N^3)$

2.7 Evolutionary Computation

2.7.1 Genetic Algorithms

An optimization problem is defined as the task of determining the best possible solution to a problem from the set all possible solutions. Mathematically, it can be defined as locating the maximum or minimum value of some function over some domain within certain constraints. This is a broad and often difficult problem, and many approaches have been demonstrated to work on some subset of optimization problems. The study of biology has introduced evolutionary computation as an approach to optimization, leading to the development of the genetic algorithm (GA) by Holland [Holland, 1975]. In a GA, as well as most evolutionary techniques, a possible solution to an optimization problem is encoded as a chromosome containing a set of genes or variables to be optimized. A population of chromosomes, each representing one possible solution, is created using a random initialization method. Each individual is evaluated by a *fitness function*, which produces a score representing how well it solves the optimization problem. Individuals are selected from the population using a *selection function* and used as the inputs to a *crossover function*, which combines features from the parent solutions and produces new child solutions. These new individuals may then be refined by a *mutation function*, which introduces random noise in order to explore new areas of the search space. The population of individuals is updated to include the new children, which usually involves replacing the parents with their children. Some form of elitism may also be used in selecting the subsequent generation, ensuring that the best-scoring individuals survive without modification. The process of selecting individuals

and producing new children constitutes one generation, and the search proceeds by computing additional generations until some stopping criteria is met. The choice of functions to carry out fitness evaluation, selection, crossover, and mutation is often problem specific and depends on the representation scheme being used.

Genetic algorithms provide a stochastic method for solving a variety of search and optimization problems [Goldberg, 1989]. In [Rodriguez & Jarur, 2005] a modified genetic algorithm is used for searching spatial configurations using a topological model. The algorithm is based on asexual reproduction, allowing a single parent solution to create a child. In contrast to an exhaustive search, the GA allows for a controlled computational cost, and can always provide a solution regardless of the problem complexity. However, the solutions of an exhaustive deterministic search are always optimal, whereas the GA may produce good, but suboptimal solutions.

2.7.2 Memetic Algorithms

There are many different variations of the standard genetic algorithm described above. In [Moscato, 1989] the concept of a memetic algorithm is introduced, which is a type of hybrid genetic algorithm. In these methods, a local search strategy is used to improve some individuals between generations. This allows the best individuals to continue improving locally, which results in shorter search times. In [Houck, et al., 1996] the hybrid GA is described in biological terms using Lamarckian evolution and the Baldwin effect. Lamarckian learning allows individuals to be replaced by their improved versions found as a result of a local search. These improvements are then preserved

through to the next generation. The Baldwin effect allows an individual to be evaluated using the fitness of its improved version, but the genetic representation remains unchanged. This is similar to what happens in actual biological systems and ensures that diversity is maintained around an optimal point. The hybrid GA is shown to converge more quickly than the standard GA on many problems. An outline for a generic memetic algorithm is shown in Figure 2.6.

Generic Memetic Algorithm
Initialize: Create initial population of individuals
While stopping criteria is not met
    Evaluate all individuals
    Evolve new population using selection/crossover/mutation
    Select a subset of individuals for local improvement, Ω_{il}
    For each individual in Ω_{il}
        Perform local improvement with probability f_{il} for a period t_{il}
        Proceed with Lamarckian or Baldwinian learning
    End For
End While

Figure 2.6 Generic memetic algorithm.

3 DESIGN OF THE MATCHING ALGORITHM

3.1 Problem Overview

Our goal is to develop an algorithm which can locate a group of objects within a geospatial image from an approximate sketch. The objects can be buildings, parking lots, or other landmarks of interest. Since the objects are disjoint and contain no information apart from a label and their spatial properties, we make use of the histograms of forces to model the spatial relationships. This provides a robust framework for capturing the relative direction, distance, scale, and to an extent the relative shapes between objects. The collection of HoF relationships between all objects in a sketch or scene are modeled as an ARG, which allows for efficient matching techniques. For this study, we chose to use hand-segmented imagery to ensure that we have the best possible ground truth. The segmented objects make up the reference set $\mathcal{R} = \{x_1, x_2, \dots, x_m\}$, where each object is given a label of either "building" or "parking lot."

A sketch of objects $S = (o_1, o_2, \dots, o_n)$ can be created in a variety of ways. If a person draws a map of his or her immediate surroundings, the resulting collection of labeled objects is a sketch which should match to some real location in the reference image (provided that they are standing in the segmented region.) This process could be automated by interpreting a natural language description of a person's surroundings [Sledge & Keller, 2009]. Regardless of how the sketch is generated, the goal of our algorithm is to find the mapping function $\Gamma: S \to \mathcal{R}$ that assigns each object of the sketch to an object in the reference set. Ideally, these will be the same objects that the sketch is intended to represent. Since there are only two different object types in this study, the

spatial relationships between objects will be the primary matching features. The best match for a given sketch is the set of objects from the reference set which most closely matches the spatial configuration of the objects in the sketch. Figure 3.1 shows an example of a sketch and the corresponding matching location within a segmented satellite image.

(a) (b)

Figure 3.1 Sketch example. (a) An example of a machine-drafted sketch. (b) The corresponding match within a segmented satellite image. Buildings are shown in red and parking lots are shown in green.

3.2 Representing Object Sets

We use an attributed relational graph to model the spatial relations between objects in a scene. Suppose that we have a scene consisting of a set of 2D objects $\mathcal{O} = \{o_1, o_2, \ldots, o_n\}$. Each object will represent a node in the ARG and the relationships between objects will be stored as edges. A single edge relationship between a pair of

objects is defined as $e_{ij} = (o_i, o_j) \in \mathcal{O} \times \mathcal{O}$. Depending on the size of $\mathcal{O}$, we may define all inter-object relationships as edges, or only some of the closest and most reasonable relationships. The set of all edges in a scene is defined as $E_{\mathcal{O}} = \{e_{ij} \mid (o_i, o_j) \in \mathcal{O} \times \mathcal{O}\}$. Each object is given a label $L_{\mathcal{O}}(o_i) = l_i \in \mathcal{L}$ where $\mathcal{L}$ is the set of all possible labels (*e.g.* "building", "parking lot", etc.). For each edge e_{ij} we define the spatial relationship as the triple $h_{ij} = (F_0^{o_i o_j}, F_2^{o_i o_j}, \varphi^{o_i o_j})$, where $F_0^{o_i o_j}$ and $F_2^{o_i o_j}$ are the constant and gravitational F-histograms between o_i and o_j, and $\varphi^{o_i o_j}$ is the main direction. (The h_{ij} notation was chosen as a mnemonic for "histogram," as the first two arguments are actually histograms.) The set of all spatial relationships is defined as $H_{\mathcal{O}} = \{h_{ij} \mid e_{ij} \in E_{\mathcal{O}}\}$, which implies that there must be a spatial relationship defined for each edge which exists in the graph. The complete ARG representation for the object set $\mathcal{O}$ is defined as $G_{\mathcal{O}} = (\mathcal{O}, E_{\mathcal{O}}, L_{\mathcal{O}}, H_{\mathcal{O}})$. An example is given in Figure 3.2.

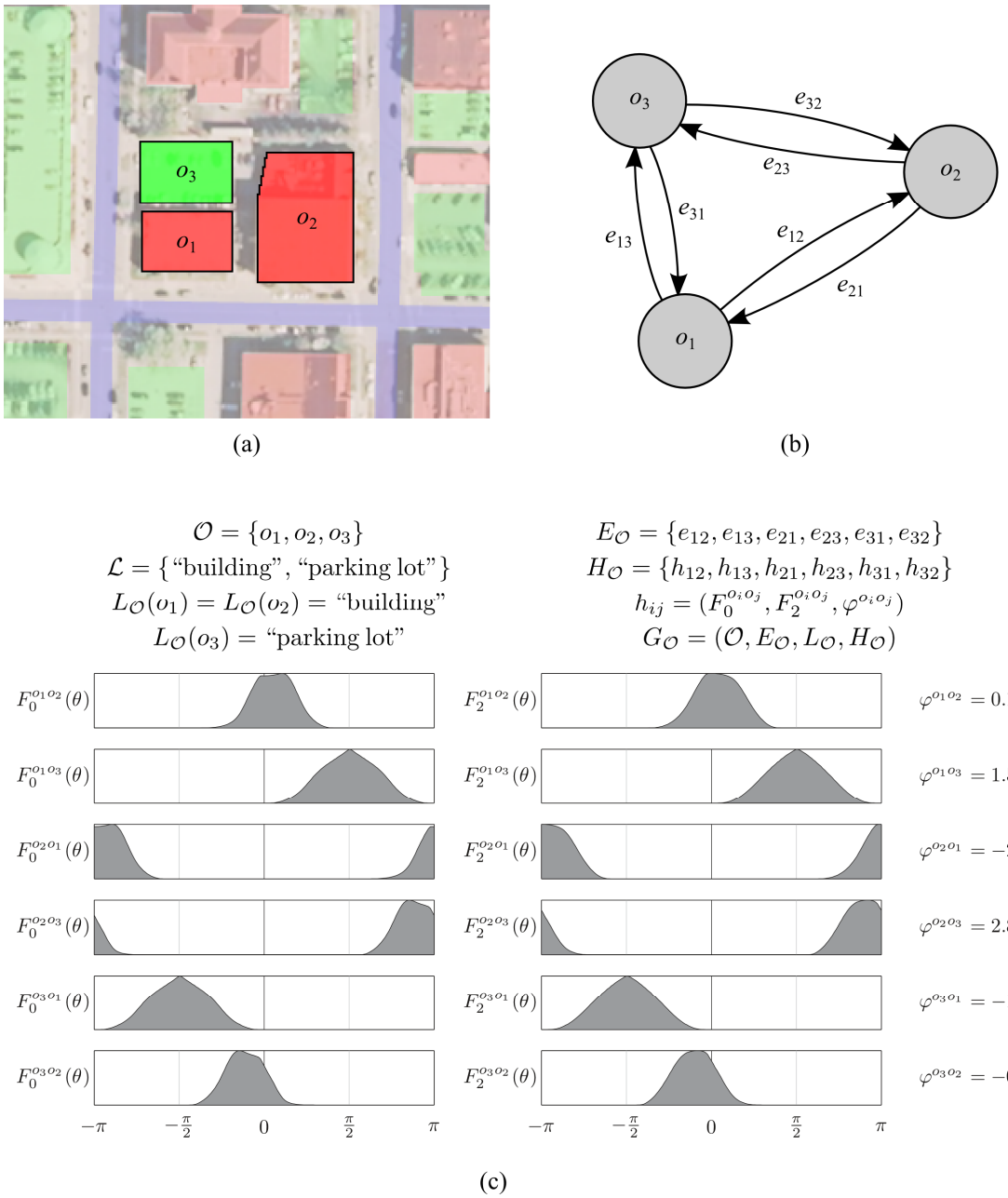


Figure 3.2 ARG representation of an object set. (a) An example of an object set $\mathcal{O} = \{o_1, o_2, o_3\}$ (b) The ARG representation of the object set. (c) Formal definition of $G_{\mathcal{O}}$ and its attributes.

Obviously since each edge represents a spatial relationship, the order of the arguments is important. “A is in direction θ of B” is not the same as “B is in direction θ of A.” The two statements contain largely the same information, however, and they are

related by the semantic inverse property of the HoF [Matsakis, et al., 2004], which states that

$$F_r^{BA}(\theta) = F_r^{AB}(\theta + \pi). \tag{3.1}$$

Since F_r^{AB} is a periodic function, this is simply a circular shifting of the histogram bins, in which no information is lost. A complete ARG for a set of n objects will have n vertices and $n \times (n-1)$ edges, with a unique edge defined between each ordered pair of vertices. We can reduce the storage requirement of the ARG representation by a factor of two if we only calculate edges (o_i, o_j) in which $i < j$, and use the semantic inverse property for all other pairs.

3.3 Comparing F-Histograms

Toward the goal of developing a similarity measure between two scenes, we begin by comparing a single pair of F-histograms. If two pairs of objects have a similar spatial configuration, then they should have similar F-histograms. Matsakis et al. [Matsakis, et al., 2004] investigated several similarity measures for F-histograms:

$$\mu_T(f_1, f_2) = \frac{\sum_\theta \min\big(f_1(\theta), f_2(\theta)\big)}{\sum_\theta \max\big(f_1(\theta), f_2(\theta)\big)}, \tag{3.2}$$

$$\mu_P(f_1, f_2) = 1 - \frac{\sum_\theta |f_1(\theta) - f_2(\theta)|}{\sum_\theta |f_1(\theta) + f_2(\theta)|}, \tag{3.3}$$

$$\mu_C(f_1, f_2) = \frac{\sum_\theta f_1(\theta) f_2(\theta)}{\sqrt{\sum_\theta f_1^2(\theta)}\sqrt{\sum_\theta f_2^2(\theta)}}. \tag{3.4}$$

Here, μ_T is a Tversky index, μ_P is a Pappis' measure, and μ_C is the normalized cross-correlation between two F-histograms, f_1 and f_2. Also, θ is a member of the finite set of angles for which the F-histograms are computed. These measures all satisfy the following properties.

$$0 \leq \mu(f_1, f_2) \leq 1 \tag{3.5}$$

$$f_1 = f_2 \Rightarrow \mu(f_1, f_2) = 1 \tag{3.6}$$

$$\mu(f_1, f_2) = \mu(f_2, f_1) \tag{3.7}$$

$$\forall \lambda \in \mathbb{R}_+^*, \mu(\lambda f_1, \lambda f_2) = \mu(f_1, f_2) \tag{3.8}$$

In addition, μ_C also satisfies

$$\forall \lambda_1 \in \mathbb{R}_+^*, \forall \lambda_2 \in \mathbb{R}_+^*, \ \mu(\lambda_1 f_1, \lambda_2 f_2) = \mu(f_1, f_2), \tag{3.9}$$

which states that the normalized cross-correlation is invariant to the relative scales of the two histograms. This is important for the task of matching a sketch to a satellite image, because the scales at which each are represented may differ by several orders of magnitude. For this reason we use μ_C in the remainder of this work.

Although the overall scaling of the y-axis values of the F-histograms do not impact the similarity measure, the x-axis values play a large role. The F-histograms of each object set are all computed with respect to a common reference angle. If two pairs of objects are defined with the same reference angle, then their F-histogram relationships can be compared directly. If, however, they are defined with different reference angles (*e.g.* by rotating one of the pairs) then one F-histogram must be shifted to match the other. We call upon the basic properties of the histograms of forces [Matsakis &

Wendling, 1999], [Matsakis, et al., 2004], which state that if a pair of objects (A, B) is rotated counter-clockwise by an angle φ, its F-histogram becomes

$$F_r^{rot(A,B)}(\theta) = F_r^{AB}(\theta - \varphi). \tag{3.10}$$

This is simply a circular shifting of the histogram bins, which allows us to compare spatial relationships defined with any orientation. Given two pairs of objects, (A, B) and (A', B') defined with reference angles ϕ and ϕ' respectively, we can compare their relative spatial relationships with the general equation

$$\begin{aligned} \mu_{Pair}(A, B, \phi, A', B', \phi') &= \beta\mu_{C0} + (1-\beta)\mu_{C2}, \text{ where} \\ \mu_{C0} &= \mu_C\left(F_0^{AB}(\theta - \phi), F_0^{A'B'}(\theta - \phi')\right), \\ \mu_{C2} &= \mu_C\left(F_2^{AB}(\theta - \phi), F_2^{A'B'}(\theta - \phi')\right). \end{aligned} \tag{3.11}$$

Here β is a weighting factor between the histograms of constant and gravitational forces, which is typically set at 0.5 to give equal weight to both F-histograms.

3.4 Comparing Object Sets

Suppose that we are given a sketch of objects $S = (o_1, o_2, \dots, o_n)$ and a reference set $\mathcal{R} = \{x_1, x_2, \dots, x_m\}$ in which $m \gg n$. Our goal is to pick a subset of objects from $\mathcal{R}$ which could match the objects in S via the injective function $\Gamma: S \to \mathcal{R}$. This can be represented as the candidate set $\Gamma = \left(x_{(1)}, x_{(2)}, \dots, x_{(n)}\right)$, $x_{(i)} \in \mathcal{R}$ such that $\Gamma(o_i) = x_{(i)}$. Notice that we avoid the general correspondence problem and assume that the order of objects in Γ is the same as in S. The task of finding Γ is the subject of our evolutionary algorithm and will be discussed further in Section 3.6. We can compare S and Γ by measuring the average similarity of the spatial relationships between each pair of objects.

If we can guarantee that both S and Γ are defined with the same orientation, then the similarity of the two object sets can be computed as

$$\Psi_1(S,\Gamma) = \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mu_{Pair}\left(o_i, o_j, 0, x_{(i)}, x_{(j)}, 0\right). \tag{3.12}$$

Here both reference angles are defined as 0, implying that no shifting of the histograms is necessary. The complexity of this computation is $O(n^2\omega)$, where n is the number of objects in each set and ω is the number of angles computed for each F-histogram.

The above expression is valid if both object sets share the same orientation, however this is usually not the case. Maps are not always drawn with the same orientation as the ground truth, often out of convenience. Take, for example, the streets of Manhattan, which are commonly drawn on maps as perfectly horizontal and vertical lines, yet a satellite image of the city shows that the island is not actually aligned in one of the cardinal directions. In order to compensate for changes in orientation between the sketch and the reference database, we rotate all of the F-histograms from the sketch by an angle $\varphi^\star$ which would give the best overall alignment with the F-histograms from the candidate set. Although the rotation could be applied to either the sketch or the candidate set, we choose to rotate the sketch to mimic how one orients a map.

The angular difference between two pairs of objects (A, B) and (A', B') is defined as the difference between their main directions $\varphi^{AB} - \varphi^{A'B'}$. By considering all of the angular differences between each unique pair of objects in S and Γ, we create a list

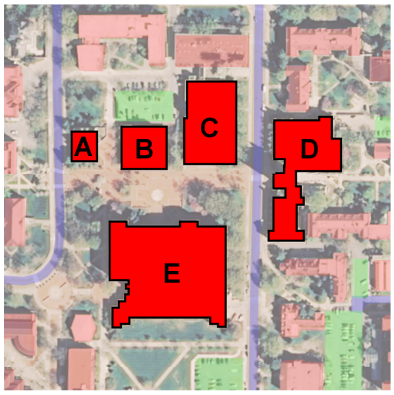
$$D = \{d_{11}, d_{12}, \ldots, d_{ij}, \ldots, d_{(n-1)n}\}, \qquad d_{ij} = \varphi^{o_i o_j} - \varphi^{x_{(i)} x_{(j)}}, \tag{3.13}$$

which represents the total mismatch between the orientations of S and Γ. For example, if D contains only 0 values, then S and Γ have the same orientation. The values of D are shifted into the range $[0, 2\pi)$ and used to determine the optimal rotation angle $\varphi^\star$ that will be applied to S. The mean and median values of D are both reasonable choices for $\varphi^\star$, with the median providing greater stability overall [Buck, et al., 2011]. Because the angles are defined on a periodic domain, it may not be possible to define a 2π range which can serve as a linear mapping to compute the median. Therefore, we pick the optimal rotation angle as the angle in D which minimizes the angular distance to all other angles in D using the following expression from [Fisher, 1993].

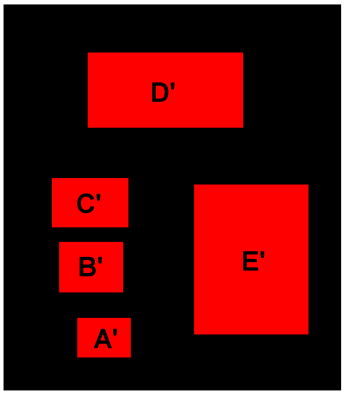
$$\varphi^\star = \underset{d_{uv} \in D}{\arg\min}[q(d_{uv})], \qquad q(d_{uv}) = \pi - \sum_{i=1}^{n} \sum_{j=i+1}^{n} \left| \pi - \left| d_{ij} - d_{uv} \right| \right| \tag{3.14}$$

Here, q is a temporary list of the total angular distances evaluated for each angle in D. An example of this process is given in Figure 3.3. Having found $\varphi^\star$, we rotate all of the F-histograms from S by a uniform angle to obtain an orientation-independent similarity measure,

$$\Psi_2(S, \Gamma) = \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mu_{Pair}\left(o_i, o_j, \varphi^\star, x_{(i)}, x_{(j)}, 0\right). \tag{3.15}$$



(a)



(b)

Object Pair		Main Direction (Ground Truth)	Main Direction (Sketch)	Angular Difference	Angular Distance
u	v	φ^{uv}	$\varphi^{u'v'}$	d_{uv}	$q(d_{uv})$
A	B	3.12	4.87	4.53	-25.39
A	C	3.30	4.81	4.78	-27.32
A	D	3.04	4.49	4.83	-27.49
A	E	2.12	3.58	4.83	-27.49
B	C	3.42	4.72	4.98	-26.73
B	D	3.02	4.36	4.94	-27.03
B	E	1.74	3.18	4.84	-27.48
C	D	2.77	4.24	4.81	-27.46
C	E	1.32	2.84	4.76	-27.21
D	E	0.62	1.99	4.91	-27.21
				$\min(q) =$	-27.49
				$\varphi^\star = 4.83 \approx$	-83°

(c)

Figure 3.3 Calculation of the optimal rotation angle, $\varphi^\star$. All angles are given in radians measured counterclockwise from the x-axis. The ground truth object set in (a) is approximated by the simplified sketch in (b), which has been rotated counterclockwise about one quarter-turn. The object correspondences between the sketch and ground truth are given, and the main direction between each unique object pair is given in (c). The first two columns of (c) list the individual object pairs, and the main directions calculated for the ground truth and sketch are given in the third and fourth columns respectively. The list of angular differences D is listed in the fifth column, which is used as the input to equation (3.14) for computing the list of angular distances, q. The angle which minimizes the angular distance to all other angles in D is chosen as the optimal rotation angle, $\varphi^\star$. Here, $\varphi^\star$ is chosen as an 83° clockwise rotation of the sketch.

3.5 Elastic Angles

As an alternative to rotating all of the F-histograms from one set by the global best rotation angle, we can exercise a little more control over the similarity measure by rotating each F-histogram individually. This gives each pair of histograms a tolerance to small directional differences. The orientation of the scene as a whole is still important, so we begin by calculating the angular difference list D in the same way as before. This gives the best rotation angle for the whole scene, $\varphi^{\star}$. Rather than rotating all histograms of the sketch by this angle, we create normalized F-histograms by rotating each histogram of both the sketch and candidate set clockwise by its main direction so that all F-histograms are centered at $\theta = 0$. Comparing normalized F-histograms removes all orientation biases, leaving only the shapes and sizes as distinguishing characteristics. To compensate for the loss of directional information, we apply a weighting factor to each histogram, defined by a fuzzy membership function $\mu_{Trap}(\theta)$ shown in Figure 3.4. The angular difference between the original histograms is used as the input to this weighting function, allowing only histograms that shared similar orientations to be considered with full weight and all others to have less weight.

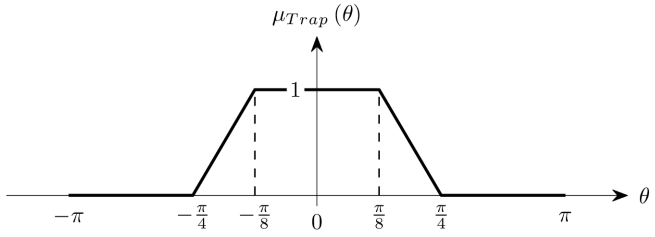


Figure 3.4 The trapezoidal weighting function used for elastic angles.

The overall similarity measure is defined as

$$\Psi_3(S,\Gamma) = \frac{2}{n(n-1)}\sum_{i=1}^{n}\sum_{j=i+1}^{n}\mu_{Trap}\left(\varphi^{\star} - d_{ij}\right) \times \mu_{Elastic}\text{, where}$$
$$\mu_{Elastic} = \mu_{Pair}\left(o_i, o_j, \varphi^{o_i o_j}, x_{(i)}, x_{(j)}, \varphi^{x_{(i)}x_{(j)}}\right) \tag{3.16}$$

The elastic angle method allows for small imperfections between two object sets. F-histograms which would not otherwise be perfectly aligned are normalized and considered with full weight. This tends to result in higher similarity values overall [Buck, et al., 2011], but allows for the small discrepancies between object sets that tend to arise when working with real data. Figure 3.5 shows an example which highlights the differences between the elastic and non-elastic methods for evaluating object set similarity.

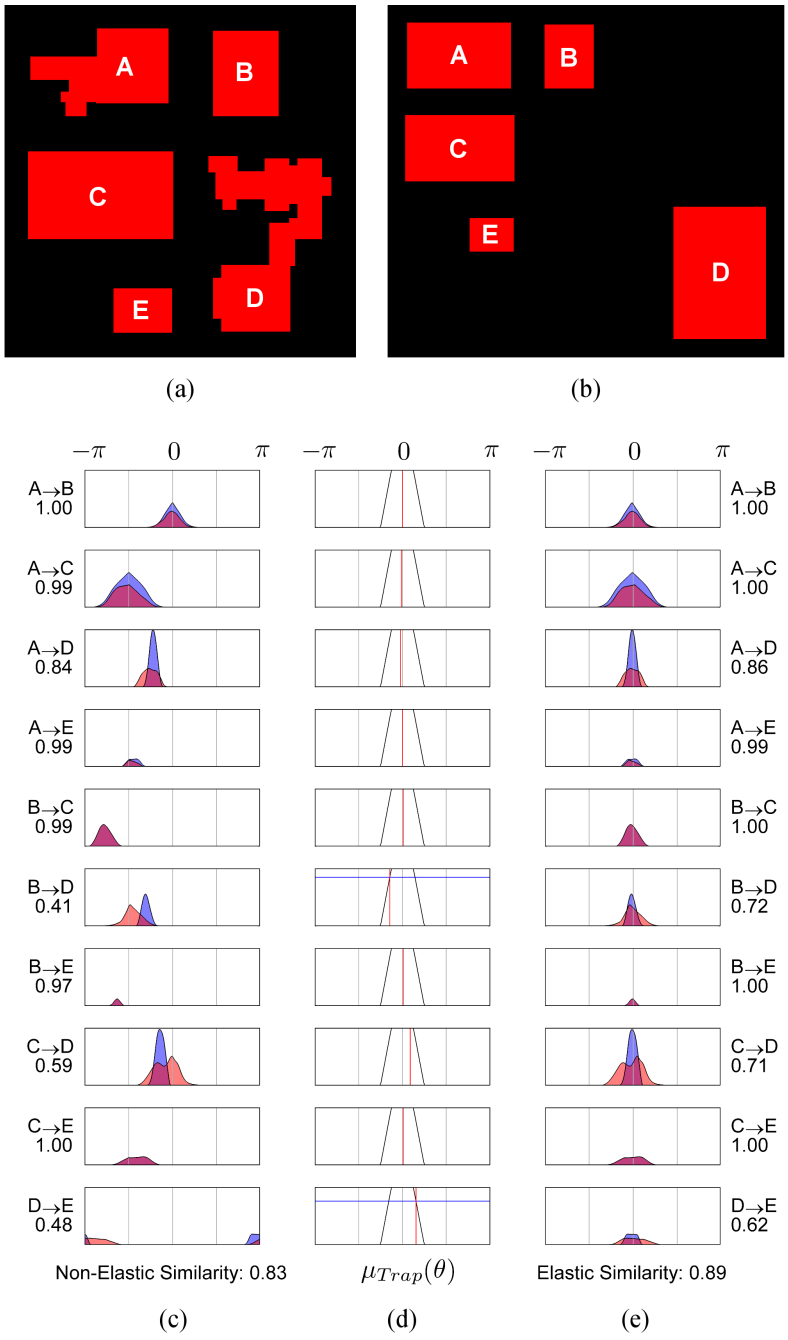
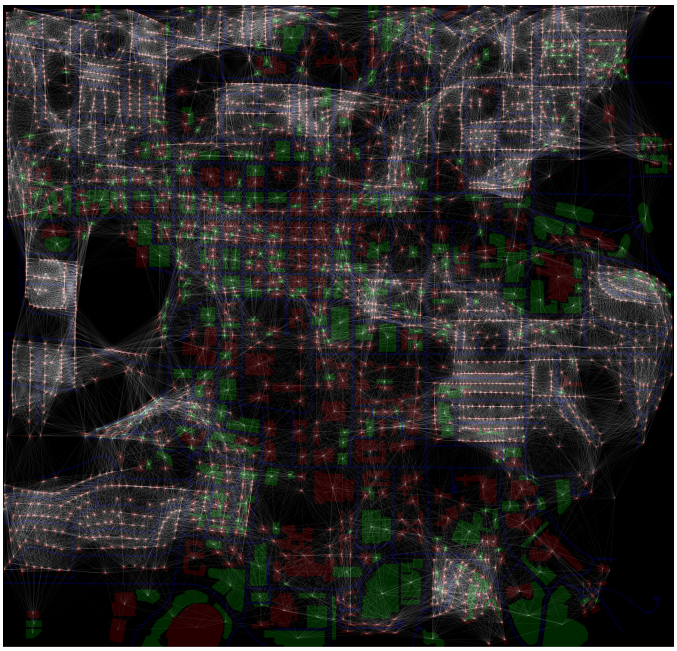


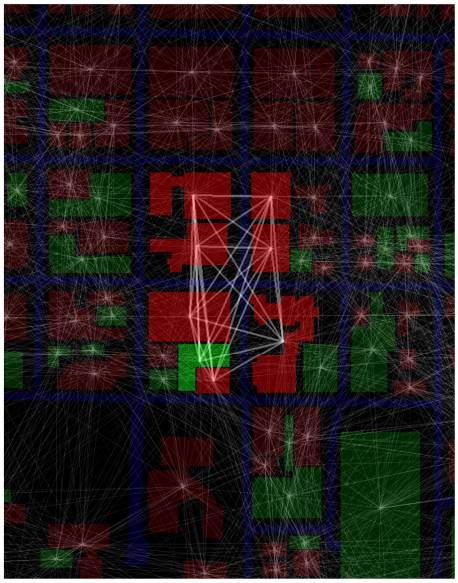
Figure 3.5 Comparison of elastic and non-elastic methods. (a) Ground truth image. (b) The sketch is a simplification of (a) with object *D* significantly misplaced. (c) shows the computation of the non-elastic similarity, where the numbers to the left of each histogram represent the individual cross-correlation values. For clarity, only the histogram of constant forces is shown, although both the constant and gravitational F-histograms are used in computing the final similarity. (d) and (e) show the computation of the elastic fitness. (d) is the weighting function $\mu_{Trap}(\theta)$ and (e) shows the normalized histograms, where the numbers to the right of each histogram represent the weighted cross-correlation values.

3.6 Overview of the Evolutionary Algorithm

The similarity measures defined above offer a way to compare two object sets based on their spatial relationships. This allows us to develop an algorithm to search a segmented satellite image for a group of objects which most closely matches an input sketch. Given a sketch $S = (o_1, o_2, \ldots, o_n)$, and a reference set $\mathcal{R} = \{x_1, x_2, \ldots, x_m\}$ in which $m \gg n$, we begin by constructing the attributed relational graphs $G_{\mathcal{R}}$ and G_S. For the sketch S, G_S is completely defined with a vertex for each object and the full set of $n \times (n-1)$ edges. When constructing $G_{\mathcal{R}}$, we create a vertex for each object, but only define some of the spatial relationships as edges. Typically, $\mathcal{R}$ contains many objects spread over a large area. Since the sketch represents only a small spatial region, we restrict the set of outgoing spatial relationships for each object in $\mathcal{R}$ to its K nearest neighbors. This prunes the search space considerably, eliminating edges between objects which are not nearby. In [Bloch, et al., 2006], several techniques for measuring the degree to which an object is between two others were investigated. Object pairs which are too far apart, or have too many objects between themselves can also be excluded from the edge list. These parameters must be chosen carefully ahead of time to ensure that the subset of $\mathcal{R}$ which we would like to see matched with S remains fully connected, so that there is an edge relationship between each pair of objects. Requiring S and the corresponding subset of $\mathcal{R}$ to be complete graphs allows us to treat this as a subgraph isomorphism problem rather than a subgraph homomorphism problem. Our experiments use a reference set of 2814 objects with a maximum of 50 neighbor connectivity, shown in Figure 3.6, which highlights $G_{\mathcal{R}}$ and the subgraph which matches G_S, the ARG representation of the sketch given in Figure 3.1.



(a)



(b)

Figure 3.6 ARG representation of the search space. (a) $G_{\mathcal{R}}$, the ARG representation of the reference set used in our experiments containing 2814 objects. The graph is superimposed over the reference set with the edges displayed in white. Buildings are shown in red and parking lots are shown in green. (b) G_S, the ARG representation of the sketch from Figure 3.1, highlighted as a subgraph of $G_{\mathcal{R}}$.

Due to the potentially enormous search space, we have chosen to use an evolutionary algorithm to match the sketch to the reference database, as it is visually intuitive and can be scaled to many different problem sizes. An individual solution, or chromosome, is a function $\Gamma: S \to \mathcal{R}$, represented as $\Gamma = \left(x_{(1)}, x_{(2)}, \dots, x_{(n)}\right),\ x_{(i)} \in \mathcal{R}$ such that $\Gamma(o_i) = x_{(i)}$. This vector represents the objects of the reference set which could potentially correspond to the sketch. We must take care when constructing Γ to ensure that no objects are duplicated and that they are all fully connected to one another. The labels of the sketch objects must also match the labels of the corresponding chromosome objects such that $L_S(o_i) = L_{\mathcal{R}}\left(x_{(i)}\right)$ for $1 \leq i \leq n$.

The search algorithm (Figure 3.7) begins by generating a population of η random individuals $P^{(0)} = \left(\Gamma_1, \Gamma_2, \dots, \Gamma_\eta\right)$. Each random individual solution is chosen by first finding the most unique label of the sketch and picking a random object from the reference set with the same label. This object becomes the seed of the chromosome, and the remaining objects are chosen randomly from the nearest neighbors of this seed such that the labels match the objects of the sketch. Closer neighbors have a higher likelihood of being chosen in order to keep the individual chromosome spatially compact. If none of the seed's nearest neighbors can satisfy the label requirements of the sketch, a new seed is chosen in a different location.

Procedure: Evolutionary Spatial Matching Algorithm
Input: $\mathcal{R}$ and S
Constants: η, τ, ρ
Initialize: Set $t = 0$ and create initial population of individuals: $P^{(0)} = (\Gamma_1, \Gamma_2, \dots, \Gamma_\eta)$
While stopping criteria is not met
$P^{(t+1)} = P^{(t)}$
$t = t + 1$
If t is a multiple of τ
Replace the lowest scoring fraction ρ of $P^{(t)}$ with new random individuals
Else
For each individual $\Gamma_P \in P^{(t)}$
Generate list of children through mutation: $\mathcal{C} = \text{mutate}(\Gamma_P)$
Select most fit child: $\Gamma_C = \arg\max_{\Gamma \in \mathcal{C}} \psi(\Gamma)$
If $\psi(\Gamma_C) > \psi(\Gamma_P)$
Replace Γ_P with Γ_C
End If
End For
End If
End While
Output: Top scoring individuals in $P^{(t)}$

Figure 3.7 Outline of the evolutionary spatial matching algorithm.

After the initial population has been created, we calculate the fitness of each individual as $\psi(\Gamma) = \Psi(\mathcal{S}, \Gamma)$ (equation (3.16)) where $\Gamma \in P^{(0)}$. During each generational cycle of the algorithm, we perform a local improvement in the form of a mutation operator on each individual chromosome in the population. The mutation operators described below each take a single parent solution Γ_P as input and return a list of possible child solutions, $\mathcal{C} = \{\Gamma_1, \Gamma_2, \dots \Gamma_z\}$. We pick the child with the highest fitness $\Gamma_C = \arg\max_{\Gamma \in \mathcal{C}} \psi(\Gamma)$ and compare against the parent, Γ_P. Whichever has the higher fitness survives to the next generation, ensuring that we always have the best solution in the local search space. Since the individuals are less likely to improve after a certain number of generations, and to increase the diversity of the search, we replace the lower scoring fraction ρ of the population with new random individuals every τ generations.

This allows us to continue searching new areas of the search space, while preserving the best solutions found thus far. The search process continues until some stopping criteria is met, usually a fitness threshold or a maximum number of generations.

3.7 Mutation Operators

Mutation operators traditionally play the role of maintaining genetic diversity within a population and are often paired with crossover operators to create each succeeding generation. However, crossover operators do not fit well with our representation scheme, since we require each individual mapping to be a fully connected subgraph. Combining objects from two separate solutions would likely result in a spatially disjoint child, which should be avoided. The mutation operators must then play the role of improving a single solution using a local search strategy. In this way, we can consider our algorithm to be a type of memetic algorithm, using an evolutionary global framework with a separate local improvement operator for each individual. In the following sections, four mutation operators for improving the spatial configuration of an individual solution to match a target sketch are introduced and an example of their application is provided.

3.7.1 Single-Object Replacement

The single-object replacement (SOR) mutation is based on the work presented in [Buck, et al., 2010] and [Buck, et al., 2011]. In this strategy, a single object from the parent is replaced by one of its nearest neighbors. Given a parent mapping function

$\Gamma_P = \left(x_{(1)}, x_{(2)}, \ldots, x_{(n)}\right)$, we cycle through each object $x_{(i)} \in \Gamma_P$ and replace it with one of its nearest neighbors. Previous versions of this algorithm randomly picked only a single object for replacement; however we found that by testing all parent objects for replacement we could improve the matching rate with only a small amount of additional overhead. This strategy of choosing all possible initializations carries over into the set reconstruction mutation methods as well. Let $\mathcal{X}$ be the set of nearest neighbors for $x_{(i)}$, such that any object $x^\star \in \mathcal{X}$ could replace $x_{(i)}$ in Γ_P and still maintain full connectivity. For each neighbor object $x^\star$, we build a mapping function Γ' which is identical to Γ_P except that $x_{(i)}$ has been replaced by $x^\star$. The function with the highest fitness is added to the list of children, $\mathcal{C}$.

Initial experiments with the SOR mutation operator [Buck, et al., 2010], [Buck, et al., 2011] revealed that the algorithm often has difficulty finding the ideal match in a region. Because each chromosome is an ordered set, an individual solution can contain all of the objects of the ideal match, but not in the right order. Successive mutations on these solutions will often stall out as objects are swapped with neighboring objects to allow for different orderings of the chromosome. We therefore consider multiple different permutations of the parent solution before applying the SOR mutation, which produces a larger set of children, but decreases the amount of stalling performed by individuals. Clearly, evaluating all possible permutations would result in a large computational overhead, so we typically use only a small number of randomly chosen permutations. This offers a balance between performing an exhaustive search for each mutation and maintaining a degree of randomness which helps prevent premature

convergence. The complete SOR mutation method is outlined in Figure 3.8. Given that the complexity of each fitness evaluation is $O(n^2\omega)$, the SOR mutation has a complexity of $O(pn^3\omega K)$, where p is the number of permutations considered, n is the number of objects in the sketch set, ω is the number of angles in each F-histogram, and K is the maximum number of nearest neighbor connections used in the reference set ARG.

Procedure: Single-Object Replacement Mutation
Input: Sketch: $S = (o_1, o_2, \dots, o_n)$
Reference set: $\mathcal{R} = \{x_1, x_2, \dots, x_m\}$
Parent: $\Gamma_P = \left(x_{(1)}, x_{(2)}, \dots, x_{(n)}\right)$
Constant: p
Initialize: $\mathcal{C} = \emptyset$
Add Γ_P to list of permutations, $\mathcal{P}$
Add p random permutations of Γ_P to $\mathcal{P}$
For Each $\Gamma \in \mathcal{P}$
  For $i = 1$ **to** n
    $\Gamma' = \Gamma$
    Get the set of nearest neighbors $\mathcal{X} \subseteq \mathcal{R}$ of the object $x_{(i)}$
    $\psi_{best} = 0$
    For Each $x^\star \in \mathcal{X}$
      Replace a single object: $\Gamma'(i) = x^\star$
      Evaluate the fitness: $\psi(\Gamma') = \Psi(S, \Gamma')$
      If $\psi(\Gamma') > \psi_{best}$ **Then**
        $x_{best} = x^\star$
      End If
    End For
    Replace with best object: $\Gamma'(i) = x_{best}$
    Add to list of children: $\mathcal{C} = \mathcal{C} \cup \Gamma'$
  End For
End For
Output: $\mathcal{C}$

Figure 3.8 Outline of the single-object replacement mutation algorithm.

3.7.2 One-Seed Set Reconstruction

The set reconstruction methods are based on the idea that the best possible solution can be reconstructed from a small starting seed of just one or two objects. Given

sketch set $S = (o_1, o_2, \dots, o_n)$, we define a partial sketch $S' = \left(o'_{(1)}, o'_{(2)}, \dots, o'_{(t)}\right) \subset S$ which only contains some of the original objects. Likewise, we define a partial mapping function $\Gamma' : S' \to \mathcal{R}$ with partial fitness $\psi(\Gamma') = \Psi(S', \Gamma')$, which only considers the specified subset of sketch objects. The idea behind the one-seed mutation method is to start with a single object $S' = \left(o'_{(1)}\right) \subset S$ and add objects one at a time until we use all of the objects in the sketch. The overall outline of the one-seed set reconstruction method is shown in Figure 3.9. Given a parent mapping function $\Gamma_P = \left(x_{(1)}, x_{(2)}, \dots, x_{(n)}\right)$, we cycle through each object $x_{(i)} \in \Gamma_P$ and use it as the seed object for a partial sketch. We then consider all possible mappings of the seed object onto an object from the parent, and create a partial solution Γ' for each one. For each partial solution, we randomly pick an unassigned sketch object $o_u \in S - o'_{(1)}$ and find the set of nearest neighbors $\mathcal{X} \subseteq \mathcal{R}$ to which o_u could be assigned while maintaining full connectivity. As with the SOR mutation, we create a set of temporary partial mapping functions, each with o_u assigned to a different neighbor object $x^\star \in \mathcal{X}$. The neighbor that produces the greatest partial fitness is added to the partial sketch S'. We continue to map the unassigned sketch objects of S to the best neighbor objects in this greedy manner until $S' = S$. Once we have a complete mapping function, we add it to the list of children, $\mathcal{C}$.

Procedure: One-Seed Set Reconstruction Mutation
Input: Sketch: $S = (o_1, o_2, \dots, o_n)$
Reference set: $\mathcal{R} = \{x_1, x_2, \dots, x_m\}$
Parent: $\Gamma_P = \left(x_{(1)}, x_{(2)}, \dots, x_{(n)}\right)$ where each $x_{(i)} \in \mathcal{R}$
Initialize: $\mathcal{C} = \emptyset$
Initialize list of index locations: $I = \{1, 2, \dots, n\}$
For Each $(i, j) \in I \times I$ such that $L_S(o_i) = L_{\mathcal{R}}\left(x_{(j)}\right)$
Clear Γ'
Create the partial ordered sketch: $S' = (o_i)$
Update remaining index locations: $I' = I - i$
Define $\Gamma'(o_i) = x_{(j)}$
Get the set of nearest neighbors $\mathcal{X} \subseteq \mathcal{R}$ of the object $x_{(j)}$
While $|I'| > 0$
Pick an index $k \in I'$ randomly
Add o_k to the end of the partially ordered sketch: $S' = (\dots, o_k)$
$\psi_{best} = 0$
For Each $x^\star \in \mathcal{X}$
Define $\Gamma'(o_k) = x^\star$
Evaluate the partial fitness: $\psi(\Gamma') = \Psi(S', \Gamma')$
If $\psi(\Gamma') > \psi_{best}$ **Then**
$x_{best} = x^\star$
End If
End For
Define $\Gamma'(o_k) = x_{best}$
Remove this index location: $I' = I' - k$
Update $\mathcal{X}$ as the nearest neighbors of the image of Γ'
End While
Add Γ' to list of children: $\mathcal{C} = \mathcal{C} \cup \Gamma'$
End For
Output: $\mathcal{C}$

Figure 3.9 Outline of the one-seed set reconstruction mutation algorithm.

The one-seed set reconstruction mutation solves many of the problems faced by the SOR mutation operator. Individuals rarely stall over a valid match without converging to a locally optimal solution. Different orderings of buildings becomes less of an issue since the entire mapping function must be reconstructed. The one-seed mutation also tends to converge faster than the SOR mutation since more of the solution is being replaced, although this can cause individuals to become trapped in sub-optimal local solutions. The complexity of the one-seed method can be derived as $O(n^5 \omega K)$,

which is greater than the SOR method, assuming that only a few permutations are considered. Because of the exponential term on n, this method is limited to relatively small sketch sizes; our experiments use sketches of five objects. The greater complexity of the one-seed method is compensated by the faster convergence rate, which will be shown in Chapter 4.

Given that we always try to recover the best alignment between the sketch and each chromosome, each partial solution may be oriented differently from its previous version. As the set reconstruction methods add additional objects, the resulting orientation of each partial solution becomes increasingly more difficult to change. When there are only two objects, the partial solution is allowed to rotate to whichever angle best matches the corresponding objects of the sketch, essentially relying only on the shape of the F-histograms to evaluate the fitness. This means that the second object of the partial solution defines the initial orientation, and the remaining objects must conform to this orientation.

3.7.3 Two-Seed Set Reconstruction

The two-seed set reconstruction mutation method is almost identical to the one-seed method with the exception that we use two seed objects instead of just one. By using two seeds, we define an edge relationship between two objects, which determines the individual's initial orientation. This allows a single mutation to explore many different possible orientations, but incurs a significant computational overhead. For a parent mapping function $\Gamma_P = \left(x_{(1)}, x_{(2)}, \ldots, x_{(n)}\right)$, we cycle through each pair of objects

$\left(x_{(i)}, x_{(j)}\right) \in \Gamma_P \times \Gamma_P$ and use them as the seed objects for a partial sketch. We then consider all possible mappings of the seed objects onto a pair of objects from the parent, and create a partial solution, Γ' for each one. This results in a complexity of $O(n^7 \omega K)$, significantly greater than any of the other methods, but with the advantage of searching many more possible mappings. Again, the high complexity restricts this method to small sketch sizes. Unlike the one-seed method, the two-seed mutation provides the option to check individual edges for compatibility. Although we don't make use of this property in our experiments, one could conceive of a representation in which edges are compared directly. This could greatly reduce the number of possible mappings and the overall complexity of the two-seed method. The remainder of the algorithm is the same as the one-seed method and is given in Figure 3.10.

Procedure: Two-Seed Set Reconstruction Mutation
Input: Sketch: $S = (o_1, o_2, \dots, o_n)$
Reference set: $\mathcal{R} = \{x_1, x_2, \dots, x_m\}$
Parent: $\Gamma_P = \left(x_{(1)}, x_{(2)}, \dots, x_{(n)}\right)$ where each $x_{(i)} \in \mathcal{R}$
Initialize: $\mathcal{C} = \emptyset$
Initialize list of index locations: $I = \{1, 2, \dots, n\}$
For Each $(i, j) \in I \times I$ such that $i \neq j$
For Each $(k, l) \in I \times I$ such that $k \neq l$
If $L_S(o_i) \neq L_{\mathcal{R}}\left(x_{(k)}\right)$ **Or** $L_S\left(o_j\right) \neq L_{\mathcal{R}}\left(x_{(l)}\right)$ **Then**
Continue
End If
Clear Γ'
Create the partial ordered sketch: $S' = \left(o_i, o_j\right)$
Update remaining index locations: $I' = I - \{i, j\}$
Define $\Gamma'(o_i) = x_{(k)}$ and $\Gamma'\left(o_j\right) = x_{(l)}$
Get the set of nearest neighbors $\mathcal{X} \subseteq \mathcal{R}$ of the image of Γ'
While $|I'| > 0$
Pick an index $m \in I'$ randomly
Add o_m to the end of the partially ordered sketch: $S' = (\dots, o_m)$
$\psi_{best} = 0$
For Each $x^\star \in \mathcal{X}$
Define $\Gamma'(o_m) = x^\star$
Evaluate the partial fitness: $\psi\ (\Gamma') = \Psi(S', \Gamma')$
If $\psi(\Gamma') > \psi_{best}$ **Then**
$x_{best} = x^\star$
End If
End For
Define $\Gamma'(o_m) = x_{best}$
Remove this index location: $I' = I' - m$
Update $\mathcal{X}$ as the nearest neighbors of the image of Γ'
End While
Add Γ' to list of children: $\mathcal{C} = \mathcal{C} \cup \Gamma'$
End For
End For
Output: $\mathcal{C}$

Figure 3.10 Outline of the two-seed set reconstruction mutation algorithm.

3.7.4 VF2 Subgraph Isomorphism

Since the sketch and reference database are both stored as attributed relational graphs, it makes sense to use existing graph matching techniques to solve the subgraph isomorphism problem. The entire evolutionary algorithm could in fact be replaced by a

general algorithm for locating subgraph isomorphisms; however the feasibility of this approach decreases as the size of the search space grows very large. Instead, we use a graph matching algorithm as a local improvement operator in the form of a mutation. This allows the size of the graph-based search to remain bounded while the evolutionary algorithm handles the overall global search.

The VF2 algorithm developed by Cordella et al. [Cordella, et al., 2004] is well suited for use as a local search mutation operator. It can handle large graphs and evaluates isomorphisms based on node and edge compatibility. The input to the algorithm is a pair of graphs, G_{NN} and G_S. Given a parent mapping function $\Gamma_P = \left(x_{(1)}, x_{(2)}, \ldots, x_{(n)}\right)$, G_{NN} is a subgraph of $G_{\mathcal{R}}$ containing the objects in Γ_P and the N nearest objects to the objects in Γ_P. G_S is the ARG of the sketch being matched. Additionally, we must define node and edge compatibility functions. The VF2 algorithm works by performing a tree search in which all possible mappings from G_{NN} onto G_S are constructed one node at a time. The addition of new nodes to the mapping function is governed by a set of feasibility rules, which require that new nodes and the edges they induce are compatible with each other. If a node or an induced edge is incompatible, that search path is discarded and a new path is considered.

Fundamentally, the VF2 graph matching algorithm is similar to the set reconstruction methods described above. A small partial mapping is grown one object at a time until a complete matching is defined. In the set reconstruction methods, the object which produces the best partial mapping fitness is added to the mapping function, whereas in the VF2 algorithm, any object which satisfies the compatibility criteria is considered for inclusion. For the VF2-based mutation, any object which produces a

partial mapping function with a partial fitness greater than a given threshold satisfies the compatibility criteria. It should be noted that computation of the partial fitness is dependent on the entire partial sketch, not just the new object and the induced edges. This is because the addition of a new object can change the overall orientation of the sketch, which can have a significant impact on the sketch fitness.

The VF2 algorithm is guaranteed to find all compatible subgraphs, provided that the compatibility of each new node is dependent only on itself and the edges it induces. When this is the case, there are $n!$ ways to build a matching subgraph of n nodes. To prevent redundancy, the graph nodes are given arbitrary index values and only nodes with index values greater than any of the nodes already present in the partial map are considered. This ensures that each possible subgraph can be reached, but only in one order. In the case of sketch matching, the compatibility of new nodes is dependent on the existing partial map, which means that the order in which objects are added can influence the partial fitness values at each stage. If the addition of an object lowers the partial fitness value below the given threshold, the search path will be discarded, even if the addition of another object would raise the partial fitness above the threshold. The VF2 algorithm was not designed to handle such cases, so the compatibility functions must be changed to accept a copy of the entire partial solution in addition to the two nodes being compared. We can minimize the risk of missing a compatible subgraph by randomizing the index order of the nearest neighbor graph and repeating the search with multiple fitness thresholds. If a compatible subgraph is not found using a high initial threshold, the search is repeated with a new random index order and a lower fitness threshold. Eventually the threshold will be low enough that at least one compatible subgraph is

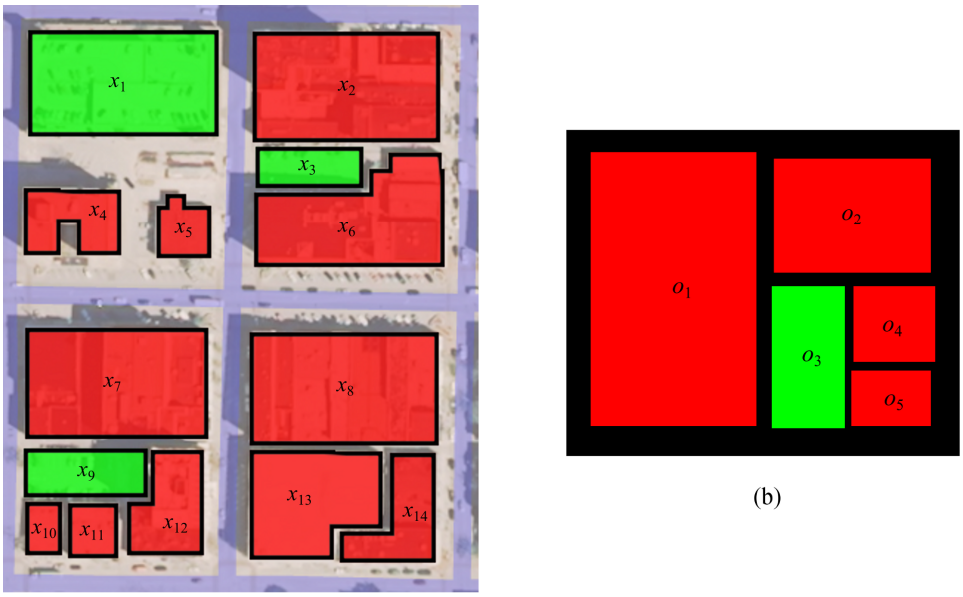
found. The results of the graph matching algorithm are returned as child solutions. The entire VF2 subgraph isomorphism mutation method is given in Figure 3.11.

Procedure: VF2 Subgraph Isomorphism Mutation
Input: Sketch: $S = (o_1, o_2, \dots, o_n)$
Reference set: $\mathcal{R} = \{x_1, x_2, \dots, x_m\}$
Parent: $\Gamma_P = \left(x_{(1)}, x_{(2)}, \dots, x_{(n)}\right)$ where each $x_{(i)} \in \mathcal{R}$
Constants: $\psi_{\min}$, δ, N
Initialize: $\mathcal{C} = \emptyset$
Construct the graph G_S from the sketch S
While $\mathcal{C} = \emptyset$
Get the set of N nearest neighbors $\mathcal{X} \subseteq \mathcal{R}$ of the parent Γ_P
Shuffle the order of $\mathcal{X}$
Construct the graph G_{NN} from $\mathcal{X}$
Run the VF2 algorithm on G_{NN} and G_S using fitness threshold $\psi_{\min}$
Add the results to $\mathcal{C}$
Let $\psi_{\min} = \psi_{\min} - \delta$
End While
Output: $\mathcal{C}$

Figure 3.11 Outline of the VF2 subgraph isomorphism mutation algorithm.

3.7.5 Mutation Example

We now present an example which demonstrates each mutation process as a chromosome converges to the ideal solution. Figure 3.12a shows an example search space containing 11 buildings shown in red, and 3 parking lots shown in green, which form the reference set. The reference set ARG is computed with an edge between every pair of objects. The sketch in Figure 3.12b is a simplified representation of the five objects in the lower-left of the reference set, rotated one quarter-turn. Our goal is to recover the mapping function $\Gamma = (x_7, x_{12}, x_9, x_{11}, x_{10})$ from a random initialization.



(a) (b)

Figure 3.12 Reference set and sketch used in the mutation example. (a) The ground truth used as the reference set. (b) The sketch to be matched.

Suppose that after initializing the algorithm, the population consists of the four individuals given in Figure 3.13.

Penultimate Population
$\Gamma_1 = (x_{12}, x_{10}, x_9, x_4, x_{11})$
$\Gamma_2 = (x_2, x_5, x_1, x_4, x_7)$
$\Gamma_3 = (x_{11}, x_6, x_3, x_7, x_8)$
$\Gamma_4 = (x_4, x_5, x_3, x_6, x_2)$

Figure 3.13 Penultimate population of the mutation example.

Notice that because the third object in the sketch is a parking lot, all of the chromosomes must also have a parking lot as the third object. A different mutation operator will be

applied to each chromosome in this example to demonstrate how each one converges to the ideal solution.

The SOR mutation is applied to $\Gamma_1 = (x_{12}, x_{10}, x_9, x_4, x_{11})$, with the main events which lead to convergence shown in Figure 3.14. First, several permutations of Γ_1 are chosen, of which $\Gamma' = (x_4, x_{12}, x_9, x_{11}, x_{10})$ is the specific permutation which could potentially match the sketch. Only a single object is incorrect, and as we cycle through each object to test for replacement, we find that replacing x_4 with x_7 produces a child chromosome with very high fitness, which is returned as the child.

SOR Mutation on Γ_1

Permute	$\Gamma' = (x_4, x_{12}, x_9, x_{11}, x_{10})$
Pick a single object	$c_i = x_4$
Get list of possible replacements	$\mathcal{X} = \{x_2, x_5, x_6, x_7, x_8, x_{13}, x_{14}\}$
Find best replacement	$x_{best} = x_7$
Replace c_i with x_{best}	$\Gamma' = (x_7, x_{12}, x_9, x_{11}, x_{10})$

Figure 3.14 Example of the SOR mutation method.

The one-seed mutation is applied to $\Gamma_2 = (x_2, x_5, x_1, x_4, x_7)$ with the main events leading to convergence shown in Figure 3.15. Each object in the sketch is evaluated as the seed object, and when o_1 is assigned to the chromosome object x_7, the remaining objects can be assigned one at a time such that the ideal solution is recovered. Similarly, the two-seed mutation is applied to $\Gamma_3 = (x_{11}, x_6, x_3, x_7, x_8)$ in Figure 3.16. Note that in this case, two sketch objects must already be assigned to the ideal reference objects in order for a complete convergence to occur in a single mutation. This occurs in this

example when o_1 is assigned to x_7 and o_4 is assigned to x_{11}, allowing the ideal solution to be formed by adding the remaining objects one at a time.

One-Seed Mutation on Γ_2	
Pick the seed object	$\Gamma'(o_1) = x_7$
Get list of neighbor objects	$\mathcal{X} = \{x_1, \dots, x_6, x_8, \dots, x_{14}\}$
Pick next sketch object randomly	$o_k = o_4$
Find best match for this object	$x_{best} = x_{11}$
Update chromosome	$\Gamma'(o_4) = x_{11}$
Pick next sketch object randomly	$o_k = o_5$
Find best match for this object	$x_{best} = x_{10}$
Update chromosome	$\Gamma'(o_5) = x_{10}$
Pick next sketch object randomly	$o_k = o_3$
Find best match for this object	$x_{best} = x_9$
Update chromosome	$\Gamma'(o_3) = x_9$
Pick next sketch object randomly	$o_k = o_2$
Find best match for this object	$x_{best} = x_{12}$
Update chromosome	$\Gamma'(o_2) = x_{12}$
Return child	$\Gamma' = (x_7, x_{12}, x_9, x_{11}, x_{10})$

Figure 3.15 Example of the one-seed set reconstruction mutation method.

Two-Seed Mutation on Γ_3	
Pick the two seed objects	$\Gamma'(o_1) = x_7$ and $\Gamma'(o_4) = x_{11}$
Get list of neighbor objects	$\mathcal{X} = \{x_1, \dots, x_6, x_8, \dots, x_{10}, x_{12}, \dots, x_{14}\}$
Pick next sketch object randomly	$o_k = o_3$
Find best match for this object	$x_{best} = x_9$
Update chromosome	$\Gamma'(o_3) = x_9$
Pick next sketch object randomly	$o_k = o_2$
Find best match for this object	$x_{best} = x_{12}$
Update chromosome	$\Gamma'(o_2) = x_{12}$
Pick next sketch object randomly	$o_k = o_5$
Find best match for this object	$x_{best} = x_{10}$
Update chromosome	$\Gamma'(o_5) = x_{10}$
Return child	$\Gamma' = (x_7, x_{12}, x_9, x_{11}, x_{10})$

Figure 3.16 Example of the two-seed set reconstruction mutation method.

The VF2 subgraph isomorphism mutation is applied to $\Gamma_4 = (x_4, x_5, x_3, x_6, x_2)$ with the main events listed in Figure 3.17. First, the graph of the sketch is built, which can be saved for use in later mutations. The neighbor objects of the parent are then shuffled and used to create the neighbor graph. These graphs are used as the input to the VF2 algorithm, which returns all subgraphs of the neighbor graph which match the sketch graph above a given threshold. Of these, the ideal solution is returned as the best child. Notice that unlike the previous methods, the parent does not need to contain any of the objects found in the child, allowing the VF2 algorithm to have the largest local search space.

VF2 Subgraph Isomorphism Mutation on Γ_4	
Build the sketch graph	$G_S = (\mathcal{O}, E_\mathcal{O}, L_\mathcal{O}, H_\mathcal{O})$
Get list of neighbor objects	$\mathcal{X} = \{x_1, x_2 \ldots, x_{14}\}$
Shuffle the order	$\mathcal{X} = \text{shuffle}(\mathcal{X})$
Build the nearest neighbor graph	$G_{NN} = (\mathcal{X}, E_\mathcal{X}, L_\mathcal{X}, H_\mathcal{X})$
Run VF2 algorithm	$\mathcal{C} = \text{match}(G_{NN}, G_S)$
Return best child	$\Gamma' = (x_7, x_{12}, x_9, x_{11}, x_{10}) \in \mathcal{C}$

Figure 3.17 Example of the VF2 subgraph isomorphism mutation method.

4 EXPERIMENTS AND RESULTS

4.1 Experiment Setup

To verify its applicability on real data, the evolutionary algorithm developed in Chapter 3 is tested with a ground truth satellite image of Columbia, MO. The image was hand segmented into a reference set $\mathcal{R}$ of 2467 buildings and 378 parking lots, shown in Figure 4.1. The reference set contains several different types of regions including some urban, suburban, and rural areas arranged in both structured road grids and less structured residential areas. The reference ARG $G_{\mathcal{R}}$ (Figure 3.6) was built by calculating the HoF relationships between each object and its 50 nearest neighbors, provided that the two objects are within 500 pixels of each other and do not contain more than five other objects in between. The latter two restrictions further reduce the overall size of the search space by removing relationships that are unlikely to match to locally confined sketches. The F-histograms were calculated using a 2 degree interval, which provides enough information to distinguish most spatial configurations. All of the calculations required to build $G_{\mathcal{R}}$ were performed *a priori*, leaving only the input sketch ARG to be computed for each search.

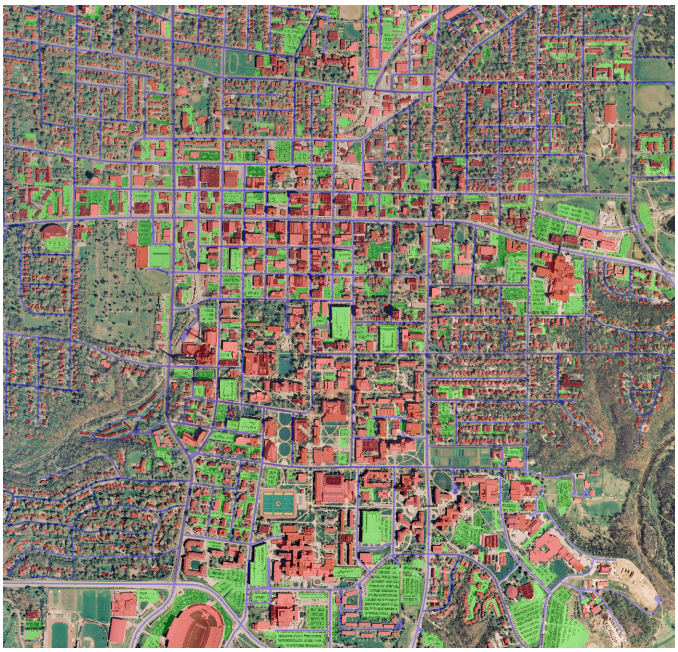
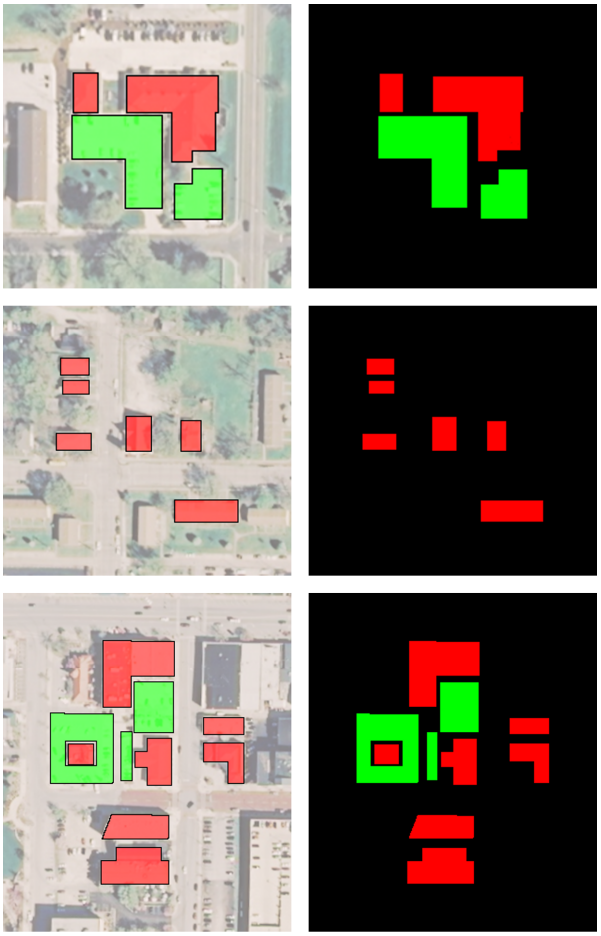


Figure 4.1 The reference set $\mathcal{R}$ used in the experiments. The set contains 2467 buildings shown in red and 347 parking lots shown in green from downtown Columbia, MO and the University of Missouri campus.

The experiments are divided into two categories, based on the type of sketch being matched. The first is a simple resubstitution search, in which the sketch is taken directly from the reference database, without modification. This serves to show that the algorithm can search the large reference set for an exact copy of the sketch. Examples of resubstitution sketches are given in Figure 4.2. The resubstitution sketches for the experiments are randomly generated and each contains a set number of objects, which can be either buildings or parking lots. They are generated by first selecting a single seed object at random from the reference set. All of the connected neighbor objects of this seed are sorted by distance and used to create the rest of the sketch. Each remaining object is chosen from the list of neighbors, with the closest object having a 50% chance of being picked, the second having a 25% chance, the third having a 12.5% chance, etc.

This ensures that the sketch is relatively compact, while still allowing for random variation. Each pair of objects in the sketch must have a relationship defined in $G_{\mathcal{R}}$ to ensure full connectivity. If a suitable set of objects cannot be found for a given seed, a new seed is randomly chosen.



(a) (b)

Figure 4.2 Examples of resubstitution sketches. (a) Original ground truth. (b) Resubstitution sketch.

The second experiment type uses simplified sketches. These are resubstitution sketches that have been simplified by reducing each object to its bounding box and

applying a random rotation to the entire sketch. When two objects' bounding boxes intersect, one object overwrites part of the other. The object with the most common label in the reference set overwrites the other, which for our experiments allows buildings to be completely surrounded by parking lots. If the two objects have the same label, the one with the greater extent (ratio of original area to bounding box area) overwrites the other. If through the simplification process an object is completely overwritten, a new resubstitution sketch is chosen to be simplified. These simplified sketches show how the algorithm can handle the imperfections and misalignments of actual hand-drafted sketches and yet provide the ability to accurately score the results. Examples of simplified sketches are given in Figure 4.3.

(a) (b)

Figure 4.3 Examples of simplified sketches. (a) Original ground truth. (b) Simplified sketch.

4.2 Comparison of the Mutation Methods

The first experiment performed compares the four different mutation operators using both resubstitution and simplified sketches. For each type of sketch, 100 random test sets were created, each containing five objects. They may contain any combination of buildings and parking lots, and are guaranteed to be a complete subgraph of $G_{\mathcal{R}}$. For each test set, the search algorithm is run 30 times using each mutation operator. This results in 3000 searches for each type of mutation. The specific algorithm parameters are

chosen to reflect the differences between each mutation operator and are summarized in Table 4.1. The SOR mutation method uses a population size of $\eta = 50$ individuals with a replacement rate of $\tau = 50$ generations and $\rho = 50\%$. Additionally, the SOR mutation method evaluates five different permutations of the parent chromosome with each mutation operation. The set reconstruction methods use the same population size as the SOR method, but with a more aggressive replacement rate of $\tau = 10$ generations and $\rho = 80\%$. These parameters were chosen after some initial experimentation, which indicated that the set reconstruction methods would often converge to locally optimal solutions much faster than the SOR method, often within only a few generations. The aggressive replacement strategy allows new solutions to compete against older stalled individuals more often, improving the overall rate of convergence. The lower replacement frequency of the SOR mutation method provides more time for individuals to perform a local search, since this method makes smaller incremental changes than the set reconstruction methods. The VF2 subgraph isomorphism mutation method uses a much smaller population size of $\eta = 10$ with a replacement rate of $\tau = 2$ generations and $\rho = 80\%$. Because the VF2 algorithm locates all matching subgraphs in a local neighborhood, there is no need to repeat the mutation multiple times in the same area. After each generation, only the top two individuals are retained. The rest of the population is replaced with new random individuals. The VF2 method uses a neighborhood size of 50 objects and an initial fitness threshold of $\psi_{\min} = 0.95$. This threshold is decremented by $\delta = 0.05$ after each local search that returns no children.

Table 4.1 Mutation Method Comparison Search Parameters

Mutation Method	Population Size (η)	Replacement Frequency (τ)	Replacement Percent (ρ)
SOR	50	50 Generations	50%
1-Seed	50	10 Generations	80%
2-Seed	50	10 Generations	80%
VF2	10	2 Generations	80%

For each search, we use the elastic angle object set comparison method with a maximum search time of 1000 generations. In an application setting, a minimum fitness threshold would be an appropriate termination criterion, which continues searching until at least one individual in the population has a fitness value greater than or equal to the threshold. However, this could terminate before the ideal match is found if the threshold is poorly chosen or if the ideal match does not have the highest fitness value. We therefore terminate the search only after at least one individual in the population is found to be identical to the correct ground truth match, or after the maximum number of generations has been reached, in order to measure the actual search time required to find the ideal match.

4.2.1 Resubstitution

The results of the mutation comparison experiment using resubstitution sketches are shown in Table 4.2, and Figure 4.4. This experiment clearly shows that the SOR mutation method simply cannot perform as well as the set reconstruction methods or the VF2 algorithm. It only finds the ideal match 51.9% of the time and takes much more time to run that any of the other methods. This is largely because of the small incremental changes that are made with each mutation. As was shown in [Buck, et al.,

2010], the SOR mutation method requires a large population size and a high generation limit. In that study, the SOR mutation method was used with population sizes ranging up to 1000 individuals and with a 10,000 generation limit. With these parameters, the SOR mutation was able to find the ideal match 94.8% of the time, however at the cost of greatly increased runtime.

The set reconstruction methods make much more drastic changes than the SOR method during each mutation step, allowing them to converge more quickly and also escape local minima with less effort. The one-seed mutation method performed well, finding the ideal match every time and requiring the fewest generations of all the methods. The two-seed mutation method found the ideal match for almost all of the tests, but took significantly longer to converge. This is to be expected due to the additional computational overhead of using a second seed. It should also be noted that for all of the methods except the SOR mutation, the average number of generations and runtime is greater than the median value, implying that a majority of the test sketches were easy to find, with only a few difficult ones.

The best method for the resubstitution experiments is the VF2 subgraph isomorphism mutation method. Like the one-seed method, the VF2 method found all of the test sets in the reference database, and with the fastest runtime of any of the mutation methods. The deterministic nature of the VF2 algorithm allows it to search a local neighborhood only one time before moving on to a new location. This proves to be a very efficient strategy, requiring fewer redundant calculations than the set reconstruction or SOR mutation methods.

Figure 4.4 graphs the results of the resubstitution experiments. In Figure 4.4 (a) and (b), the maximum and mean population fitness values are plotted for each generation, averaged over all of the tests. Figure 4.4a shows that the top fitness values increase very quickly and then level off, implying that reasonable, though perhaps not ideal, solutions can be found very quickly using any of the mutation methods. Figure 4.4b gives the average fitness value of the population, showing the drops which occur at regular intervals corresponding to the replacement rates of the algorithms. The SOR method can be seen to have the slowest improvement rate of any of the methods, which explains its low score. In Figure 4.4 (c) and (d), the average number of generations and runtime for each method is plotted for each test. Since the mutation methods each use different parameters, the runtime is a more reliable comparison criteria than the number of generations, which can be misleading. The VF2 algorithm performed the best for almost all the tests, and the SOR mutation method did the worst.

Table 4.2 Mutation Method Comparison with Resubstitution Sketches Results

Mutation Method	Percent Found	Average Generations	Median Generations	Average Time (s)	Median Time (s)
SOR	51.9%	652	931	2594	2865
1-Seed	100%	14	5	159	44
2-Seed	99.4%	36	12	1703	494
VF2	100%	17	9	40	14

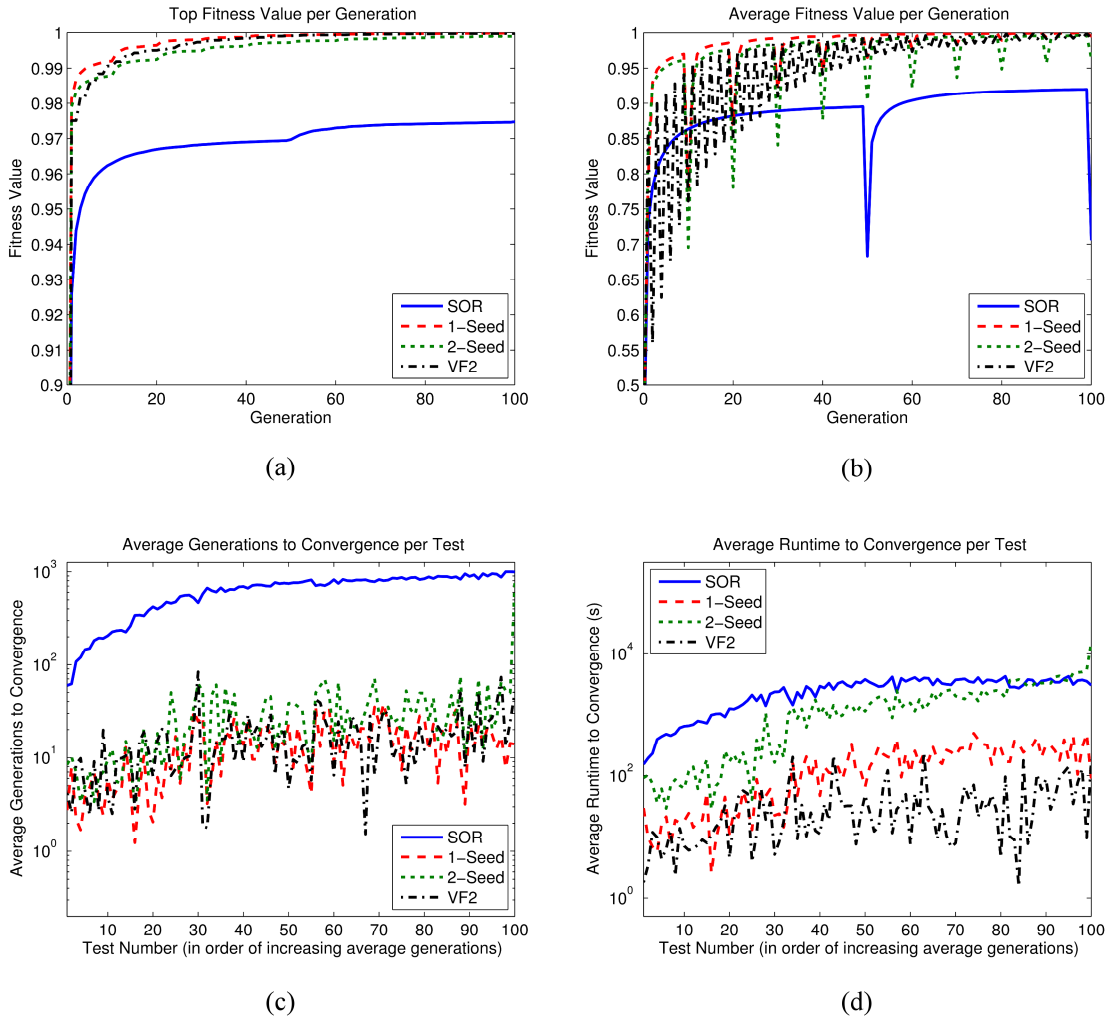


Figure 4.4 Results of the resubstitution experiments. (a) Top fitness value per generation. (b) Average fitness value per generation. (c) Average generations to convergence per test. (d) Average runtime to convergence per test.

Figure 4.5 shows two examples of the resubstitution experiment. In both examples, a set of five objects is chosen from the reference set for use as the ground truth. These objects are copied directly into the sketch and used for matching with each of the three mutation methods. The top five results found over all runs of each set are listed with their respective fitness values. The top match for each experiment is the ideal recovered match and the remaining results all share a similar spatial configuration. For

example, the second-best result of the first set in Figure 4.5 could match the sketch via a 180° rotation. The building which is completely surrounded by a parking lot in the sketch is surrounded on three sides in the second-best match and nearly all four sides in the fourth-best match. The second example set in Figure 4.5 produced very high scoring results, which could each match the sketch with a rotation.

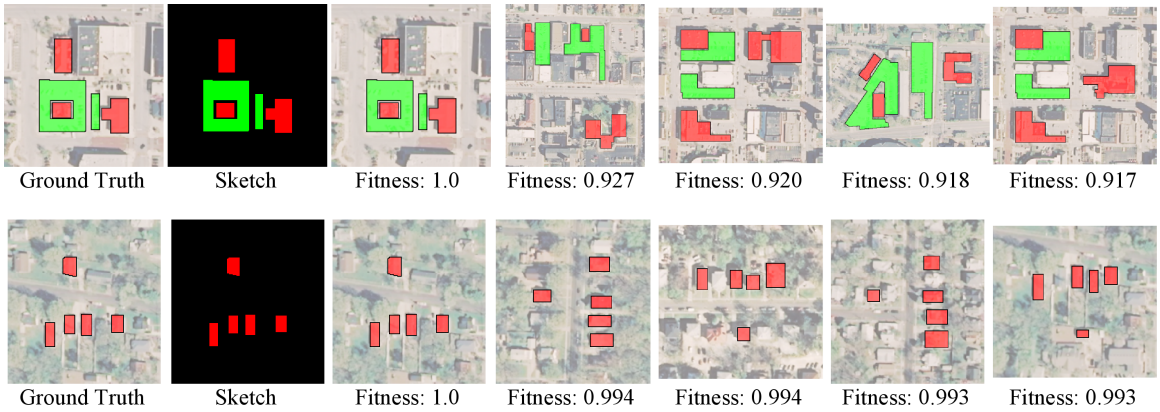


Figure 4.5 Examples of top matches for the resubstitution sketches. Buildings are shown in red and parking lots are shown in green. In both examples, the top match is the correct mapping to the ground truth set, and the remaining high scoring matches all share similar spatial configurations.

4.2.2 Simplified Sketches

The results of the mutation comparison experiment using simplified sketches are given in Table 4.3 and Figure 4.6. Again, the SOR mutation method performed rather poorly, and all of the methods had lower convergence rates and longer search times with the simplified sketches than with the resubstitution sketches. This can be attributed to the more complex search that occurs when the sketch does not perfectly match the ground truth. The one-seed method found the highest number of ideal matches, although the VF2 algorithm had the fastest average runtime. An interesting result is the particularly

long runtime of the two-seed set reconstruction mutation. The two-seed method converged to the correct match slightly less often than the one-seed method, which implies that the greater complexity of the two-seed method is unjustified. Although the two-seed method considers all possible edge assignments, the single starting seed of the one-seed method appears to be sufficient for matching even the simplified sketch configurations. Arguably, the added flexibility of the elastic angle object set comparison method allows the one-seed method to handle arbitrary orientations of the sketch sets. However, not all of the test sets were easy to find. The significant difference between the mean and median number of generations and the total runtime of the set reconstruction methods shows that they both have difficulty finding the ideal match for a small portion of the test sets. This is also shown by the sudden rise in runtime and generations to convergence for the last few tests in Figure 4.6c and Figure 4.6d.

Table 4.3 Mutation Method Comparison with Simplified Sketches Results

Mutation Method	Percent Found	Average Generations	Median Generations	Average Time (s)	Median Time (s)
SOR	49.1%	681	1000	3127	3710
1-Seed	97.0%	60	12	539	103
2-Seed	96.5%	81	21	6404	1231
VF2	94.0%	81	11	216	19

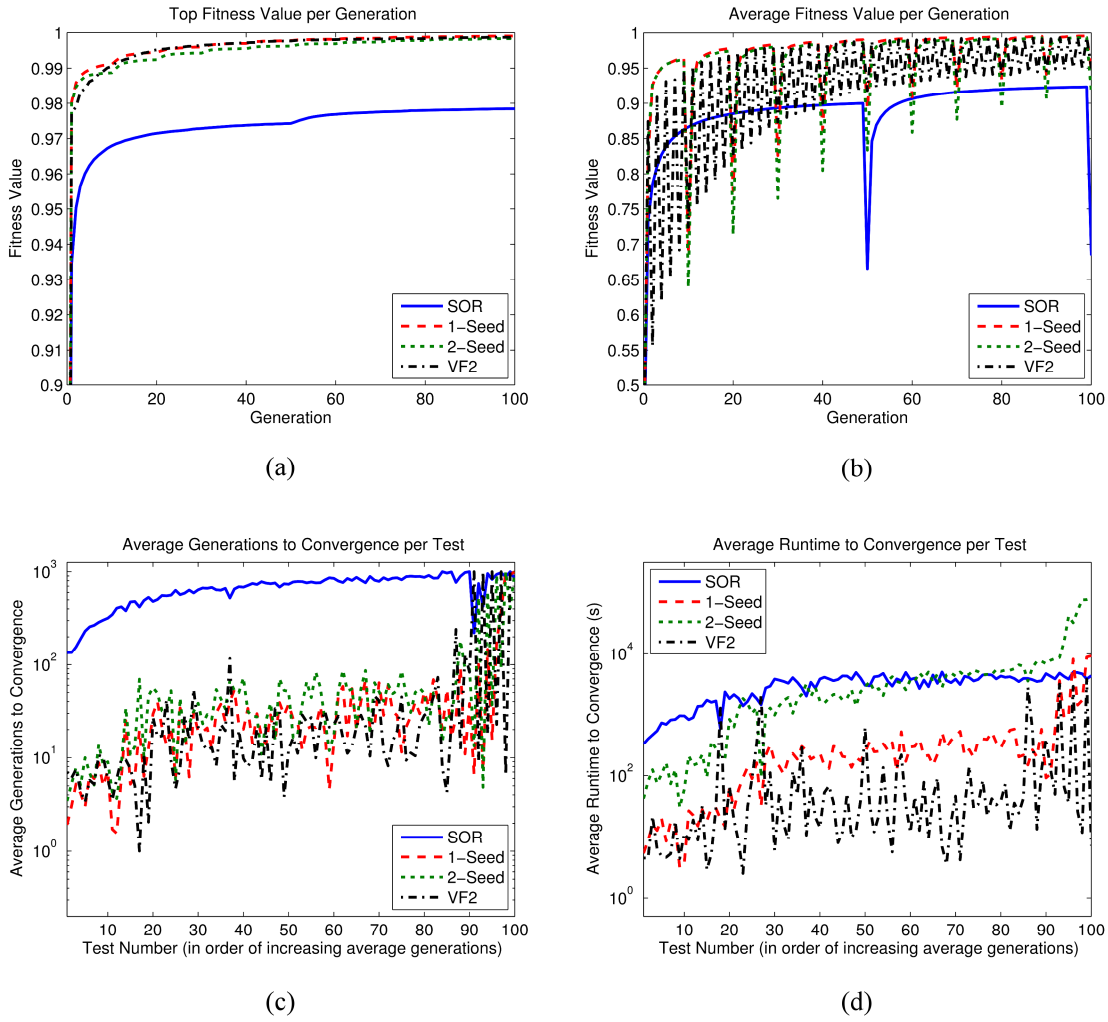


Figure 4.6 Results of the experiments with simplified sketches. (a) Top fitness value per generation. (b) Average fitness value per generation. (c) Average generations to convergence per test. (d) Average runtime to convergence per test.

Figure 4.7 shows two examples of experiments with simplified sketches. The ground truth sets are chosen in the same way as the resubstitution experiments, and each sketch is created by simplifying and rotating the objects. The sketches are then matched back onto the reference set using each of the mutation operators. The top five results found for each sketch are listed along with their corresponding fitness values. In the first example, the top match is the ideal mapping to the ground truth, and the remaining high-

scoring results all share a similar spatial configuration. The orientation of the set with respect to the sketch has little impact on the fitness value. The second example shows a set which did not converge to the ideal match, although all of the top-scoring results have a similar spatial configuration with the sketch. The ideal match for this configuration has a fitness of 0.976 due to the simplification process, which is lower than the fitness of the results shown. This indicates that although the search was unable to meet our criteria of finding the actual ground truth location, it was able to provide several alternate locations with high fitness.

Figure 4.7 Examples of top matches for the simplified sketches. Buildings are shown in red and parking lots are shown in green. The top match is the correct mapping to the ground truth set in the first example, whereas the ideal match is not one of the top scoring matches for the second example. All of the top matches for each example share a similar spatial configuration.

4.3 Impact of Sketch Size

The results of the first experiment show that the best mutation methods are the one-seed set reconstruction method and the VF2 subgraph isomorphism method. Both of these had very high convergence rates and relatively short runtimes. The second

experiment investigates the impact of the number of objects in the sketch using these two mutation methods. For sketch sizes of 4, 6, 8, 10, and 12 objects, 100 resubstitution and 100 simplified sketches are created from the Columbia reference database. Each sketch is matched using both the one-seed and VF2 mutation operators 10 times apiece. Altogether, this results in 1000 searches with each mutation operator for each sketch size. The algorithm parameters are identical to the first experiment, with the exception of a 100 generation limit rather than 1000 generations. This is done to limit the total search time for particularly large sketches. Again, the elastic angle method is employed and the search stops if the ideal match is found.

4.3.1 Resubstitution

The results of the sketch size experiment using resubstitution sketches are given in Table 4.4. The table shows some interesting trends, some of which are expected, while others are not. In general, both mutation methods had high convergence rates for sketch sizes less than 10 objects, and performed worse for larger sketches. The one-seed method had slightly higher convergence rates for all but the smallest sketch size of 4 objects. The search times increase exponentially as the sketch size increases. For 8 objects or fewer, the VF2 method ran the quickest, however for more than 8 objects, the one-seed method was faster. There is a significant disparity between the runtimes and convergence rates of the two mutation methods for 12 objects, implying that the one-seed mutation method is better suited for handling large sketch sizes.

Table 4.4 Sketch Size Comparison with Resubstitution Sketches Results

Mutation Method	Number of Objects in Sketch	Percent Found	Average Generations	Median Generations	Average Time (s)	Median Time (s)
1-Seed	4	95.1%	22	11	89	40
	6	98.5%	12	3	207	40
	8	99.6%	9	2	494	83
	10	94.8%	13	2	1258	150
	12	86.2%	20	2	3304	502
VF2	4	98.7%	16	9	26	10
	6	96.6%	19	11	81	24
	8	98.1%	19	11	189	62
	10	90.8%	26	13	1777	563
	12	76.5%	39	23	4986	2617

Figure 4.8 shows how the fitness values of the population change over time for each sketch size. The trends are very similar to the mutation comparison experiment, where the top fitness value rises very quickly and then levels off. Large sketch sizes result in lower fitness values overall, and a greater disparity between the two mutation methods. The one-seed method consistently has the same or higher fitness values than the VF2 method for all sketch sizes. Figure 4.9 shows the impact of sketch size on search time and generations required for convergence. The graphs show that larger sketches take longer to converge and require more generations. The largest sketch sizes of 10 and 12 objects had some tests which failed to converge at all. Apart from these few tests, the number of generations required for the VF2 algorithm increased as the sketch size increased, whereas the number of generations for the one-seed actually decreased. This shows that the one-seed method is particularly well suited for handling large sketch sizes, whereas the VF2 method does better for small sketches. This is reflected in the search times, where the VF2 method performs faster for all but the largest sketches.

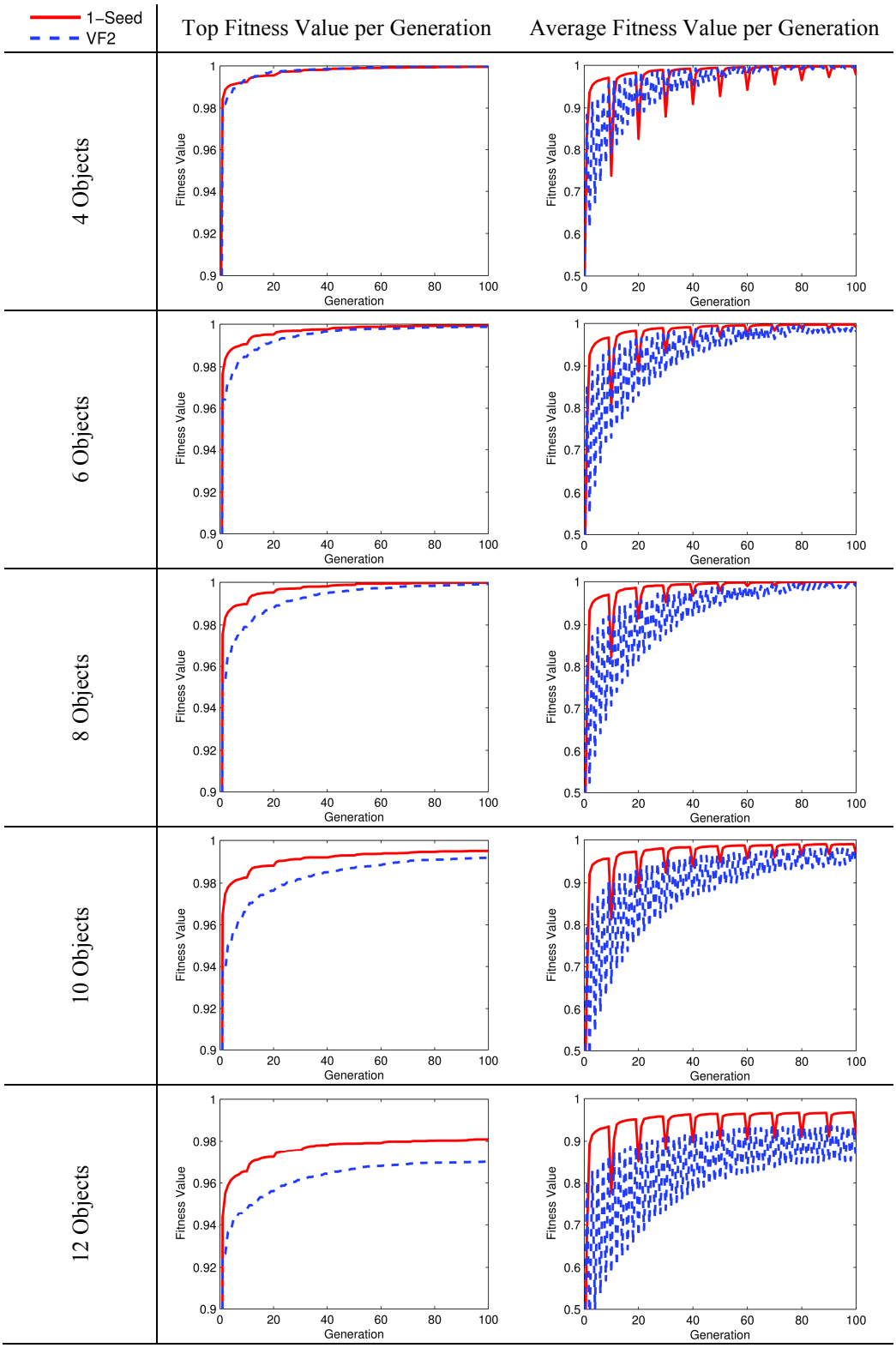


Figure 4.8 Impact of sketch size on fitness values for resubstitution sketches.

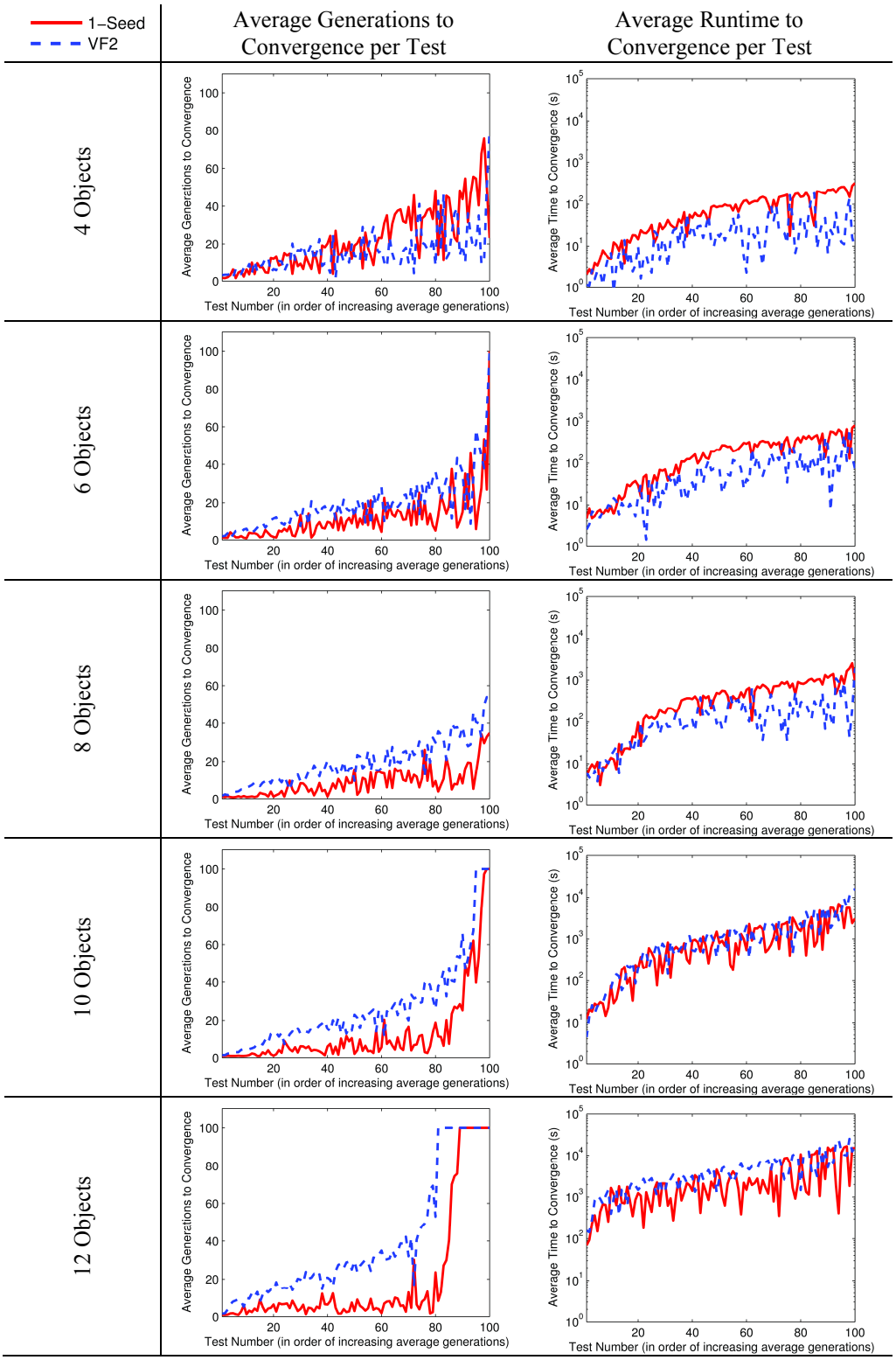


Figure 4.9 Impact of sketch size on convergence time for resubstitution sketches.

4.3.2 Simplified Sketches

The results of the sketch size experiments using simplified sketches are given in Table 4.5. Recall that the sketches for each sketch size are generated independently for both the resubstitution and simplified cases, so one should be careful when comparing the two cases directly. Overall, the trends for the simplified sketches are similar to the resubstitution sketch results, but more prominent. The convergence rates are lower across the board, which should be expected due to the simplification process. An interesting trend that was less visible with resubstitution sketches shows that very small sketches can be difficult to match. For sketches with only 4 objects, both the one-seed and the VF2 mutation methods found about 80% of the test sketches. While not as good as the resubstitution sketches, this is still a fairly high recall rate and could still produce results valuable to a human analyst. The lower convergence rates for small sketches may be because there are many configurations which could potentially match just 4 objects. Likewise, this might explain why there is an increase in convergence rates for very large sketches with 12 objects. There are fewer object configurations in the reference database which have compatible labels and are fully connected for large sketches, which reduces the size of the search space.

Table 4.5 Sketch Size Comparison with Simplified Sketches Results

Mutation Method	Number of Objects in Sketch	Percent Found	Average Generations	Median Generations	Average Time (s)	Median Time (s)
1-Seed	4	80.6%	36	18	151	60
	6	95.6%	16	5	318	84
	8	93.4%	18	5	771	194
	10	81.7%	28	7	2171	693
	12	87.2%	20	3	4317	874
VF2	4	80.4%	32	15	97	18
	6	94.3%	22	11	143	48
	8	86.0%	31	17	744	206
	10	69.2%	43	23	5927	1308
	12	78.8%	38	20	11009	3506

Figure 4.10 again shows the effect of sketch size on the population fitness values over time. The graphs are very similar to the resubstitution results in Figure 4.8 with lower fitness values in general for larger sketches and the one-seed method outperforming the VF2 in terms of population fitness. Figure 4.11 shows how the simplified sketches performed in terms of runtime and generations required to converge. The trends are similar to the resubstitution experiments in Figure 4.9, but requiring more time and generations, as verified by Table 4.5. A greater percentage of sketches failed to converge, as indicated by the large jumps in the average generation plots toward the last few tests for each sketch size. This is likely due to the rough simplification, which may make it difficult to ever recover the original ground truth location. The one-seed method again requires fewer generations, but takes more time for all but the largest sketch sizes. From these results, we can conclude that an ideal sketch size contains about 6 objects. With this size sketch, the VF2 mutation method provides the fastest search, although the one-seed method may produce higher scoring results. As the sketch size increases, the one-seed mutation method is preferable as the VF2 method quickly becomes intractable.

One possible strategy for matching a large sketch would be to break it into smaller sub-sketches and match each of these individually. However, this brings additional problems such as determining the best sub-sketches to use and how to recombine them into a complete solution.

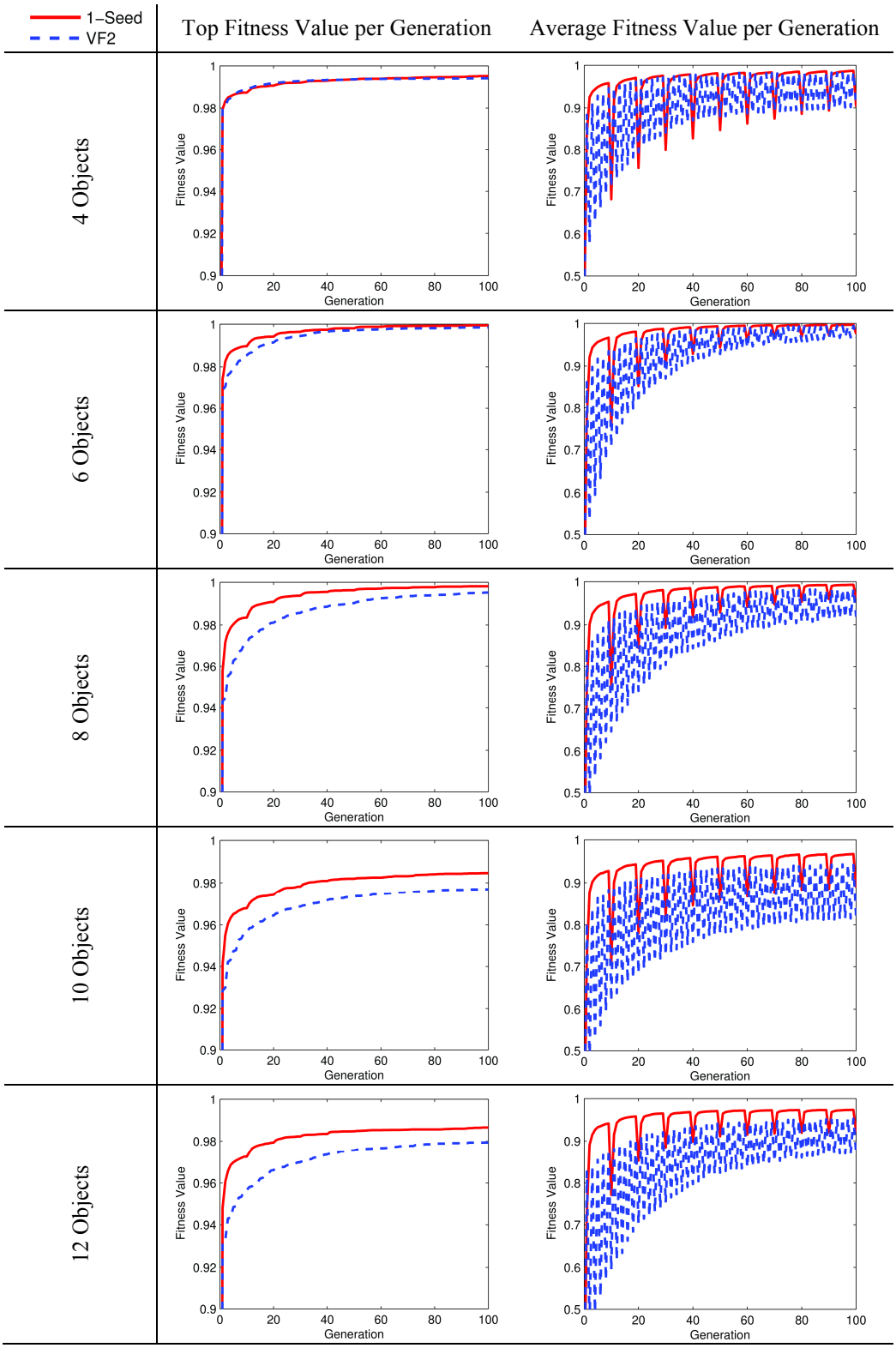


Figure 4.10 Impact of sketch size on fitness values for simplified sketches.

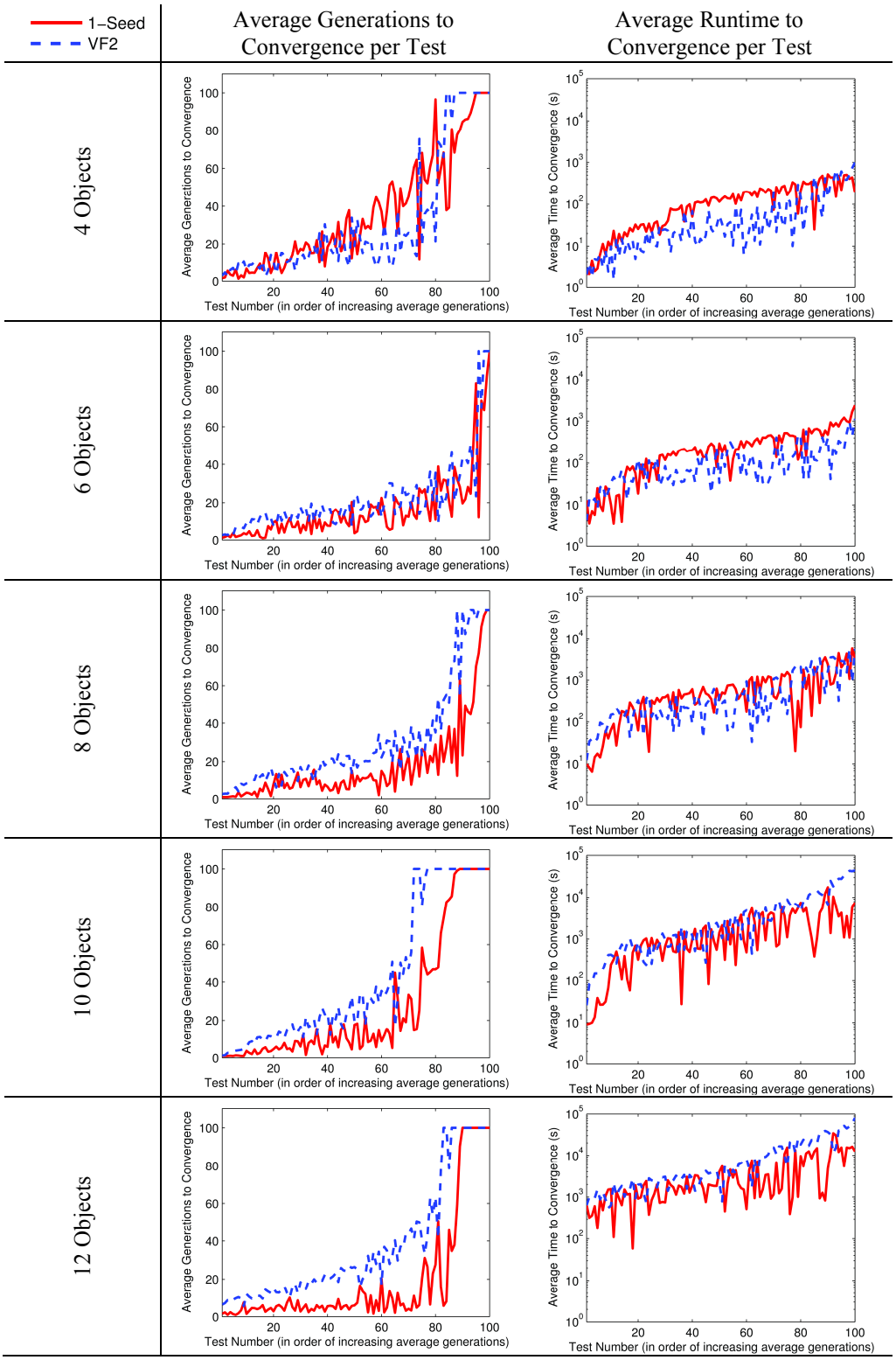


Figure 4.11 Impact of sketch size on convergence time for simplified sketches

4.4 Real-World Example

As a final example of the usefulness of the matching framework we have developed, consider the following example. The sketch in Figure 4.12 was hand-drafted to indicate a particular region of the Columbia reference image. The sketch was used as the input to the matching algorithm using the one-seed mutation method, a population size of 50 individuals, and a 10 generation limit. These parameters limit the search time to an amount reasonable to a human analyst. The top 10 matches found after 10 generations are shown in Figure 4.13. The top result is, in fact, the location that the sketch is intended to represent, rotated 90 degrees. The remaining results all share a similar configuration, with seven buildings and a parking lot. Some of the results are rotated to odd angles, but if the underlying road network and individual building orientations are ignored, one could imagine that the sketch might be matched to these results.

The invariance to rotation and the individual object shapes is one of the key characteristics of this matching algorithm. Significant simplifications have been made to both the size and shape of the sketch objects, yet the algorithm is still able to find the ideal matching location using relative directional spatial relationships. One future challenge for this work will be to incorporate road networks and other object features to help guide the search and ensure that the results make sense.

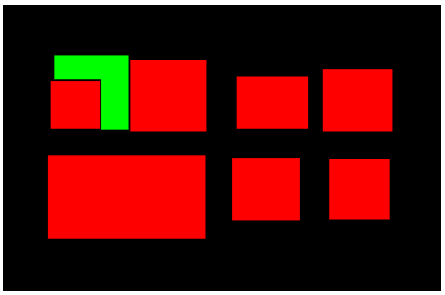


Figure 4.12 Hand-drafted example sketch.

Figure 4.13 Top 10 matching locations of the real-world example sketch.

5 CONCLUSION

Spatial relationships play an important role in describing scene configurations. A sketch can be described solely in terms of the spatial relationships between objects and still contain enough information to locate a matching copy within a large geospatial database. The evolutionary framework we have developed performs this search and provides many adjustable parameters to satisfy specific problem constraints. Each of the mutation operators provides a different search strategy which affects the performance of the algorithm. It was determined through experimentation that the one-seed set reconstruction method and the VF2 subgraph isomorphism hybrid method provide the fastest and most reliable searches. The VF2 method is optimized for small sketch sizes, whereas the one-seed method performs better for large sketches. Both methods have very high convergence rates and will yield high-scoring matches very quickly, even if the ideal match cannot be found.

The matching algorithm sits at the end of the T_2S pipeline, allowing sketches which are created from linguistic descriptions to be matched onto actual ground truth satellite imagery. This has many applications in the geospatial intelligence community, such as locating a person from just a linguistic description of his or her surroundings. Future work in this area may incorporate additional reasoning abilities to be able to infer a person's intent or future location.

The evolutionary framework of the matching algorithm suggests that it will be able to scale up to reference databases several orders of magnitude larger than the one used for these experiments. At these scales, it may be helpful to include additional

matching criteria beyond simple spatial relationships. In urban and suburban environments, road networks can provide a useful constraint for allowable match locations, and additional labels can help to quickly limit the search space. For example, sketches containing more descriptive labels such as, “restaurant”, “church”, or “house” could restrict searches to certain areas.

It seems advisable to continue the use of fuzzy methods in designing future search algorithms. The inherent ambiguity of building a sketch from linguistic descriptions and matching it to a real-world location requires techniques which can handle uncertainty. As additional features are added as search criteria, a fuzzy aggregation operator can be used to determine a single matching score. Although the methods here are presented specifically for the case of matching geospatial objects, they could be applied to any problem in which a specific spatial configuration is to be found from within a much larger database of objects.

REFERENCES

[Allen, et al., 2008] J. F. Allen, M. Swift, and W. de Beaumont, "Deep semantic analysis of text," in *Proceedings of the 2008 Conference on Semantics in Text Processing*, Venice, Italy, 2008, pp. 343-354.

[Barnea & Silverman, 1972] D. I. Barnea and H. F. Silverman, "A Class of Algorithms for Fast Digital Image Registration," *Computers, IEEE Transactions on,* vol. 21, pp. 179-186, 1972.

[Bloch, 1999] I. Bloch, "Fuzzy relative position between objects in image processing: A morphological approach," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 21, pp. 657-664, 1999.

[Bloch, et al., 2006] I. Bloch, O. Colliot, and R. M. Cesar, Jr., "On the ternary spatial relation 'Between'," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on,* vol. 36, pp. 312-327, 2006.

[Brown, 1992] L. G. Brown, "A survey of image registration techniques," *ACM Comput. Surv.,* vol. 24, pp. 325-376, 1992.

[Bruns & Egenhofer, 1996] H. T. Bruns and M. Egenhofer, "Similarity of spatial scenes," in *Seventh International Symposium on Spatial Data Handling (SDH '96)*, Delft, the Netherlands, 1996, pp. 31-42.

[Buck, et al., 2010] A. R. Buck, J. M. Keller, and M. Skubic, "A modified genetic algorithm for matching building sets with the histograms of forces," in *Evolutionary Computation (CEC), 2010 IEEE Congress on*, 2010, pp. 1-7.

[Buck, et al., 2011] A. R. Buck, J. M. Keller, M. Skubic, M. Detyniecki, and T. Baerecke, "Object set matching with an evolutionary algorithm," in *Computational Intelligence for Security and Defense Applications (CISDA), 2011 IEEE Symposium on*, Paris, France, 2011, pp. 43-50.

[Chan & Vese, 2001] T. F. Chan and L. A. Vese, "Active contours without edges," *Image Processing, IEEE Transactions on,* vol. 10, pp. 266-277, 2001.

[Conte, et al., 2004] D. Conte, P. Foggia, C. Sansone, and M. Vento, "Thirty years of graph matching in pattern recognition," *International Journal of Pattern Recognition and Artificial Intelligence,* vol. 18, pp. 265-298, 2004.

[Cordella, et al., 1998] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, "Subgraph transformations for the inexact matching of attributed relational graphs," *Computing,* vol. 12, pp. 43-52, 1998.

[Cordella, et al., 1999] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, "Performance evaluation of the VF graph matching algorithm," in *Proceedings of the International Conference on Image Analysis and Processing*, 1999, pp. 1172-1177.

[Cordella, et al., 2004] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, "A (sub)graph isomorphism algorithm for matching large graphs," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 26, pp. 1367-1372, 2004.

[Cremers, et al., 2006] D. Cremers, N. Sochen, and C. Schnörr, "A multiphase dynamic labeling model for variational recognition-driven image segmentation," *International Journal of Computer Vision,* vol. 66, pp. 67-81, 2006.

[De Castro & Morandi, 1987] E. De Castro and C. Morandi, "Registration of translated and rotated images using finite fourier transforms," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 9, pp. 700-703, 1987.

[Dubois & Jaulent, 1987] D. Dubois and M. C. Jaulent, "A general approach to parameter evaluation in fuzzy digital pictures," *Pattern Recognition Letters,* vol. 6, pp. 251-259, 1987.

[Egenhofer & Franzosa, 1991] M. Egenhofer and R. Franzosa, "Point-set topological spatial relations," *International Journal of Geographical Information Systems,* vol. 5, pp. 161-174, 1991.

[Egenhofer, 1997] M. J. Egenhofer, "Query processing in spatial-query-by-sketch," *Journal of Visual Languages and Computing,* vol. 8, pp. 403-424, 1997.

[Fisher, 1993] N. I. Fisher, *Statistical analysis of circular data*. Cambridge, England; New York, NY, USA: Cambridge University Press, 1993.

[Freeman, 1975] J. Freeman, "The modeling of spatial relations," *Computer Graphics and Image Processing,* vol. 4, pp. 156-171, 1975.

[Goldberg, 1989] D. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[Holland, 1975] J. H. Holland, *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.

[Houck, et al., 1996] C. Houck, J. Joines, and M. Kay, "Utilizing lamarckian evolution and the baldwin effect in hybrid genetic algorithms," Meta-Heuristic Research and Applicat. Group, Depart. Ind. Eng., North Carolina State Univ., Raleigh, CA, NCSU-IE Tech. Report 96-01,1996.

[Keller, et al., 1999] J. M. Keller, P. Gader, and W. Xiaomei, "LADAR scene description using fuzzy morphology and rules," in *Computer Vision Beyond the Visible Spectrum: Methods and Applications, 1999. (CVBVS '99) Proceedings. IEEE Workshop on*, 1999, pp. 120-129.

[Keller & Sztandera, 1990] J. M. Keller and L. Sztandera, "Spatial relations among fuzzy subsets of an image," in *Uncertainty Modeling and Analysis, 1990. Proceedings., First International Symposium on*, 1990, pp. 207-211.

[Keller & Wang, 1995] J. M. Keller and X. Wang, "Comparison of spatial relation definitions in computer vision," in *Proceedings of ISUMA - NAFIPS '95 The Third International Symposium on Uncertainty Modeling and Analysis and Annual Conference of the North American Fuzzy Information Processing Society*, 1995, pp. 679-684.

[Keller & Wang, 2000] J. M. Keller and X. Wang, "A fuzzy rule-based approach to scene description involving spatial relationships," *Computer Vision and Image Understanding,* vol. 80, pp. 21-41, 2000.

[Krishnapuram, et al., 1993] R. Krishnapuram, J. M. Keller, and Y. Ma, "Quantitative analysis of properties and spatial relations of fuzzy image regions," *Fuzzy Systems, IEEE Transactions on,* vol. 1, pp. 222-233, 1993.

[Lowe, 2004] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision,* vol. 60, pp. 91-110, 2004.

[Marjamaa, et al., 2001] J. Marjamaa, O. Sjahputera, J. M. Keller, and P. Matsakis, "Fuzzy scene matching in LADAR imagery," in *Fuzzy Systems, 2001. The 10th IEEE International Conference on*, 2001, pp. 692-695.

[Mark & Egenhofer, 1994] D. M. Mark and M. J. Egenhofer, "Modeling spatial relations between lines and regions: combining formal mathematical models and human subjects testing," *Cartography and Geographic Information Systems,* vol. 21, pp. 195-212, 1994.

[Matsakis, et al., 2004] P. Matsakis, J. M. Keller, O. Sjahputera, and J. Marjamaa, "The use of force histograms for affine-invariant relative position description," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 26, pp. 1-18, 2004.

[Matsakis, et al., 2001] P. Matsakis, J. M. Keller, L. Wendling, J. Marjamaa, and O. Sjahputera, "Linguistic description of relative positions in images," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on,* vol. 31, pp. 573-588, 2001.

[Matsakis & Wendling, 1999] P. Matsakis and L. Wendling, "A new way to represent the relative position between areal objects," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 21, pp. 634-643, 1999.

[Miyajima & Ralescu, 1994] K. Miyajima and A. Ralescu, "Spatial organization in 2D segmented images: Representation and recognition of primitive spatial relations," *Fuzzy Sets and Systems,* vol. 65, pp. 225-236, 1994.

[Moscato, 1989] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms," Caltech Concurrent Computation Program, Publication Report 790,1989.

[Nedas & Egenhofer, 2008] K. A. Nedas and M. J. Egenhofer, "Spatial-scene similarity queries," *Transactions in GIS,* vol. 12, pp. 661-681, 2008.

[Parekh, 2007] G. Parekh, "Scene matching between a quantitative map and a qualitative hand drawn sketch," M.S., University of Missouri-Columbia, 2007.

[Parekh, et al., 2007] G. Parekh, M. Skubic, O. Sjahputera, and J. M. Keller, "Scene matching between a map and a hand drawn sketch using spatial relations," in *Robotics and Automation, 2007 IEEE International Conference on*, 2007, pp. 4007-4012.

[Reddy & Chatterji, 1996] B. S. Reddy and B. N. Chatterji, "An FFT-based technique for translation, rotation, and scale-invariant image registration," *Image Processing, IEEE Transactions on,* vol. 5, pp. 1266-1271, 1996.

[Riklin-Raviv, et al., 2007] T. Riklin-Raviv, N. Kiryati, and N. Sochen, "Prior-based segmentation and shape registration in the presence of perspective distortion," *International Journal of Computer Vision,* vol. 72, pp. 309-328, 2007.

[Rodriguez & Jarur, 2005] M. A. Rodriguez and M. C. Jarur, "A genetic algorithm for searching spatial configurations," *Evolutionary Computation, IEEE Transactions on,* vol. 9, pp. 252-270, 2005.

[Rosenfeld, 1979] A. Rosenfeld, "Fuzzy digital topology," *Information and Control,* vol. 40, pp. 76-87, 1979.

[Rosenfeld, 1983] A. Rosenfeld, "On connectivity properties of grayscale pictures," *Pattern Recognition,* vol. 16, pp. 47-50, 1983.

[Rosenfeld, 1984] A. Rosenfeld, "The fuzzy geometry of image subsets," *Pattern Recognition Letters,* vol. 2, pp. 311-317, 1984.

[Shackelford & Davis, 2003a] A. K. Shackelford and C. H. Davis, "A hierarchical fuzzy classification approach for high-resolution multispectral data over urban areas," *Geoscience and Remote Sensing, IEEE Transactions on,* vol. 41, pp. 1920-1932, 2003.

[Shackelford & Davis, 2003b] A. K. Shackelford and C. H. Davis, "A combined fuzzy pixel-based and object-based approach for classification of high-resolution multispectral data over urban areas," *Geoscience and Remote Sensing, IEEE Transactions on,* vol. 41, pp. 2354-2363, 2003.

[Shapiro & Haralick, 1981] L. G. Shapiro and R. M. Haralick, "Structural descriptions and inexact matching," *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 3, pp. 504-519, 1981.

[Sjahputera, 2004] O. Sjahputera, "Object registration in scene matching based on spatial relationships," Ph. D., University of Missouri-Columbia, 2004.

[Sjahputera & Keller, 2005a] O. Sjahputera and J. M. Keller, "Particle swarm over scene matching," in *Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE*, 2005, pp. 108-115.

[Sjahputera & Keller, 2005b] O. Sjahputera and J. M. Keller, "Possibilistic c-means in scene matching," in *Fourth International Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT)*, 2005, pp. 669-675.

[Sjahputera & Keller, 2007] O. Sjahputera and J. M. Keller, "Scene matching using F-histogram-based features with possibilistic C-means optimization," *Fuzzy Sets Syst.,* vol. 158, pp. 253-269, 2007.

[Sjahputera, et al., 2003] O. Sjahputera, J. M. Keller, and P. Matsakis, "Scene matching by spatial relationships," in *Fuzzy Information Processing Society, 2003. NAFIPS 2003. 22nd International Conference of the North American*, 2003, pp. 149-154.

[Sjahputera, et al., 2000] O. Sjahputera, J. M. Keller, P. Matsakis, P. Gader, and J. Marjamaa, "Histogram-based scene matching measures," in *Fuzzy Information Processing Society, 2000. NAFIPS. 19th International Conference of the North American*, 2000, pp. 392-396.

[Skubic, et al., 2003] M. Skubic, C. Bailey, and G. Chronis, "A sketch interface for mobile robots," in *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, 2003, pp. 919-924 vol.1.

[Skubic, et al., 2004] M. Skubic, S. Blisard, C. Bailey, J. A. Adams, and P. Matsakis, "Qualitative analysis of sketched route maps: translating a sketch into linguistic descriptions," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on,* vol. 34, pp. 1275-1282, 2004.

[Sledge, et al., 2011] I. Sledge, J. Keller, S. Wenbo, and C. Davis, "Conflation of vector buildings with imagery," *Geoscience and Remote Sensing Letters, IEEE,* vol. 8, pp. 83-87, 2011.

[Sledge & Keller, 2009] I. J. Sledge and J. M. Keller, "Mapping natural language to imagery: Placing objects intelligently," in *Fuzzy Systems, 2009. FUZZ-IEEE 2009. IEEE International Conference on*, 2009, pp. 518-523.

[Svedlow, et al., 1978] M. Svedlow, C. D. McGillem, and P. E. Anuta, "Image Registration: Similarity Measure and Preprocessing Method Comparisons," *Aerospace and Electronic Systems, IEEE Transactions on,* vol. 14, pp. 141-150, 1978.

[Tsai & Fu, 1979] W.-H. Tsai and K.-S. Fu, "Error-correcting isomorphisms of attributed relational graphs for pattern analysis," *Systems, Man and Cybernetics, IEEE Transactions on,* vol. 9, pp. 757-768, 1979.

[Ullmann, 1976] J. R. Ullmann, "An algorithm for subgraph isomorphism," *J. ACM,* vol. 23, pp. 31-42, 1976.

[Winston, 1975] P. Winston, "Learning structural descriptions from examples," in P. Winston (ed.), *The Psychology of Computer Vision*, McGraw-Hill, New York, 1975.

[Wong, 1978] R. Y. Wong, "Sequential scene matching using edge features," *Aerospace and Electronic Systems, IEEE Transactions on,* vol. 14, pp. 128-140, 1978.