A Memetic Algorithm for Matching Spatial Configurations With the Histograms of Forces

Andrew R. Buck, *Student Member, IEEE,* James M. Keller, *Fellow, IEEE,* and Marjorie Skubic, *Member, IEEE*

***Abstract*—In this paper, we present an approach for modeling and comparing small sets of 2-D objects based on their spatial relationships. 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 on the specific problem of matching a sketched map containing several 2-D objects to hand-segmented satellite imagery. We define a similarity measure between the spatial configurations of two object sets, which uses attributed relational graphs to represent scene information. Objects are represented as graph nodes and edges are defined by the histograms of forces between object pairs. We develop a memetic algorithm based on a $(\mu + \lambda)$ evolution strategy to solve this scene-matching problem with three domain-specific local search operators that are compared experimentally.**

***Index Terms*—Attributed relational graph (ARG), histogram of forces (HoF), memetic algorithms, scene matching, text-to-sketch (T_2S).**

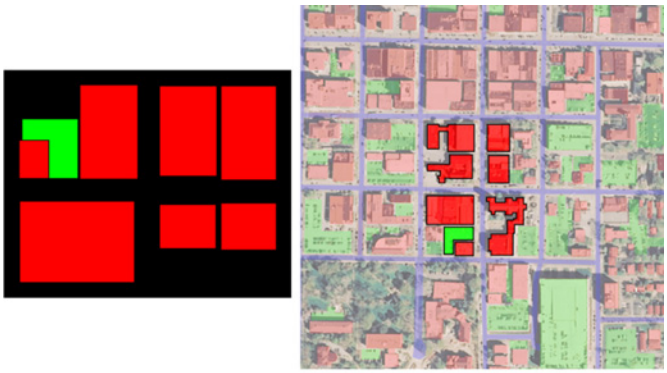


Fig. 1. (a) Example of a machine-drafted sketch and (b) its corresponding match within a segmented satellite image. Buildings are shown in red (dark gray) and parking lots are shown in green (light gray).

I. Introduction

A GEOGRAPHIC information system (GIS) that stores and manipulates spatial information is often required to perform the task of scene matching. A scene can be defined as a certain configuration of objects or image features that may have some real-world origin. Scene matching is a high-level task that applies computer vision and pattern recognition techniques to find corresponding regions in multiple images. From a GIS point of view, scene matching can be used to identify multiple views of the same scene [1] or to perform a query by sketch [2], in which a small target sketch must be found within a much larger image. For this application, a sketch is defined as a specific spatial configuration of 2-D objects that is to be found in a much larger GIS database of annotated object locations. This can be used to match a sketched map of object locations to a real-world location using segmented satellite imagery (see Fig. 1).

This paper is motivated by a recent grant from the U.S. National Geospatial-Intelligence Agency (NGA) aimed at tackling the inverse problem of linguistic scene description, in which a set of one or more spatial descriptions is used to construct an approximate sketch of object locations [3]. This is called text-to-sketch (T_2S) by the NGA. The input to the T_2S system is a set of linguistic spatial descriptions. Descriptions often relate two objects based on their relative position, such as "I see a large building to the left," or "There is a parking lot directly behind the building to my right." Buildings, parking lots, and other descriptive entities that appear in the descriptions are drawn by the T_2S system as 2-D objects in an image. Apart from a small collection of labels, the only defining features of these sketches are the shapes, sizes, and spatial relationships between the objects. Being built from linguistic descriptions, the sketches are rarely a completely accurate representation of the actual ground-truth objects; thus, our model must be tolerant of variations in shape, size, scale, and position.

The core requirement of scene matching is the definition of a measure that assesses the similarity of two scenes. For GIS systems, this is usually done by evaluating the similarity of the two spatial configurations. Since sketches are often constructed from qualitative descriptions rather than exact dimensions, we seek ways in which to represent this qualitative spatial information. Most methods use some combination of topology, orientation, size, shape, and distance to represent spatial relationships [4]. Several crisp similarity measures have been defined between scenes using topological models based

Manuscript received November 25, 2011; revised July 17, 2012 and October 2, 2012; accepted October 10, 2012. Date of publication November 12, 2012; date of current version July 25, 2013. This work was supported by the U.S. National Geospatial Agency NURI under Grant HM 1582-08-1-0020.

The authors are with the Department of Electrical and Computer Engineering, University of Missouri, Columbia, MO 65211 USA (e-mail: arb9p4@mail.missouri.edu; kellerj@missouri.edu; skubicm@missouri.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2012.2226889

1089-778X/$31.00 © 2012 IEEE

on minimum bounding rectangles [5], orientation graphs [6], and by counting the number of gradual changes required to transform one scene into another [7]. One drawback of these methods is that the basic topological framework must be substantially appended with notions of direction and distance in order to handle situations in which nearly all objects are disjointed.

In contrast to these crisp methods, the histograms of forces (HoF) [8] provide a fuzzy framework for expressing the relative position between 2-D objects based on directional relationships. The HoF have been shown to be affine invariant [9], allowing for a robust similarity measure between two scenes that can handle arbitrary changes in rotation, scale, and translation. In [1] and [10], the HoF are used to generate scene descriptors representing all of the spatial relationships between objects in a scene. A correspondence map is built using a possibilistic C-means clustering method, which associates pairs of objects between scenes. The HoF method is compared to the Fourier–Mellin transform method of image registration [11], and shown to be more robust to variations in viewing geometries. These scene-matching methods focus solely on establishing a correspondence and confidence value between two predefined scenes, however, and do not provide a way to search for the existence of one scene within another. Our method uses the descriptive power of the HoF to define the object correspondence between scene objects within a larger evolutionary framework.

In this paper, we focus exclusively on the HoF method of capturing spatial information in a scene and show how it can be used for the task of scene matching. We develop a similarity measure based on the concept of an attributed relational graph (ARG) [12], [13], in which scene objects become the nodes of a graph and the HoF relationships between objects become edges. By representing both a query sketch and a reference GIS database as ARGs, scene matching can be described as an approximate subgraph-matching problem in which we seek to find a subgraph of the reference ARG that has maximum similarity to the sketch ARG. This approach was used by Berretti *et al*. [14] to perform image retrieval and by Nedas and Egenhofer [15] for querying spatial scenes. In the latter, the search problem is cast as a constraint satisfaction problem [16] in which a set of variables representing the scene objects must be mapped to a specific set of possible reference objects such that the induced ARG is isomorphic to the query ARG. The notion of an error-tolerant subgraph isomorphism is used to allow for the types of partial constraints that arise in real searches. This represents a combinatorial optimization problem, for which an exhaustive search would quickly become intractable on large databases.

Evolutionary algorithms have been shown to be well suited to solving optimization problems, particularly in noisy or uncertain environments [17], which makes them attractive methods for searching spatial configurations. Rodríguez and Jarur [18] developed a genetic algorithm that uses a measure based on topological relationships and the distance between objects to evaluate the similarity of object pairs, but does not consider directional relationships. Papadias *et al*. [19] combined a hill-climbing strategy with an evolutionary approach for this problem using a crisp relation scheme. Although these methods were shown to be effective in their own domains, our work focuses on the application of the HoF to the problem of scene matching, and a thorough comparison is left for future work. The method used in [19] is similar to a memetic algorithm [20], which adds an individual refinement step to the standard evolutionary algorithm. This is an effective search strategy for many different types of problems [21], particularly those that operate on graphs. In [22], evidence is given that a domain-specific heuristic can improve a genetic algorithm for the maximum clique problem, and in [23], a memetic algorithm is used to find a solution to the capacitated arc-routing problem. We have designed three novel local improvement search operators that are used within a memetic algorithm intended for searching spatial configurations. This allows for the incorporation of domain-specific knowledge into a parallel search procedure with a population of candidate solutions covering a large geographic search area.

The remainder of this paper is organized as follows. In Section II, we define a similarity measure that uses an HoF–ARG model for representing spatial relationships. Section III outlines the memetic framework with Section IV describing the local search methods in detail. Section V gives our experimental results using the proposed method, and our conclusions are made in Section VI.

II. Evaluating Spatial Similarity

In this section, we discuss the development of a similarity measure between two spatial scenes. We use the notion of the HoF as attributes for ARGs and define ways to compare both individual force histograms and complete ARGs. The issue of orientation independence is addressed and a novel comparison method that we call elastic angles is introduced.

A. Histograms of Forces

The relative position of a pair of 2-D objects A and B can be represented by the set of forces acting between them. For every direction θ, we calculate the sum of elementary forces acting between A and B in direction θ (see Fig. 2). These forces can be aggregated into the F-histogram $F_r^{AB}(\theta)$, which maps $\mathbb{R} \rightarrow \mathbb{R}^+$ and represents the degree of support for the proposition, "A is in direction θ of B." Provided that A and B are both nonempty regions and we compute θ on a fine enough scale, F_r^{AB} should have at least one element greater than zero. We calculate the magnitude of the individual forces 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.

B. Main Direction

Often, we need to reduce the spatial relationship to a single scalar direction φ^{AB}, which is the main direction between the

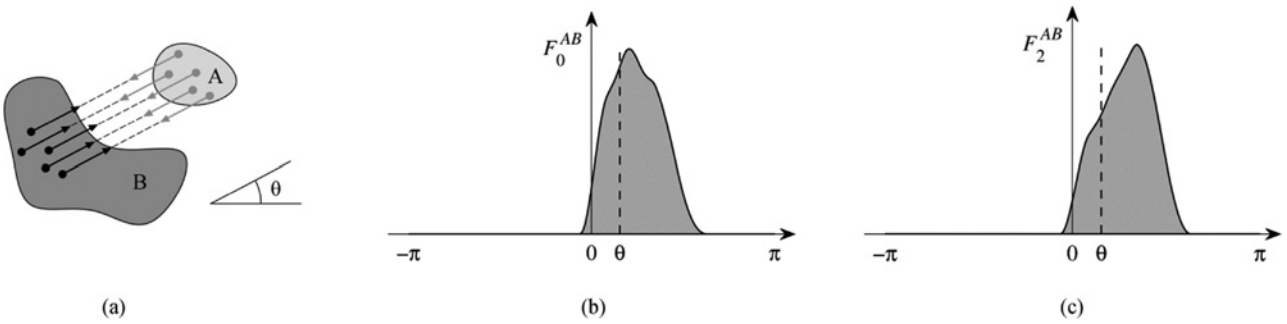


Fig. 2. (a) Force histogram F_r^{AB} is the scalar resultant of elementary forces exerted by the points of A on those of B. Each one pulls B in direction θ. (b) Histogram of constant forces ($r = 0$) is one representation of the spatial relationship between A and B providing a global perspective. (c)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

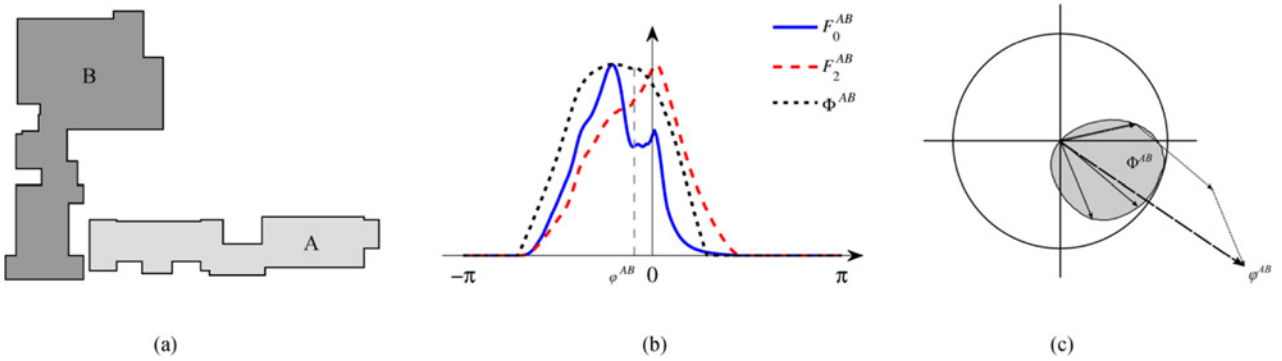


Fig. 3. (a) Pair of objects for which the constant and gravitational force histograms indicate different primary directions. (b) F_0^{AB} and F_2^{AB} histograms can be combined into the main direction histogram, Φ^{AB}. The centroid of this histogram gives the scalar main direction φ^{AB}, which is 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.

objects A and B. Matsakis *et al.* [24] presented a method that uses both the F_0 and F_2 histograms for assessing the degree of truth of the statement, "A is in direction θ of B." This is especially important in cases where the F_0 and F_2 histograms would by themselves indicate different primary directions, such as in Fig. 3. By using a common value for the main direction, we can treat the F_0 and F_2 histograms as a pair with a single reference axis. For each angle θ, the forces of F_r^{AB} are categorized as effective, contradictory, or compensatory. Contradictory forces are those that oppose the proposition, "A is in direction θ of B." Compensatory forces are chosen from the noncontradictory forces to balance the pull of the contradictory forces. Any remaining forces are labeled as effective forces, and are used to compute four statistical values, $a_0^{AB}(\theta)$, $a_2^{AB}(\theta)$, $b_0^{AB}(\theta)$, and $b_2^{AB}(\theta)$. Here, a_r^{AB} represents the combined degree of truth that "A is in direction θ of B" according to the F-histogram F_r^{AB}, and b_r^{AB} represents the percentage of all forces that are effective. Details of this computation can be found in [24]. By evaluating all directions, we define the combined force histogram as

$$\Phi^{AB}(\theta) = \max\{a_0^{AB}(\theta), \min\{a_2^{AB}(\theta), b_0^{AB}(\theta)\}\}. \quad (1)$$

Skubic *et al.* [25] define the main direction φ^{AB} as the direction θ for which $\Phi^{AB}(\theta)$ is maximum. We chose to use a more robust approach in which we define φ^{AB} to be the direction of the centroid of the combined force histogram, Φ^{AB}. Because Φ^{AB} is defined on a periodic domain, we must use polar vector summation to ensure that all directions are treated equally [26]. This is especially true for cases in which A surrounds B or vice versa in which there is no suitable 2π range of Φ^{AB} that could serve as a linear mapping. We define the main direction as

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

where $\text{atan2}(y, x) : \mathbb{R}\times\mathbb{R} \to [0, 2\pi)$ is the two-argument variation of the arctangent function. Although we could certainly use the centroid of either the F_0 or F_2 histogram to represent the primary direction, the main direction interpretation defined above gives a unified framework that is more consistent with the natural human interpretation.

C. Representing Object Sets

Suppose that we have a set of 2-D objects $\mathcal{O} = \{o_1, o_2, \ldots, o_n\}$, each with a label defined by the function $L(o_i) = l_i \in \mathcal{L}$, where $\mathcal{L}$ is the set of all possible labels. For any pair of objects $(o_i, o_j) \in \mathcal{O}$, we can represent the directional relationship between them with the F-histograms $F_0^{o_i o_j}$ and $F_2^{o_i o_j}$, and we can also compute the main direction $\varphi^{o_i o_j}$. As the number of objects grows, it becomes convenient to represent the set as an ARG in which each vertex and edge is assigned a set of attributes. We can represent each object as a vertex in an ARG, and assign the relationships between objects as edge attributes. Let us define the ARG for a set of 2-D objects $\mathcal{O}$

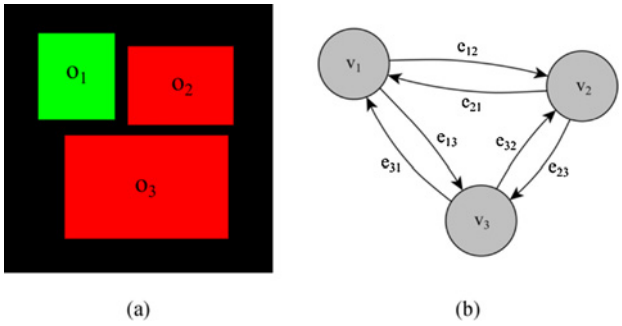


Fig. 4. (a) Example of an object set $\mathcal{O} = \{o_1, o_2, o_3\}$. (b) ARG representation of $\mathcal{O}$ is $G_{\mathcal{O}} = (V_{\mathcal{O}}, E_{\mathcal{O}})$, where $V_{\mathcal{O}} = \{v_1, v_2, v_3\}$ and $E_{\mathcal{O}} = \{e_{12}, e_{13}, e_{21}, e_{23}, e_{31}, e_{32}\}$. Each vertex v_i is a pair (o_i, l_i) where l_1 = parking lot and $l_2 = l_3$ = building. Each edge e_{ij} is a triple $(F_0^{o_i o_j}, F_2^{o_i o_j}, \varphi^{o_i o_j})$ containing the calculated F-histograms and main direction between each object pair.

as $G_{\mathcal{O}} = (V_{\mathcal{O}}, E_{\mathcal{O}})$ where $V_{\mathcal{O}}$ is the set of vertices and their attributes, and $E_{\mathcal{O}}$ is the set of edges and their attributes. A vertex $v_i = (o_i, l_i) \in V_{\mathcal{O}}$ is a pair containing an object $o_i \in \mathcal{O}$ and its label l_i. An edge $e_{ij} = (F_0^{o_i o_j}, F_2^{o_i o_j}, \varphi^{o_i o_j}) \in E_{\mathcal{O}}$ is a triple that connects two vertices in the graph and contains the histograms of constant and gravitational forces, as well as the main direction between those two objects. 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 pair of vertices. An example ARG representation of an object set is given in Fig. 4.

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." However, the two statements contain largely the same information, and we can relate them with the semantic inverse property of the HoF [9], which states that

$$F_r^{BA}(\theta) = F_r^{BA}(\theta + \pi). \tag{3}$$

Since F_r^{AB} is a periodic function, this is equivalent to a circular shifting of the histogram bins, in which no information is lost. We can reduce the storage requirement of our 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.

D. Comparing Histograms

Toward the goal of developing a similarity measure between the spatial configurations of object sets, 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.* [9] investigated several similarity measures for F-histograms of which we choose the cross correlation for its invariance to scale. The cross-correlation of two individual histograms, h_1 and h_2, is defined as

$$\mu_C(h_1, h_2) = \frac{\sum_\theta h_1(\theta) h_2(\theta)}{\sqrt{\sum_\theta h_1^2(\theta)} \sqrt{\sum_\theta h_2^2(\theta)}} \tag{4}$$

which is guaranteed to be in the range [0, 1] provided that both h_1 and h_2 contain no elements less than zero, and at least one element greater than zero. A cross-correlation of 1 implies a perfect match, whereas 0 implies that the support of h_1 and h_2 have no common elements.

A special feature of the cross-correlation is that it is independent of the scale of h_1 and h_2. This means that we do not need to normalize the y-axis values of the F-histograms. The x-axis values, however, are defined with respect to the reference angle where $\theta = 0$. 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 HoF [8], [9], which state that if a pair of objects (A, B) is rotated counter clockwise by an angle φ, its F-histogram becomes

$$F_r^{\text{rot}(A,B)}(\theta) = F_r^{AB}(\theta - \varphi). \tag{5}$$

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

$$\mu_{\text{Pair}}(A, B, \phi, A', B', \phi') = \beta \mu_{C0} + (1 - \beta)\mu_{C2} \tag{6}$$

where

$$\mu_{C0} = \mu_C(F_0^{AB}(\theta - \phi), F_0^{A'B'}(\theta - \phi'))$$
$$\mu_{C2} = \mu_C(F_2^{AB}(\theta - \phi), F_2^{A'B'}(\theta - \phi')).$$

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.

E. Comparing Object Sets

Suppose that we are given two sets, each containing the same number of objects in a defined order $S = (o_1, o_2, \ldots, o_n)$ and $S' = (o'_1, o'_2, \ldots, o'_n)$. We can compare S and S' by measuring the average similarity of the spatial relationships between each pair of objects. Note that we avoid the general correspondence problem and assume that the object order is the same in both sets. If we can guarantee that both S and S' are defined with the same orientation, then we can compute the similarity of the two object sets as

$$\Psi_{\text{Static}}(S, S') = \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mu_{\text{Pair}}(o_i, o_j, 0, o'_i, o'_j, 0). \tag{7}$$

Here both reference angles are defined as zero, implying that no shifting of the histograms is necessary. The output range of this similarity measure is in the range [0, 1] and the complexity to compute it given S and S' 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.

Although the above expression is valid if both object sets are defined with respect to the same reference angle, we typically do not have any way to guarantee that this condition holds. Consider the case of matching a sketched map of roads and buildings to ground truth imagery. 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 two object sets, we rotate all of the F-histograms from S' by the angle φ^* that would give the best overall alignment with the F-histograms from S.

We define the angular difference between two pairs of objects (A, B) and (A', B') as the difference between their main directions, $\varphi^{AB} - \varphi^{A'B'}$. The angular difference between each unique pair of objects in S and S' makes a list of angular differences

$$D = \{d_{11}, d_{12}, \ldots, d_{ij}, \ldots, d_{(n-1)n}\}, \quad d_{ij} = \varphi^{o_i o_j} - \varphi^{o'_i o'_j} \tag{8}$$

which represents the total mismatch between the orientations of the two sets. The values of D are shifted into the range $[0, 2\pi)$ and used to determine the optimal rotation angle φ^* that will be applied to S'. The mean and median values of D are both reasonable choices for φ^*, with the median providing greater stability overall [27]. Because the angles are defined on a periodic domain, it may not be possible to define a 2π range that can serve as a linear mapping to compute the median. Therefore, we pick the optimal rotation angle as the angle in D that minimizes the angular distance to all other angles in D using the following expression from [26]:

$$\varphi^* = \arg \min_{d_{uv} \in D} [q(d_{uv})] \tag{9}$$

where

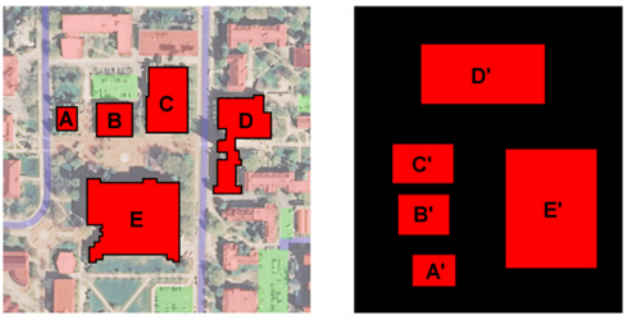
$$q(d_{uv}) = \pi - \sum_{i=1}^{n} \sum_{j=i+1}^{n} |\pi - |d_{ij} - d_{uv}||.$$

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 Fig. 5. Having found φ^*, we rotate all of the F-histograms from S' by a uniform angle to obtain an orientation-independent similarity measure

$$\Psi_{\text{Rotate}}(S, S') = \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mu_{\text{Pair}}(o_i, o_j, 0, o'_i, o'_j, \varphi^*). \tag{10}$$

F. Elastic Angles

One problem with rotating one entire object set by a uniform angle is that there is not always a single angle that best represents the ideal rotation required to match two given object sets. *Elastic angles* are a way to be more flexible with the orientation normalization. Rather than rotating all of the F-histograms from one set by the single optimal rotation angle, we rotate each F-histogram individually and apply a separate weight to each one based on how different each rotation is from the optimal angle. This gives each pair of histograms a tolerance to small directional differences. We begin by calculating the angular difference list D in the same way as before to compute the best rotation angle, φ^*. Rather than rotating all histograms of one set by this angle, we compare



(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 ≈	-83°

(c)

Fig. 5. Example of the calculation of the optimal rotation angle, φ^*. All angles are given in radians measured counterclockwise from the x-axis. (a) Ground truth object set approximated by (b) simplified sketch, which has been rotated counterclockwise about one quarter turn. (c) Object correspondences between the sketch and ground truth, and the main direction between each unique object pair, are shown. 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 between main directions D is listed in the fifth column, which is used as the input to (9) for computing the list of angular distances, q. The angle that minimizes the angular distance to all other angles in D is chosen as the optimal rotation angle φ^*. Here, φ^* is chosen as an 83° clockwise rotation of the sketch.

the normalized histograms of both sets. A normalized F-histogram is one that has been rotated clockwise by its main direction so that it becomes 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 the fuzzy weighting function $\mu_{\text{Trap}}(\theta)$ shown in Fig. 6. The angular difference between the original histograms is used as the input to this weighting function, allowing only histograms that originally shared similar orientations to be considered with full weight and all others to have less weight. The overall similarity measure is defined as

$$\Psi_{\text{Elastic}}(S, S') = \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mu_{\text{Trap}}(\varphi^* - d_{ij}) \times \mu_{\text{Pair}}(o_i, o_j, \varphi^{o_i o_j}, o'_i, o'_j, \varphi^{o'_i o'_j}). \tag{11}$$

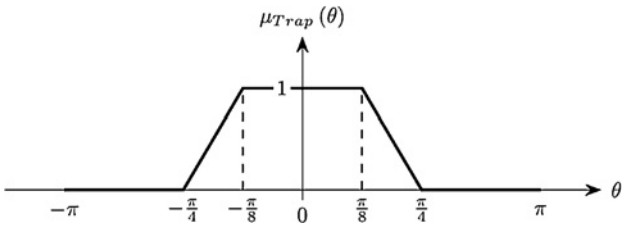


Fig. 6. Trapezoidal weighting function used for elastic angles.

The elastic angle method allows for small imperfections between two object sets. The trapezoidal membership function allows F-histograms that would not otherwise be perfectly aligned to still be considered with full weight. This tends to result in higher similarity values overall [27], but allows for the small discrepancies between object sets that tend to arise when working with real data. Fig. 7 shows an example that highlights the differences between the elastic and nonelastic methods for evaluating object set similarity.

III. Memetic Algorithm

Having defined a similarity measure between two object sets, we now turn to the task of scene matching between a target sketch and a large reference database. With both object sets represented as ARGs, this can be viewed as an approximate subgraph-matching problem, in which we seek to find a subgraph of the reference database that has a high degree of similarity to the target sketch. The optimal solution to this problem is the subgraph that has the highest similarity to the target sketch. Formally, we define the problem as follows. 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 represent a potential solution to this problem as a vector of objects from the reference set $\overrightarrow{\Gamma} = (x_{(1)}, x_{(2)}, \ldots, x_{(n)})$, $x_{(i)} \in \mathcal{R}$ for $1 \leq i \leq n$. We use this vector to define the correspondence between sketch objects and reference objects such that sketch object $o_i \in S$ is matched with reference object $x_{(i)} \in \overrightarrow{\Gamma}$, which we also notate as Γ_i. The solution to the optimization problem is the vector of reference set objects that maximizes the similarity measure, $\Psi_{\text{Elastic}}(S, \overrightarrow{\Gamma})$.

First, the input sketch and reference set are modeled as the ARG G_S and $G_{\mathcal{R}}$, respectively. For the sketch, 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 by eliminating edges between objects that are not close to each other. Bloch *et al.* [28] investigated several techniques for measuring the degree to which an object is between two other objects. Object pairs that are too far apart or have too many objects between themselves can also be excluded from the edge list. Our experiments use a reference set of 2845 objects with a maximum of 50 neighbor connectivity, shown in Fig. 8(a), which overlays $G_{\mathcal{R}}$ on the segmented image of Columbia, MO, and Fig. 8(b), which focuses on the subgraph that matches G_S, the ARG representation of the sketch given in Fig. 1.

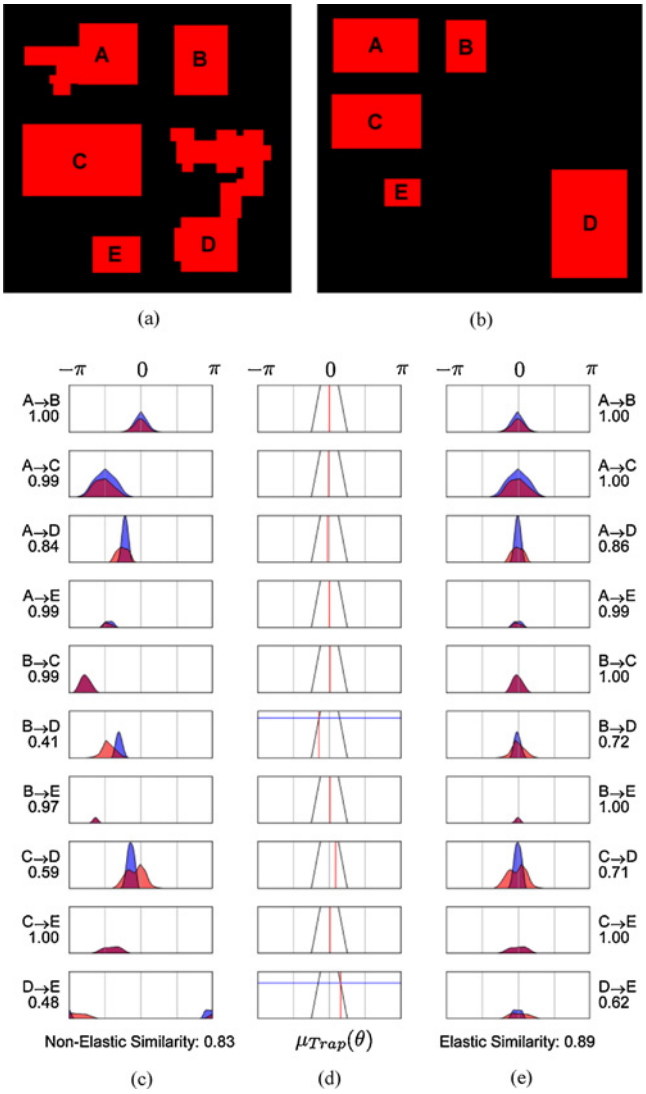
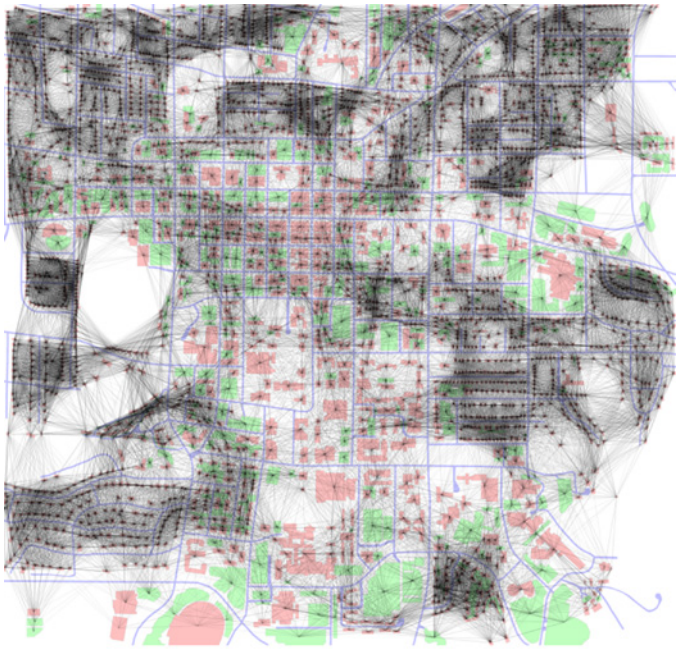
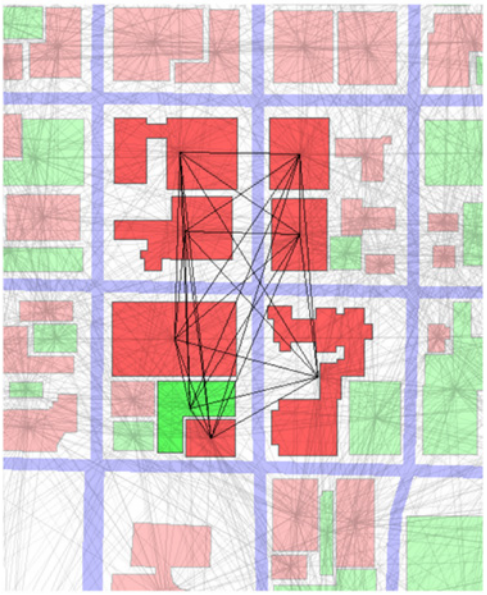


Fig. 7. Comparison of the elastic and nonelastic methods for evaluating object set similarity. (a) Subset of the example in Fig. 1. (b) Simplification of (a) with object D significantly misplaced. (c) Computation of the nonelastic similarity, where the numbers to the left of each histogram represent the individual cross-correlation values. For clarity, only the histograms of constant forces are shown, although both the constant and gravitational F-histograms are used in computing the final similarity. The red (dark gray) histograms are computed from (a) and the blue (light gray) histograms are computed from (b). (d) and (e) Computation of the elastic fitness. (d) Weighting function $\mu_{\text{Trap}}(\theta)$ and (e) normalized histograms, where the numbers to the right of each histogram represent the weighted cross-correlation values.

Removing edges from the reference ARG adds an important constraint to the optimization problem. Recall that our similarity measure Ψ_{Elastic} requires two complete ARGs. If the induced ARG formed by the potential solution $\overrightarrow{\Gamma}$ is not a complete graph, we cannot evaluate its similarity to S using this method. Additional F-histograms would need to be computed, which are typically all precomputed for large reference graphs. Although there are ways for comparing



(a)



(b)

Fig. 8. (a) $G_{\mathcal{R}}$, the ARG representation of the reference set used in our experiments containing 2845 objects from Columbia, MO. The graph is superimposed over the reference set with darker areas indicating regions of high connectivity. Buildings are shown in red (dark gray) and parking lots are shown in green (light gray). (b) Zoomed-in view of G_S, the ARG representation of the sketch from Fig. 1, shown as a subgraph of $G_{\mathcal{R}}$.

incomplete graphs using subgraph homomorphism techniques that could prove useful for matching large sketches, we limit our discussion here to the case in which any potential solution $\overrightarrow{\Gamma}$ forms a complete graph in the reference ARG. Object labels provide another constraint that must be considered in the design of our matching algorithm. In this paper, since we deal with only two labels, we require that the labels of any potential solution perfectly match the labels of the sketch such that $L(o_i) = L(\Gamma_i)$ for $1 \leq i \leq n$. An alternative approach would be to include label compatibility in the similarity measure. This could be done by using a fuzzy measure, which would provide greater flexibility when more labels are in use. However, our strict label-preserving requirement allows the search algorithm to explicitly seek out good solutions. As we shall see, these constraints provide the guiding principles for the design of the local search operators and affect how we design the memetic framework.

There are several reasons why an evolutionary method is appropriate for this problem. From the viewpoint of a scene-matching analyst, it is often desirable to be presented with a list of high-ranking solutions rather than a single top scoring match. Furthermore, the best match for an analyst may not necessarily correspond exactly with the defined similarity measure. An iterative population-based approach allows many potential solutions to compete and improve over time, giving an analyst the opportunity to provide user-driven input as to which matches are good and when to terminate the search. Although we do not use a human feedback component in our algorithm, this idea could be explored in future research. Finally, an evolutionary method is inherently parallel, allowing it to be scaled to very large problems through the use of parallel hardware. Each execution of our search algorithm is single-threaded, but this also represents a potential future research direction.

The term *memetic algorithm* was coined by Moscato [29] as a method of combining the general search strategy of an evolutionary algorithm with an individual refinement step. This allows for a population-based method with local search operators that can be designed to be problem specific. In general, an evolutionary algorithm (EA) starts by creating a population of individual solutions, or chromosomes, which cooperate and compete over multiple iterations to find good solutions to a given problem. Each chromosome encodes a solution to the problem as a set of genes, or variables, which can be set to take on certain values. The basic operators in an EA are selection, which picks individuals for breeding and survival; recombination, which combines genetic material from multiple individuals; and mutation, which is used to introduce new genetic material. A memetic algorithm adds an additional local search operator that can be applied to individuals to improve their fitness value or solution quality. The ability to incorporate problem-specific knowledge gives rise to many different adaptations of memetic algorithms to various problem domains [21]. To maximize the synergy of global exploration and local refinement, domain knowledge can be applied to the design of the problem representation, the evolutionary and local search operators, and the fitness evaluation [30].

The label-preserving and complete graph constraints described above limit our ability to use standard mutation and crossover operators for the scene-matching problem. Recall that an individual solution is represented as a vector of reference objects, $\overrightarrow{\Gamma}$. Blindly swapping the genes of two individuals through crossover or introducing a new random object through mutation would likely result in an incomplete or mislabeled graph, which would violate our constraints. As

discussed in [31], constraints can be handled either directly or indirectly in an evolutionary algorithm. Indirect methods tend to involve modifying the fitness values with a penalty function, whereas direct methods place restrictions on what types of solutions are allowed. Problem-specific operators are often created to repair infeasible individuals or prevent their creation [32]. Although graph-based crossover operators have been developed for the fields of genetic programming [33] and artificial neural networks [34], a crossover operator for this problem would need to produce children from parents that often form disjoint graphs in the reference ARG with no common nodes or edges. In order to ensure that the child chromosomes form complete graphs, they must be confined to a single region of the reference database. The children would only contain genetic material from both parents if the two parents were located very close to each other in the reference database. If the two parents were far apart, a child generated through crossover would need to be either very close to one of the parents or somewhere in the region between them. The former case is best left to the local search operator, which still maintains some degree of randomness, and the latter is essentially a new random individual. In the spirit of utilizing all possible domain knowledge, we have opted to design operators that are specific to this problem domain. Specifically, we have designed an initialization function for generating random individuals and three local search operators that take the place of crossover and mutation. The initialization function is used throughout the search process as a way of increasing exploration, whereas the local search operators are used for exploitation. Each local search operator takes a single parent and produces a set of multiple children that all share at least one common element with the parent. The details of these operators will be discussed in Section IV.

The outline of our search procedure is given in Algorithm 1. The algorithm requires as inputs a reference set $\mathcal{R}$ and a target sketch S along with predefined constants, μ (population size), λ (number of children generated each generation), τ (maximum age of each individual), and μ_{Elite} (number of top-ranking individuals preserved each generation). The search procedure starts by generating an initial population of μ random individuals. Each random individual solution is chosen by first finding the label of the target sketch that appears the least often in the reference set, and then picking a random object from the reference set with that label. This object becomes the seed of the chromosome and the remaining objects of the chromosome 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 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.

During every generation t, each of the μ individuals in the population $P^{(t)}$ generates a set of λ children through the local search operator, which are added to the current population. The population is then sorted based on individual fitness, defined by the similarity measure $\Psi_{\text{Elastic}}(S, \overrightarrow{\Gamma})$, and any duplicates are removed to prevent saturation. A few elite individuals, μ_{Elite}, are copied directly to the next generation

Algorithm 1 Memetic Algorithm for Matching Spatial Configurations

Input:
$\mathcal{R}$ and S
Constants: μ, λ, τ, μ_{Elite}
Initialize:
Set $t = 0$
Create initial population of individuals: $P^{(0)} = (\overrightarrow{\Gamma}^1, \overrightarrow{\Gamma}^2, \ldots, \overrightarrow{\Gamma}^\mu)$
While stopping criteria is not met
  For each individual $\overrightarrow{\Gamma}^P \in P^{(t)}$
    If age $(\overrightarrow{\Gamma}^P) > \tau$ and $\overrightarrow{\Gamma}^P$ is not one of the top μ_{Elite} individuals
      Replace $\overrightarrow{\Gamma}^P$ with a new random individual
  End If
  Generate list of children through local search:
    $\mathcal{C} = \text{local\_search}(\overrightarrow{\Gamma}^P)$
  Add the top λ children in $\mathcal{C}$ to $P^{(t)}$
  End For
  Sort $P^{(t)}$ and remove duplicates
  Add top μ_{Elite} individuals to $P^{(t+1)}$
  While $|P^{(t+1)}| < \mu$
    Pick $\overrightarrow{\Gamma}^i$ from $P^{(t)}$ without replacement using roulette-wheel selection
    Add $\overrightarrow{\Gamma}^i$ to $P^{(t+1)}$
  End While
  $t = t + 1$
  Increment age of each $\overrightarrow{\Gamma} \in P^{(t)}$
End While
Output: Top scoring individuals in $P^{(t)}$

$P^{(t+1)}$ and the remaining individuals are chosen using roulette-wheel selection until $|P^{(t+1)}| = \mu$. The age of each surviving solution is incremented and should an individual's age reach a certain threshold value τ, it is replaced with a new random individual. Restarting certain individuals in this way can be seen as a type of strong or heavy mutation [20] that forces individuals that have converged to local optima to move to a new location. The new individuals are given an opportunity to perform a local search before being required to compete against the existing population. This strong mutation may be omitted for the top few individuals in the population in order to retain the global best solutions found. Once the stopping criteria have been met, the search terminates and the top-ranking individuals are returned as the solutions.

The memetic algorithm we have designed loosely resembles a $(\mu + \lambda)$ evolution strategy [35]. A key difference is that we perform selection only when deciding which individuals survive to subsequent generations, and not in choosing which individuals to reproduce. This is done in order to give each chromosome an opportunity to perform a local search during each generation. In contrast to a simple parallel local search strategy, each parent produces multiple children that compete with one another for survival, ensuring that the pool of possible solutions for the next generation is larger than the set of each parent's single best offspring. We remove duplicate solutions to maintain population diversity and to reduce the amount of

redundant computation. Fitness-proportionate selection is used after preserving a few elite individuals to allow lower scoring solutions a chance to survive. If a parent solution is chosen to survive, it will undergo the local search operation multiple times, possibly producing the same set of potential children. The age counter forces these individuals to move to a new location after a set number of repeat local searches so that the algorithm continues to search new locations. This jump away from local optima is performed for all but the top few elite individuals, which are allowed to survive intact as the best global solutions. A common convergence criterion is to terminate the search if the top solution does not change for a certain number of generations.

IV. Local Search Operators

The local search operators developed for this memetic algorithm can be seen as a type of weak or light mutation. Given a parent chromosome, each operator produces a set of children that are confined within a local neighborhood of the parent. This is done by performing the core operations once for each possible choice of replacement objects or seed objects. The operators all use some random variation in producing offspring that are intended to be more similar to the target sketch than the parent. The following sections describe the three operators we have developed and provide an example of their application.

A. Single-Object Replacement

The single-object replacement (SOR) operator is based on the work presented in [27] and [36]. In this strategy, a single object from the parent is replaced by one of its nearest neighbors. Given a parent chromosome $\overrightarrow{\Gamma}^P = (\Gamma_1^P, \Gamma_2^P, \ldots, \Gamma_n^P)$, we cycle through each object $\Gamma_i^P \in \overrightarrow{\Gamma}^P$ and replace it with one of its nearest neighbors. Let $\mathcal{X}$ be the set of nearest neighbors for the object Γ_i^P, such that any object $x^* \in \mathcal{X}$ could replace Γ_i^P and still ensure that $\overrightarrow{\Gamma}^P$ forms a complete subgraph of the reference ARG. For each neighbor object x^*, we build a potential child solution $\overrightarrow{\Gamma}^C$ that is identical to $\overrightarrow{\Gamma}^P$ except that Γ_i^P has been replaced by x^*. The potential child with the highest fitness is added to the list of output children, $\mathcal{C}$.

Our initial experiments with the SOR operator [27], [36] revealed that the algorithm can often have difficulty finding the ideal match in a region. Because each chromosome is an ordered vector, an individual solution can contain all of the objects of the ideal match, but not in the right order. The SOR operator must be applied to these solutions multiple times, swapping chromosome objects with their neighbors to allow different orderings of the chromosome. We therefore consider multiple different permutations of the parent vector before applying the SOR operator. This produces a larger set of children, but decreases the number of times the SOR operator must be applied to each individual in order to reach a local optimum. 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 and maintaining a degree of randomness to help prevent premature convergence. The complete SOR method is given in Algorithm 2. Given that the complexity of each fitness evaluation is $O(n^2\omega)$, the SOR operator 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 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.

Algorithm 2 Single-Object Replacement Search Operator

Input:
Sketch: $S = (o_1, o_2, \ldots, o_n)$
Reference set: $\mathcal{R} = \{x_1, x_2, \ldots, x_m\}$
Parent: $\overrightarrow{\Gamma}^P = (\Gamma_1^P, \Gamma_2^P, \ldots, \Gamma_n^P)$ where each $\Gamma_i^P \in \mathcal{R}$
Constant: p
Initialize:
$\mathcal{C} = \phi$
Add $\overrightarrow{\Gamma}^P$ to list of permutations, $\mathcal{P}$
Add p random permutations of $\overrightarrow{\Gamma}^P$ to $\mathcal{P}$
For Each $\overrightarrow{\Gamma} \in \mathcal{P}$
  For $i = 1$ **to** n
    $\overrightarrow{\Gamma}^C = \overrightarrow{\Gamma}$
    Get the set of nearest neighbors $\mathcal{X} \subseteq \mathcal{R}$ of the object Γ_i^P
    $f_{\text{best}} = 0$
    For Each $x^* \in \mathcal{X}$
      Replace a single object: $\Gamma_i^C = x^*$
      Evaluate the fitness: $f(\overrightarrow{\Gamma}^C) = \Psi_{\text{Elastic}}(S, \overrightarrow{\Gamma}^C)$
      If $f(\overrightarrow{\Gamma}^C) > f_{\text{best}}$ **Then**
        $x_{\text{best}} = x^*$; $f_{\text{best}} = f(\overrightarrow{\Gamma}^C)$
      End If
    End For
    Replace with best object: $\Gamma_i^C = x_{\text{best}}$
    Add to list of children: $\mathcal{C} = \mathcal{C} \cup \overrightarrow{\Gamma}^C$
  End For
End For
Output: $\mathcal{C}$

B. 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 a complete sketch $S = (o_1, o_2, \ldots, o_n)$, we define a partial sketch $S' = (o'_{(1)}, o'_{(2)}, \ldots, o'_{(u)}) \subset S$ that only contains some of the original objects. Likewise, we define a partial solution as a vector $\overrightarrow{\Gamma}' = (x_{(1)}, x_{(2)}, \ldots, x_{(u)})$, $x_i \subset \mathcal{R}$ for $1 \leq i \leq u$, which associates each object of the partial sketch with an object from the reference set. The partial fitness $f(\overrightarrow{\Gamma}') = \Psi_{\text{Elastic}}(S', \overrightarrow{\Gamma}')$ only considers the specified subset of original sketch objects. The idea behind the one-seed method is to start with a single object $S' = (o'_{(1)}) \subset S$ and add objects one at a time until all of the objects in the sketch have been used. The overall outline of the one-seed set-reconstruction method is given in Algorithm 3. Given a parent chromosome $\overrightarrow{\Gamma}^P = (\Gamma_1^P, \Gamma_2^P, \ldots, \Gamma_n^P)$, we cycle through each object $\Gamma_i^P \in \overrightarrow{\Gamma}^P$ and use it as the seed object for a partial sketch. We then consider all possible assignments of an

Algorithm 3 One-Seed Set-Reconstruction Search Operator

Input:
Sketch: $S = (o_1, o_2, \ldots, o_n)$
Reference set: $\mathcal{R} = \{x_1, x_2, \ldots, x_m\}$
Parent: $\overrightarrow{\Gamma}^P = (\Gamma_1^P, \Gamma_2^P, \ldots, \Gamma_n^P)$ where each $\Gamma_i^P \in \mathcal{R}$
Initialize:
$\mathcal{C} = \phi$
Initialize list of index locations: $I = \{1, 2, \ldots, n\}$
For Each $(i, j) \in I \times I$ such that $L(o_i) = L(\Gamma_j^P)$
Clear $\overrightarrow{\Gamma}'$
Create the partial ordered sketch: $S' = (o_i)$
Update remaining index locations: $I' = I - i$
Set $\Gamma_i' = \Gamma_j^P$
Get the set of nearest neighbors $\mathcal{X} \subseteq \mathcal{R}$ of the object Γ_j^P
While $|I'| > 0$
Pick an index $k \in I'$ randomly
Add o_k to the end of the partially ordered sketch:
$S' = (\ldots, o_k)$
$f_{\text{best}} = 0$
For Each $x^* \in \mathcal{X}$
Set $\Gamma_k' = x^*$
Evaluate the partial fitness: $f(\overrightarrow{\Gamma}') = \Psi_{\text{Elastic}}(S', \overrightarrow{\Gamma}')$
If $f(\overrightarrow{\Gamma}') > f_{\text{best}}$ **Then**
$\mathcal{X}_{\text{best}} = x^*$; $f_{\text{best}} = f(\overrightarrow{\Gamma}')$
End If
End For
Set $\Gamma_k' = x_{\text{best}}$
Remove this index location: $I' = I' - k$
Update $\mathcal{X}$ **as the nearest neighbors of** $\overrightarrow{\Gamma}'$
End While
Add $\overrightarrow{\Gamma}'$ **to list of children:** $\mathcal{C} = \mathcal{C} \cup \overrightarrow{\Gamma}'$
End For
Output: $\mathcal{C}$

object from the target sketch onto the seed object, and create a partial solution $\overrightarrow{\Gamma}'$ for each one. For each partial solution, we randomly pick an unassigned sketch object $o_i \in S - S'$ and find the set of nearest neighbors $\mathcal{X} \subseteq \mathcal{R}$ to which o_i could be assigned while maintaining full connectivity. Similar to the approach of the SOR operator, we create a set of temporary partial solutions, each with o_i assigned to a different neighbor object $x^* \in \mathcal{X}$. The neighbor that produces the greatest partial fitness is added to the partial sketch S'. We continue to match the unassigned sketch objects of S to the best neighbor objects in this greedy manner until $S' = S$. Once we have a complete chromosome, we add it to the list of children, $\mathcal{C}$.

The one-seed set-reconstruction method solves many of the problems faced by the SOR operator. Individuals rarely stay in a single area without converging to a locally optimal solution. Different orderings of buildings is less of an issue since the entire solution is reconstructed. The one-seed operator also tends to converge faster than the SOR operator since more of the solution is being replaced, although this can cause individuals to become trapped in suboptimal 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 small number of permutations are used for the latter. 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 we will show in Section V.

Each partial solution is rotated to give the best alignment between the sketch and the chromosome. As the set-reconstruction methods add additional objects to the partial sketch, 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 mostly defines the initial orientation, and the remaining objects conform to this orientation.

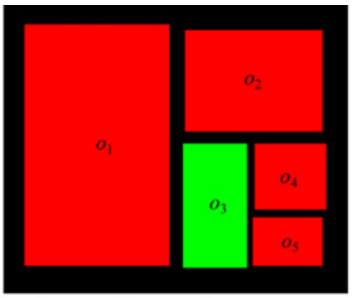
C. Two-Seed Set Reconstruction

The two-seed set-reconstruction method is almost identical to the one-seed method with the exception that two-seed objects are used 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 run of the operator to explore many different possible orientations, but incurs a significant computational overhead. For a parent solution $\overrightarrow{\Gamma}^P = (\Gamma_1^P, \Gamma_2^P, \ldots, \Gamma_n^P)$, we cycle through each pair of objects $(\Gamma_i^P, \Gamma_j^P) \in \overrightarrow{\Gamma}^P \times \overrightarrow{\Gamma}^P$ and use them as the seed objects for a partial sketch. We then consider all possible assignments of a pair of objects from the target sketch onto the seed objects, and create a partial solution, $\overrightarrow{\Gamma}'$ for each one. This results in a complexity of $O(n^7 \omega K)$, significantly greater than the other two 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 operator provides the option to check individual edges for compatibility. Although we do not make use of this property in our experiments, one could conceive of a representation that includes additional edge attributes and requires that all matched edges are compatible. 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 Algorithm 4.

D. Mutation Example

We now present an example that demonstrates each local search operator as a chromosome converges to the ideal solution. Fig. 9(a) shows an example search space containing 11 buildings shown in red (dark gray) and three parking lots shown in green (light gray), which form the reference set. The reference set ARG in this example is computed with an edge between every pair of objects. The sketch in Fig. 9(b) is a simplified representation of the five objects in the lower-left of the reference set, rotated one quarter-turn to the left. Our goal is to recover the ideal solution $\overrightarrow{\Gamma}^* = (x_7, x_{12}, x_9, x_{11}, x_{10})$ from the current population. Fig. 9(c) lists the penultimate

(a)



(b)

Penultimate Population
$\vec{\Gamma}^1 = (x_{12}, x_{10}, x_9, x_4, x_{11})$
$\vec{\Gamma}^2 = (x_2, x_5, x_1, x_4, x_7)$
$\vec{\Gamma}^3 = (x_{11}, x_6, x_3, x_7, x_8)$

(c)

SOR Mutation on $\vec{\Gamma}^1$	
Permute	$\vec{\Gamma} = (x_4, x_{12}, x_9, x_{11}, x_{10})$
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}	$\vec{\Gamma} = (x_7, x_{12}, x_9, x_{11}, x_{10})$

(d)

One-Seed Mutation on $\vec{\Gamma}^2$	
Pick the seed object	$\Gamma'_1 = x_7$
Get list of neighbor objects	$\mathcal{X} = \{x_1, \ldots, x_6, x_8, \ldots, x_{14}\}$
Pick next sketch object randomly	$o_k = o_4$
Find best match for this object	$x_{best} = x_{11}$
Update chromosome	$\Gamma'_4 = x_{11}$
Pick next sketch object randomly	$o_k = o_5$
Find best match for this object	$x_{best} = x_{10}$
Update chromosome	$\Gamma'_5 = x_{10}$
Pick next sketch object randomly	$o_k = o_3$
Find best match for this object	$x_{best} = x_9$
Update chromosome	$\Gamma'_3 = x_9$
Pick next sketch object randomly	$o_k = o_2$
Find best match for this object	$x_{best} = x_{12}$
Update chromosome	$\Gamma'_2 = x_{12}$
Return child	$\vec{\Gamma}' = (x_7, x_{12}, x_9, x_{11}, x_{10})$

(e)

Two-Seed Mutation on $\vec{\Gamma}^3$	
Pick the two seed objects	$\Gamma'_1 = x_7$ and $\Gamma'_4 = x_{11}$
Get list of neighbor objects	$\mathcal{X} = \{x_1, \ldots, x_6, x_8, \ldots, x_{10}, x_{12}, \ldots, x_{14}\}$
Pick next sketch object randomly	$o_k = o_3$
Find best match for this object	$x_{best} = x_9$
Update chromosome	$\Gamma'_3 = x_9$
Pick next sketch object randomly	$o_k = o_2$
Find best match for this object	$x_{best} = x_{12}$
Update chromosome	$\Gamma'_2 = x_{12}$
Pick next sketch object randomly	$o_k = o_5$
Find best match for this object	$x_{best} = x_{10}$
Update chromosome	$\Gamma'_5 = x_{10}$
Return child	$\vec{\Gamma}' = (x_7, x_{12}, x_9, x_{11}, x_{10})$

(f)

Fig. 9. Example of the three different local search operators on a simple matching problem. (a) Ground truth reference set. (b) Sketch for this example. Buildings are shown in red (dark gray) and parking lots are shown in green (light gray). Our goal is to recover the (c) ideal solution $\vec{\Gamma}^*(x_7, x_{12}, x_9, x_{11}, x_{10})$ from the penultimate population shown. We have chosen this population to illustrate the final search step before convergence for each of the operators. (d) The SOR operator is applied to $\vec{\Gamma}^1$ with the main steps leading to convergence listed. (e) and (f) Main steps of the one-seed and two-seed methods as they are applied to $\vec{\Gamma}^2$ and $\vec{\Gamma}^3$, respectively.

Algorithm 4 Two-Seed Set-Reconstruction Search Operator

Input:
Sketch: $S = (o_1, o_2, \ldots, o_n)$
Reference set: $\mathcal{R} = \{x_1, x_2, \ldots, x_m\}$
Parent: $\overrightarrow{\Gamma}^P = (\Gamma_1^P, \Gamma_2^P, \ldots, \Gamma_n^P)$ where each $\Gamma_i^P \in \mathcal{R}$
Initialize:
$\mathcal{C} = \phi$
Initialize list of index locations: $I = \{1, 2, \ldots, n\}$
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 $L(o_i) \neq L(\Gamma_k^P)$ **Or** $L(o_j) \neq L(\Gamma_l^P)$ **Then**
      Continue
    End If
    Clear $\overrightarrow{\Gamma}'$
    Create the partial ordered sketch: $S' = (o_i, o_j)$
    Update remaining index locations: $I' = I - \{i, j\}$
    Set $\Gamma_i^{'} = \Gamma_k^P$ and $\Gamma_j^{'} = \Gamma_l^P$
    Get the set of nearest neighbors $\mathcal{X} \subseteq \mathcal{R}$ of $\overrightarrow{\Gamma}'$
    While $|I'| > 0$
      Pick an index $u \in I'$ randomly
      Add o_u to the end of the partially ordered sketch:
        $S' = (\ldots, o_u)$
      $f_{\text{best}} = 0$
      For Each $x^* \in \mathcal{X}$
        Set $\Gamma_u^{'} = x^*$
        Evaluate the partial fitness: $f(\overrightarrow{\Gamma}') = \Psi_{\text{Elastic}}(S', \overrightarrow{\Gamma}')$
        If $f(\overrightarrow{\Gamma}') > f_{\text{best}}$ **Then**
          $x_{\text{best}} = x^*$; $f_{\text{best}} = f(\overrightarrow{\Gamma}')$
        End If
      End For
      Set $\Gamma_u' = x_{\text{best}}$
      Remove this index location: $I' = I' - u$
      Update $\mathcal{X}$ as the nearest neighbors of $\overrightarrow{\Gamma}'$
    End While
    Add $\overrightarrow{\Gamma}'$ to list of children: $\mathcal{C} = \mathcal{C} \cup \overrightarrow{\Gamma}'$
  End For
End For
Output: $\mathcal{C}$

population of chromosomes before the final local search operator. Note 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.

The SOR operator is applied to $\overrightarrow{\Gamma}^1 = (x_{12}, x_{10}, x_9, x_4, x_{11})$, with the main events that lead to convergence shown in Fig. 9(d). First, several permutations of $\overrightarrow{\Gamma}^1$ are chosen, of which $\overrightarrow{\Gamma} = (x_4, x_{12}, x_9, x_{11}, x_{10})$ is the specific permutation that 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 one of the possible offspring. The one-seed operator is applied to $\overrightarrow{\Gamma}^2 = (x_2, x_5, x_1, x_4, x_7)$ with the main events leading to convergence shown in Fig. 9(e). Each object in the parent is considered 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 operator is applied to $\overrightarrow{\Gamma}^3 = (x_{11}, x_6, x_3, x_7, x_8)$ in Fig. 9(f). Note that in this case, two sketch objects must already be mapped to the ideal reference objects in order for a complete convergence to occur in a single operation. This occurs in this example when o_1 is mapped to x_7 and o_4 is mapped to x_{11}, allowing the ideal solution to be formed by adding the remaining objects one at a time.

V. Experiments And Results

To verify the methods presented in this paper, we performed a series of experiments using a ground truth satellite image of Columbia, MO. The image was hand segmented into a reference set of 2467 buildings and 378 parking lots, shown in Fig. 10. We chose to use a hand-segmented reference database to ensure that the proposed method can work in a best-case scenario. Although the HoF are very forgiving of small shape deformities and have been used in small-scale scene matching on real imagery [9], the use of an automatically segmented reference database introduces many new degrees of uncertainty and is an area of future research. The reference ARG 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 are new for this paper and significantly reduce the overall size of the search space. The neighborhood size is dependent on the largest expected sketch size. In order to ensure that a group of objects can be returned as a solution, they must all form a complete graph in the reference set. This means that the most distant objects in the sketch must still be neighbors in the reference ARG. On the other hand, a larger neighborhood size increases the overall complexity of the algorithm. We found that for our image, 50 nearest neighbors allowed most sketches of five objects to be represented as complete graphs in the reference set.

The computational complexity of building the reference ARG depends on several factors, mainly the time to compute the nearest neighbors of each object (including the restrictions based on distance and objects in-between), and the time to compute the F-histograms. Assuming that we use the Euclidian distance between object centroids to evaluate distance, the K nearest neighbors of each of m reference objects can be computed with complexity $O(Km \log m)$ [37]. Computation of the in-between relationship and F-histograms assumes that objects are represented in raster format; our reference image has dimensions of 4667×4467 pixels. For any pair of objects, we denote N as the number of pixels in the bounding box that spans both objects, and N_1 and N_2 as the number of pixels belonging to the first and second objects, respectively. The in-between relationship is computed using fuzzy visibilities, which has a complexity of $O(NN_1N_2)$ [28]. The maximum complexity of the F-histogram computation is given as $O(N\sqrt{N})$ [8]. The F-histograms in our experiments were calculated using a two-degree interval, which we believe provides enough information to distinguish most spatial configurations. All of the calculations required to build the

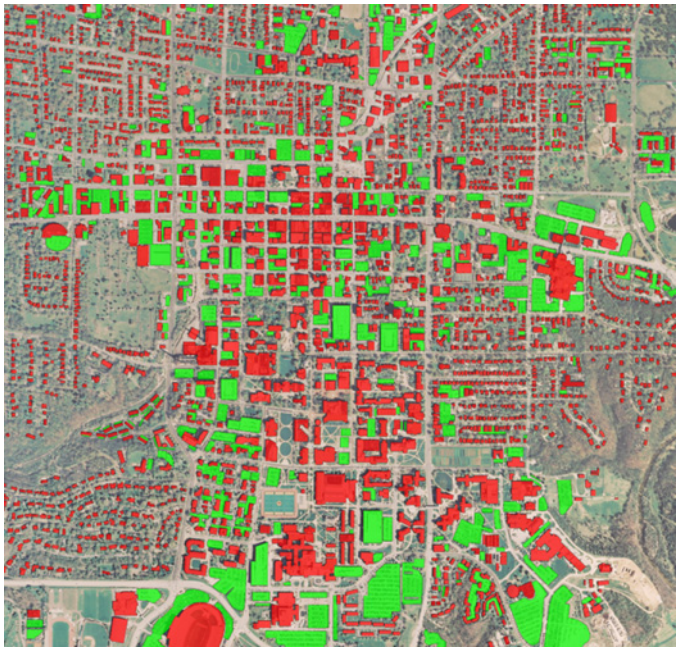


Fig. 10. Reference set $\mathcal{R}$ used for our experiments. The set contains 2467 buildings shown in red (dark gray) and 378 parking lots shown in green (light gray) from downtown Columbia, MO, and the University of Missouri campus.

reference ARG were performed *a priori*, allowing the same reference ARG to be used for all of the experiments.

Two sets of example sketches were created representing exact resubstitution and simplified search cases. For both types, we created 100 sketches of random locations from the satellite image, each containing five objects. First, a random object is selected from the reference set as the seed object, which can be either a building or a parking lot. The remaining objects are chosen from the neighbors of the seed object in such a way that closer objects are more likely to be chosen, and object labels have no effect. Each sketch set is guaranteed to be a complete subgraph of $G_{\mathcal{R}}$ and the building to parking lot ratio of each set is determined by the overall distribution in the reference database. Because there are over seven times as many buildings as parking lots in our reference set, most of our example sketches have only one or two parking lots, if any at all. The resubstitution sketches contain objects exactly as they appear in the reference database, whereas the simplified sketches reduce each object to its bounding box and then rotate the entire sketch by a random angle. The sketch ARG is always a complete graph, so the computational complexity to construct G_S for each example sketch can be given as $O(n^2 N\sqrt{N})$, where n is the number of sketch objects and N is the number of pixels in the sketch. Like the reference ARG, the sketch ARGs are computed *a priori* to allow for efficient testing of multiple search configurations.

Our goal for each search is to recover the original sketch location using the proposed memetic algorithm. We record whether the original sketch location is found in the top 1, 5, 10, and 50 results. This is an objective evaluation for a problem that often requires a subjective interpretation of the results. Often, many locations can match a given input sketch, causing the true original location to appear lower in the ranked search results. Because the resubstitution sketches come directly from the reference database, the original locations of these sketches, if found, will always be the highest scoring result with maximum fitness. These experiments serve to verify our method, showing that we can find the original sketch location when there is no ambiguity in the spatial configuration. The simplified sketches, however, may not match the original sketch location as well as some other locations, lowering the ranking of the original sketch location in the results. These experiments with simplified sketches are designed to show that our method can handle arbitrary sketch rotations and many of the ambiguities that appear in actual human queries or in the automatically generated output of the T_2S system.

In an analytical environment, the scene-matching results may indicate regions that should be targeted for additional analysis. The best result according to an analyst may not correspond to the top match, as our simplified sketch experiments try to show. For this reason, it is important to always provide a list of top-scoring results that a human analyst can choose from. Ideally, these solutions will all be from different regions of the search space. This can be accomplished through the use of diversity mechanisms such as crowding, niching, and fitness sharing. These approaches ensure that the population remains diverse enough to continually explore new areas and allow the top-ranked results to contain less fit solutions that might otherwise be missed. We use a basic strategy in our experiments that explicitly prohibits the existence of duplicate solutions in the population, although this can still produce very similar solutions. The application of additional diversity mechanisms to our algorithm is an area of ongoing future research.

We evaluated a total of 11 different search configurations, detailed in Table I. All of the test configurations were run ten times on each of the 100 sketches of the appropriate sketch type. The full list of common algorithm parameters is given in Table II. These parameters were chosen through initial experimentation to balance algorithm complexity and search time. We use a population of 50 chromosomes, which is large enough to have an individual in each neighborhood of the search area, but not so large that the search time becomes unreasonable. As was shown in [36], a larger population size leads to a greater percentage of the original locations being recovered, but results in a longer computation time. Our elite size is set to five individuals, which means that we always retain the top five solutions during each search. This value could be adjusted depending on how many results we are interested in observing at the end of the search. We allow a maximum of six children to be created for each parent, which is a value suggested in [20], although this could likely be decreased since the top children are always returned for each parent. The maximum chromosome age is set to 10 generations, which gives most individuals enough time to find a local optimum, but does not spend too much time repeating searches in the same area. Finally, five permutations were chosen for the SOR operator to give a small number of alternate chromosome orderings without adding too much computational burden.

The test configurations vary the sketch type, local search operator, and convergence criteria. The experiments with re-

TABLE I
TEST CONFIGURATIONS AND RESULTS

Test Number	Sketch Type	Local Search Operator	Min Stall Generations	Max Generations	% Found in Top 1 (%)	% Found in Top 5 (%)	% Found in Top 10 (%)	% Found in Top 50 (%)	Average Evaluations	Average Time (s)
1	Resubstitution	SOR	1000	1000	62.0	62.0	62.0	62.0	$1.54 \times 10^8 \pm 3.89 \times 10^7$	3367 ± 849
2	Resubstitution	1-Seed	1000	1000	99.8	99.8	99.8	99.8	$2.18 \times 10^8 \pm 9.01 \times 10^7$	7814 ± 3196
3	Resubstitution	2-Seed	100	100	88.2	88.2	88.2	88.2	$1.24 \times 10^8 \pm 6.90 \times 10^7$	4703 ± 2660
4	Simplified	SOR	10	1000	4.6	4.9	4.9	5.4	$3.34 \times 10^6 \pm 1.40 \times 10^6$	72 ± 31
5	Simplified	SOR	100	1000	21.4	23.0	23.0	23.0	$2.57 \times 10^7 \pm 1.15 \times 10^7$	556 ± 249
6	Simplified	SOR	1000	1000	42.4	48.3	48.5	48.5	$1.54 \times 10^8 \pm 3.64 \times 10^7$	3423 ± 857
7	Simplified	1-Seed	10	1000	39.7	44.6	44.8	44.8	$3.52 \times 10^6 \pm 1.83 \times 10^6$	125 ± 64
8	Simplified	1-Seed	100	1000	59.1	69.0	69.0	69.0	$2.84 \times 10^7 \pm 1.44 \times 10^7$	1008 ± 508
9	Simplified	1-Seed	1000	1000	68.9	80.9	80.9	80.9	$2.22 \times 10^8 \pm 8.76 \times 10^7$	7896 ± 3239
10	Simplified	2-Seed	10	100	33.0	38.6	38.6	38.7	$2.08 \times 10^7 \pm 1.37 \times 10^7$	793 ± 522
11	Simplified	2-Seed	100	100	59.9	72.8	73.0	73.0	$1.25 \times 10^8 \pm 6.74 \times 10^7$	4883 ± 2651

TABLE II
COMMON ALGORITHM PARAMETERS

Population size	μ	50
Max children per parent	λ	6
Elite size	μ_{Elite}	5
Max chromosome age	τ	10
Number of sketch objects	n	5
Nearest neighbor connectivity	K	50
SOR permutations	p	5

substitution sketches are intended to verify our search method in an environment where the target sketch is the global best solution. We are interested in knowing whether or not the algorithm will ever converge to this result, so we run all of the resubstitution experiments for the maximum number of generations. For the SOR and one-seed operators this is set to 1000 generations, but due to the added complexity, the two-seed operator is set to a maximum of 100 generations. It should be noted that we could have used a target fitness threshold of 1.0 to stop searching since all of the resubstitution sketches are perfect matches; however, to avoid potential rounding errors and to maintain consistency with the simplified experiments, we opted instead to use the generation limit. The simplified experiments vary the local search operator and the stop condition, set as the minimum number of stall generations in which the top solution does not change. When this is set equal to the maximum number of generations, the convergence criteria is the same as the resubstitution experiments.

The experiments were run on three computers running 64-bit versions of Windows 7, each with eight logical processors clocked at 2.8–2.93 GHz and with 12 GB of RAM. To evaluate the computational complexity of each method, we measure the runtime of each search and the number of times the cross-correlation operator is called. Note that this does not include the time required to compute the reference and sketch ARGs, which are all computed *a priori*. We then evaluate the percentage of tests that found the original sketch location ranked in the top 1, 5, 10, and 50 results. We see from Table I that given the maximum amount of search time for the resubstitution experiments, the one-seed operator found the original sketch location nearly every time. The two-seed and SOR methods found the original sketch location less often, but also did not spend as much time searching. The two-seed method, however, only evaluated 100 generations as opposed to 1000 generations for the other two methods. Had the two-seed experiments been allowed to run for 1000 generations, the search time would likely be ten times longer. Because the two-seed operator considers all possible edge assignments rather than the single object assignments of the one-seed, the search time increases exponentially.

The results of the experiments with simplified sketches show that if the original sketch location is found, it is usually within the top five results. Overall, we see that using a greater number of stall generations as our stop condition results in a higher recall rate, but longer search times. The two-seed method takes the most time, followed by the one-seed method, and finally the SOR method finishes in the shortest time. The one-seed and two-seed methods, however, found the original sketch location more often than the SOR method. This is likely due to the greater amount of change that occurs with each local search operation, allowing newly generated offspring to quickly compete with the existing population. The recall rate of the one-seed method is comparable to that of the two-seed method given the same convergence criteria, but the one-seed method takes far less time. This implies that the added complexity of the two-seed method to handle edge relationships is unjustified, as the flexibility of the elastic angle similarity measure allows the one-seed method to handle simplified sketches with arbitrary orientations.

The total number of evaluations and runtime statistics for each of the experiments is shown in Fig. 11. An interesting observation seen in these results is that values tend to be skewed toward the high ranges of each test. This implies that a small portion of the tests performed much better than average. We propose that this may be due to the presence of parking lots in some of the sketches, which would place additional constraints on the search. To investigate this possibility, we look at test number 2, which had a near perfect recall rate, and separate the experimental results based on how many parking lots appear in the test sketches. This is shown in Table III, which shows that 61 out of 100 test sketches contained no parking lots at all, followed by 23 sketches with a single parking lot, and 16 sketches with two or more parking lots. These sketches were randomly sampled from the reference database and show that parking lots are relatively uncommon compared to buildings for our data. Because parking lots are less common in our reference set, sketches that contain

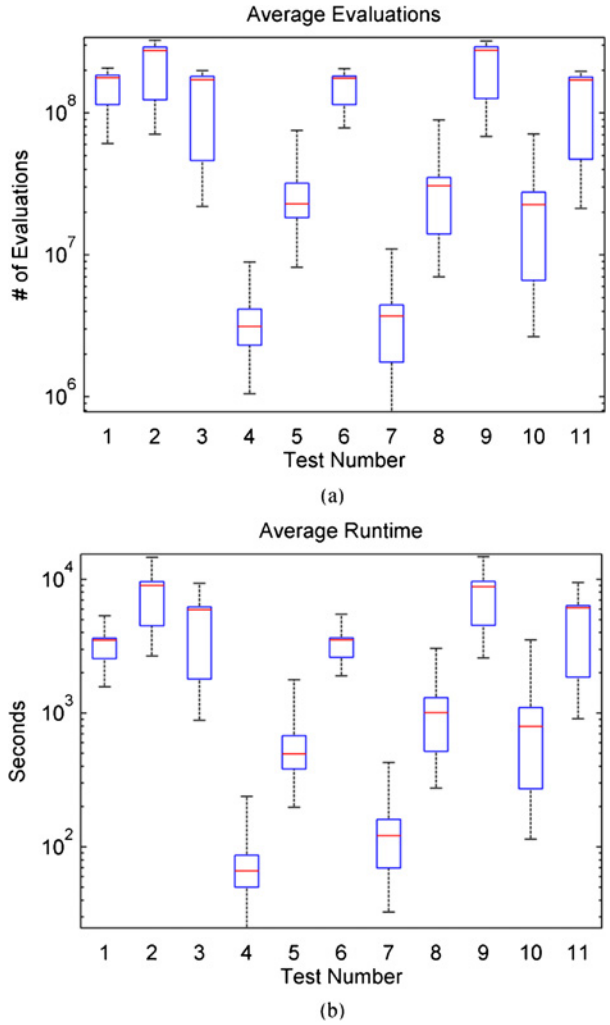


Fig. 11. (a) Average number of cross-correlation evaluations required for each of the experimental configurations. (b) Average runtime for each of the experimental configurations. Box plots give the minimum, maximum, upper and lower quartile, and the median values from all tests.

TABLE III
EFFECT OF PARKING LOTS ON SEARCH TIME USING TEST #2

Number of Parking Lots	Test Sketches (Out of 100)	Average Evaluations	Average Time (s)
0	61	$2.88 \times 10^8 \pm 1.35 \times 10^7$	$10\,102 \pm 1609$
1	23	$1.26 \times 10^8 \pm 5.36 \times 10^6$	4923 ± 758
2	12	$8.22 \times 10^7 \pm 2.48 \times 10^6$	3242 ± 475
3	3	$7.37 \times 10^7 \pm 2.26 \times 10^6$	3025 ± 480
4	1	$8.18 \times 10^7 \pm 6.87 \times 10^5$	3969 ± 613

more parking lots take less time to finish searching due to the additional constraints. This demonstrates how important additional labels can be in reducing the overall size of the search space.

In Fig. 12, we show several examples of the scene-matching experiments using the one-seed local search operator run for the maximum number of generations. The first two examples use resubstitution sketches that are exact copies of the ground truth location. In both of these, the top result is the original sketch location and all of the results share a similar spatial configuration. In the first example, all of the top matches could match the sketch via a rotation and the spacing between the buildings is well preserved. The top results of the second example show only two different locations, but with a slightly different set of buildings in each one. In the alternate location, the building that is completely surrounded by a parking lot in the sketch is surrounded by a parking lot on three sides. In the last two examples, the sketch is a simplified and rotated version of the ground truth. The third example contains all rectangular objects, so the simplification process is just a rotation. In this case, the original sketch location is recovered as the top match. However, the sketch in the last example is a significantly simplified and rotated version of the ground truth location. Here, the original sketch location is the third best match because there are other locations that, upon evaluation, have a higher similarity with the simplified sketch.

VI. CONCLUSION AND FUTURE WORK

The HoF provide a useful framework for representing the relative position between a pair of objects using directional relationships. In this paper, we presented a method of representing the spatial configuration of a group of objects using an ARG composed of object labels and HoF relations. We developed a similarity measure to compare two sets of object configurations based on the concept of elastic angles, which seeks to normalize the orientation differences between two sketches using the main direction between pairs of objects. Finally, we defined three local search operators that can be used in a memetic framework to perform the task of scene matching between a target sketch and a reference database. Of these three local search operators, the one-seed set-reconstruction method gave the best tradeoff between complexity and recall rate.

Being a population-based method, the evolutionary algorithm we developed can scale relatively easily to larger problems. The overall complexity of each generation of the algorithm can be given as $O(\mu O_{\mathrm{LSO}})$, where O_{LSO} is the complexity of the chosen local search operator. The overall complexity is not dependent on the size of the reference set, but instead depends mainly on the population size, sketch size, and number of nearest neighbor connections in the reference set. This means that with adequate hardware, the reference set could be several orders of magnitude larger than our experimental database. Multiple searches could also be run in parallel, with the final results aggregated at the end. This could be used to search a larger geospatial area or a large collection of separate images.

Our proposed use of the memetic spatial matching algorithm is to take sketches created by the T_2S system and anchor them to real world ground truth locations. A complete T_2S system would be a useful tool for geospatial data analysts, as it could provide several possible locations for a person based on a linguistic description of their surroundings. Obviously, this presents an opportunity for additional ambiguity in the scene description and sketch construction. Along with imperfections in automatically segmented satellite imagery, the complete T_2S pipeline presented several sources of uncertainty, leading to

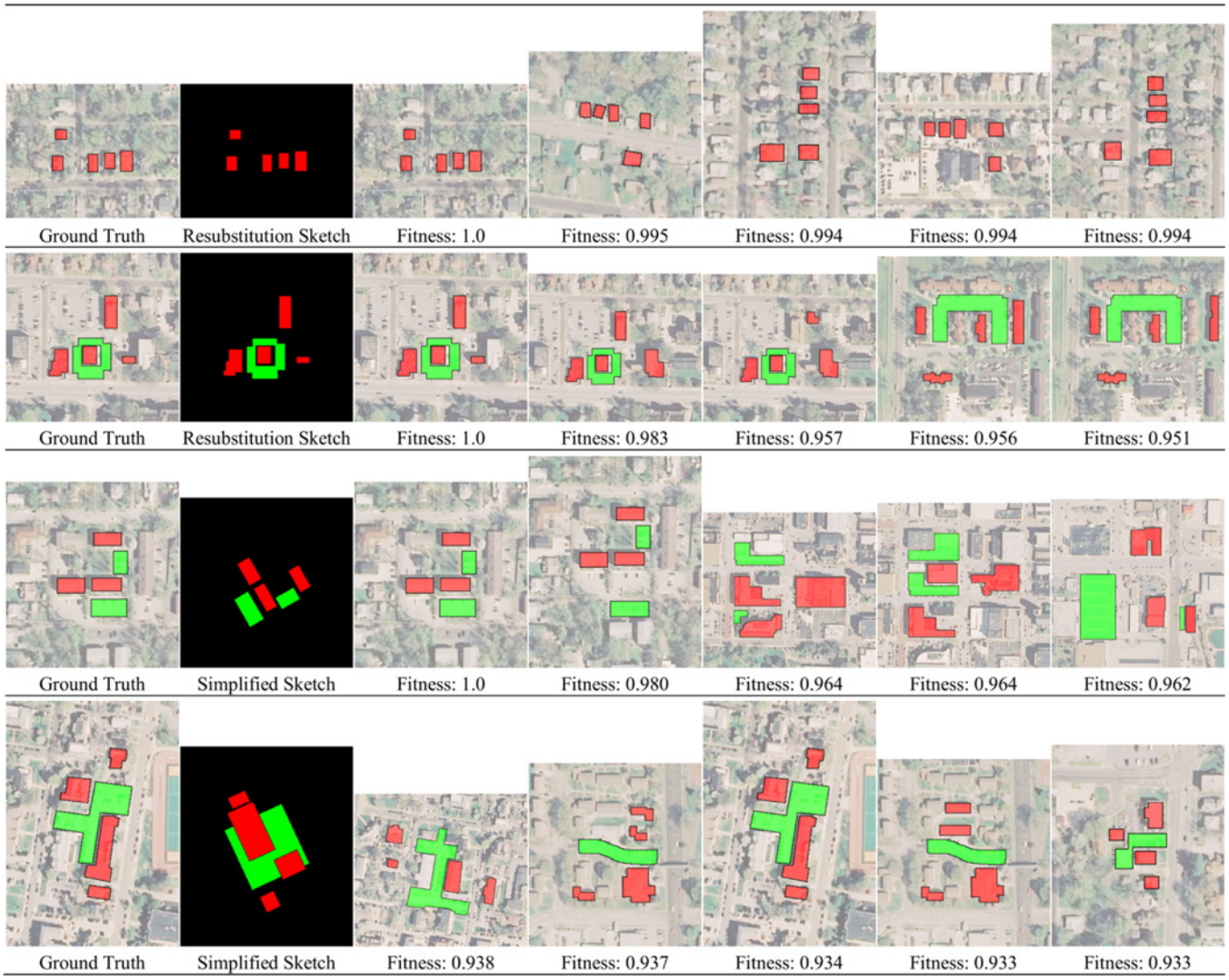


Fig. 12. Examples of the top matches found for the scene-matching experiments using the one-seed local search operator run for the maximum number of generations. Buildings are shown in red (dark gray) and parking lots are shown in green (light gray). In the top two examples with resubstitution sketches and the third example with a simplified sketch, the top match is the correct mapping to the ground truth set. In the last example, the ground truth set is the third result because there are other locations that better match the simplified sketch. All of the top matches have very high fitness values and share similar spatial configurations.

our use of fuzzy-based methods such as the HoF. Alternate applications, such as automated printed circuit analysis, may also find this scene-matching method useful. The memetic framework presented in this paper can be applied to those problems, in which a specific spatial configuration is to be found from within a larger database of objects.

There are several directions for future work in this area. Some diversity mechanisms could be integrated into our existing algorithm relatively easily, and alternate algorithms for approximate subgraph matching could provide additional search strategies. Uncertainty is already modeled to some degree through the use of force histograms, and our model could be extended to include incomplete information in the target sketch. Error-correcting subgraph homomorphisms offer one possible way to make use of incomplete sketches, which may contain unlabeled or missing objects. Additional attributes on the nodes and edges of the ARG have been shown to improve search performance by pruning the search space, and features such as road networks and additional object labels could be added to further improve the matching accuracy of the algorithm.

REFERENCES

[1] O. Sjahputera and J. M. Keller, "Scene matching using f-histogram-based features with possibilistic C-means optimization," *Fuzzy Sets Syst.*, vol. 158, no. 3, pp. 253–269, 2007.

[2] M. J. Egenhofer, "Query processing in spatial-query-by-sketch," *J. Visual Languages Comput.*, vol. 8, no. 4, pp. 403–424, Aug. 1997.

[3] I. J. Sledge and J. M. Keller, "Mapping natural language to imagery: Placing objects intelligently," in *Proc. IEEE Int. Conf. Fuzzy Syst.*, Aug. 2009, pp. 518–523.

[4] A. G. Cohn and S. M. Hazarika, "Qualitative spatial representation and reasoning: An overview," *Fundam. Inform.*, vol. 46, nos. 1–2, pp. 1–29, Jan. 2001.

[5] F. Godoy and A. Rodríguez, "A quantitative description of spatial configurations," in *Proc. Spatial Data Handling*, Jul. 2002, pp. 299–311.

[6] V. N. Gudivada and V. V. Raghavan, "Design and evaluation of algorithms for image retrieval by spatial similarity," *ACM Trans. Inform. Syst.*, vol. 13, no. 2, pp. 115–144, Apr. 1995.

[7] H. T. Bruns and M. Egenhofer, "Similarity of spatial scenes," in *Proc. 7th Int. Symp. Spatial Data Handling*, no. 4A. Aug. 1996, pp. 31–42.

[8] P. Matsakis and L. Wendling, "A new way to represent the relative position between areal objects," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 7, pp. 634–643, Jul. 1999.
[9] P. Matsakis, J. M. Keller, O. Sjahputera, and J. Marjamaa, "The use of force histograms for affine-invariant relative position description," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 1, pp. 1–18, Jan. 2004.
[10] O. Sjahputera and J. M. Keller, "Possibilistic c-means in scene matching," in *Proc. 4th Int. Conf. Eur. Soc. Fuzzy Logic Technol.*, Sep. 2005, pp. 669–675.
[11] B. S. Reddy and B. N. Chatterji, "An FFT-based technique for translation, rotation, and scale-invariant image registration," *IEEE Trans. Image Process.*, vol. 5, no. 8, pp. 1266–1271, Aug. 1996.
[12] W.-H. Tsai and K.-S. Fu, "Error-correcting isomorphisms of attributed relational graphs for pattern analysis," *IEEE Trans. Syst., Man Cybern.*, vol. 9, no. 12, pp. 757–768, Dec. 1979.
[13] M. A. Eshera and K.-S. Fu, "An image understanding system using attributed symbolic representation and inexact graph-matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 5, pp. 604–618, Sep. 1986.
[14] S. Berretti, A. Del Bimbo, and E. Vicario, "The computational aspect of retrieval by spatial arrangement," in *Proc. 15th Int. Conf. Pattern Recognit.*, vol. 1. Sep. 2000, pp. 1047–1051.
[15] K. A. Nedas and M. J. Egenhofer, "Spatial-scene similarity queries," *Trans. GIS*, vol. 12, no. 6, pp. 661–681, 2008.
[16] V. Kumar, "Algorithms for constraint satisfaction problems: A survey," *AI Mag.*, vol. 13, no. 1, pp. 32–44, 1992.
[17] Y. Jin, S. Member, and J. Branke, "Evolutionary optimization in uncertain environments: A survey," *IEEE Trans. Evol. Comput.*, vol. 9, no. 3, pp. 303–317, Jun. 2005.
[18] M. A. Rodríguez and M. C. Jarur, "A genetic algorithm for searching spatial configurations," *IEEE Trans. Evol. Comput.*, vol. 9, no. 3, pp. 252–270, Jun. 2005.
[19] D. Papadias, M. Mantzourogiannis, and I. Ahmad, "Fast retrieval of similar configurations," *IEEE Trans. Multimedia*, vol. 5, no. 2, pp. 210–222, Jun. 2003.
[20] P. Moscato and C. Cotta, "A gentle introduction to memetic algorithms," in *Handbook of Metaheuristics*. Secaucus, NJ: Kluwer Academic, 2003, pp. 105–144.
[21] N. Krasnogor and J. Smith, "A tutorial for competent memetic algorithms: Model, taxonomy, and design issues," *IEEE Trans. Evol. Comput.*, vol. 9, no. 5, pp. 474–488, Oct. 2005.
[22] A. Singh and A. K. Gupta, "A hybrid heuristic for the maximum clique problem," *J. Heuristics*, vol. 12, nos. 1–2, pp. 5–22, Mar. 2006.
[23] K. Tang, Y. Mei, and X. Yao, "Memetic algorithm with extended neighborhood search for capacitated arc routing problems," *IEEE Trans. Evolu. Comput.*, vol. 13, no. 5, pp. 1151–1166, Oct. 2009.
[24] P. Matsakis, J. M. Keller, L. Wendling, J. Marjamaa, and O. Sjahputera, "Linguistic description of relative positions in images," *IEEE Trans. Syst., Man, Cybern. B Cybern.*, vol. 31, no. 4, pp. 573–588, Aug. 2001.
[25] M. Skubic, S. Blisard, C. Bailey, J. A. Adams, and P. Matsakis, "Qualitative analysis of sketched route maps: Translating a sketch into linguistic descriptions," *IEEE Trans. Syst., Man, Cybern. B Cybern.*, vol. 34, no. 2, pp. 1275–1282, Apr. 2004.
[26] N. I. Fisher, *Statistical Analysis of Circular Data*. New York: Cambridge Univ. Press, 1993.
[27] A. R. Buck, J. M. Keller, M. Skubic, M. Detyniecki, and T. Baerecke, "Object set matching with an evolutionary algorithm," in *Proc. IEEE Symp. Comput. Intell. Security Defense Appl.*, Apr. 2011, pp. 43–50.
[28] I. Bloch, O. Colliot, and R. M. Cesar, Jr., "On the ternary spatial relation 'between'," *IEEE Trans. Syst., Man, Cybern. B Cybern.*, vol. 36, no. 2, pp. 312–327, Apr. 2006.
[29] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: Toward memetic algorithms," Caltech Concurrent Computation Program, California Inst. Technol., Pasadena, CA, Tech. Rep. 826, 1989.
[30] X. Chen, Y.-S. Ong, M.-H. Lim, and K. C. Tan, "A multi-facet survey on memetic computation," *IEEE Trans. Evol. Comput.*, vol. 15, no. 5, pp. 591–607, Oct. 2011.
[31] B. G. W. Craenen, A. E. Eiben, and E. Marchiori, "How to handle constraints with evolutionary algorithms," in *The Practical Handbook of Genetic Algorithms Applications*, L. Chambers, Ed. London, U.K.: Chapman and Hall, 2001, pp. 341–362.
[32] M. C. Riff-Rojas, "Evolutionary search guided by the constraint network to solve CSP," in *Proc. IEEE Int. Conf. Evol. Comput.*, Apr. 1997, pp. 337–342.
[33] R. Poli, W. B. Langdon, and N. F. Mcphee, *A Field Guide to Genetic Programming*. Published via http://lulu.com and freely available at http://www.gp-field-guide.org.uk, 2008.
[34] X. Yao, "Evolving artificial neural networks," *Proc. IEEE*, vol. 87, no. 9, pp. 1423–1447, 1999.
[35] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies: A comprehensive introduction," *Natural Comput.*, vol. 1, no. 1, pp. 3–52, Mar. 2002.
[36] A. R. Buck, J. M. Keller, and M. Skubic, "A modified genetic algorithm for matching building sets with the histograms of forces," in *Proc. IEEE Congr. Evol. Comput.*, Jul. 2010, pp. 1–7.
[37] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, "An optimal algorithm for approximate nearest neighbor searching in fixed dimensions," *J. ACM*, vol. 45, no. 6, pp. 891–923, Nov. 1998.

Andrew R. Buck (S'11) received the B.Sc. degree in both electrical and computer engineering in 2009, and the M.Sc. degree in computer engineering in 2012, both from the University of Missouri, Columbia, where he is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering.

His current research interests include fuzzy systems, evolutionary algorithms, spatial relationships, and computer vision.

James M. Keller (F'00) received the Ph.D. degree in mathematics from the University of Missouri, Columbia, in 1978.

He holds the Curators' Professorship with the Electrical and Computer Engineering and Computer Science Departments, University of Missouri, Columbia. He is also the R. L. Tatum Professor with the College of Engineering. His industrial and government funding sources include Electronics and Space Corporation, Union Electric, Geo-Centers, National Science Foundation, the Administration on Aging, The National Institutes of Health, NASA/JSC, the Air Force Office of Scientific Research, the Army Research Office, the Office of Naval Research, the National Geospatial Intelligence Agency, the Leonard Wood Institute, and the Army Night Vision and Electronic Sensors Directorate. He has co-authored more than 400 technical publications. His current research interests include computational intelligence, fuzzy set theory and fuzzy logic, neural networks, and evolutionary computation with a focus on problems in computer vision, pattern recognition, and information fusion, including bioinformatics, spatial reasoning in robotics, geospatial intelligence, sensor and information analysis in technology for eldercare, and landmine detection.

Dr. Keller is a Fellow of the International Fuzzy Systems Association (IFSA), and a Past President of the North American Fuzzy Information Processing Society (NAFIPS). He was a recipient of the 2007 Fuzzy Systems Pioneer Award and the 2010 Meritorious Service Award from the IEEE Computational Intelligence Society. He finished a full six-year term as the Editor-in-Chief of the IEEE TRANSACTIONS ON FUZZY SYSTEMS, followed by being the Vice President for Publications of the IEEE Computational Intelligence Society from 2005 to 2008, and since then an elected CIS Adcom Member. He is the IEEE Technical Activities Board Transactions Chair and a member of the IEEE Publication Review and Advisory Committee. Among many conference duties over the years, he was the General Chair of the 1991 NAFIPS Workshop and the 2003 IEEE International Conference on Fuzzy Systems.

Marjorie Skubic (M'97) received the Ph.D. degree in computer science from Texas A&M University, College Station, in 1997, where she specialized in distributed telerobotics and robot programming by demonstration.

She is currently a Professor with the Department of Electrical and Computer Engineering, University of Missouri, Columbia, with a joint appointment in computer science. In addition to her academic experience, she has spent 14 years within the industry working on real-time applications such as data acquisition and automation. In 2006, he established the Center for Eldercare and Rehabilitation Technology, University of Missouri, and serves as the Center Director for this interdisciplinary team. Her current research interests include sensory perception, computational intelligence, spatial referencing interfaces, human–robot interaction, and sensor networks for eldercare.