A Graph-Based Memetic Approach to Sketch Geolocation

Andrew R. Buck, *Student Member, IEEE,* and James M. Keller, *Fellow, IEEE*
Department of Electrical and Computer Engineering
University of Missouri
Columbia, Missouri, USA

***Abstract*—This paper presents a memetic algorithm for the task of sketch geolocation. Small sets of 2D objects having real-world origin are depicted as sketches, which capture their spatial configuration. These are matched to a much larger geospatial reference scene using a memetic algorithm, which combines both global and local search strategies. Sets are represented as attributed relational graphs in which objects are graph nodes and the spatial relationships between objects, defined by the histograms of forces, are graph edges. We define a similarity measure between two such graphs and describe two local search operators. The first is a greedy operator based on our previous work and the second is a new approach based on the VF2 subgraph isomorphism algorithm. Our experiments show that both methods can be successfully applied to this domain.**

***Keywords*—attributed relational graphs; histograms of forces; memetic algorithm; scene matching; Text-to-Sketch**

I. Introduction

Scene matching is the process of finding a correspondence between the objects of two scenes. For geographic information systems (GIS), this is a common task which, for example, may be used to identify multiple views of the same scene [1] or to perform a *query by sketch* [2]. In this paper we focus on the second task, in which a small target sketch representing a specific spatial configuration of 2D objects must be found within a much larger geospatial reference database.

This work 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. Given a set of one or more spatial descriptions, the Text-to-Sketch (T_2S) system constructs an approximate sketch of object locations [3]. 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 entities used in the descriptions are drawn by the T_2S system as objects in an image. The objects themselves are only rough estimates of the actual ground truth, so most of the scene information is stored as relative spatial information.

We use the histograms of forces (HoF) [4] to capture the relative position between two objects. The HoF have been shown to be affine-invariant [5], allowing for the development of a robust similarity measure that can handle arbitrary changes in rotation, scale, and translation. The HoF have also been shown to be tolerant of shape variations, which is important for matching approximate sketches. We use an *attributed relational graph* (ARG) representation [6], [7] to describe the spatial configuration of a set of objects, which stores object attributes as graph nodes and the spatial relationship between objects as graph edges. Scene matching can then be viewed as an approximate subgraph matching problem in which we seek to find a subgraph of the reference ARG that has maximum similarity to the target sketch ARG.

This is an extension of our previous work in [8], where we developed a memetic algorithm to perform this task. An evolutionary framework was used in conjunction with several local search operators, including a single object replacement strategy [9], [10], and two set reconstruction operators. In this paper, we introduce a new local search operator based on the VF2 subgraph isomorphism algorithm [11], which utilizes the relational structure of this problem. The top performing one-seed set reconstruction operator from our previous work is compared to the VF2 operator to identify the strengths and weaknesses of each one.

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

II. Graphs of Spatial Relationships

In this section, we define a graph representation of the spatial content of a scene using the histograms of forces. This representation encodes each scene object as graph node and uses the graph edges to represent the spatial relationship between objects. These graphs are called attributed relational graphs, and we define a similarity measure between two such graphs.

A. Histograms of Forces

The histograms of forces [4], [5], provide a measure of the spatial relationship between a pair of 2D objects. For every direction θ, we calculate the sum of elementary forces acting between objects A and B and aggregate these forces into the force histogram $F_r^{AB}(\theta)$, shown in Fig. 1. This function maps $\mathbb{R} \rightarrow \mathbb{R}^+$ and represents the degree of support for the statement "A is in direction θ from B." The magnitude of the

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

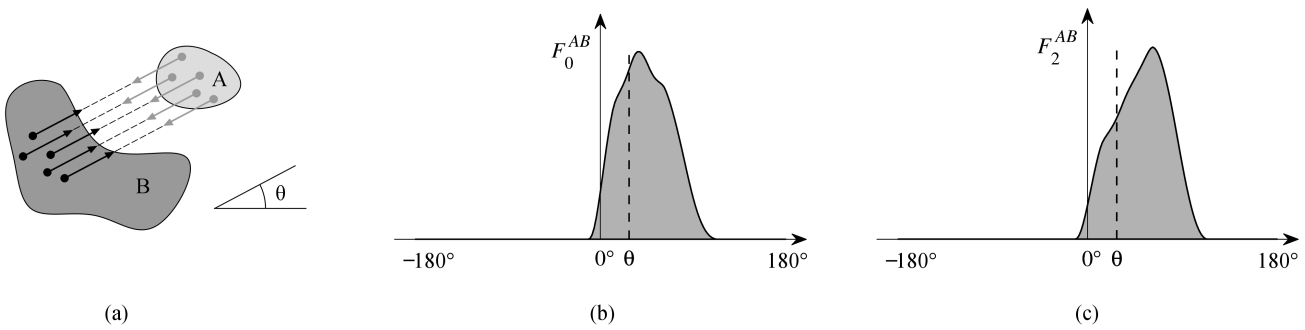

Fig. 1. (a) A force histogram F_r^{AB} is the scalar resultant of elementary forces exerted by the points of A on those of B. Each one pulls B in direction θ. (b) The histogram of constant forces ($r = 0$) is one representation of the spatial relationship between A and B providing a global perspective. (c) The histogram of gravitational forces ($r = 2$) is another possible representation, which is more sensitive to nearby points.

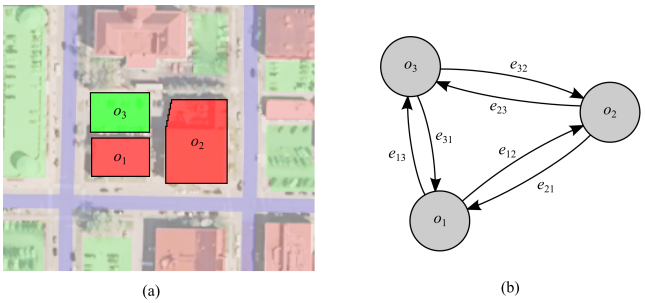

Fig. 2. An example scene (a) and its corresponding ARG (b).

forces is calculated as an inverse ratio of d^r, where d is the distance between objects 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 objects. 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.

We represent a force histogram (F-histogram) as a vector of 180 real numbers indicating the magnitude of the histogram for every direction θ, discretized into 2 degree intervals. For every pair of constant and gravitational F-histograms, we compute the *main direction* φ^{AB} which is the single scalar direction that best represents the direction between A and B. This value can be chosen as a weighted centroid of the two histograms, or as is the case with our experiments, it can be chosen using the method described in [8] and [12], which decomposes the two histograms into effective, contradictory, and compensatory forces. To facilitate the comparison of two F-histograms, we define a normalized F-histogram $\overline{F}_r^{AB}$ where $\overline{F}_r^{AB}(0) = F_r^{AB}(\varphi^{AB})$. This amounts to a shifting of the histogram bins, which, according to he properties defined in [5], is equivalent to rotating the pair of objects such that the degree to which A is to the right of B is maximized. The triple $\left(\overline{F}_0^{AB}, \overline{F}_2^{AB}, \varphi^{AB}\right)$ then represents the entirety of our knowledge regarding the spatial relationship between A and B.

B. Attributed Relational Graphs

Given a scene $\mathcal{X}_S$ consisting of a spatial configuration of N 2D objects, we define an index o_i and label l_i for each object. Additionally, we define the spatial relationship between the pair of objects (o_i, o_j) as the triple $h_{ij} = \left(\overline{F}_0^{ij}, \overline{F}_2^{ij}, \varphi^{ij}\right)$. The attributed relational graph representing the scene $\mathcal{X}_S$ is defined as the 4-tuple $G_S = (V_S, E_S, L_S, H_S)$, where V_S is the node set of object indices, E_S is the set of edges between vertices, L_S is the set of node attributes containing a label l_i for each node $o_i \in V_S$, and H_S is the set of edge attributes containing a spatial relationship h_{ij} for each edge $e_{ij} = (o_i, o_j) \in E_S$. Fig. 2 shows an example of a scene and its corresponding ARG.

Depending on the magnitude of N, we may decide to compute the edge relationship between only some of the scene objects. This is the case for the reference scene $\mathcal{X}_R$, which may contain several thousand objects. When computing G_R, we restrict the set of outgoing edges from each node to that object's K nearest neighbors. We can provide other pruning heuristics, such as limiting the distance between object pairs and restricting the number of objects which can lie between the two objects in question. For an overview of how to compute the latter, we refer readers to [13].

An ARG is complete if there is an edge relationship between every pair of nodes. A target scene $\mathcal{X}_T$ that is to be found within the reference scene $\mathcal{X}_R$ is defined as a complete ARG G_T. A potential solution to the scene matching problem is a subgraph of the reference ARG G_R that could match G_T. A subgraph of G_R is defined as $G_\Gamma \subset G_R$, where G_Γ contains only some of the objects defined in G_R and the corresponding edges and attributes. In order to evaluate the quality of a potential solution, we require it to be isomorphic to the target ARG—that is, each potential solution must be a complete ARG. Were it not for the attributes associated with each graph, we could define our scene matching problem as a subgraph isomorphism search. Since the definition of isomorphism between two ARGs is somewhat vague, we instead refer to the scene matching problem as that of approximate subgraph matching, in which we seek to find a subgraph of the reference ARG G_R which has a high degree of similarity to the target ARG G_T. We define a similarity measure between two ARGs in the following section.

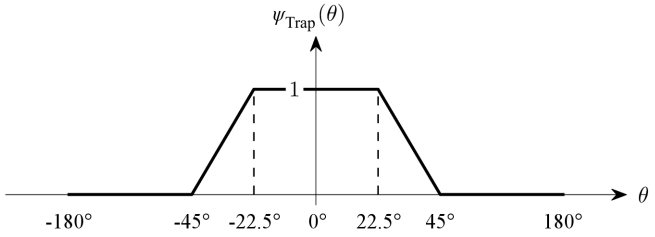

Fig. 3. Trapezoidal weighting function.

C. Comparing Graphs

We now define a similarity measure that we call *elastic angles* [10], which can be applied to two ARGs under the following circumstances:

- Both ARGs must be complete graphs of the same size.
- The correspondence between objects is assumed to be defined by the vector order of the object indices.

Under these conditions, the two ARGs G_1 and G_2 will each contain N nodes and $N(N-1)$ edges. Because of the semantic inverse property of the histograms of forces [5], which states that an inverse relationship is equivalent to a 180 degree shift of the histograms, we can reduce the required computation by considering only edges e_{ij} for which $i < j$. The resulting two lists of edge attributes are notated as $\widetilde{H}_1$ and $\widetilde{H}_2$, each of length $N(N-1)/2$. The ith element of $\widetilde{H}_1$ is referenced as $\widetilde{h}_{1i} = \left(\widetilde{F}_0^{1i}, \widetilde{F}_2^{1i}, \widetilde{\varphi}^{1i}\right)$ with a similar notation for $\widetilde{H}_2$.

We begin by defining the similarity between two normalized F-histograms $\widetilde{F}_r^{1i}$ and $\widetilde{F}_r^{2i}$ as their cross-correlation,

$$\psi_{CC}\left(\widetilde{F}_r^{1i}, \widetilde{F}_r^{2i}\right) = \frac{\sum_\theta \widetilde{F}_r^{1i}(\theta)\widetilde{F}_r^{2i}(\theta)}{\sqrt{\sum_\theta \left(\widetilde{F}_r^{1i}(\theta)\right)^2 \sum_\theta \left(\widetilde{F}_r^{2i}(\theta)\right)^2}}. \quad (1)$$

The similarity of the constant and gravitational F-histograms are weighted equally, giving a histogram similarity score that does not yet include orientation information from the main directions,

$$\psi_{\text{Hist}}\left(\widetilde{h}_{1i}, \widetilde{h}_{2i}\right) = \tfrac{1}{2}\psi_{CC}\left(\widetilde{F}_0^{1i}, \widetilde{F}_0^{2i}\right) + \tfrac{1}{2}\psi_{CC}\left(\widetilde{F}_2^{1i}, \widetilde{F}_2^{2i}\right). \quad (2)$$

The main direction terms of each ARG encode the scene orientation with respect to a global reference angle. In order to compare two ARGs that may have been defined with different reference angles, we must determine the optimal rotation angle that produces the best alignment. We compute a vector of angular differences by subtracting the corresponding main direction angles between ARGs,

$$\mathbf{D} = \left(d_1, \ldots, d_{N(N-1)/2}\right), \qquad d_i = \widetilde{\varphi}^{1i} - \widetilde{\varphi}^{2i}. \quad (3)$$

We choose the optimal rotation angle $\varphi^\star$ as the median value of $\mathbf{D}$, computed on a periodic domain. This is equivalent to picking the angle from $\mathbf{D}$ which minimizes the distance to all other angles [14],

$$\varphi^\star = \underset{d_i \in \mathbf{D}}{\arg\min} \left[180^\circ - \sum_{d_j \in \mathbf{D}} \left| 180^\circ - |d_j - d_i| \right| \right]. \quad (4)$$

The similarity between the two ARGs is defined as a weighted average of the histogram similarity scores, where the weight for each one is defined by how far the histograms would need to be rotated in order to be aligned with the optimal rotation angle. We define a symmetric trapezoidal function ψ_{Trap} (Fig. 3) to determine the weighting factor and define the overall ARG similarity as

$$\Psi(G_1, G_2) = \frac{\sum_{i=1}^{N(N-1)/2} \psi_{\text{Trap}}\left(\varphi^\star - d_i\right) \psi_{\text{Hist}}\left(\widetilde{h}_{1i}, \widetilde{h}_{2i}\right)}{N(N-1)/2}. \quad (5)$$

This measure takes a value between 0 and 1, with 1 indicating a perfect match between ARGs. The complexity of computing the similarity is on the order of $O(N^2)$.

III. A Memetic Algorithm

Given a geospatial reference database as a reference scene $\mathcal{X}_R$ and a target scene $\mathcal{X}_T$, which is constructed from an input sketch, our goal is to find a region of the reference database that closely matches the target scene. Because the single top scoring match according to our similarity measure may not coincide with the best match according to an analyst, we are interested in returning a list of the top scoring matches, which may come from several different regions of the reference database. An evolutionary algorithm (EA) is an appropriate tool for this problem, as it can maintain a population of potential solutions, and also have an adjustable stop condition based on factors decided by the analyst. In this paper, we use a memetic algorithm for matching spatial configurations [8] that combines the global search strategy of an EA with an individual local search operator.

A memetic algorithm [15] is an extension of an EA that adds a local refinement step to improve individual solutions during the evolutionary process. In general, an EA starts by creating a population of random individuals that cooperate and compete over time to find good solutions to a given problem. The basic operators of an EA are selection, recombination, and mutation. The selection operator ensures that good solutions survive to subsequent generations while still allowing individuals of lower fitness to occasionally survive. Recombination serves to improve the existing population by combining the best attributes of individuals, and mutation is performed occasionally to maintain population diversity and to search unexplored areas of the search space. The local search operator introduced by the memetic approach increases the fitness of an individual solution by utilizing domain-specific knowledge. Proper algorithm design requires a good balance between exploration provided by mutation, and exploitation provided by recombination and the local search operator.

Prior to the start of the algorithm, the reference ARG G_R is computed from the reference scene $\mathcal{X}_R$ and the target ARG G_T is computed from the target scene $\mathcal{X}_T$. We define the number of target objects in $\mathcal{X}_T$ as N and the number of reference objects in $\mathcal{X}_R$ as M with $M \gg N$. For convenience, we represent the objects of the reference scene as the vector $\mathbf{R} = (x_1, \ldots, x_M)$ and those of the target scene as the vector $\mathbf{T} = (o_1, \ldots, o_N)$. A potential solution, or individual, is represented as the vector $\mathbf{\Gamma} = \left(x_{(1)}, \ldots, x_{(N)}\right) \subset \mathbf{R}$ where each $x_{(i)}$ is an object in $\mathbf{R}$ that should be matched with the target object o_i. The ith object of the individual is notated as Γ_i, and we define the label attribute of an object x_i as $L(x_i)$. Given a potential solution in vector form, it is straightforward to build the ARG representation G_Γ as a subgraph of G_R and compute the similarity with the target ARG, which we use as the fitness function for the memetic algorithm, $f(\mathbf{\Gamma}) = \Psi(G_T, G_\Gamma)$.

Ultimately, this is a combinatorial optimization problem with a specific set of constraints. Only potential solutions that form complete ARGs can be compared to the target scene, and we also require that the labels of the target objects match those of the potential solution. We enforce these constraints by ensuring that the population always contains valid individuals. This implies that solutions will be spatially compact, due to the pruning of distant object relationships in the reference ARG. Recombination is difficult in this domain because the solutions can be spaced far apart in the search space. Swapping some of the objects from two individuals with standard crossover operators will almost always yield an invalid solution. Similarly, mutation cannot blindly replace an object from an individual with a random one from the reference scene, as this will also tend to result in an invalid solution. We therefore delegate the role of exploration to the random initialization operator, which returns a valid solution at a random location in the search space, and the role of exploitation to the local search operator, which uses domain-specific knowledge to improve a single solution.

The outline of our search procedure is given in Algorithm 1. The algorithm requires reference and target ARGs as input along with predefined constants μ (population size), τ (replacement rate), and ρ (replacement percent). The search procedure starts by generating an initial population of μ random individuals. Each random individual is chosen by first finding the label of a target object that appears least often in the reference scene, and then picking a random object from the reference scene with that label. This becomes the base object of the individual, and the remaining objects are chosen from its nearest neighbors such that the labels match those of the target scene. Closer neighbors have a higher likelihood of being chosen in order to keep the individual spatially compact. If none of the base object's neighbors can satisfy the label requirements of the target scene, a new base object is chosen in a different location.

After creating the initial population, the local search operator is applied to each individual. These newly formed child solutions replace their parents if they have a greater fitness value, utilizing domain knowledge to move closer to a locally optimal solution. After a few generations, the repeated use of the local search operator makes it likely that many of the individuals in the population have converged to local optima. To encourage exploration, we replace the least fit fraction ρ of the population with new random individuals every τ generations. This can be seen as a type of strong mutation, which allows new regions of the search space to be explored while ensuring that the best solutions remain in the population. The algorithm can be summarized as a parallel local search procedure that keeps a record of the best results. By casting it in a memetic framework, we can better understand how each operator affects the balance of exploration and exploitation.

Algorithm 1 Memetic Algorithm for Scene Matching

Input:
  G_R and G_T
  Constants: μ, τ, ρ
Initialize:
$t = 0$
Create initial population of individuals:
  $P^{(0)} = \left(\mathbf{\Gamma}^1, \ldots, \mathbf{\Gamma}^\mu\right)$
While stopping criteria is not met **Do**
  $P^{(t+1)} = P^{(t)}$
  $t = t + 1$
  If t is a multiple of τ **Then**
    Replace the lowest scoring fraction ρ of $P^{(t)}$ with new random individuals
  Else
    For Each individual $\mathbf{\Gamma}^P \in P^{(t)}$ **Do**
      Generate child $\mathbf{\Gamma}^C = \text{local\_search}\left(\mathbf{\Gamma}^P\right)$
      If $f\left(\mathbf{\Gamma}^C\right) > f\left(\mathbf{\Gamma}^P\right)$ **Then**
        Replace $\mathbf{\Gamma}^P$ with $\mathbf{\Gamma}^C$
      End If
    End For
  End If
End While
Output: Top scoring individuals in $P^{(t)}$

IV. Local Search Operators

The local search operators we have designed for scene matching in this domain each take a target and reference scene as input, along with a single parent solution, and return a child solution from the reference scene that should be more similar to the target scene. Each operator is restricted to the local neighborhood of the parent, and while the algorithms are mostly deterministic, both contain some degree of randomness. This implies that although the best solution is usually found, repeated applications of the algorithm on the same solution may yield different results. The following sections describe the greedy one-seed set reconstruction operator introduced in [8], and a new operator based on the VF2 algorithm for finding subgraph isomorphisms [11].

A. One-Seed Set Reconstruction

The one-seed set reconstruction method is based on the idea that the best solution in a local region can be reconstructed from a small starting seed of a single object. Given a complete

Algorithm 2 One-Seed Set Reconstruction Search Operator

Input:
Target scene: $\mathbf{T} = (o_1, \ldots, o_N)$
Reference scene: $\mathbf{R} = (x_1, \ldots, x_M)$
Parent: $\mathbf{\Gamma}^P = \left(\Gamma_1^P, \ldots, \Gamma_N^P\right)$ where each $\Gamma_i^P \in \mathbf{R}$
Initialize:
$C_{\text{best}} = 0$
Initialize list of index locations: $I = \{1, \ldots, N\}$
For Each $(i, j) \in I \times I$ such that $L(o_i) = L(\Gamma_j^P)$ **Do**
Clear $\mathbf{\Gamma}'$
Create the partial target scene: $\mathbf{T}' = (o_i)$
Update remaining index locations: $I' = I - i$
Set $\Gamma_i' = \Gamma_j^P$
Get the set of valid nearest neighbors $\mathcal{N} \subseteq \mathrm{R}$ of Γ_j^P
While $|I'| > 0$ **Do**
Pick an index $k \in I'$ randomly
Add o_k to the partial target scene: $\mathbf{T}' = (\ldots, o_k)$
$f_{\text{best}} = 0$
For Each $x^\star \in \mathcal{N}$ **Do**
Set $\Gamma_k' = x^\star$
Evaluate the partial fitness $f(\mathbf{\Gamma}') = \Psi(G_{T'}, G_{\Gamma'})$
If $f(\mathbf{\Gamma}') > f_{\text{best}}$ **Then**
$x_{\text{best}} = x^\star; \quad f_{\text{best}} = f(\mathbf{\Gamma}')$
End If
End For
Set $\Gamma_k' = x_{\text{best}}$
Remove this index location: $I' = I' - k$
Update $\mathcal{N}$ as the nearest neighbors of Γ'
End While
If $f(\mathbf{\Gamma}') > C_{\text{best}}$ **Then**
$\mathbf{\Gamma}^C = \mathbf{\Gamma}'; \quad C_{\text{best}} = f(\mathbf{\Gamma}')$
End If
End For
Output: $\mathbf{\Gamma}^C$

target scene $\mathbf{T} = (o_1, \ldots, o_N)$, we define a partial target scene $\mathbf{T}' = \left(o_{(1)}, \ldots, o_{(n)}\right) \subset \mathbf{T}$, which only contains some of the original objects. Likewise, we define a partial solution as a vector $\mathbf{\Gamma}' = \left(x_{(1)}, \ldots, x_{(n)}\right) \subset \mathbf{R}$, which associates each object of the partial target scene with an object from the reference scene. The partial fitness $f(\mathbf{\Gamma}') = \Psi(G_{T'}, G_{\Gamma'})$ only considers the specified subset of original target objects.

The overall outline of the one-seed set reconstruction method is given in Algorithm 2. Given a parent solution $\mathbf{\Gamma}^P = \left(\Gamma_1^P, \ldots, \Gamma_N^P\right)$, we cycle through each object $\Gamma_i^P \in \mathbf{\Gamma}^P$ and use it as the seed object. We then consider all target objects with the same label attribute and create a partial solution $\mathbf{\Gamma}'$ for each one. For each partial solution, we randomly pick an unassigned target object $o_i \in \mathbf{T} - \mathbf{T}'$ and find the set of valid nearest neighbors $\mathcal{N} \subseteq \mathbf{R}$ to which o_i could be assigned while keeping $G_{\Gamma'}$ a complete ARG. For each neighbor, we create a set of temporary partial solutions, each with o_i assigned to a different neighbor object $x^\star \in \mathcal{N}$. The neighbor which produces the greatest partial fitness is added to the partial solution and o_i is added to the partial target scene. We continue to match the unassigned target objects in this greedy manner until all target objects have been assigned. The complete solution with the highest fitness among all solutions generated from different starting seeds is returned as the child solution.

Algorithm 3 VF2 Subgraph Isomorphism Search Operator

Input:
Target graph: G_T
Reference scene: $\mathbf{R} = (x_1, \ldots, x_M)$
Parent: $\mathbf{\Gamma}^P = \left(\Gamma_1^P, \ldots, \Gamma_N^P\right)$ where each $\Gamma_i^P \in \mathbf{R}$
Constants: $f_{\min}, \delta, K$
Initialize:
$\mathbf{\Gamma}^C = \emptyset$
While $\mathbf{\Gamma}^C = \emptyset$ **Do**
Get the set of K valid nearest neighbors $\mathcal{N} \subseteq \mathbf{R}$ of $\mathbf{\Gamma}^P$
Build the neighbor graph G_N with random node order
Run the VF2 algorithm to find the set of subgraphs $\mathcal{C} \subset G_N$ that are compatible with G_T using fitness threshold $f_{\min}$
If $\mathcal{C}$ is empty **Then**
$f_{\min} = f_{\min} - \delta$
Else
Set $\mathbf{\Gamma}^C$ as the vector form of the best solution in $\mathcal{C}$
End If
End While
Output: $\mathbf{\Gamma}^C$

Given that the complexity of a single fitness evaluation is $O(N^2)$, the worst-case complexity of the one-seed method can be derived as $O(N^5 K)$, where K is the maximum neighborhood size of the reference scene. In practice, the complexity tends to be lower since object labels prune the number of possible seeds, and evaluating the fitness of a partial solution requires fewer operations than for a complete solution. Even so, this method is only appropriate for relatively small values of N.

B. VF2 Subgraph Isomorphism

The VF2 subgraph isomorphism algorithm [11] is well suited for matching attributed graphs. The algorithm uses a state space representation that encodes each possible mapping between two graphs as a search state. A set of syntactic feasibility rules based on graph structure and semantic feasibility rules based on attribute compatibility are employed to prune the search tree. Given a reference graph G_R and a target graph G_T, the VF2 algorithm begins by exploring all feasible mappings of a single object from the target graph onto an object from the reference graph. Feasible object pairs are added to each mapping one at a time using a depth-first search until no more feasible pairs exist, or a subgraph isomorphism is found. To prevent visiting the same state multiple times, an

arbitrary total order relation is defined on the graph nodes so that any potential mapping is reachable from only a single previous state.

The success of the VF2 algorithm depends on the ability to prune the search tree by ignoring the addition of infeasible object pairs. A key assumption is that the overall compatibility of two graphs can be determined by evaluating the compatibility of each node and edge pair independently via node and edge compatibility functions. This is not possible for our similarity measure since each edge in the graph contributes to the overall orientation. We therefore modify the VF2 algorithm slightly so that the feasibility of a new object pair is computed by evaluating the partial fitness of the partial solution resulting from the addition of the object pair. The pair is considered feasible if the partial fitness is greater than a minimum fitness threshold, $f_{\min}$. Evaluating the feasibility in this way changes the assumptions of the VF2 algorithm and no longer guarantees that all compatible subgraph isomorphisms will be found. For example, a complete solution may be feasible, but it will not be reached if an intermediate state is determined to be infeasible. To ameliorate this effect, we randomize the index order of the graph nodes for each run of the algorithm, which changes the search path to any mapping state.

Our VF2 local search operator is outlined in Algorithm 3. Given a parent solution $\mathbf{\Gamma}^P = \left(\Gamma_1^P, \ldots, \Gamma_N^P\right)$, we construct a neighbor graph $G_N \subseteq G_R$ containing the K nearest neighbors of G_Γ. The VF2 algorithm is then used to find all subgraphs of G_N which are compatible with the target graph G_T using a fitness threshold $f_{\min}$. If no solutions are found, the fitness threshold is decremented by a value δ and the VF2 algorithm is run again. This continues until at least one solution is found, and of these, the solution with the highest fitness is returned as the child solution.

The complexity of the VF2 algorithm can be stated in terms of the best and worst cases. For a neighbor graph with K nodes and a target graph with N nodes, the cost of exploring a single state can be done with a complexity of $O(K+N)$ with an additional complexity of $O(N^2)$ for the fitness evaluation. The algorithm will visit N states in the best case scenario and $K!$ states in the worst case. This gives best-case and worst-case complexities of $O\big(N(K+N^2)\big)$ and $O\big(K!(K+N^2)\big)$ respectively. We rely on experimental evaluation to determine the average complexity.

V. Experiments

We test our method on a hand-segmented database of Columbia, MO containing 2467 buildings and 378 parking lots shown in Fig. 4. The reference ARG was built a priori 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 5 other objects in-between. Two sets of test sketches were randomly generated representing direct resubstitution and transformed input sketches. The direct resubstitution sketches contain objects taken directly from the reference scene with no transformation applied. These sketches verify that a match can be found under ideal conditions. The transformed sketches take a direct resubstitution sketch and reduce all of the objects to their bounding boxes before applying a global random rotation to the whole scene. These sketches approximate the imprecision that would be present in a sketch generated by hand or by the T_2S system. For both types we consider sketches of 4, 6, 8, 10, and 12 objects, and for each sketch size, we generate 100 test sketches from random locations in the reference scene.

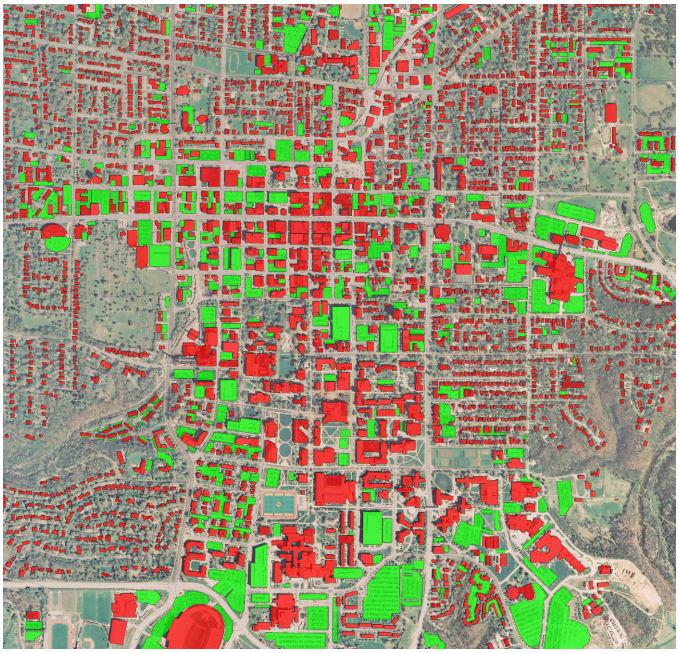

Fig. 4. The geospatial reference database used for our experiments. The set contains 2467 buildings shown in red (dark gray) and 378 parking lots shown in green (light gray) from downtown Columbia, MO and the University of Missouri campus.

TABLE I. Algorithm Parameters

	1-Seed	**VF2**
Population Size (μ)	50	10
Replacement Frequency (τ)	10 Gen.	2 Gen.
Replacement Percent (ρ)	80%	80%
Nearest Neighbor Connectivity (K)	50	50
Initial Minimum Fitness Threshold ($f_{\min}$)	N/A	0.95
Fitness Threshold Step Size (δ)	N/A	0.95
Maximum Number of Generations	100 Gen.	100 Gen.

The memetic algorithm is run with the reference scene 10 times for each sketch using either the one-seed or VF2 local search operators. The search stops when the original sketch location is found or after 100 generations. We then record how many of the tests were correctly matched to the original location within the allowed search time. The complete list of algorithm parameters is given in Table I. For the one-seed operator, we use a population size of 50 individuals with a replacement rate of 80% every 10 generations. For the VF2 operator, due to the greater computational overhead, we use a smaller population size of 10 individuals with a more aggressive replacement strategy of 80% every 2 generations. Additionally, for the VF2 operator we start with a minimum fitness threshold of 0.95 and a fitness step size of 0.05. Although these parameters are not identical, they are good choices for each operator based on preliminary experiments, and allow our comparison to evaluate a best-case scenario for

TABLE II. RESULTS OF THE MEMETIC ALGORITHM FOR MATCHING SPATIAL CONFIGURATIONS

		Direct Resubstitution Sketches					Transformed Sketches				
Local Search Method	*Objects in Sketch*	*% Correctly Matched*	*Generations*		*Time (seconds)*		*% Correctly Matched*	*Generations*		*Time (seconds)*	
			Mean	Std. Dev.	Mean	Std. Dev.		Mean	Std. Dev.	Mean	Std. Dev.
1-Seed	4	95.1%	22	16	89	77	80.6%	36	30	151	139
	6	98.5%	12	13	207	183	95.6%	16	18	318	367
	8	99.6%	9	8	494	460	93.4%	18	22	771	998
	10	94.8%	13	21	1258	1531	81.7%	28	35	2171	2790
	12	86.2%	20	32	3304	4282	87.2%	20	31	4317	6012
VF2	4	98.7%	16	12	26	33	80.4%	32	33	97	176
	6	96.6%	19	15	81	104	94.3%	22	20	143	183
	8	98.1%	19	12	189	259	86.0%	31	28	744	1086
	10	90.8%	26	24	1777	2705	69.2%	43	38	5927	9809
	12	76.5%	39	33	4986	4864	78.8%	38	32	11 009	15 040

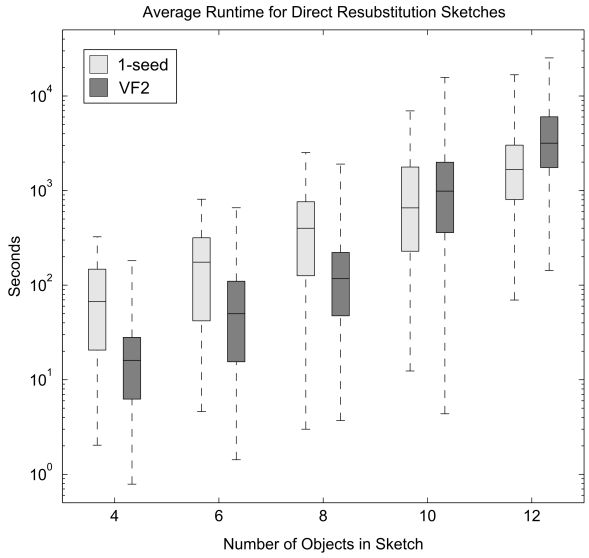

Fig. 5. Average runtime of the experiments with direct resubstitution sketches, showing the minimum, maximum, upper and lower quartile, and median values for each configuration.

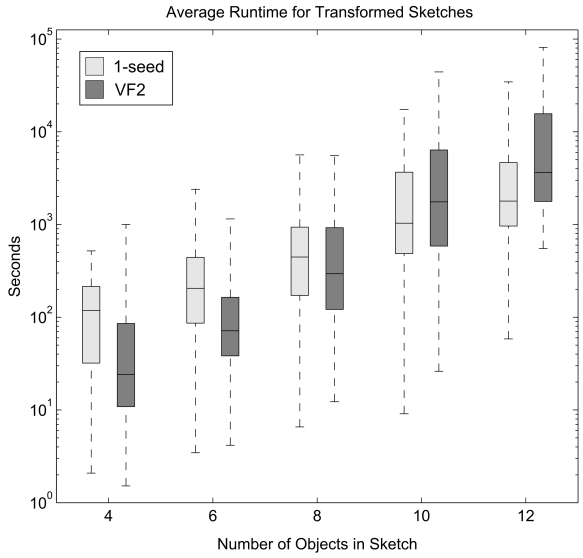

Fig. 6. Average runtime of the experiments with transformed sketches, showing the minimum, maximum, upper and lower quartile, and median values for each configuration.

each operator. All of our experiments were implemented in C++ and conducted in parallel on a 64-bit computer running Windows 7 with 8 logical processors clocked at 2.8 GHz and 12 GB of RAM.

The results of our experiments are given in Table II. In general, the direct resubstitution sketches had a greater number of tests that found the correct match and also had faster runtimes. This is to be expected since the transformed sketches will not always find the original sketch location to be the most similar match. This is especially true if the sketch contains only a few objects, or if it is a common configuration such as a row of buildings. Another observation is that the match rate tends to decrease if the number of sketch objects is very large or small. Large sketches can be difficult to match in the allotted time due to the greater number of variables, whereas small sketches often match to so many places in the reference scene that the correct match is less likely to be found.

The average runtimes of the experiments with direct resubstitution sketches are given in Fig. 5 and the average runtimes of the experiments with transformed sketches are given in Fig. 6. These box plots show the minimum, maximum, upper and lower quartile, and median values of the 1000 tests in each configuration. In general larger sketches have a greater runtime, although there is a very large spread between the best and worst cases. The VF2 method also appears to perform slightly better than the one-seed method for small sketch sizes, whereas the one-seed method is slightly better for large sketch sizes.

Finding the original sketch location is only one measure of success for this algorithm. An analyst may wish to find all of the close-matching locations or the input sketch may not originate from the reference scene. This requires a more qualitative assessment of the results. Two example search results are given in Fig. 7. The first row shows the top five results of the algorithm on a direct resubstitution sketch using the VF2 operator and the second row shows the top five results

Fig. 7. Examples of the top matches found using our memetic algorithm. The top row shows a direct resubstitution sketch and the top five results found with the VF2 local search operator, and the bottom row shows a transformed sketch and the top five results found with the one-seed local search operator.

on a transformed sketch using the one-seed operator. In both examples, the ground truth location is returned as the top result and the remaining solutions all share a similar spatial configuration with the input sketch. Note that the results are all independent of the scale and rotation of the input sketch.

VI. Conclusions and Future Work

Our experiments have demonstrated the capability of the HoF-ARG model for representing spatial relationships. The memetic algorithm for matching spatial configurations is a useful framework that allows separate development of local and global search strategies. Both the one-seed and VF2 local search operators work well on this problem with the one-seed operator performing slightly better for transformed sketches and the VF2 operator performing slightly better for small sketches. The runtime of both methods becomes prohibitive as the sketch size grows large.

There are several directions for future work in this area. The effect of neighborhood size on the VF2 operator has not been fully explored. As the size of the neighborhood approaches the size of the search space, scene matching can be done with a single run of the VF2 algorithm, however we suspect this approach may not scale to very large reference scenes. The global search portion of the memetic algorithm could be improved by employing a $(\mu + \lambda)$ evolution strategy with appropriate diversity mechanisms. Finally, additional attributes on the nodes and edges of the ARG would help prune the search space and improve search performance.

References

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

[2] M. J. Egenhofer, "Query processing in spatial-query-by-sketch," *J. of Visual Languages and Computing*, 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. 2009 IEEE Int. Conf. Fuzzy Systems (FUZZ-IEEE 2009)*, Jeju Island, Korea, Aug. 2009, pp. 518–523.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] A. R. Buck, J. M. Keller, and M. Skubic, "A memetic algorithm for matching spatial configurations with the histograms of forces," *IEEE Trans. Evol. Comput.*, Nov. 2012, in press.

[9] ——, "A modified genetic algorithm for matching building sets with the histograms of forces," in *Proc. 2010 IEEE Congr. Evolutionary Computation (CEC)*, Barcelona, Spain, Jul. 2010, pp. 1–7.

[10] A. R. Buck, J. M. Keller, M. Skubic, M. Detyniecki, and T. Baerecke, "Object set matching with an evolutionary algorithm," in *Proc. 2011 IEEE Symp. Computational Intell. Security and Defense Applicat. (CISDA)*, Paris, France, Apr. 2011, pp. 43–50.

[11] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento, "A (sub)graph isomorphism algorithm for matching large graphs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 10, pp. 1367–1372, Oct. 2004.

[12] 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*, vol. 31, no. 4, pp. 573–588, Aug. 2001.

[13] I. Bloch, O. Colliot, and R. M. Cesar Jr., "On the ternary spatial relation 'between'," *IEEE Trans. Syst., Man, Cybern. B*, vol. 36, no. 2, pp. 312–327, Apr. 2006.

[14] N. I. Fisher, *Statistical Analysis of Circular Data*. Cambridge, England; New York, NY: Cambridge University Press, 1993.

[15] 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.