Abstract—Geospatial Intelligence analysis involves the combination of multi-source information expressed in logical form (as sentences or statements), computational form (as numerical models of physics or other processes), and sensor data (as measurements from transducers). Each of these forms has its own way to describe uncertainty or error: e.g., frequency models, algorithmic truncation, floating point roundoff, Gaussian distributions, etc. We propose BRECCIA, a Geospatial Intelligence analysis system, which receives information from humans (as logical sentences), simulations (e.g., weather or environmental predictions), and sensors (e.g., cameras, weather stations, microphones, etc.), where each piece of information has an associated uncertainty; BRECCIA then provides responses to user queries based on a new probabilistic logic system which determines a coherent overall response to the query and the probability of that response; this new method avoids the exponential complexity of previous approaches. In addition, BRECCIA attempts to identify concrete mechanisms (proposed actions) to acquire new data dynamically in order to reduce the uncertainty of the query response. The basis for this is a novel approach to probabilistic argumentation analysis.

I. INTRODUCTION

A. Geospatial Intelligence (GEOINT)

Current knowledge-based GEOINT systems do not incorporate a broad notion of uncertainty quantification (UQ), although such a capability would allow decision makers to make more informed decisions, or to acquire more data before coming to conclusions. In addition, it would be better if system responses were provided with an explanation of how they were derived, as well as how the uncertainty was determined. This can be the result of sensor error, computational error, human error, etc., and the best models should be selected at each time step in order to reduce the variance on quantities of interest. In addition, intelligence, surveillance and reconnaissance support systems should generate dynamic path planning solutions which can include constraints on time, energy, or uncertainty reduction. The automatic generation of constraints arising from the various models can be used to inform the deployment of data measurement systems. The application studied here is UAV (Unmanned Aerial Vehicle) surveillance and reconnaissance in urban areas. Some work has been done in this general area (e.g., see [19] for a novel guidance law in windy urban environments combining pursuit and line-of-sight laws, and [35] for a multi-cost UAV mission path planner).

Exploiting Dynamic Data Driven Application Systems (DDDAS) for large-scale, geographically distributed scenarios promises significant advantages, and we envision an approach that combines various types of information with associated uncertainty to enable model-driven active data acquisition. Figure 1 shows our proposed overall organization of a dynamic data-driven GEOINT application system (called BRECCIA after a type of rock formed from several mineral pieces held together in a fine-grained matrix). Typical geographic visual and data products include: maps, charts, digital files, imagery and vector information. Value added items include: data verification, correction, updates, densification, reformating, orthorectification, map finishing, seismic activity, intelligence reports, and additional categories of content [31].

We describe here two major novel research results: (1) the combination of formal probabilistic logic methods with state-of-the-art physics-based uncertainty quantification methods, and (2) uncertainty driven active information data acquisition, demonstrated by UAV path planning, to optimize performance or to resolve contradictory information. The probabilistic logic method is a re-formulation of the approach described in [23] (although Boole [5] first proposed it); see [15] for details. Basically, Nilsson’s method requires first solving the SAT problem (i.e., find all consistent truth

---

1This material is based upon work supported by the Air Force Office of Scientific Research under award number FA9550-17-1-0077.
BRECCIA is a dynamic uncertainty monitoring and reduction system; that is, uncertainty can arise due to a change in local conditions (e.g., the weather may make movement difficult) or new information may be available (e.g., obscurants in the air, new interesting sites). As a consequence, steps are taken to reduce the uncertainty of assessments; for example, an unmanned aerial vehicle may overfly a zone to get visual confirmation of damage assessments, or a swarm of quadrotors may be sent to provide surveillance or reconnaissance in an urban area. Transparent and precise reasons are offered to the user to explain uncertainty conditions and how they may be resolved.

The study and analysis of geospatial data has progressed from simple Geographic Information Systems (GIS) (spatially organized layers of digital data with associated functional elements [9]) to broader geocomputations (see [1], [18]). Such systems involve geospatial data, large-scale computation, and a willingness to apply data mining techniques in order to build models based on experience in addition to analytical theories. This involves GIS as well as artificial intelligence, high performance computing and underlying science models. At its core, this is a major paradigm shift which allows large-scale data exploitation. We believe that GEOINT systems need to be centered in such a modern framework: this means a cloud-based and scalable system with a user-friendly interface (see [22]).

Example applications include decision support for snow removal [34], environmental science [33] knowledge base integration for GIS [24], multiagent systems using GIS data for planning purposes [29], and the exploitation of machine learning and data mining methods in this domain [11], [27], [38], [39] Large-scale geospatial big data projects and systems are relevant as well [7], [13], [32], [37]. We have previously developed the basic computational framework for a DDDAS-based cloud architecture for small-scale structural health monitoring of aircraft [40].

II. THE BRECCIA SYSTEM

A. Current Implementation Details

BRECCIA is designed using a well documented multi-agent, Belief-Desire-Intention (BDI) framework called Jason [6]. Jason includes an interpreter for an extended version of the AgentSpeak(L) [28] language, which provides a Prolog-like grammar. Both the style, resembling a natural language application, and the operational semantics of the extended language that enable a data-driven architecture, fit well with the proactive and reactive goals of BRECCIA. Each agent in the BRECCIA system is composed of a backward chaining inference module (see [25] for a formal justification of modularity in BDI programming languages) with a probabilistic logic component. This module, and probabilistic component, form the most abstract fusion implementation in the BRECCIA system. To elaborate, this system may include many functional fusion modules that agents specialize for a particular application. For example, an agent that specializes in managing a particular unmanned air vehicle (UAV) requires a fusion module for estimating target detection. However, the human agent, a “user” in BRECCIA, requires information in more abstract terms to support the decision process. For example, a user may query the system by asking for the probability of mission success. Among the many bits of information that support this assertion, with their associated probabilities, and one of which is whether the target was detected, BRECCIA would respond as follows:

\[
\text{mission\_success}[p(0.9), \\
\text{justification}([\text{target\_detected, no\_damage, ...}])]
\]

The probabilistic logic component facilitates the propagation of quantitative uncertainties when making inferences from the knowledge base, all while maintaining the justifications for such inferences. Probabilities of sentences in the knowledge base, which may be interpreted as uncertainty, are represented as beliefs with annotations (a feature of the extended AgentSpeak language). For example, the belief, “Bob thinks it will rain today with 0.9 certainty,” may be represented in Jason as follows:

\[
\text{will\_rain\_today}[p(0.9), \text{source}(\text{bob})]
\]

If the application requires a data-driven event, for example to notify a user that their base assumptions about a prior simulation have changed, the program simply requires a plan in the following syntax:

\[
+\text{will\_rain\_today}[p(X)] : X < 0.8 \leftarrow \text{!tell\_bob}
\]

This plan says that if the certainty for “will rain today” dips below 0.8, then the agent should adopt the goal to tell Bob. This example demonstrates the clarity with which a data-driven application is programmed in this framework. Furthermore, the BDI system allows for the dynamic prioritization of goals (termed intentions once they are adopted by the agent); high priority goals can interrupt low priority intentions and become the current intention. This is a critical feature for a reactive system such as BRECCIA; changing assumptions about the current state of the environment should force replanning. In Jason this concept is represented as the plan’s “context,” shown in the above example after the colon asserting that \( X < 0.8 \). Another plan may exist in the agent to tell a higher authority when the probability dips below 0.5, for example.

The algorithm for backward chaining with probabilistic logic is shown in Algorithm 1. Inference rules with probabilistic logic are stored as material implications in BRECCIA in the following format:

\[
\text{mat\_imp}(\text{consequent}, [\text{anteecedents}]) \leftarrow [p(\text{probability})]
\]

As a concrete example, if the implication \( a \land b \implies c \) should be stored in the knowledge base with probability 0.9, then it is defined in the following syntax:

\[
\text{mat\_imp}(c, [a, b])\leftarrow[p(0.9)]
\]
Algorithm 1 matches each antecedent in the given material implication with a belief in the agent’s knowledge base and forms a new list with the current sentence probabilities. If an antecedent is an inferred belief that has yet to be calculated, then the current intention is suspended and a new intention is generated to resolve the constraint. The process of generating intentions for unresolved constraints uses the built-in (non-probabilistic) backward chaining that comes with Jason. However, this process is automatic and simply requires the following plan that responds to unresolved beliefs that have a material implication rule (notice the context after the colon) and adds the goal to run Algorithm 1:

```
+?X : mat_cnf(X,_) ← !backchain(X)
```

Once the list of probabilities have compiled, the non-linear probabilistic logic algorithm (described in section III) is executed to calculate the probability for the inferred belief. Finally the inferred belief is added to the agent’s knowledge base. The process of belief-revision (when probabilities of antecedents change) currently utilizes the same algorithm, except first a list of justifications, stored in the belief annotations, is compiled until the most abstract inference is located. Belief-revision in the BRECCIA system is an active area of research.

### B. Path Planning

BRECCIA includes an RRT* path planner [8], [17] that provides an asymptotically optimal path between two states (in this case between the launch and recovery sites). In this specific implementation, the goal is to fuse uncertainties in the environment and vehicle models into an estimate of the probability of success of a selected path in terms of power, flight time constraints, etc. Consider a simplified case of the Raven UAV tasked with monitoring a location in an urban environment.

The `simulate_path` method calculates the cost of the path between two locations in an environment at incremental steps during the path planning procedure. In the current implementation the cost is a representation of the flight time required to traverse between the two locations in one-meter increments. In this simplified model, the component of drag in the direction of the travel is added to the vehicle’s velocity to calculate a ground speed. The path cost is then calculated from the ground speed, hence higher ground speeds are favorable.

```
Data: 𝑣₁ ← start_position; 𝑣₂ ← end_position
Result: path_cost
path_velocity ← 𝑣₂ − 𝑣₁;
air_velocity ← cruise_speed + 𝑣-path_velocity;
for each meter in path do
    wind_velocity ← sample_wind(position);
    ground_speed ← cruise_speed + 𝑣-wind_velocity-air_velocity;
    time_burned ← (ground_speed)⁻¹;
    if time_burned > max_flight_time then
        path_cost ← inf;
        break;
    else
        path_cost ← path_cost + time_burned;
end
```

The critical step in Algorithm 2 is sampling from the wind model. Data from the wind model is captured in Matlab’s `griddedInterpolant` object to enable the path planner to sample from any location. Each wind sample is purposely corrupted by Gaussian noise with the given variance from the wind model as determined in the vortex simulation (see below). BRECCIA then runs the path planner multiple times to calculate a variance in path costs. Figure 2 shows the final path discovered by RRT using vortex simulation data.

The path planner was run thirty times for the operational scenario shown in Figure 2 with an assumed wind model variance of 0.5. One example of a resulting path is shown in Figure 2 as a dashed line between waypoints. A histogram of resulting flight times and a fitted normal distribution is shown in Figure Figure 3. Based on the resulting model, the mean flight time is approximately 612 seconds with a standard deviation of 27 seconds. The resulting 90th percentile flight-time is 653 seconds and is shown marked on the cumulative distribution function in Figure 4. These values are propagated to BRECCIA’s argumentation system for uncertainty fusion.

The key attribute of this path-sampling strategy is that the uncertainty in path optimality is included in the resulting flight-time model. Hence, the parameters that control RRT*, such as the number of iterations, may be adjusted by BRECCIA to achieve more or less uncertainty in the final result. The benefit of this is the ability to re-plan in real-time while maintaining awareness of the probability of mission success.
initially provided to the path planning algorithm by means of a simple 2D particle model simulation. The approach is based on the detailed description given by Greenspan [14]. It is assumed that the air mass is comprised of \( N \) particles, \( \vec{P}_i, i = 1 \ldots N \), each with mass \( m \). A system of coupled ODEs describes the motion of each particle:

\[
\vec{F}_i = m \frac{\partial^2 \vec{r}_i}{\partial t^2}, \quad i = 1 \ldots N
\]

where \( \vec{F}_i \) is the force on particle \( i \), and \( \vec{r}_i \) is the position vector of particle \( i \). Note that:

\[
\vec{F}_i = \vec{F}^{**}_i + \vec{F}^*_{i}
\]

where \( \vec{F}^{**}_i \) is a long range force (gravity and \( g = 980 \)), and \( \vec{F}^*_{i} \) is a short range force that holds within specified distance \( D \):

\[
\vec{F}^*_{ij,k} = \left[ -\frac{G}{(r_{ij,k})^p} + \frac{H}{(r_{ij,k})^q} \right] \vec{r}_{ij,k}
\]

To obtain values for the mission simulation, 2576 points are used in a square area where it is assumed that three sides are closed and one open (the top). A wind (\( V = [-10, 0] \)) passing by the top produces the cavity flow. The parameters are set to: \( G = 0 \), \( p = 3 \), \( H = 100 \), \( q = 5 \), \( D = 0.35 \) (the initial distance between particles is 0.25), and \( \delta t = 0.0001 \).

A snapshot of the state of the particles and the trajectory of a sample particle after 12,000 steps in the simulation are shown in Figure 5 on the left and right, respectively. A Gaussian noise model on the individual particle forces is used with \( \sigma^2 = 0.0001 \).

III. PROBABILISTIC LOGIC

Here we address the problem of finding a suitable representation for uncertainty associated with logical sentences. Although several approaches have been proposed in the past (see [2], [10], [12], [16], [21], [26], [36], [30]), they generally have some significant drawbacks. Usually, these have to do with the computational complexity of the semantics of the sentences (i.e., finding the set of consistent truth assignments is exponential in the number of sentences, or for Domingos, exponential in the number of cliques in the Markov graph [3]).

We have developed a new approach which computes the probabilities of the atoms in the sentences, and in terms of these, provides a solution for \( Pr(Q \mid KB) \), where \( Q \) is the query and \( KB \) is the knowledge base set of sentences (see [15] for details). Moreover, the knowledge of the probabilities of the atoms allows us to determine where the most uncertain part of the argument lies, and to
allocate resources to lower that uncertainty, thus decreasing
the uncertainty of the query. This is done by exploiting
the probability of a disjunctive clause, and developing a set
of equations from the sentences and their probabilities, and then
solving those equations (the number of equations equals the
number of sentences).

Our approach to probabilistic logic starts with an analysis
of Nilsson’s method [23] \(^2\). Given a set of \(n\) sentences,
\(S = \{S_1, S_2, \ldots, S_n\}\), in the propositional calculus, where
\(\{S_1, \ldots, S_{n-1}\}\) is the KB and \(S_n\) is the query, he first finds
the set of models of the sentences (i.e., the set of truth value
assignments to the sentences that are consistent using the
general semantic tree [20] for a set of sentences). In our new
approach [15], we avoid the exponential complexity of most
other algorithms by solving for the logical variable (atom)
probabilities directly as follows.

First, we assume that the sentences are given in conjunc-
tional normal form. This means that each sentence is a disjunct
of literals (an atom or its negation). Our second assumption is
that \(Pr(P \land Q) = Pr(P)Pr(Q)\); note that if this assumption
is violated, our methods also allow the bounds on the
probability to be determined. Next, we find the set of logical
atoms (i.e., variables) in \(S\); let \(A = \{A_1, A_2, \ldots, A_k\}\) be
this set. In this case the probability of a sentence can be
computed from the probability of its literals as follows:

\[
Pr(L_1 \lor L_2 \lor \ldots \lor L_p) = Pr(L_1) + Pr(L_2 \lor \ldots \lor L_p) - Pr(L_1)Pr(L_2 \lor \ldots \lor L_p),
\]

where the probabilities of clauses on the right hand side are
computed recursively.

Assuming that the logical (random) variables are inde-
pendent, each sentence gives rise to a (usually) nonlinear
equation defined by the recursive probability of the
disjunctive clause as defined above. The resulting set of
equations can be solved using standard nonlinear solvers
(e.g., \textit{fsolve} in Matlab), and a set of consistent values for
the probabilities of the atoms determined. Of course, one
problem with the nonlinear solver approach is that it may
not find a solution, even when one or more exist. Thus, our
current approach is to solve all equations that have a single
unknown (recursively), and then use an iterative method to
find a set of atom probabilities which produce the correct
sentence probabilities.

IV. EXPERIMENTS

Here we describe a scenario which uses a Raven, man-
portable, hand-launched small unmanned aerial vehicle (see
Figure 6). The mission is described in Figure 7 and consists
of going to a specified location (Named Area of Interest -
NAI), loitering there while reconnoitering some points of
interest, then going to the recovery location. Note that the
area is similar to our simulation scenario in that there are

\(^2\)Note that Nilsson’s method for propositional calculus is the same as that
proposed by George Boole in the 1800’s [4], [5].

three closed sides formed by the buildings, and the fourth
side is open. Assume the KB has sentences related to the
use of several Raven platforms for a mission, and a subset
of KB sentences are extracted that form an argument for
using the specific platform called Raven_1; then the argument
sentences and their origins are as follows (the sources of
information, i.e., which \textit{BRECCIA} component produced them
are given in parentheses):

1. Raven_1 Platform Available (Maintenance Reports)
2. Raven_1 Air Control Measures OK (Mission Plan)
3. Wind<17 Knots (Weather Report)
4. Precipitation Low (Weather Report)
5. Visibility OK (Weather Report)
6. Temperature between [0,90] (Weather Report)
7. (3 \& 4 \& 5 \& 6) \rightarrow Weather_OK
8. Target-Loiter distance < 7miles (Mission Plan)
9. Raven_1 Electro-Optical (Mission Plan)
10. Raven_1 Infra-Red (Mission Plan)
11. (9 and 10) \rightarrow Collection Requirement Done
12. Raven_1 Power OK (Path Planning)
13. Raven_1 Battery OK (Maintenance Reports)
14. Raven_1 Speed Known (Path Planning)
15. Raven_1 Altitude Known (Path Planning)
16. Raven_1 Loiter Time Known (Path Planning)
17. (12 \& 13 \& 14 \& 15 \& 19 \& 16 \& 21) \rightarrow Path_OK
18. Raven_1 Crew Available (Mission Plan)
19. Raven_1 Route Time Known (Path Planning)
20. Air Defense Threat Known (Mission Plan)
21. Named Areas of Interest Defined (Mission Plan)
22. (1 \& 2 \& Weather_OK \& Collection Requirements Specified \& Path_OK \& 18 \& 20) \rightarrow Raven_1 Mission_OK
23. (Query) Raven_1 Mission_OK?

Note that sentences 7, 11, 17, and 22 are human specified
rules. This leads to 23 CNF clauses for sentences 1 to 22,
and 1 for the query. Note that the probabilities for the individ-
ual sentences come from either human attribution (e.g.,
Raven_1 platform available), or from noise models in the
data (velocity vectors have Gaussian noise as determined by
the simulations). Furthermore, note that some sentences are
comprised of only a single literal, and thus, the probability
of the associated atom is the probability of the sentence.
However, the probability of some atoms (e.g., Path_OK in
sentence 17) is implicit and must be found as part of the
solution for the set of nonlinear equations arising from the sentences. The probabilities for the path planning sentences arise from simulations as shown in Figure 2.

As a preliminary test of the probabilistic logic computation, all KB sentences were assigned the (same) value ranging from 0.9 to 1.0 in steps of 0.02, and the resulting probabilities assigned to the query were [0.7406 0.8373 0.9011 0.9452 0.9768 1.0000]. As can be seen, when all the sentences are true (probability 1), the query is true with probability 1.

Now consider the case where sentences are provided by the following specialized agents in a BRECCIA system and their assigned sentences.

\[(\text{mission planner}, \text{uav manager}, \text{weather monitor}) \in \text{Breccia} \]
\[(2, 8, 9, 10, 11, 12, 13, 15, 16, 17, 18, 19, 22) \in \text{mission planner} \]
\[(1, 13) \in \text{uav manager} \]
\[(3, 4, 5, 6, 7) \in \text{weather monitor} \]

To facilitate inference across agents, material implications that include external propositions include the special annotation "$\text{ask(}\text{agent}\text{)}$". Intentions are then automatically generated to query agents across the BRECCIA network. A mission is simulated with every sentence probability set to 0.9, and Figure 8 shows the simulation console output. BRECCIA finds the following values for the implicit atoms: \(\text{Pr(Weather OK)} = 0.8476, \text{Pr(Collection Requirement Done)} = 0.8765, \text{Pr(Path OK)} = 0.7909, \text{and Pr(Mission OK)} = 0.7406\). Thus, to increase the probability of mission success, it is essential to increase the probability of Path OK. For example, replanning to ensure that the weather info in sentences 3–6 and power info in 12–13 has probability 0.95, raises the \(\text{Pr(Mission OK)} \) to 0.7560. The user must decide if it is worth investing resources to improve that information. The advantage of BRECCIA, is that deeper insight into the reasons for the overall probability of success can be known.

V. CONCLUSIONS AND FUTURE WORK

BRECCIA, a dynamic geospatial information analysis system, is described which provides a unified probabilistic framework for multi-source data uncertainty. The major contributions here are: (1) an effective probabilistic logic methodology, and (2) an experimental GEOINT system which allows the specification and combination of uncertain data from a wide variety of information sources, including the ability to determine specific actions to lower the uncertainty of the likelihood of statements of interest.

Future work includes:

- the extension of the knowledge base to first order logic,
- a more in-depth demonstration of the argumentation capabilities of BRECCIA,
- the addition of other information services (e.g., the use of available 3D urban wind models),
- the inclusion of cost models to provide cost-benefit analysis for the user in making decisions,
- more advanced belief revision,
- the further improvement of the interaction between BRECCIA and the information sources, and
- field testing with UAV reconnaissance missions.

REFERENCES


