AKSED: adaptive knowledge-based system for event detection using collaborative unmanned aerial vehicles

X. Sean Wang[a], Byung Suk Lee[a], Firooz Sadjadi[b]

[a]University of Vermont, Burlington, VT, USA 95405
[b]Lockheed Martin Tactical Systems, Eagan, MN, USA 55121

ABSTRACT

Advances in sensor technology and image processing have made it possible to equip unmanned aerial vehicles (UAVs) with economical, high-resolution, energy-efficient sensors. Despite the improvements, current UAVs lack *autonomous* and c*ollaborative* operation capabilities, due to limited bandwidth and limited on-board image processing abilities. The situation, however, is changing. In the next generation of UAVs, much image processing can be carried out onboard and communication bandwidth problem will improve. More importantly, with more processing power, collaborative operations among a team of autonomous UAVs can provide more *intelligent event detection* capabilities. In this paper, we present ideas for developing a system enabling target recognitions by collaborative operations of autonomous UAVs. UAVs are configured in three stages: manufacturing, mission planning, and deployment. Different sets of information are needed at different stages, and the resulting outcome is an optimized event detection code deployed onto a UAV. The envisioned system architecture and the contemplated methodology, together with problems to be addressed, are presented.

Keywords: knowledge-based system, event detection, unmanned aerial vehicle

1. FUTURE UNMANNED AERIAL VEHICLES

Unmanned Aerial Vehicles (UAVs) are remotely piloted or self-piloted aircraft that can carry cameras, sensors, communications equipment or other payloads [7, 14]. In a recent Defense Science Board report [5], UAVs have been attributed to "provide vastly improved acquisition and more rapid dissemination of Intelligence, Surveillance and Reconnaissance (ISR) data".

During the past several decades, advances in sensor technology and image processing have made it possible to apply sensors to detect targets that are fixed or moving under a variety of environmental conditions. Moreover, as sensors are becoming miniaturized and their production costs reduced, it has become increasingly feasible to equip UAVs with economical, high-resolution, energy-efficient sensors [5].

Despite the continuous improvements, current UAVs lack *autonomous* and *collaborative* operation capabilities [5], due to limited bandwidth and limited on-board image processing abilities. The situation, however, is changing. In the next generation of UAVs, much image processing can be carried out onboard and communication bandwidth problem will improve. More importantly, with more processing power, collaborative operations among a team of autonomous UAVs can provide more *intelligent event detection* capabilities.

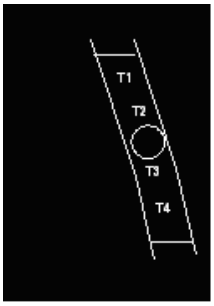
By an *event* we mean a complex situation that can be derived from the processing results (which we call *signatures*) extracted from images taken by UAVs. Probabilistic, contextual and other reasoning techniques will be necessary to derive an event from signatures. For example, a signature may be a soldier appearing in a scene, and an event may be an increasing number of soldiers appearing in the same scene – the staging of a troop. In another example, a signature may be a moving vehicle, and an event may be multiple vehicles moving in a row, with approximately equal distance among them – a convoy. The following example illustrates collaborative UAVs detecting complex events.

Example 1. (See Figure 1.) In this example, since UAVs equipped with only visual sensors are less expensive, many of them are deployed to scout an interested area; in contrast, since UAVs with infrared (IR) sensors are more expensive,

only a few of them are deployed to respond to requests only in special scenarios. Now, a UAV with visual sensors detects a possible convoy in the current image (see the subfigure a and the subfigure b; the image in b is the result of, for instance, contextual scene analysis of the image in a.) However, for conclusion with high enough detection confidence, it needs to see an additional convoy unit with certain statistical confidence in a particular location in the image. This missing unit may be present but obscured in the visual image by dust or fog. The UAV then communicates with a UAV with IR sensors, which then takes an IR image of the region of interest, to detect the missing unit in the convoy (see the subfigure c).

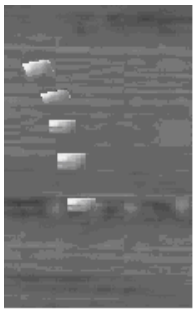
(a) Visual sensor image.

The middle convoy unit is obscured in the image.



(b) Image analysis result.

The rectangle is a potential convoy. T1, T2, T3, and T4 are potential vehicles. The circle is a possible undetected



(c) IR sensor image.

The middle convoy unit shows up in this image.

Figure 1:. Convoy detection example.

Another example of collaborative use of visual and IR imagery to detect complex events is shown in the following figure.

(a) Visible band image of a scene.

This image shows a truck. The dust cloud is hiding the airplane and people.

(b) Infrared image of same scene.

This image shows portions (high temperature areas) of a truck, an airplane and several people in the background.

Figure 2: Visual and IR images of a scene.

The above examples show a powerful and effective way of using collaborative UAVs to achieve effective event detection. Obviously, it is only one of many possible ways. Many different strategies can be designed for different types of events and for different situations. Thus, the following questions and observations arise naturally:

1. How do we configure UAVs so that it uses a particular strategy when detecting a particular type of event in a particular situation? It may not be realistic to build multiple strategy variations into all UAVs at manufacturing time so that we are ready for all foreseeable situations. Instead, certain strategies may need to be coded into UAVs at

deployment time, based on the situation at hand for a particular mission. In this case, we need an event detection coding environment that is flexible enough for mission-time strategy development. Furthermore, for each event definition, it will be desirable that the system automatically generates collaborative detection strategies (as in the convoy detection example) with minimum help from the users.

2. Suppose that, in addition to the existence of the Infrared (IR) UAV, there is also a Synthetic Aperture Radar (SAR) UAV available in the above convoy detection problem. How does the visual UAV decide whose help is more desirable between an IR UAV and a SAR UAV? How does the UAV reason about the situation and decide? What kind of "knowledge" does a UAV need and have in order to make such decisions? Even for a simple decision like whether to transmit certain messages back to the controller, due to the limited spectrum available in a particular theatre (and multiple UAVs in action) we will need to build into the UAV the necessary knowledge. How can we provide the UAV with a flexible system coded with such knowledge and reasoning rules?

2. ENVISIONED SYSTEM ARCHITECTURE

Inspired by the above questions and observations, we envision a system that is highly configurable for event-detection with future collaborative UAVs. We call the system an *adaptive knowledge-based system for even detection,* or *AKSED* (pronounced 'axed') for short. See Figure 3.

With AKSED, a UAV may be configured at *three* different stages. In general, at *manufacturing time*, UAV basic capabilities (<1> in Figure 3), such as feature extraction algorithms using signal and image processing techniques, communication capabilities, and maneuvering capabilities, are provided as software modules. These modules can be coded in efficient programming languages like C and FORTRAN. At *mission planning time*, event descriptions (<2> in Figure 3) are provided in a high-level declarative language (provided by AKSED). At *deployment time,* situation descriptions and quality-of-service (QoS) requirements (<3> in Figure 3) are given to the system through a checkbox and form-filling interface (provided by AKSED). Given all these inputs provided in different stages, AKSED generates deployment code with necessary knowledge bases for onboard reasoning. The knowledge bases are in the form of base facts and inference rules. This division of tasks into three stages allows coding and testing of difficult and detailed algorithms to be done at manufacturing time, and allows flexible event-detection strategies to be coded easily at mission planning and deployment time, when coding resources (e.g., time and expertise) may be lacking.

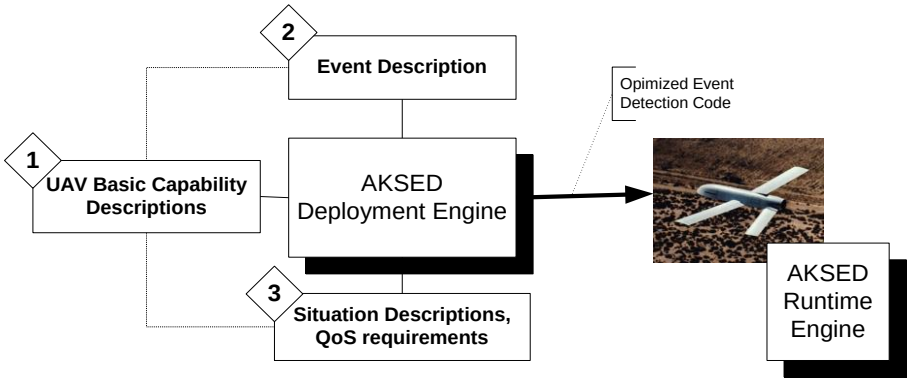


Figure 3. AKSED system for collaborative UAVs.

The AKSED runtime engine is responsible to execute the event detection code deployed on a UAV. The event detection code also contains necessary knowledge for the reasoning engine. Figure 4 shows its architecture. The runtime engine continuously monitors the external scene with sensors. The event detection manager controls the sensors and other components of the UAV. Feature extraction algorithms (e.g., image processing) will generate necessary features to be used by the reasoning engine. The reasoning engine is responsible to perform necessary probabilistic, contextual and other reasoning and QoS optimization (based on the event detection code). Possible output of the runtime system is a

detected event report (to the UAV controller) and some recommendation for collaborative UAVs. The runtime engine will also receive recommendations from other UAVs as input.

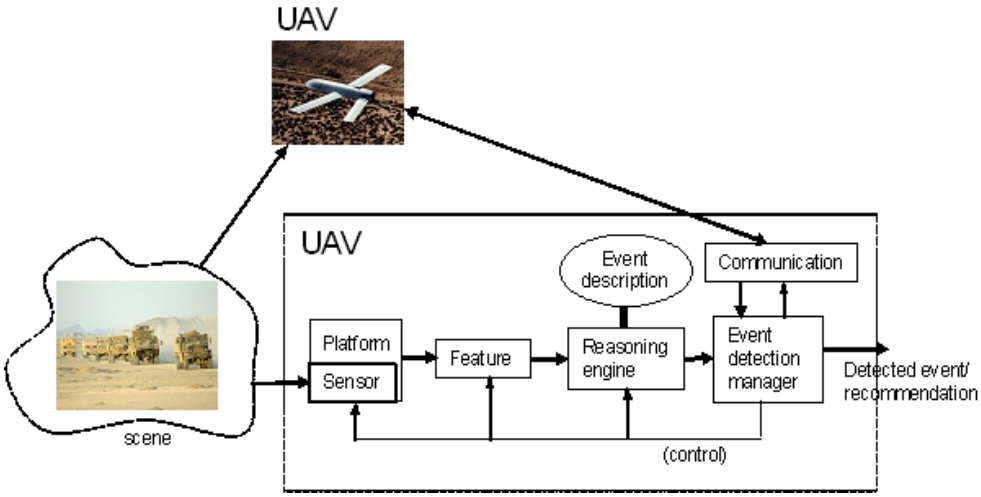


Figure 4:. AKSED runtime engine architecture.

3. PROJECT OBJECTIVES

The project ideas are being established at the present time. The objective is to develop an AKSED system with the above system architecture. Specifically, the project will involve the following steps to materialize AKSED.

1. Study and explore the language for UAV/sensor capability description. This language is to interface between the lower-level image processing and UAV operation code.
2. Study and explore the language for event description. This language will have the ability to incorporate probabilistic, contextual and other reasoning. The goal is to describe the event to be detected using the UAV/sensor capabilities (from 1 above) and its own reasoning capability (afforded by its logic language constructs).
3. Develop a deployment engine prototype.
4. Develop a runtime engine prototype.
5. Develop a mockup UAV simulation to demonstrate the system ability.

4. METHODOLOGY

The central component of AKSED is the *AKSED Deployment Engine* (See Figure 3). It takes the three kinds of descriptions (i.e., basic capabilities, event, situation and QoS) as inputs and produces optimized event detection code (for UAVs) as the output to be executed by the AKSED runtime engine. In this section we outline key methodologies needed in each part of the AKSED system. The concrete methodologies are subject to change as they mature, and are open to suggestions.

4.1 UAV basic capability descriptions

For UAV basic capability description, our methodology is to use *probabilistic predicates* in the following form:

$$p(x_1, \ldots, x_n): [v_1, v_2]$$

where p represents the capability name with parameters $x_1, \ldots, x_n$, and v_1 and v_2 are the minimum and maximum confidences (i.e., probabilities that the predicate is true) given the parameter values. For example, the predicate

$$VehicleInIRimage(x_1, y_1, x_2, y_2): [v_1, v_2]$$

means that the IR sensor recognizes a vehicle with a confidence of at least v_1 and at most v_2 in the rectangular region denoted by the two coordinates (x_1, y_1) and (x_2, y_2). In many cases, either $v_1=0$ or $v_2=1$. Specifically, for the $[v_1, 1]$ case, we have at least v_1 probability that the predicate is true; while for the $[0, v_2]$ case, we have at most $v2$ probability that the predicate is true. These predicates will be used in an event description (see Section 4.2 below) to describe complex events.

The above notation has its root in the probabilistic logic programming [6, 13]. Note that deterministic predicates are special cases of probabilistic predicates with $[1, 1]$ (i.e., True) or $[0, 0]$ (i.e., False).

There are two possible implementations of capability predicate evaluations: *push* and *pull.* The push method will let the sensors (and other UAV operations) generate data for the feature extractor to populate a database with values of predicates (including all the corresponding parameter values and confidence values); then, the runtime reasoning engine simply looks at (i.e., query) the database to deduce events using the rules given in the event description. In the pull method, the sensors (and other UAV operations) will accept commands from the reasoning engine to produce values of the corresponding predicates. In this pull mode, it is possible that some of the parameters and/or the confidence values are *partially* instantiated. In the convoy detection example, the visual sensors are in the push mode (since they are on board the UAV) while the IR sensor is in the pull mode (since it is on another UAV). In this case, the reasoning engine will send a command to the IR UAV to see if indeed it can determine a vehicle at the said location with certain confidence. The predicate sent to the IR UAV is partially instantiated with only the location information, and the IR UAV returns with the confidence instantiated. How to determine which UAV capability should be in a pull or push mode is a research problem, and should be related to the events to be detected.

4.2 Event descriptions

We will study the use of *probabilistic logic programming* [6, 13] combined with techniques from *Datalog with constraints* [10, 15] to represent complex events. Probabilistic logic programming provides the basis for probabilistic reasoning, while the Datalog with constraints provides the basic tools to handle spatiotemporal information. For example, the following rule describes the Convoy event:

(1) $Convoy(x, y, x', y'):[v_1, v_2] \leftarrow (Vehicle(x_1, y_1), \ldots, Vehicle(x_5, y_5)):[v_1, v_2], Spatial\_constraint$

In the above, *Spatial_constraint* is a formula that relates all the location coordinates such that the distance between each pair of consecutive vehicles is about the same, and (x, y, x', y') is the smallest rectangle region containing all the five vehicles. This rule reads that if we have the confidence of $[v_1, v_2]$ to have detected five vehicles (with said spatial relationship among them), then we have the same confidence of having detected a convoy. To connect the above rule to the UAV capabilities, we may set up the following rules:

(2) $Vehicle(x, y):[v, v'] \leftarrow VehicleInVisualImage(x_1, y_1, x_2, y_2):[v, v'], Spatial\_constr$

(3) $Vehicle(x, y):[v, v'] \leftarrow IRUAV(x_1, y_1, x_2, y_2):[v, v'], Spatial\_constr$

In the above, the *Spatial_constr* specifies that (x, y) is within the rectangular region indicated by the two coordinates (x_1, y_1) and (x_2, y_2).

A question arises on how to determine the confidence interval of recognizing a five-vehicle convoy (required by rule 1) from the confidence intervals of recognizing individual vehicles (rules 2 and 3). This is in the domain of probabilistic reasoning. Simple independence assumption (i.e., each vehicle appears independently) may not be a realistic assumption. Probabilistic logic programming [6, 13] provides some basic techniques for solving this problem. In this project, since we use a combined language of probabilistic logic programming and Datalog with constraints, we will need to study the syntax and semantics of the above rules, as well as strategies to compute combined probabilities. Computational complexity of this language (or its sublanguages) is also an important topic.

Note that the *IRUAV* predicate is not from sensors. Instead, it is from an underlying communication capability of the UAV. When rule 3 is invoked, the UAV is to communicate with an IR UAV to determine if there is a vehicle at certain

location with certain confidence. This is a quite costly operation and should be used judiciously. This is the topic of situation description and QoS requirements below.

4.3 Situation descriptions and QoS requirements

Situation description is expressed as metadata given to predicates, like the *evaluation cost* and *availability*. For example, we may specify the cost of *IRUAV* predicate (in rule 3) to be much more expensive than the *VehicleInVisualImage* predicate (in rule 2). This will guide the reasoning engine to evaluate the *VehicleInVisualImage* predicate before *IRUAV* predicate whenever it has a choice between the two. Another situation description may indicate that *IRUAV* predicate is not available at all; in this case, IR UAV is not an option when detecting the obscured vehicle. We will study situation descriptions needed for various event detection tasks.

QoS is also expressed as metadata specifying *user preferences* on the outcomes of event detection. An example is a preference between the speed of event detection and the precision of detected event. QoS requirements affect the generation of event detection code as well as its execution. We will study what kind of metadata and QoS requirement are necessary and sufficient for various event detection tasks.

4.4 AKSED deployment engine

For the output (detection code) generation, we will achieve code optimization through query *optimization* and *transformation* techniques, including *QoS based* query planning and transformation. The basic idea is to provide multiple execution possibilities of the same set of rules so that at runtime one is chosen based on the situation. This provides a clear advantage, namely, that the runtime engine does not have to perform the costly operations of choosing plans from vast possibilities (but only from the execution plans deployed into it) and that possibilities have already been explored to make sure that any execution plan used by the runtime engine will be a good one. The deployment engine also needs to take the resource availability of the UAV into consideration. For example, if IR UAV is not available, then rule 3 will be disabled.

Furthermore, there is a clear optimization problem of how to balance the complexity of the deployed code and the possibility of runtime optimization. On one hand, run time optimization can be done best when the actual data have become available but, in this case, the deployed code may have to be very complex and hence take longer to run. On the other hand, some runtime optimization opportunities may be lost if we do not have complete "intelligence" included in the deployed code. We will study this issue further in this project.

4.5 AKSED runtime engine

The *AKSED runtime engine* will be based on optimized evaluation strategies for constraint-based probabilistic Datalog programs, which is the output of the AKSED deployment engine. The runtime engine is also responsible to manage the limited memory and bandwidth based on the user QoS requirements. A challenging aspect of this part of the project is to deal with *data streams* [4]. Indeed, when a UAV capability predicate is in a push mode, the reasoning engine will receive a stream of facts (i.e., feature predicate values with probability intervals). Many techniques for data stream processing [1, 8, 11] will be applicable. In particular, we will be able to take advantage of certain continuity property of the data (e.g., same vehicle appearing at slightly different locations at consecutive times) in order to expedite the processing. In addition, *feature prediction* will be useful to provide some pre-processing before some specific features appear in images. Forecasting techniques [2, 3] and predictive data mining [9, 12] techniques will be useful for such feature prediction. These are all research problems we need to address in this project.

5. UNIQUENESS

To our knowledge, this is the first project for enhancing UAV operations to the next level. If successful, it will enable UAVs to cooperatively and autonomously detect a complex event in an adaptive and efficient way. We believe the three-stage (i.e., manufacturing, mission planning, and deployment) approach of this project is very useful. Indeed, in contrast to monolithic development efforts resulting in UAVs that are not easily customizable, this approach allows fast and

flexible event definition with a high-level declarative language using concepts closer to the mission at hand. This way, UAVs can be quickly adjusted based on the mission requirements to detect events that were not predicted at the manufacturing time. The use of QoS concept is also unique and beneficial since UAVs will need to decide how to balance different requirements autonomously.

6. CONCLUSIONS

In this paper, ideas for developing a system (called AKSED) for detecting events using UAVs have been presented. The envisioned UAVs operate autonomously and collaboratively in their reconnaissance mission. The operation is adaptive to the situations and the quality of service (QoS) requirements. The necessary intelligence (or knowledge) for these operations is instilled in the code loaded onto the runtime engine. The knowledge is tuned for specific mission the UAVs are launched to perform. The code to be loaded onto the runtime engine is generated by the deployment engine. This engine generates an optimal code made to adapt to various situations and QoS requirements that are expected during a specific mission. The system is characterized by its separation of UAV configurations into three stages of manufacturing, mission planning, and deployment. This separation facilitates efficient and economical development efforts. We believe this will be the first project aiming to develop a system supporting a collaborative UAV mission in this manner.

AKNOWLEDGMENTS

The efforts of the University of Vermont authors are partially supported by the US National Science Foundation through Grant No. NSF IIS-0415023.

REFERENCES

[1] S. Babu, J. Widom, "Continuous Queries over Data Streams," *SIGMOD Record*, vol. 30, pp. 109-120, 2001.
[2] G. R. Brown, *Smoothing, Forecasting, and Prediction*: Prentice-Hall, 1963.
[3] C. Chatfield, *Time-Series Forecasting*: Chapman and Hall/CRC, 2001.
[4] N. Chaudhuri, Kevin Shaw, Mahdi Abdelguerfi (editors), *Stream Data Management*: Springer, 2005.
[5] Defense Science Board, "Defense Science Board Study on Unmanned Aerial Vehicles and Uninhabited Combat Aerial Vehicles," 2004.
[6] A. Dekhtyar, V. S. Subrahmanian, "Hybrid Probabilistic Programs," *Journal of Logic Programming*, vol. 43, pp. 187-250, 2000.
[7] P. G. Fahlstrom, Thomas J. Gleason, *Introduction to UAV Systems*, 2 ed. Columbia, MD: UAV Systems, Inc., 1998.
[8] M. E. Garofalakis, Johannes Gehrke (Editor), Rajeev Rastogi (Editor), *Data Stream Management: Processing High-Speed Data Streams (Data-Centric Systems and Applications)*, 1st ed: Springer, 2006.
[9] J. a. M. K. Han, *Data Mining: Concepts and Techniques*: Morgan Kaufmann, 2001.
[10] P. C. Kanellakis, Gabriel M. Kuper, Peter Z. Revesz, "Constraint Query Languages," *Journal of Computer & Systems Sciences*, vol. 51, pp. 26-52, 1995.
[11] S. Madden, M.J. Franklin, "Fjording the Stream: An Architecture for Queries over Streaming Sensor Data," presented at the International Conference on Data Engineering, 2002.
[12] T. Mitchell, *Machine Learning*: Mc-Graw Hill, 1997.
[13] R. Ng, and V. S. Subrahmanian, "Probabilistic logic programming," *Information and Computation*, vol. 101, pp. 150-201, 1992.
[14] J. Pike, "Unmanned Aerial Vehicles (UAVs)," http://www.fas.org/irp/program/collect/uav.htm, 2005.
[15] P. Z. Revesz, "Constraint Databases: A Survey," in *Semantics in Databases*, 1995, pp. 209-246.