Prospective Architectures for Onboard vs Cloud-based Decision Making for Unmanned Aerial Systems

Shankar Sankararaman[1], and Christopher Teubert[2]

[1] *SGT Inc., NASA Ames Research Center, Moffett Field, CA, 94035, USA*
shankar.sankararaman@nasa.gov

[2] *NASA Ames Research Center, Moffett Field, CA, 94035, USA*
christopher.a.teubert@nasa.gov

ABSTRACT

This paper investigates propsective architectures for decision-making in unmanned aerial systems. When these unmanned vehicles operate in urban environments, there are several sources of uncertainty that affect their behavior, and decision-making algorithms need to be robust to account for these different sources of uncertainty. It is important to account for several risk-factors that affect the flight of these unmanned systems, and facilitate decision-making by taking into consideration these various risk-factors. In addition, there are several technical challenges related to autonomous flight of unmanned aerial systems; these challenges include sensing, obstacle detection, path planning and navigation, trajectory generation and selection, etc. Many of these activities require significant computational power and in many situations, all of these activities need to be performed in real-time. In order to efficiently integrate these activities, it is important to develop a systematic architecture that can facilitate real-time decision-making. Four prospective architectures are discussed in this paper; on one end of the spectrum, the first architecture considers all activities/computations being performed onboard the vehicle whereas on the other end of the spectrum, the fourth and final architecture considers all activities/computations being performed in the cloud, using a new service known as "Prognostics as a Service" that is being developed at NASA Ames Research Center. The four different architectures are compared, their advantages and disadvantages are explained and conclusions are presented.

1. INTRODUCTION

The importance of unmanned aerial vehicles and systems has steadily increased in the past ten to fifteen years. Both the Federal Aviation Administration (FAA) and the National Aeronautics and Space Administration (NASA) have shown significant interest in the development of technologies for unmanned aerial vehicles and unmanned traffic management systems.

It is expected that there will be a significant increase in unmanned aerial traffic and therefore, the overall safety of the United States National Airspace System needs to analyzed carefully. An Unmanned Aerial System (UAS) will have access to civilian air space only when the safety of the airspace, property, and the vehicle itself can be guVaranteed. Further, traffic management becomes increasingly complicated in urban environments where low altitude flight control and safety is of critical importance.

Several researchers have been focusing the development of various technologies that would ultimately enable the systematic inclusion of unmanned systems into the airspace (McAree & Chen, 2013). These technologies include sensing and data logging (Berni, Zarco-Tejada, Suárez, & Fereres, 2009; Corke et al., 2004; Everaerts et al., 2008; Sharp, Shakernia, & Sastry, 2001; Shakernia, Vidal, Sharp, Ma, & Sastry, 2002), fault tolerant flight control (Coombes, McAree, Chen, & Render, 2012; Ducard, 2009; B. S. Kim & Calise, 1997; H. J. Kim, Shim, & Sastry, 2002; Stepanyan & Krishnakumar, 2017), simultaneous localization and mapping (SLAM) (Hening, Ippolito, Krishnakumar, Stepanyan, & Teodorescu, 2017; Caballero, Merino, Ferruz, & Ollero, 2009; Wang et al., 2008; Gupte, Mohandas, & Conrad, 2012; J.-H. Kim & Sukkarieh, 2003), obstacle detection and avoidance (Sinopoli, Micheli, Donato, & Koo, 2001; Shim, Chung, & Sastry, 2006), optimal power management (K. Kim, Kim, Lee, & Kwon, 2011; Marsh et al., 2001; Saha et al., 2011), path planning and trajectory design (D'Souza, 2017; Yang & Kapila, 2002; Jun & DAndrea, 2003; Rysdyk, 2006; Sigurd & How, 2003), searching and tracking (Chakrabarty, Morris, Bouyssounouse, & Hunt, 2017), autonomous decision-making (Ollero & Maza, 2007;

Ruff, Narayanan, & Draper, 2002), unmanned traffic management (Jang, Ippolito, Sankararaman, & Stepanyan, 2017), etc. In addition, it is also important to study the overall design (Cai, Feng, Chen, & Lee, 2008) that can systematically integrate the aforementioned technologies.

According to Koperdaker (Kopardekar, 2014), the near-term goal (1-5 years) is to safely enable low-altitude airspace and UAS operations while the long-term goal (10-15 years) is to safely enable massive increases in airspace density and UAS operations. These goals are particularly challenging because there is a significant amount of uncertainty and numerous factors that are constantly and dynamically evolving in urban environments (Krishnakumar et al., 2017). As a result, developing a methodology for safe, autonomous decision-making is a challenging problem.

Sankararaman and Kalmanje (Sankararaman & Kalmanje, 2017) presented a probabalistic framework for decision-making under uncertainty in order to aid autonomous operation of sUAS. This framework focuses on the identification, assessment, and prediction of various risk-factors (such as obstacle collision, untimely battery drain, etc.) that affect the operation of sUAS. Then, it uses probabilistic methods for estimating the likelihood of occurrence of risk-factors and aids decision-making under uncertainty. While the framework was presented in a mathematical manner, it is still necessary to develop a software architecture that can systematically implement the proposed decision-making framework.

This paper investigates propsective architectures for enabling real-time decision-making for unmanned aerial systems. The first architecture supports all computations on board, it may be time-consuming due to lack of sufficient onboard hardware that support computing. The second architecture is considered from the point of view of cloud-based computing where, time-consuming computations can be outsourced from the vehicle to the cloud. The advatnages and the disadvantages of both of these architectures are discussed in detail

The rest of this paper is organized as follows. Section 2 discusses the various risk-factors in flight, and explains how to model such risk-factors in order to aid decision-making. Prediction of risk-factors is an important component of decision-making; such prediction can either be performed onboard (for instance, using the Generic Software Architecture for Prognostics, that is developed and open-sourced by NASA Ames Research Center) or on the cloud (using "Prognostics as a Service", i.e., Paas, being developed at NASA Ames Research Center). Section 3 uses the prediction of risk-factors to enable decision-making and summarizes the decision-making framework developed by Sankararaman and Krishnakumar (Sankararaman & Kalmanje, 2017). Section 4 discusses multile architectures that can implement the aforementioned decision-making framework, and discusses the advantages and disadvantages of each of these architectures. Fi-

nally, Section 5 concludes the paper and discusses possible directions for further research.

2. RISK FACTORS IN FLIGHT AND MODELING

There are several risk-factors that affect the flight of unmanned systems. Decision-making needs to take into account the occurrence of risk-factors, compute the likelihood of such occurrence in the future, and proactively make plans at the time of decision-making. For instance, one risk-factor could refer to complete discharging of the battery prior to the completion of the trajectory; another risk-factor could refer to the impending collision of the unmanned system against a static/dynamic obstacle.

Consider a given trajectory and a generic time of prediction t_P at which it is necessary to calculate the likelihood of a particular risk-factor continuously as a function of future time ($\forall \ t > t_P$). In order to achieve this goal, it is necessary to model the evolution of the UAS continuously as a function of time along with the evolution of external factors related to the risk-factor. For instance, in the case of a collision against a dynamic obstacle, it may be necessary to model the evolution of the position of the UAS continuously as a function of time (based on the planned trajectory), and the anticipated position of the dynamic obstacle (which is typically uncertain if the trajectory of the obstacle is unknown and can only be approximately quantified based on its position and velocity as estimated by the sensors on the UAS).

2.1. Modeling the Evolution of State With Respect to the Risk-Factor

Consider the state space model which is used to continuously predict the state of the system, as:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(t, \mathbf{x}(t), \boldsymbol{\theta}(t), \mathbf{u}(t), \mathbf{v}(t)) \tag{1}$$

where $\mathbf{x}(t) \in \mathbb{R}^{n_x}$ is the state vector, $\boldsymbol{\theta}(t) \in \mathbb{R}^{n_\theta}$ is the parameter vector, $\mathbf{u}(t) \in \mathbb{R}^{n_u}$ is the input vector, $\mathbf{v}(t) \in \mathbb{R}^{n_v}$ is the process noise vector, $\mathbf{f}$ is the state equation, and t is the continuous time variable. Note that the above state vector is not necessarily equal to the aerodynamic state of the UAS (measured in terms of position, attitude, etc.); instead, this state vector is directly related to the risk-factor under consideration. If collision against a dynamic obstacle is a risk-factor, then this state vector contains the position of the UAS. On the other hand, if battery-charge draining is a risk-factor, then this state vector contains the charge of the battery of the UAS. Note that all the quantities in Eq. 1 are uncertain in nature and need to be treated probabilistically (Sankararaman, 2015).

The state vector at time t_P, i.e., $\mathbf{x}(t)$ (and the parameters $\boldsymbol{\theta}(t)$, if they are unknown) is (are) estimated using output data collected until t_P. Let $\mathbf{y}(t) \in \mathbb{R}^{n_y}$, $\mathbf{n}(t) \in \mathbb{R}^{n_n}$, and $\mathbf{h}$ de-

note the output vector, measurement noise vector, and output equation respectively. Then,

$$\mathbf{y}(t) = \mathbf{h}(t, \mathbf{x}(t), \boldsymbol{\theta}(t), \mathbf{u}(t), \mathbf{n}(t)) \qquad (2)$$

Typically, filtering approaches such as Kalman filtering, particle filtering, etc. may be used for such state estimation (Sankararaman, Daigle, & Goebel, 2014).

Having estimated the state at time t_P, Eq. 1 is used to predict the future states of the component/system. This differential equation can be discretized and used to predict $\mathbf{x}(t)$ for all $t > t_P$.

2.2. Modeling the Risk-Factor

Risk-Factors can be expressed in terms of a binary constraint function $c^H(\mathbf{x}(t), \boldsymbol{\theta}(t), \mathbf{u}(t)) = 1$ that maps a given point in the joint state-parameter space given the current inputs, $(\mathbf{x}(t), \boldsymbol{\theta}(t), \mathbf{u}(t))$, to the Boolean domain $\mathbb{B} \triangleq [0, 1]$. Without loss of generality, $c^H(\mathbf{x}(t), \boldsymbol{\theta}(t), \mathbf{u}(t))$ can be written as $c^H(t)$; $c^H(t) = 1$ implies that the risk-factor is encountered at time t whereas $c^H(t) = 0$ implies that the risk-factor is not encountered at time t.

At any generic time of prediction t_P, note that the constraint function $c^H(t)$ associated with each risk-factor is a function of t. Therefore, the approach needs to forecast all available information until future time t in order to predict the occurrence of the risk-factor. Thus, it needs all information (states, parameters, and inputs in Eq. 1) between time t_P and t.

2.3. Likelihood of Risk-Factor and Prediction of Time of Occurrence

Typically, there are two quantities of interest, in the context of risk-factor prediction:

1. Time of Occurrence: At any time of prediction t_P, it is useful to know the future time at which the risk-factor will be encountered. Let $T^H(t_P)$ denote this quantity. This information can be helpful in determining the amount of time remaining so that corrective action may be taken. However, due to the uncertainties involved, this quantity is a probability distribution.

2. Likelihood of Occurrence of the Risk-Factor as a function of time: At any time of prediction t_P, it is also useful to know the likelihood of the occurence of risk-factor as a function of future time. This likelihood is denoted as $P_t^H(t_P)$; note that this is a trajectory as a function of future time t and changes with the time of prediction t_P.

First, at any t_P, the time of occurrence of a risk-factor (that is, the future time at which the risk-factor will be encountered) can be written as:

$$T^H(t_P) \triangleq \inf\{t \in \mathbb{R} : t \geq t_P \wedge c^H(t) = 1\}. \qquad (3)$$

It can be easily seen that $T^H(t_P)$ depends on the state at time of prediction, future inputs/parameters, etc., which are uncertain in nature; in order to calculate the probability distribution of $T^H(t_P)$, it is necessary to systematically propagate the aforementioned uncertain quantities and quantify their effect on the probability distribution of $T^H(t_P)$. Second, $P_t^H(t_P)$, i.e., the likelihood of the risk-factor at future time t (predicted at time t_P) can be expressed as $P(c^H(t) = 1)$.

The computation of both the probability distribution of $T^H(t_P)$ and the probability $P(c^H(t) = 1)$ can be accomplished using Monte Carlo sampling-based techniques, analytical techniques based on first-order and second-order reliability methods, or hybrid methods involving machine learning approaches (Haldar & Mahadevan, 2000; Sankararaman, 2015).

While onboard computational resources can be used to compute the aforementioned two quantities, it is also possible to send the relevant information to the cloud which hosts PaaS (Prognostics as a Service), to perform future predictions and perform these computations. Each of these approaches have their own advantages, and disadvantages. These issues will be discussed in detail in the following section, by analyzing multiple computational/software architectures

3. DECISION-MAKING FRAMEWORK

This section summarizes the decision-making framework presented earlier by Sankararaman and Krishnakumar (Sankararaman & Kalmanje, 2017).

3.1. Goal of Decision-Making: Trajectory Selection

An ideal decision-making algorithm should autonomously work in conjunction with the path planner (that generates trajectories) to identify whether a given trajectory is safe or not. In order to achieve this goal, the decision-making algorithm leverages information available from various sources as shown in Fig. 1, and identifies safe trajectories for real-time flight.

As seen from Fig. 1 (Krishnakumar et al., 2017), a trajectory is classified as follows:

1. Safe

   (a) Nominally safe: The likelihoods of risk-factors are extremely low

   (b) Off-nominal but safe: The likelihood of risk-factors are higher than the nominal scenario, but still low enough to be considered safe

2. Unsafe: The likelihood of risk-factors are considerably high.

The limits for likelihood demarcating (1) the nominally safe scenario and the off-nominally safe scenario; and (2) the safe
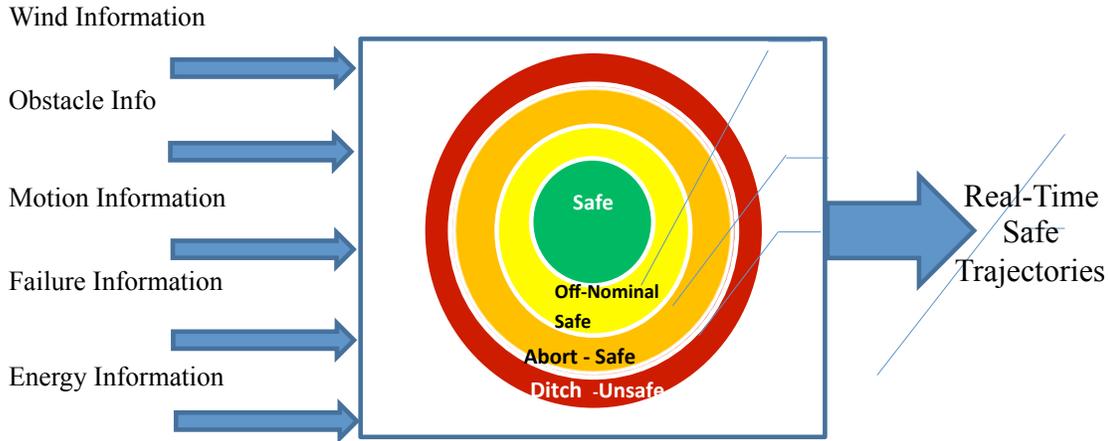
Figure 1. Goal of Decision-Making: Identify Safe Trajectory

and unsafe scenarios need to be assigned based on computing the costs/risk associated with each risk-factor.

If a trajectory is unsafe, then it is necessary to identify whether it is possible to generate a trajectory that can:

1. Abort the mission and return the UAS safely to a landing site; (or)

2. Abort the mission and ditch the UAS without any loss of private and/or public property.

These four different types of trajectories, i.e., nominally safe, off-nominal but safe, abort and return to base, and abort and ditch, are identified in Fig. 1. Note that the scope of this paper is limited to identifying whether a given trajectory is safe or not; further classification and aspects of decision-making will be considered in future work.

3.2. Risk-Factors

While there are different types of risk-factors that are associated with flights in urban environments, they can be broadly classfied into two categories, as shown in Fig. 2 (Krishnakumar et al., 2017).

As seen from Fig. 2, risk-factors may arise simply out of uncertainties (inherent variability, lack of information, etc. due to GPS Denied, degraded sensors, dynamic obstacles, etc.) or due to vehicular performance constraints (such as rapidly draining battery, lack of control, etc.) The decision-making system needs to assess all risk-factors as far as possible, assimilate information from the sensors, and select trajectories. Note that the decision-making is both risk-informed (since it calculates the likelihood of risk-factors along with the associated risk) and safety-assured (selects only those trajectories that are considered "safe", i.e., the likelihood of a risk-factor is far below a critical limit and hence the operation is considered safe).

3.3. Decision-Making through Information Fusion

Given a trajectory, and a risk-factor, how should the determine whether the trajectory is safe? Modern reliability analysis (Hohenbichler & Rackwitz, 1983) defines safety using the so-called limit state function, i.e., a curve of demarcation between a predefined "safe region" and an "unsafe region".

In simple scenarios, the idea of the limit state can be viewed in terms of capabilities (C) and requirements (R). When capabilities of a system are more than its requirements, then the system is said to be safe; otherwise, the system is considered to be unsafe. The limit state is then represented by the equation that implies capabilities are equal to requirements $C - R = 0$.

In more realistic scenarios, the limit state can be represented as a generic function $G(\boldsymbol{X}) = 0$, where $\boldsymbol{X}$ represents the vector of quantities that affect the limit state. In the context of this paper, $\boldsymbol{X}$ may potentially include (depending on the risk-factor under consideration) wind information, obstacle information, vehicular information (including motion, dynamics, and properties), energy information, and trajectory information, as shown in Fig. 3. Without loss of generality, the region represented by the curve $G(\boldsymbol{X}) > 0$ can be assumed to be the safe region, and the region represented by the curve $G(\boldsymbol{X}) < 0$ can be assumed to the be the unsafe region.

As mentioned earlier in Section 1, it is likely that elements contained in the vector $\boldsymbol{X}$ are all uncertain quantities and hence, these are represented as probability distributions in Fig. 3. It is therefore necessary to compute the probability $(P(G) < 0)$, and this probability corresponds to the likelihood of the risk-factor under consideration. It is important to compute this likelihood continuously as a function of future time (starting with the time of prediction) until the end of the
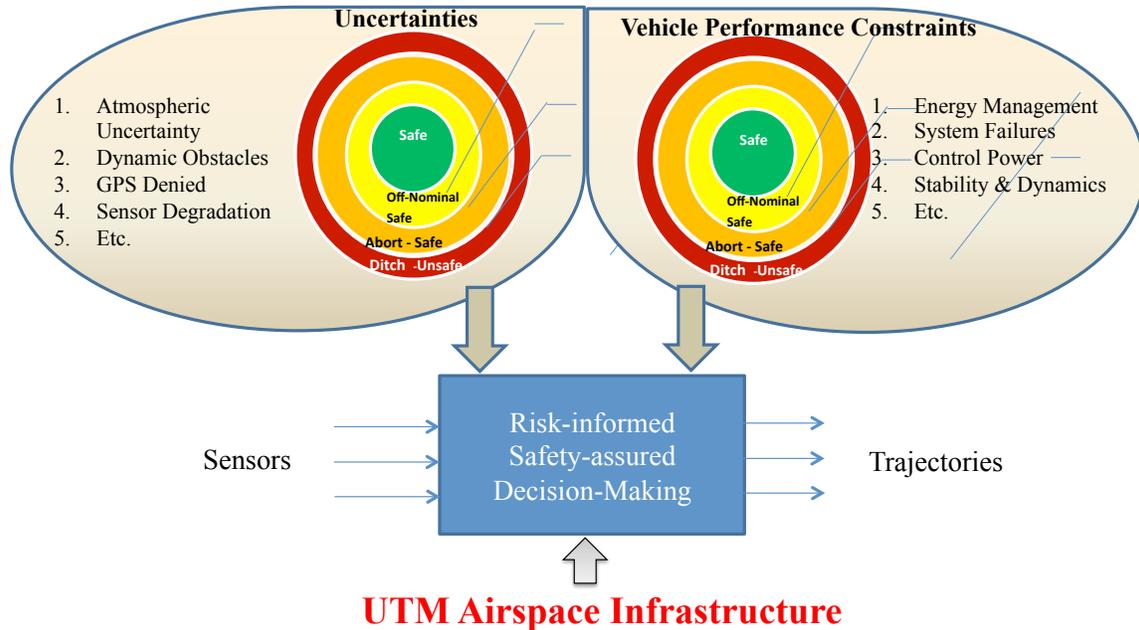
Figure 2. Risk-Factors in Low-Altitude Urban Environment Flight

trajectory under consideration. This computation is discussed in detail in the following section.

4. ARCHITECTURES FOR DECISION-MAKING

There are three primary functions in a prognostics-enabled decision making architecture: risk factor prediction (prognostics), a decision maker, and a trajectory generator. These units are described in greater detail in the previous section.

Each of the three primary functions in prognostics-enabled decision making can be either hosted on-board or remotely. On-board hosting reduces risk by eliminating dependency on a communication method, but it requires user to host a computer capable of performing complex prognostics and prognostic decision making activities. This might not be possible because of size, weight, or power constraints (SWaP). Remote prognostics allows users to share resources efficiently, have access to greater computational resources, and upgrade capabilities easily, but communication between the vehicle and the remote server requires time and energy.

Architectures exist for performing the risk factor prediction functions on-line or remotely. The Generic Software Architecture for Prognostics (GSAP) is often used to create an on-board prognostics application. GSAP is a general, object-oriented, cross-platform software framework and support library for prognostics technologies. GSAP implements many of the functions and algorithms used in prognostics as part of a prognostics support library. The GSAP framework implements and enforces the prognostic process. A standard inter-

face is supplied for integrating new models and algorithms, and for integrating the system into data sources (sensors) and sinks (displays, decision support tools, etc.). Users are then able to create a prognostic application by integrating their algorithms, models, and interfaces to their systems into the GSAP framework, with possible integration onboard or off-board a vehicle or other asset.

At the highest level, GSAP consists of two parts: The Prognostics Framework and the Prognostics Support Library. These are compiled with any user layer contributions to build a prognostics application, as shown in Fig. 4. In this case the user-layer includes *prognosers* for the risk factor prediction.

Alternatively the Prognostics As-A-Service (PaaS) architecture could be used to support remote prognostics. PaaS is a GSAP-enabled Prognostics Application (See Fig. 4) for performing prognostics remotely, as-a-service. A RESTful API is used to communicate with the Prognostics Application.

The efficiency of each option is explored in the following sections.

4.1. Assessment of Architecture Efficiency

The cost of transferring between on-board and remote functions can be represented by Eq. 4. Here L is average latency, B is bandwidth, and b is the amount of bits transferred. Here, average latency is defined as the time it takes one bit to travel through the network. This is equal to the propagation delay (time to move through the physical layer) plus the queuing
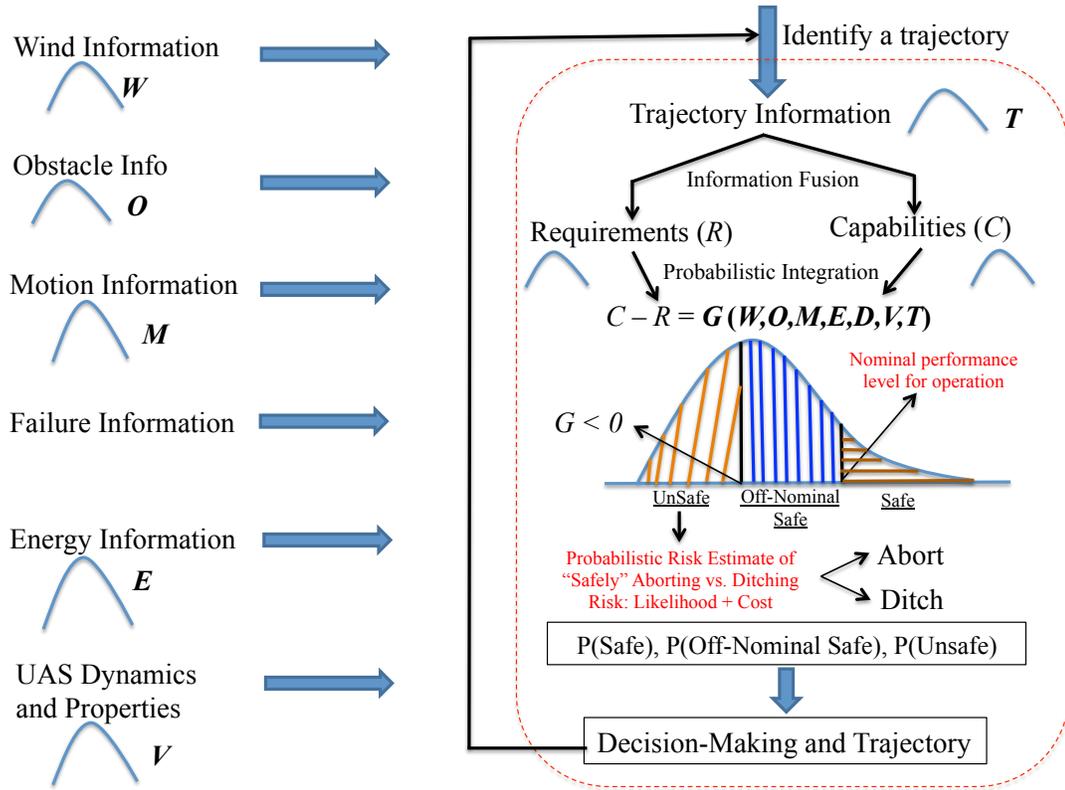
Figure 3. Framework for Decision-Making
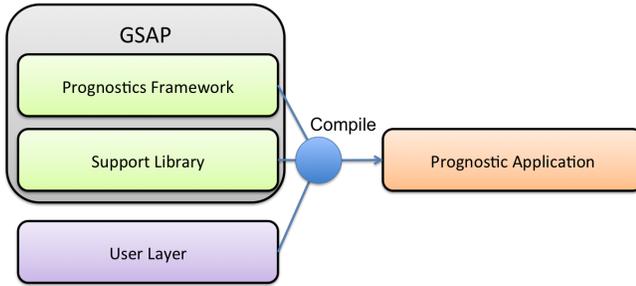
Figure 4. Using GSAP

delay (delay in routers due to queuing) plus the processing delay (time to process the packet). Note that for most of the team's past applications an average latency of around 0.1 seconds could be expected. Part of this was due to the complexity of the connection. Some applications will have less.

$$t_{net} = n(L + b/B) \qquad (4)$$

Using this equation, the total step time is represented in Eq. 5. Here n is the number of times data is sent back and forth. Note that this is true because none of the actions can be done in parallel. That is, the output of path generation depends on the output of decision making, which depends on the output of prognostics.

$$t_t = t_{net} + t_p + t_{dm} + t_{tg} \qquad (5)$$

All the variables used in these equations are described below:

t_t Total time to complete one step of UAS decision making

t_p Time to perform risk factor prediction

t_{dm} Time to perform decision making

t_g Time to perform trajectory generation

t_{net} Network cost: the time required for data transmission over the network

L Latency: time it takes one bit to travel through the network

B Bandwidth: Capacity of the network (in bits per second, bps)

b Bits transferred (data plus headers)

n Number of times that transfer must be made

The below sections outline the advantages and disadvantages of the four architectures considered.

4.1.1. Architecture 1: All On-Board

The first architecture option under consideration is to host all three primary functions on-board the vehicle. This architecture is illustrated in Fig. 5.
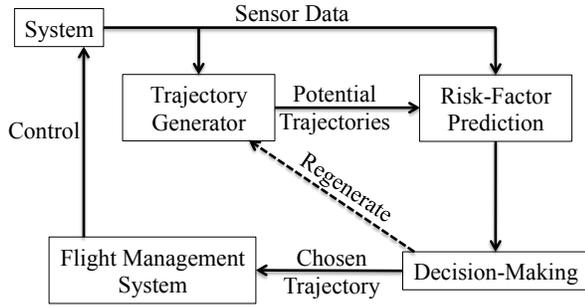
Figure 5. On-Board Architecture for Decision-Making

The advantages and disadvantages of this architecture are described below:

1. **Advantages**
   (a) Less risk through reduced dependence on communications
   (b) Reduced the communication requirements (bandwidth, etc). This reduces the mass and cost of communication equipment.
   (c) Reduced communication time between the three functions
2. **Disadvantages**
   (a) Need to host computer capable of performing calculations on-board
   (b) Increased difficulty in upgrading components

A major consideration in total step time (t_t). For this example there is no network delay (t_{net}), so the step time can be considered just the processing time, shown in Eq. 6. This is considered the baseline for this activity.

$$t_t = t_p + t_{dm} + t_{tg} \tag{6}$$

4.1.2. Architecture 2: Remote Prognostics

The second architecture option under consideration is to host risk-factor prediction remotely using PaaS, while continuing to host trajectory generation and decision making on-board. This architecture is illustrated in Figure 6.

The advantages and disadvantages of this architecture are described below:
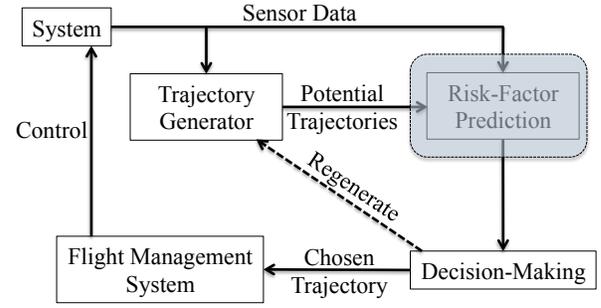
Figure 6. Cloud-based Architecture for DM using PaaS

1. **Advantages**
   (a) Partially hosted remotely: This allows resource sharing, and access to more powerful machines
   (b) Somewhat reduced weight, size, and power (SWaP) of on-board computer
2. **Disadvantages**
   (a) Data has to be transmitted every iteration of the Prediction, Decision Making, and Trajectory Generation.
   (b) Network needs: Need to host a network hardware capable of robustly handling communications
   (c) Increased risk: Decision making now becomes dependent on the network connection

For this architecture network delay is significant. It has three parts, as seen in Eq. 7. The three parts are time to communicate sensor information to PaaS ($t_{S \rightarrow P}$), time to communicate prognostic results to the decision maker ($t_{P \rightarrow D}$), and time to communicate trajectories to test to PaaS ($t_{G \rightarrow P}$). The number of iterations of the prediction, decision making, trajectory generation process is signified by n_{cyc}, which can be one in the case of an ideal trajectory generator. The complete step-time can be seen in Eq. 8. Here t'_p is the computation time for running risk factor prediction remotely, which will likely be less than that of running risk factor prediction on-board ($t'_p < t_p$).

$$t_{net} = t_{S \rightarrow P} + n_{cyc} * (t_{P \rightarrow D} + t_{G \rightarrow P}) \tag{7}$$

$$t_t = t_{net} + t'_p + t_{dm} + t_{tg} \tag{8}$$

4.1.3. Architecture 3: Remote Prognostics and Path Generation

The third architecture option under consideration is to host risk-factor prediction (using PaaS) and path generation remotely, while continuing to host decision making on-board. This architecture is illustrated in Fig. 7.
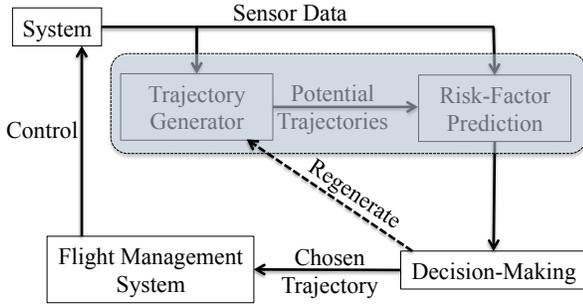
Figure 7. Cloud-based Architecture for DM using PaaS

Figure 8. Cloud-based Architecture for DM using PaaS

The advantages and disadvantages of this architecture are described below:

1. **Advantages**
   (a) More hosted remotely: This allows resource sharing, and access to more powerful machines
   (b) More reduced weight, size, and power (SWaP) of on-board computer
2. **Disadvantages**
   (a) Data has to be transmitted every iteration of the Prediction, Decision Making, and Trajectory Generation.
   (b) Network needs: Need to host a network hardware capable of robustly handling communications
   (c) Increased risk: Decision making now becomes dependent on the network connection

For this architecture network delay is significant. Network delay has three parts, as seen in Equation 9. The three parts are time to communicate sensor information to PaaS ($t_{S \to P}$), time to communicate prognostic results to the decision maker ($t_{P \to D}$), and time to decision making instructions to the path generator ($t_{D \to G}$). The number of iterations of the prediction, decision making, trajectory generation process is signified by n_{cyc}. The complete step-time can be seen in Equation 9. Here t'_{tg} is the computation time for running trajectory generation remotely, which will likely be less than that of running trajectory generation on-board ($t'_{tg} < t_{tg}$).

$$t_{net} = t_{S \to P} + n_{loops}(t_{P \to D} + t_{D \to G}) \qquad (9)$$

$$t_t = t_{net} + t'_p + t_{dm} + t'_{tg} \qquad (10)$$

4.1.4. Architecture 4: All Remote

The final architecture option under consideration is to host all critical funcitons remotely. This architecture is illustrated in Fig. 8.

The advantages and disadvantages of this architecture are described below:

1. **Advantages**
   (a) Low network costs: Only have to transmit twice: once to receive the sensor data, a second time to send the results of decision making
   (b) Most hosted remotely: This allows resource sharing, and access to more powerful machines
   (c) Most reduced weight, size, and power (SWaP) of on-board computer
2. **Disadvantages**
   (a) Network needs: Need to host a network hardware capable of robustly handling communications
   (b) Increased risk: Decision making now becomes dependent on the network connection

For this architecture network delay is much lower than the architectures 1 and 2. Network delay has two parts, as seen in Eq. 11. The three parts are time to communicate sensor information to PaaS ($t_{S \to P}$) and time to decision making results to the aircraft ($t_{D \to F}$). Note that this transfer of information is only done once per iteration, elimination the n_{cyc} term. The complete step-time can be seen in Eq 11. Here t'_{dm} is the computation time for running decision making remotely, which will likely be less than that of running decision making on-board ($t'_{dm} < t_{dm}$).

$$t_{net} = t_{S \to P} + t_{D \to F} \qquad (11)$$

$$t_t = t_{net} + t'_p + t'_{dm} + t'_{tg} \qquad (12)$$

5. CONCLUSION AND FUTURE WORK

The decision of architectures to choose for UAS decision making is highly dependent on a) the ability of the vehicle to host computers capable of performing these computations,

b) the quality (reliability, bandwidth, latency) of communications, and c) risk tolerance of the mission.

In all cases there are advantages to hosting operations remotely. These include: shared resources (one server can host capabilities for multiple vehicles), reduced on-board hosting requirements, simplified updating, and increased capabilities. There are also disadvantages, such as increased network requirements, increased program risk due to reliance on networking, and, sometimes, increased step time when network times (t_{net}) are greater than function step duration improvements ($t - t'$) from running remotely.

For example: For a small Unmanned Aerial Systems (sUAS), hosting computers capable of accurately and precisely performing this kind of decision making technologies may not be feasible. These systems might be forced to use completely remote hosting. On the other end of the spectrum, critical UAS may not tolerate relying on network connections for decision making. These vehicles may decide to host on-board, unless they can establish a very reliable connection.

One thing that is clear is that splitting the core functionality of UAS decision making between on-board and remote operations will result in severe network delays, due to the number of iterations (n_{cyc}) of the predict, decision make, trajectory generation loop performed. For that reason, it seems likely that systems that require quick decision making should either host all three functions on-board or remotely, not split them.

The next steps for this research include benchmarking algorithm performance on different class computers and network performance to provide numbers for a few examples of these trade-offs. Future work could also include exploring reduced-operation decision making client hosted on-board that can take over in the case of a network failure in a completely-remote architecture. Finally, future work includes testing and benchmarking of the PaaS architecture with a real UAS system to discover other design concerns.

REFERENCES

Berni, J. A., Zarco-Tejada, P. J., Suárez, L., & Fereres, E. (2009). Thermal and narrowband multispectral remote sensing for vegetation monitoring from an unmanned aerial vehicle. *Geoscience and Remote Sensing, IEEE Transactions on*, *47*(3), 722–738.

Caballero, F., Merino, L., Ferruz, J., & Ollero, A. (2009). Unmanned aerial vehicle localization based on monocular vision and online mosaicking. *Journal of Intelligent and Robotic Systems*, *55*(4-5), 323–343.

Cai, G., Feng, L., Chen, B. M., & Lee, T. H. (2008). Systematic design methodology and construction of UAV helicopters. *Mechatronics*, *18*(10), 545–558.

Chakrabarty, A., Morris, R. A., Bouyssounouse, X., & Hunt, R. (2017). An integrated system for autonomous search and track with a small unmanned aerial vehicle. In *AIAA Information Systems-AIAA Infotech@ Aerospace* (p. 0671).

Coombes, M., McAree, O., Chen, W.-H., & Render, P. (2012). Development of an autopilot system for rapid prototyping of high level control algorithms. In *Control (CONTROL), 2012 UKACC International Conference on* (pp. 292–297).

Corke, P., Hrabar, S., Peterson, R., Rus, D., Saripalli, S., & Sukhatme, G. (2004). Autonomous deployment and repair of a sensor network using an unmanned aerial vehicle. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on* (Vol. 4, pp. 3602–3608).

D'Souza, S. N. (2017). Developing a Generalized Trajectory Modeling Framework for Small UAS Performance in the Presence of Wind. In *AIAA Information Systems-AIAA Infotech@ Aerospace* (p. 0447).

Ducard, G. J. (2009). *Fault-tolerant flight control and guidance systems: Practical methods for small unmanned aerial vehicles*. Springer Science & Business Media.

Everaerts, J., et al. (2008). The use of unmanned aerial vehicles (UAVs) for remote sensing and mapping. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *37*, 1187–1192.

Gupte, S., Mohandas, P. I. T., & Conrad, J. M. (2012). A survey of quadrotor unmanned aerial vehicles. In *Southeastcon, 2012 proceedings of ieee* (pp. 1–6).

Haldar, A., & Mahadevan, S. (2000). *Probability, reliability, and statistical methods in engineering design*. John Wiley.

Hening, S., Ippolito, C. A., Krishnakumar, K. S., Stepanyan, V., & Teodorescu, M. (2017). 3D LiDAR SLAM Integration with GPS/INS for UAVs in Urban GPS-Degraded Environments. In *AIAA Information Systems-AIAA Infotech@ Aerospace* (p. 0448).

Hohenbichler, M., & Rackwitz, R. (1983). First-order concepts in system reliability. *Structural safety*, *1*(3), 177–188.

Jang, D.-S., Ippolito, C. A., Sankararaman, S., & Stepanyan, V. (2017). Concepts of airspace structures and sys-

tem analysis for uas traffic flows for urban areas. In *AIAA Information Systems-AIAA Infotech@ Aerospace* (p. 0449).

Jun, M., & DAndrea, R. (2003). Path planning for unmanned aerial vehicles in uncertain and adversarial environments. In *Cooperative control: models, applications and algorithms* (pp. 95–110). Springer.

Kim, B. S., & Calise, A. J. (1997). Nonlinear flight control using neural networks. *Journal of Guidance, Control, and Dynamics*, *20*(1), 26–33.

Kim, H. J., Shim, D. H., & Sastry, S. (2002). Nonlinear model predictive tracking control for rotorcraft-based unmanned aerial vehicles. In *American Control Conference, 2002. Proceedings of the 2002* (Vol. 5, pp. 3576–3581).

Kim, J.-H., & Sukkarieh, S. (2003). Airborne simultaneous localisation and map building. In *Robotics and automation, 2003. proceedings. icra'03. ieee international conference on* (Vol. 1, pp. 406–411).

Kim, K., Kim, T., Lee, K., & Kwon, S. (2011). Fuel cell system with sodium borohydride as hydrogen source for unmanned aerial vehicles. *Journal of Power Sources*, *196*(21), 9069–9075.

Kopardekar, P. H. (2014, April). *Unmanned Aerial System (UAS) Traffic Management (UTM): Enabling Low-Altitude Airspace and UAS Operations* (Tech. Rep. No. NASA/TM2014218299). Moffett Field, CA 94035, USA: NASA Ames Research Center.

Krishnakumar, K., Ippolito, C., Kopardekar, P., Melton, J., Stepanyan, V., Sankararaman, S., & Nikaido, B. (2017). Safe Autonomous Flight Environment (SAFE50) for the Notional Last "50 ft" of Operation of "55 lb" Class of UAS. In *AIAA SciTech Conference, Grapevine, Texas, USA*.

Marsh, R., Vukson, S., Surampudi, S., Ratnakumar, B., Smart, M., Manzo, M., & Dalton, P. (2001). Li-Ion batteries for aerospace applications. *Journal of power sources*, *97*, 25–27.

McAree, O., & Chen, W.-H. (2013). Artificial situation awareness for increased autonomy of unmanned aerial systems in the terminal area. *Journal of Intelligent & Robotic Systems*, 1–11.

Ollero, A., & Maza, I. (2007). *Multiple heterogeneous unmanned aerial vehicles*. Springer Publishing Company, Incorporated.

Ruff, H. A., Narayanan, S., & Draper, M. H. (2002). Human interaction with levels of automation and decision-aid fidelity in the supervisory control of multiple simulated unmanned air vehicles. *Presence: Teleoperators and virtual environments*, *11*(4), 335–351.

Rysdyk, R. (2006). Unmanned aerial vehicle path following for target observation in wind. *Journal of guidance, control, and dynamics*, *29*(5), 1092–1100.

Saha, B., Koshimoto, E., Quach, C., Hogge, E., Strom, T., Hill, B., & Goebel, K. (2011). Predicting battery life for electric UAVs. *AIAA Infotech@ Aerospace*.

Sankararaman, S. (2015). Significance, interpretation, and quantification of uncertainty in prognostics and remaining useful life prediction. *Mechanical Systems and Signal Processing*, *52*, 228–247.

Sankararaman, S., Daigle, M. J., & Goebel, K. (2014). Uncertainty quantification in remaining useful life prediction using first-order reliability methods. *Reliability, IEEE Transactions on*, *63*(2), 603–619.

Sankararaman, S., & Kalmanje, K. (2017). Towards a computational framework for autonomous decision-making in unmanned aerial vehicles. In *AIAA Information Systems-AIAA Infotech@ Aerospace* (p. 0446).

Shakernia, O., Vidal, R., Sharp, C. S., Ma, Y., & Sastry, S. (2002). Multiple view motion estimation and control for landing an unmanned aerial vehicle. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on* (Vol. 3, pp. 2793–2798).

Sharp, C. S., Shakernia, O., & Sastry, S. S. (2001). A vision system for landing an unmanned aerial vehicle. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on* (Vol. 2, pp. 1720–1727).

Shim, D. H., Chung, H., & Sastry, S. S. (2006). Conflict-free navigation in unknown urban environments. *Robotics & Automation Magazine, IEEE*, *13*(3), 27–33.

Sigurd, K., & How, J. (2003). UAV trajectory design using total field collision avoidance. *American Institute of Aeronautics and Astronautics*.

Sinopoli, B., Micheli, M., Donato, G., & Koo, T. J. (2001). Vision based navigation for an unmanned aerial vehicle. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on* (Vol. 2, pp. 1757–1764).

Stepanyan, V., & Krishnakumar, K. S. (2017). Estimation, navigation and control of multi-rotor drones in an urban wind field. In *AIAA Information Systems-AIAA Infotech@ Aerospace* (p. 0670).

Wang, J., Garratt, M., Lambert, A., Wang, J. J., Han, S., & Sinclair, D. (2008). Integration of gps/ins/vision sensors to navigate unmanned aerial vehicles. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *37*, 963–970.

Yang, G., & Kapila, V. (2002). Optimal path planning for unmanned air vehicles with kinematic and tactical constraints. In *Decision and Control, 2002, Proceedings of the 41st IEEE Conference on* (Vol. 2, pp. 1301–1306).