

Encounter-Based Simulation Architecture for Detect-And-Avoid Modeling

Mohamad Refai * and Michael Abramson[†] and Seungman Lee[‡]
Crown Consulting, Inc., Moffett Field, California, 94035

Gilbert Wu [§]
NASA Ames Research Center, Moffett Field, California, 94035

This paper presents an encounter-based simulation architecture developed at NASA to facilitate flexible and efficient Detect and Avoid modeling in parametric or tradespace studies on large data sets. The basic premise of this tool is that large-scale input data can be reduced to a set of ‘canonical encounters’ and that using the reduced data in simulations does not lead to loss of fidelity. A canonical encounter is specified as ownship and intruder flight portions potentially resulting in a loss of well clear along with a set of properties that characterize the encounter. The advantages of using canonical encounters include faster simulations, reduced memory footprint, ability to select encounters based on user-specified criteria, shared encounters across multiple teams, peer-reviewed encounters, and a better understanding of the input data set, to name a few.

Glossary

CPA	Closest Point to Approach
DAA	Detect-and-Avoid
DWC	DAA Well Clear
HITL	Human-in-the-Loop
HMD	Horizontal Miss Distance
JADEM	Java Architecture for DAA Extensibility and Modeling
Low SWaP	Low Size, Weight, and Power
LoWC	Loss of Well Clear
MOPS	Minimum Operational Performance Standards
NAS	National Airspace System
NMAC	Near Mid-Air Collision
RADES	84th Radar Evaluation Squadron
UAS	Unmanned Aircraft Systems
VFR	Visual Flight Rules
VMD	Vertical Miss Distance

I. Introduction

SUCCESSFUL integration of Unmanned Aircraft Systems (UAS) into the National Airspace System (NAS) is predicated upon maintaining the same level of safety and performance as achieved by current manned operations. To ensure that safety and performance goals are achievable, various data collection and analysis methods are employed in developing UAS Minimum Operational Performance Standards (MOPS) [1]. These methods include first principle engineering analyses, human-in-the-loop simulations, live flight tests, and fast-time simulations. Regardless of the

*Senior Engineer, M/S 210-8, mohamad.s.refai@nasa.gov

[†]Senior Engineer, M/S 210-8, Member AIAA, michael.abramson@nasa.gov

[‡]Senior Scientist, M/S 210-8, AIAA Member, seungman.lee@nasa.gov

[§]Aerospace Engineer, Aviation Systems Division, M/S 210-10, Member AIAA, gilbert.wu@nasa.gov

methodologies employed, a main challenge is identifying representative data sets to use in foundational studies and safety and performance assessments. For system-wide assessments, the challenge is two-fold. First, UAS mission data is not available at the anticipated densities. Second, UAS will operate in managed and unmanaged airspace, so, Visual Flight Rules (VFR) traffic data is also required.

An approach to provide representative data sets was developed by MIT Lincoln Laboratory [2], wherein radar data from the 84th Radar Evaluation Squadron (RADES) is used to estimate VFR initial conditions and trajectories. The data are sampled to create a database of statistical features of these flights. These statistics are then sampled to create new trajectories with characteristics consistent with the original trajectories. Encounters are modeled by sampling one or more trajectories that satisfy ‘proximity’ criteria. The model therefore generates encounters representing VFR to VFR encounters.

The approach taken at NASA has been to develop a set of UAS mission profiles that represent today’s view of future expected UAS operations and missions [3]. Nineteen missions and their corresponding geographical locations and operating schedules were developed with extensive contributions from subject matter experts [4]. To model intruder traffic, RADES data for 21 days of varying traffic density were selected and processed to provide smoothed VFR trajectories [5]. These VFR and UAS data have been used to conduct several foundational NAS-wide and parametric fast-time simulations in support of MOPS development [3, 6–10]. While this approach has proven quite valuable to MOPS development, it suffers from several shortcomings, which include:

- The total duration of encounters is a small fraction of the time of flight, so a full end-to-end simulation is wasteful.
- UAS maneuvers, when employed, alter the remaining UAS trajectory and the characteristics of following encounters, making data comparison difficult.
- VFR data is rather extensive and consequently, simulations consume considerable resources, despite various optimizations, and results include considerably more data than is typically required.
- A priori scenario selection is not possible.
- The data are not currently shared mainly due to size and computational requirements.
- Simulation results can be moderately sensitive to flight mechanics models, computational optimizations, etc; this complicates comparisons of alternative approaches.

This paper describes an encounter-based simulation architecture aimed at addressing these challenges. Simply put, this new approach uses the RADES and UAS mission data to identify and persist potential encounters, which are subsequently used in various Detect and Avoid (DAA) simulations and analyses. The process of identifying encounters also generates a set of corresponding encounter properties, which can be used to pre-select encounters of interest to a particular study. This approach marks a shift from past NAS-wide simulations by focusing on the events of interest and simulating only the relevant portions of flight instead of full end-to-end departure to destination modeling.

In what follows, we discuss this approach and its benefits in more detail. The next section defines encounters and scenarios and describes the encounter-based simulation architecture. Section III describes canonical encounter detection. Section IV discusses encounter properties and their uses. Section V describes scenario selection using encounter properties and typical uses cases. Sample results are provided in Section VI. Finally, concluding remarks are presented in Section VII.

II. Encounter-Based Architecture

The new architecture is based on a decomposition of the processing steps used in our research when running NAS-wide fast time simulations. This decomposition is depicted in Fig. 1 and will henceforth be referred to as a “processing pipeline”. The figure is simplified to depict unmitigated simulations and omits miscellaneous other optimizations. In preparation for running DAA on a given ownship, we first predict UAS trajectory states using a flight mechanics model. We then determine if an alert is likely using a series of gross filters and subsequently create the encounters to be processed. Noise is then optionally added before running DAA alerting and guidance algorithm and persisting the results. A subsequent post-processing step is used to generate metrics for reporting and analysis purposes. This processing pipeline has the following characteristics:

- The pipeline is fixed and requires code modification to change.
- Any time a simulation is executed, the Java Architecture for DAA Extensibility and Modeling (JADEM) [11] portion of the pipeline is executed in full even when the resulting perturbed encounters are unchanged between runs.
- Encounters are not persisted in a standard format.
- The pipeline lacks a mechanism to control or select the types of encounters processed.

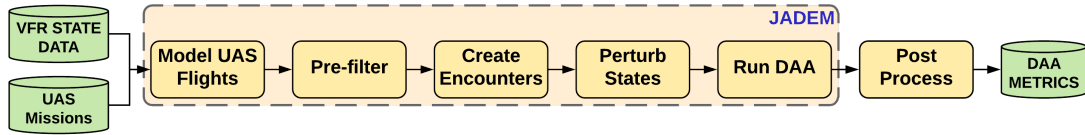


Fig. 1 Current processing pipeline

By refactoring these processing steps into independent modules, we can eliminate wasteful processing, change order of steps where applicable, and allow for additional steps to be interspersed where desired. Furthermore, by modularizing the processing pipeline, we allow it to be composited from available modules or alternative module implementations. Fig. 2 depicts one realization of a modular processing pipeline that applies a perturbation module and a scenario selection module prior to running DAA with various settings for a parametric or tradespace study. See Section V.A for other examples. In the figure, we persist and load data in between all processing steps but it should be noted that this is not a requirement. UAS missions are modeled using a flight mechanics model and saved as trajectory states. VFR and UAS states are then processed to detect encounters, which are subsequently perturbed. In the scenario selection step, a subset of the resulting perturbed encounters are selected for DAA processing. Finally, the DAA data is processed to generate metrics of interest. UAS mission modeling, encounter generation, perturbations, and scenario selection are executed once, whereas DAA can be executed multiple times for different configurations.

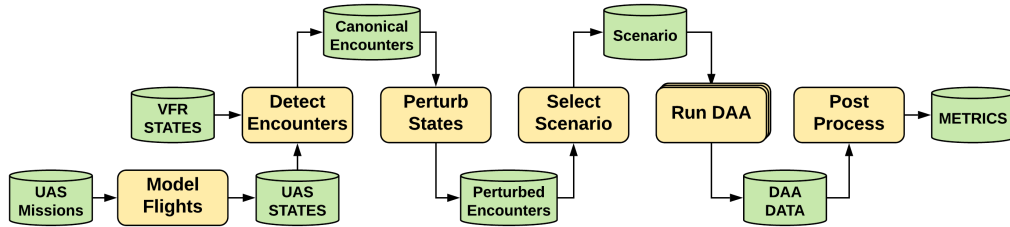


Fig. 2 Modular processing pipeline

Encounters For our purposes, we define encounters to include trajectory portions from one ownship and one or more intruders that can potentially cause loss of well clear, an alert, or peripheral guidance [1] with the ownship. Peripheral guidance is guidance information corresponding to an intruder that is not itself alerted [1, §2.2.4.4]. In addition to traffic data, a set of properties describing encounter characteristics of interest are also included in the encounter definition (see Section IV for details). This facilitates selective processing of encounters as dictated by research needs (see Sections IV.A and V). Note that an encounter, as defined here, includes one ownship only, so multiple ownship coordinated maneuvers are not modeled. A “canonical” encounter is one that is generated from source data such as VFR traffic and UAS data.

Pairwise Encounters The above definition of an encounter is quite general but for many foundational studies, the interaction with multiple intruders is of less interest; instead single intruder interactions are desired. For these cases, pairwise encounters composed of one ownship and one intruder are more appropriate. Such encounters are referred to as pairwise encounters. Note that multi-intruder encounters can be created by composition of pairwise encounters of the same ownship that overlap temporally. Pairwise encounters can therefore be considered as fundamental building blocks for creating more general or complex encounters.

Scenarios A scenario is a means to specify the encounters to be processed by DAA. It is defined as the set of encounters that satisfy specified criteria. For example, a scenario may include only those encounters whose ownship

speed is within a specified range. Scenarios are typically generated by selecting from one or more sets of encounters such as those obtained from data for different days. Scenarios can also be created by filtering encounters from other scenarios. In addition to the list of encounters, a scenario specification includes a reference to the source data and the encounter selection criteria used to generate the scenario. The source data can be another scenario, multiple scenarios, or the set of VFR and UAS data. Referencing source data and selection criteria in the scenario specification provides traceability of scenario data; this is depicted in Fig. 3.

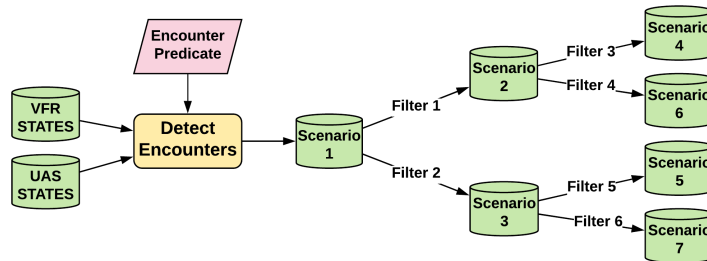


Fig. 3 Scenario Tree

1. A Note About Encounter Ownership Trajectory Data

An encounter’s ownership trajectory data can be represented as a set of trajectory states or as a mission profile, which is typically a portion of the full mission profile. In most simulations, trajectory state data is sufficient. However, if a simulation requires mission recapture, then the relevant portion of the mission profile must also be included in the encounter.

A. Scalable Architectures

The encounter-based architecture described above is particularly suited to scalable and big data architectures. Scalable concurrent execution is much simplified given that encounters are independent of one another. The modular nature of the pipeline also facilitates understanding of the data volumes to be processed at each step and the typical processing times involved. This simplifies data and processing architecture design dictated by research needs. Hybrid systems can be considered, which leverage scalable architectures for some processing steps and big data architectures for others. These can be combined to result in a flexible end-to-end solution from data acquisition to metric reports.

III. Encounter Detection

The goal of the core encounter detection algorithm is to extract encounters from input ownship and intruder flight data.

A. Encounter Detection Criteria

Ownship and intruder flights for an encounter are selected by applying user-defined predicates (boolean expressions) to pairs of ownship and intruder states. The simplest predicate is a cylindrical “hockey puck” (a static disc). If this predicate is used, the encounter is determined by comparing horizontal distance d_{horiz} and vertical distance d_{vert} between ownship and intruder states for the same time with specified horizontal and vertical separation thresholds d_{horiz}^* , d_{vert}^* .

$$(d_{horiz} < d_{horiz}^*) \wedge (d_{vert} < d_{vert}^*) \quad (1)$$

d_{horiz}^* , d_{vert}^* in condition 1 should be large enough to ensure that any alerts and well clear violations will result in an encounter. Fig. 4 shows one of the alerting structures that have been evaluated for UAS. The Buffered Well Clear Criteria (separation standards) in this table include a combination of Horizontal Miss Distance (HMD), vertical

separation, and modified tau-separation. The time to Closest point of Approach (CPA) t_{CPA} from Alerting Time Threshold column is more conservative and easier to use to estimate the minimal acceptable values for d_{horiz}^* and d_{vert}^* .





Symbol	Name	Buffered Well Clear Criteria	Alerting Time Threshold	Aural Alert
	Warning Alert	$DMOD = HMD^* = 0.75$ nmi $Z_{THR} = 450$ ft $T_{mod} = 35$ s	25 s ($t_{CPA} \sim 60$ s)	"Traffic, Maneuver Now"
	Corrective Alert	$DMOD = HMD^* = 0.75$ nmi $Z_{THR} = 450$ ft $T_{mod} = 35$ s	55 s ($t_{CPA} \sim 90$ s)	"Traffic, Avoid"
	Preventive Alert	$DMOD = HMD^* = 1.0$ nmi $Z_{THR} = 700$ ft $T_{mod} = 35$ s	55 s ($t_{CPA} \sim 90$ s)	"Traffic, Monitor"
	Traffic	In surveillance field of regard	-	-

Fig. 4 An example of alerting structure

The largest distance between ownship and intruder causing an alert corresponds to situation of head-on encounter. This distance can be estimated as a product of approach speed ΔV_{horiz} , which is an intruder's speed relative to ownship, and the time to CPA from Alerting Time Threshold column in Figure 4. In simulations considered in this paper, the air speed of UAS (ownship) and VFR traffic (intruders) does not exceed 300 knots that corresponds to $\Delta V_{horiz} < 600knots$. The largest t_{CPA} in Alerting Time Threshold column for Preventive or Corrective Alert (Figure 4) is 90 seconds. Therefore, the minimal value for $d_{horiz}^* = \Delta V_{horiz} \cdot t_{CPA} = 15$ nmi.

A similar estimate based on assumption of relative vertical speed ΔV_{vert} not exceeding 6000 fpm leads to the minimal value for $d_{vert}^* = \Delta V_{vert} \cdot t_{CPA} = 9000$ ft.

Therefore, the values $d_{horiz}^* = 20$ nmi and $d_{vert}^* = 10000$ ft or higher can be used for generating encounters if target application is simulations with alerting structure similar to shown in Figure 4.

B. Encounter Detection Algorithm

The encounter detection algorithm uses ownship and intruder trajectories represented as time series position and velocity data. These trajectories may start and end at any time and need not be contiguous; therefore, data gaps are allowed.

The algorithm proceeds as follows (see Fig. 5):

- 1) Pre-filter aircraft states to reduce the amount of computations.
- 2) Apply predicates to identify encounters.
- 3) Post-filter to remove unusable encounters such as those with very short duration.



Fig. 5 Encounter Detection

Pre-filtering is done in two steps. The goal of first step (the coarse filter) is to quickly identify the intruders, which could in principle come close to each ownship within a specified look-ahead time. This is done by mapping all initial intruder horizontal positions to a fixed horizontal grid, as described in [11].

Further pre-filtering is done by skipping time-steps on the remaining flights. The amount of time to skip is estimated from the time it would take ownship and intruder to be within the horizontal and vertical thresholds in a worst case if

they have instantly turned to a collision course. Note that the amount of time that can be skipped, and hence the number of computations at prefiltering step, does not depend on time step.

Once the amount of time-to-skip becomes close to a time step, the prefiltering ends and further calculations are repeated at every time step. All sequential intruder states that satisfy the condition defined by Eq. 1 are included in the encounter for this intruder. The encounter may include more than one intruder if there is a time overlap between trajectory segments of different intruders that satisfy the condition of Eq. 1 for the same ownship.

The last optional step of encounter detection algorithm involves filtering out the encounters shorter than a specified minimum duration (e.g. 30 seconds).

The remaining encounters are persisted to the data store along with a corresponding scenario specification (see Section V for more details).

IV. Encounter Properties

A number of properties are defined as a part of encounter detection process and persisted for each encounter. The most important of these properties are listed below.

EncounterID	a unique numeric identifier of this encounter
OwnshipAcID	the callsign (ACID) of ownship flight
OwnshipAcType	the aircraft type of ownship used to determine its performance in trajectory generation process
EncounterStartTime	encounter start time defined as the first time when condition given in Eq. 1 was satisfied at least for one intruder
EncounterEndTime	encounter end time defined as the last time when condition given in Eq. 1 was satisfied at least for one intruder
EncounterDuration	encounter duration defined as the difference between EndTime and StartTime
OwnshipFlightPhase	the ownship's flight phase during the encounter
OwnshipAirspaceClass	the ownship's airspace class during the encounter
Intruder.IntruderAcID	the intruder's callsign (ACID)
Intruder.IntruderFlightPhase	the intruder's flight phase during the encounter
Intruder.OwnshipSpeedAtCpa	ownship speed at CPA
Intruder.OwnshipAltitudeAtCpa	ownship altitude at CPA
Intruder.IntruderSpeedAtCpa	intruder speed at CPA
Intruder.IntruderAltitudeAtCpa	intruder altitude at CPA
Intruder.SlantCpa.HorizontalDistance	horizontal distance d_{horiz} between ownship and intruder at CPA
Intruder.SlantCpa.VerticalDistance	vertical distance d_{vert} between ownship and intruder at CPA
Intruder.SlantCpa.Time	time of CPA
Intruder.MinHmd	minimum Horizontal Miss Distance (HMD) [11] over encounter duration
Intruder.MinVmd	minimum Vertical Miss Distance (VMD) over encounter duration; the VMD is defined here as zero for vertically converging flights and as altitude difference between ownship and intruder states otherwise
Intruder.HorizontalClosureRate	horizontal closure rate \dot{d}_{horiz} at CPA
Intruder.ViolatedNmac	indicates whether an intruder is in a NMAC defined as $(d_{horiz} < 500ft) \wedge (d_{vert} < 100ft)$

The “.” notation is used to indicate the grouping of related properties. It is also used in a search path when looking up the properties in “filters” described in IV.A. In particular, all intruder-specific properties are grouped by intruder.

In addition to these “general properties” that don't depend on specific separation standards, several “DAA properties” can be defined, such as the properties of Loss of Well Clear (LoWC) and alerts of different “levels” corresponding to a particular alerting structure, such as shown in Fig. 4. These DAA properties are calculated by processing results generated by JADEM. The processing pipeline for generating all encounter properties is shown in Fig. 6.

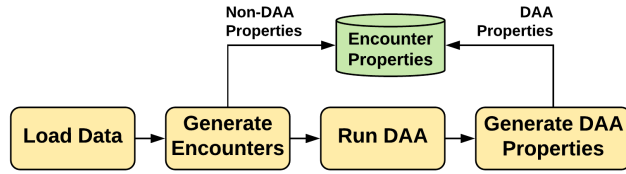


Fig. 6 Encounter Properties Generation

A. Encounter Filtering

The main use of encounter properties is to select the encounters of interest for a particular study or application. The technique used for this involves creating user-defined “filters” and is referenced herein as “encounter filtering”.

A filter is defined as one or more predicates (boolean expressions). Each predicate compares a property specified via its path in the property files to a constant (threshold) or list of constants with appropriate units. The supported comparison operators are equality, strict and non-strict inequality, and “ContainedIn” and “NotContainedIn” operators. The last two operators are used to check whether the property has one of several specified values defined in a list. The predicates can be combined using AND and OR logical operators.

For instance, a user may define a filter to select encounters for a particular `OwnshipAcType`. Another example could be selecting all encounters with `OwnshipAltitudeAtCpa` below certain altitude in feet for `EncounterDuration` above a specified threshold in seconds that have a `ViolatedNmac = false` (i.e. don’t result in NMAC). One especially useful filter is selecting encounters with `Intruder.MinHmd` and `Intruder.MinVmd` properties that don’t exceed HMD^* and Z_{THR} thresholds (Figure 4), since only these encounters can alert or result in LoWC.

Applying the filter to a scenario results in another scenario including only a subset of encounters that passed the filter. This process is schematically shown in Fig. 7.

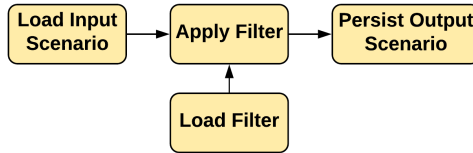


Fig. 7 Encounter Filtering

V. Scenario Selection

This section describes how the encounter detection and filtering methodology introduced in Sections III and IV.A is typically used in simulations (see Fig. 3 for a depiction of this process).

The first step is to generate scenarios for available VFR and UAS data sets. These scenarios include all pairwise canonical encounters detected using a conservative predicate as described in III.A. As a result, these scenarios typically include many more encounters than are strictly required for a particular study. For instance, many encounters may not result in any events of interest, such as alert, LoWC, or NMAC.

To “tailor” scenarios to a study, researchers create filters to be used to select encounters of interest as described in Section IV.A. Filters may be defined using DAA independent properties or they may be defined using properties generated by running a scenario through JADEM as shown in Fig. 6. In the latter case, JADEM is typically run with conservative settings that form a superset of all the design parameters required for the study. The encounter filtering process shown in Fig. 7 can be repeated several times using different filters.

The resulting scenario is processed through JADEM to generate DAA data, which are subsequently processed to generate aggregate metrics for analysis and reporting. Note that the most computationally expensive operation, namely,

the generation of canonical encounters and computation of their properties, need only be performed once; the resulting encounters can then be used for many different studies.

A. Use Cases and Benefits

The benefits of this approach can be illustrated by several use cases.

- 1) **Evaluating alternate DAA Well Clear (DWC) definitions.** In this case, the desired encounters are those that will generate an alert for at least one of the DWC definitions being evaluated. Therefore, the canonical encounters are processed once through JADEM using the most conservative DWC definition, which covers all the DWC definitions. The resulting alerted encounters are then selected for use in the comparative study.
- 2) **Selecting Low Size, Weight, and Power (SWaP) encounters.** Low SWaP sensors are typically installed on relatively small and slow UAS flying below 10000 ft. Therefore, low SWaP encounters can be selected using filters composed from predicates on OwnshipAcType, OwnshipSpeedAtCpa, OwnshipAltitudeAtCpa, and possibly others, depending on specific goals of the study.
- 3) **Selecting Terminal Area encounters.** Terminal area encounters can be selected using such properties as OwnshipAirspaceClass, OwnshipAltitudeAtCpa, and OwnshipFlightPhase.
- 4) **Selecting flight-phase-specific encounters.** For studies focused on characterization of encounters by flight phase (e.g. level-level, level-climb, climb-descent, etc), encounters can be selected using OwnshipFlightPhase and IntruderFlightPhase properties. It should be noted, however, that IntruderFlightPhase may be difficult to determine from non-cooperative flight data.
- 5) **Validating canonical encounters.** To experimentally validate the encounters generated by the encounter detection process, one could compare JADEM results using original VFR and UAS data to the results using canonical encounters generated for a test case. Should the results differ meaningfully, the encounter predicate is adjusted and the process repeated. This is depicted in Fig. 8.

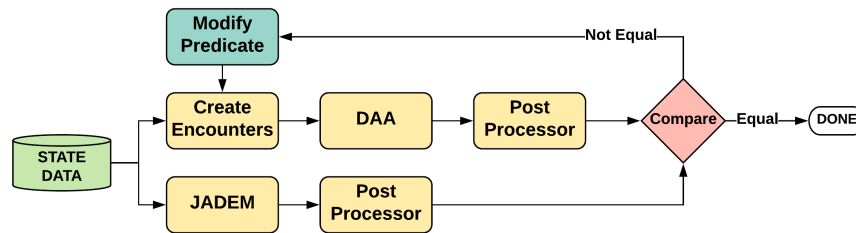


Fig. 8 Validating Canonical Encounters

VI. Results

This section compares the performance of the encounter-based approach to that of the “old” approach, which modeled flights from departure to destination.

The comparisons were performed using 21 days of VFR traffic with a parametric suite of 96 individual configurations each. The baseline computation times, using the old approach, were estimated from an average day using one representative configuration rather than from the full suite of 2016 simulation runs. The processing pipeline used in this comparison study is depicted in Fig. 9.

The overall approach can be summarized in four steps:

- 1) Detect pairwise encounters.
- 2) Filter out all encounters that will not violate any of the Low SWaP DWC definitions; this is accomplished through the use of a conservative DWC that covers all the Low SWaP DWC definitions.
- 3) Combine the resulting encounters into a single scenario.
- 4) Process the scenario through DAA and generate the metrics.

The first step is to generate the trajectories for all UAS flights; trajectory generation uses the model described in [11]. The flight times are subsequently adjusted for each simulation day. For each of the 21 days, pairwise encounters are

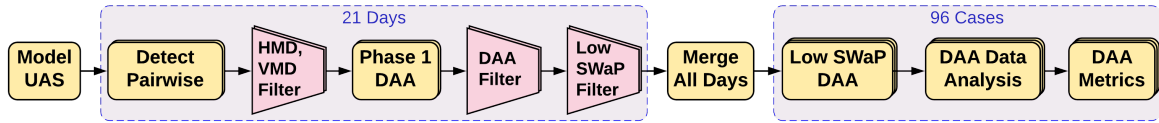


Fig. 9 Experiment Pipeline

detected along with their properties, which include HMD, VMD, and UAS and VFR speeds and altitudes at CPA. To filter all redundant encounters that will have no chance of violating the Low SWaP DWC definitions, we use the DWC definition published in the Phase 1 MOPS. This is accomplished as follows:

- 1) *HMD & VMD Filter*: Create a filter that compares an encounter’s minimum HMD and VMD against the HMD and Vertical Separation thresholds given in Phase 1 DWC. This creates a scenario of the remaining encounters.
- 2) *Phase 1 DAA*: Process the resulting scenario through JADEM configured for Phase 1 DWC alerting.
- 3) *DAA Filter*: Use the alert results to filter encounters, thus creating a scenario containing only those encounters that alert on Phase 1 DWC.
- 4) *Low SWaP Filter*: Apply a speed and altitude filter on the resulting scenario to remove all encounters whose speeds and altitudes exceed the Low SWaP performance limits. The resulting scenario will include only Low SWaP encounters that alert on Phase 1 DWC.

The resulting scenarios for the 21 days are small enough that they can be combined into a single scenario. This is done in order to reduce the number of simulations, since we are only interested in the overall results not those of individual days. As a result, we only have to contend with 96 instead of 2,016 (96×21) simulations.

Table 1 summarizes the salient computational performance figures. For each step in the pipeline (cyan colored rows), the table reports the average time per run, the number of runs, the total time for all runs, and where applicable, the total number of input encounters, the total number of output encounters, and the average time per input encounter. Also reported in the table are the estimated times (salmon colored rows) had the simulations been run using JADEM as shown in Fig. 1. These time estimates were obtained from a single run and scaled to all 2,016 runs. The main takeaways from the table are that filtering is key to improving the overall performance and that using the new processing pipeline reduced the total computation time five-fold.

Table 1 Performance Metrics

Processing Step	Average Time (hr)	#	Total Time (hr)	Total Input Encounters	Total Output Encounters	Time (ms) Per Encounter
Trajectory Generation	0.75	1	0.75	–	–	–
Encounter Detection	1.23	21 days	25.9	–	–	–
HMD/VMD Filter (Phase 1)	0.42	21 days	8.8	9,692,090	2,106,610	3.3
Phase 1 DAA	1.93	21 days	40.6	2,106,610	–	69
DAA Filter (Phase 1)	0.14	21 days	2.9	2,106,610	129,394	5.0
Low SWaP Filter	0.0	21 days	0.0	129,394	82,524	0.7
Scenario Merge	0.0	1 scenario	0.0	82,524	82,524	0.0
Low SWaP DAA	3.60	96 cases	346	7,922,304	–	157
DAA Data Analysis	8.69	96 cases	834	7,922,304	–	379
Metrics	0.00	96 cases	0.2	–	–	–
<i>Total</i>	–	–	1,259	–	–	–
Legacy DAA Processing	1.8	21×96	3,651	–	–	–
Legacy DAA Data Analysis	1.5	21×96	3,024	–	–	–
<i>Legacy Total</i>	–	–	6,675	–	–	–

In Table 1, all I/O operations were to the file system, with I/O over the network consuming approximately twice as much time as local I/O. In the early stages of the pipeline, where the data volumes are large, the processing time is dominated by I/O. This is especially true for encounter detection where computations consumed only 20% of the total time. Note that all calculations were performed without any parallelization.

VII. Conclusion

NASA uses JADEM to conduct fast-time parametric and tradespace studies in support of UAS integration in the NAS. These studies use NAS-wide VFR data and projected UAS missions. Simulations are conducted on the full airspace data, modeling flights from departure to destination. As a result, simulations are time consuming, require substantial computational resources, and are wasteful since the events of interest represent a small fraction of flight time. Furthermore, research studies often require selective processing of encounters but the current tools do not provide mechanisms for selecting encounters a priori. Finally, sharing the voluminous NAS-wide data with our partners is cumbersome. To address these issues, JADEM was refactored to an encounter-centric processing pipeline composed of a set of decoupled modules. This design has been shown to have the following advantages:

- Reduced memory footprint and processing time.
- Flexible scenario generation.
- Flexible DAA processing pipeline, with swappable modules.
- Standardized Encounter, Scenario, and Filtering representations.
- Canonical encounters that can be shared across multiple teams.
- Increased transparency.

Acknowledgments

We wish to express gratitude to our fellow DAA team developers for their tireless efforts to realize the encounter-based simulation architecture.

References

- [1] SC-228, *Detect and Avoid Minimum Operational Performance Standards Phase I (DAA MOPS)*, RTCA, Washington, DC, 2017.
- [2] Weinert, A. J., Harkleroad, E. P., Grith, J., Edwards, M. W., and Kochenderfer, M. J., "Uncorrelated Encounter Model of the National Airspace System, Version 2.0," Tech. Rep. ATC-404, MIT Lincoln Laboratory, Lexington, Massachusetts, August 2013.
- [3] Lee, S., Park, C., Thippavong, D., Isaacson, D., and Santiago, C., "Evaluating Alerting and Resolution Performance of a UAS Detect-And-Avoid (DAA) System," Tech. Rep. NASA/TM-2016-219067, NASA Ames Research Center, Moffett Field, California, February 2016.
- [4] Ayyalasomayajula, S., Wieland, F., Trani, A., and Hinze, N., "Unmanned Aircraft System Demand Generation and Airspace Performance Impact Prediction," *Proceedings of the 32nd IEEE Digital Avionics Systems Conference*, IEEE, Syracuse, NY, 2013.
- [5] Park, C., Lee, H., and Musaffar, B., "Radar Data Tracking Using Minimum Spanning Tree-Based Clustering Algorithm," *11th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference (AIAA-2011-6825)*, AIAA, Virginia Beach, VA, 2011.
- [6] Lee, S., Park, C., Johnson, M., and Mueller, E., "Investigating Effects of "Well Clear" Definitions on UAS Sense-And-Avoid Operations," *2013 AIAA Aviation Technology, Integration, and Operations Conference*, AIAA, Los Angeles, CA, 2013.
- [7] Park, C., Lee, S., and Mueller, E., "Investigating Detect-and-Avoid Surveillance Performance for Unmanned Aircraft Systems," *2014 AIAA Aviation Technology, Integration, and Operations Conference*, AIAA, Atlanta, GA, 2014.
- [8] Johnson, M., Mueller, E., and Santiago, C., "Characteristics of a Well Clear Definition and Alerting Criteria for Encounters between UAS and Manned Aircraft in Class E Airspace," *10th USA/Europe ATM Research and Development Seminar, ATM2015*, ATM Seminar, Lisbon, Portugal, 2015.
- [9] Cone, A., Thippavong, D., Lee, S., and Santiago, C., "UAS Well Clear Recovery against Non-Cooperative Intruders using Vertical Maneuvers," *17th AIAA Aviation Technology (AIAA-2017-4382)*, AIAA, Denver, Colorado, 2017.

- [10] Thippavong, D., Cone, A., and Lee, S., “Ensuring Interoperability between UAS Detect-and- Avoid and Manned Aircraft Collision Avoidance,” *Twelfth USA/Europe Air Traffic Management Research and Development Seminar, ATM2017*, ATM Seminar, Seattle, Washington, 2017.
- [11] Abramson, M., Refai, M., and Santiago, C., “The Generic Resolution Advisor and Conflict Evaluator (GRACE) for Unmanned Aircraft Detect-And-Avoid Systems,” Tech. Rep. NASA/TM-2017-219507, NASA Ames Research Center, Moffett Field, California, 2017.