

# Chapter 7

## Simulations

The chapter presents the set of simulations developed according to a test plan (sec. 7.1). Test configuration (sec. ??) targets at exercising and evaluating proposed framework. The test cases are grouped in the following sections:

1. *Non-cooperative test cases* (sec. ??).
2. *Cooperative test cases* (sec. ??).
3. *Test cases conclusion* (sec. ??).
4. *Reach set approximation performance tests* (sec. ??).

### 7.1 Test Plan

The *Avoidance requirements* are given in (sec. ??), namely:

1. *Safety Margin Enforcement* (sec. ??) - keep UAS safe depending on situation.
2. *Path Tracking* (sec. ??) - track mission is given by a set of *waypoints* in the manner of *Energy Efficiency* (sec. ??).

These are given as nominal behavior (sec. ??), further enhanced by rule-based behavior (sec. ??).

The *Navigation requirements*, out of this scope, are given in (sec. ??). These are satisfied by *Mission Control Run* (sec. ??).

#### 7.1.1 Testing approach

The purpose of this section is to show complex scenarios, not unit testing of framework functionality. The focus is on *borderline* cases for typical situations in an *expected environment*. The *mode switch* between *Navigation* and *Emergency Avoidance*.

The *Tests* are designed to focus on particular functionality in specific *operational environment* with main *obstacle/weather/intruder feature* with environment induced *constraints*. There is also *UTM* factor and *Navigation penalty*.

**Operational Environment** is classified according to:

1. *Operation space* - important for *Low Altitude Operations*, the difficulty of *Avoidance Maneuvers* is proportionally increasing with *Obstacle density*. There are following main categories
  - a. *Rural environment* - the relief and man-made structures are sparsely spread around the *operation space*; the UAS is operating on *very low altitude* ( $\leq 50$  feet).
  - b. *Urban environment* - the concentration of the man-made structures are much higher, and they are more incorporated into land relief pattern, the UAS is operating on *very low altitude*.
  - c. *Open air* - the concentration of ground structures is very low, the concentration of *cooperative* and *non-cooperative intruders* is increased, the UAS is operating in altitude ranging from *50 feet* to *space border*. This brings us to:
2. *Airspace category* - when *Operation Space* pattern is categorized as *Open air* and depending on *altitude above mean sea level*. The UTM is *designed authority* for controlled airspace in current *F/G class airspace*.
  - a. *Controlled* - Open air where authority is present. The cases when *Authority* is not enforced due to the UTM malfunction, *C2 link loss* or other cause are not considered.
  - b. *Non-Controlled* - Open air operation space where is no central arbiter to determine or enforce traffic attendants behavior.

**Static obstacles:** Static obstacles with various features detectable by main *LiDAR* sensor. The main purpose is to show avoidance capabilities combined with heavy restrictions imposed by *soft* and *hard* constraints. The original purpose of our approach was to provide robust framework for static obstacle avoidance. Three tests with increasing obstacle density and navigation complexity are delivered.

**Operational Space Constraints** depends mainly on the *operational environment*. The standard set of constraints were taken into account for our test cases:

1. *Rural, Urban environment (low altitude)* are geo-fencing zones, ground (hard constraints), non-controlled airspace altitudes (soft constraints).
2. *Non-controlled airspace constraints (open air)* are geo-fencing zones (hard constraints), restricted airspace (hard constraint), weather (soft/hard constraint), controlled airspace (hard constraint), very low altitude border (soft constraints).
3. *Controlled airspace constraints (open air)* are restricted airspace (hard constraint), weather (soft/hard constraint), non-controlled airspace boundary (hard constraints), UTM Directives (hard constraints).

**Air Traffic Attendants:**

1. *Non-cooperative UAS* (Intruder) - there are some intruders with some degree of authority, size and *severity*. There were three test cases for non-cooperative intruders. Non-cooperative Intruders can be categorized as following based on behavior:
  - a. *Chaotic* intruders usually tend to behave unpredictable, for example, bird or *UAS in distress*, for this type of intruders *Maneuver Uncertainty Intersection Model* is used (sec. ??).
  - b. *Harmonic* intruder usually follows long straight paths, for example, UAS converging to waypoint, for this type of intruder *Body Volume Intersection Model* is used. (sec. ??).

*Cooperative UAS* (Intruder) - there are cooperative intruders who are obeying authority (UTM) or follow *common consensus*. The work focus on *UTM* authority implementation in four test cases. These test cases are reflecting the traffic management situations essential for successful UTM collision management

**Weather** impose *soft* and *hard* space constraint, which can be moving or static. The *soft constraint avoidance* is covered by *hard constraint avoidance*. The *static constrained area* is covered by *static obstacle avoidance* capability due to the *data fusion procedure* [1]. The only case which is not covered is *Moving constrained area*; small constraints can be covered by intruder models. The ideal candidate is a *storm*, because it covers quite a large area, the clouds are constantly moving, and severity is changing with time.

**UTM:** The *UAS Traffic Management* service should be implemented in *controlled airspace* by 2035. It is necessary to study impact of UTM services on the *Detect and Avoid* systems like ours.

The most basic service is *Identity provider* which should be implemented by 2020.

Then there are *location services*, which are necessary for coordinated collision avoidance, these were implemented in our solution up to necessary level for *Rules Of the Air* implementation.

*Mission tracking* is service tracking deviations from *declared mission plan* and *actual execution*. These statistics were used in all tests to track deviations from the reference trajectory.

*Directives* for *Traffic management* and *Collision prevention* are implemented as the functional life cycle of *Position notification* (sec. ??), *Collision Case* (sec. ??) for UTM. The directive handling is implemented as *Rule engine* (sec. ??) on UAS side.

**Navigation:** Navigation algorithm is depending on *Navigation mode*. UAS is usually in *Navigation mode* most of the time, despite this fact, UAS was forced into *Emergency Avoidance Mode* most of the time in test cases. The navigation complexity has been divided into following categories:

1. *Open space* - UAS has visibility to goal waypoint most of the time; there are no traps.
2. *Hidden waypoint* - UAS does not have visibility to goal waypoint, most of the time; there are irregular traps sometimes.
3. *Maze solving* - UAS line of sight for goal waypoint is hindered by multiple obstacles, there are irregular traps often.
4. *Rule following* - UAS navigation capabilities are constrained by rule enforcement.

### 7.1.2 Test Cases Summary

*Test cases* are summarized in (tab. 7.1).

| <i>Test Case Name</i> | <i>Operational Environment</i>                         | <i>Air Traffic Attendants</i> | <i>Weather</i> | <i>UTM</i> | <i>Navigation</i> | <i>Scenario</i>                           |
|-----------------------|--|-------------------------------|----------------|------------|-------------------|---|
| Building Avoidance    | Non-controlled (Rural)<br>$4 \times \text{buildings}$  | -                             | -              | -          | Open space        | Fly mission around four buildings         |
| Slalom                | Non-controlled (Rural)<br>$14 \times \text{buildings}$ | -                             | -              | -          | Hidden waypoint   | Navigate to hidden waypoint               |
| Maze                  | Non-controlled (Urban)<br>$30 \times \text{buildings}$ | -                             | -              | -          | Maze structure    | Solve maze with multiple curves           |
| Storm                 | Non-controlled (Rural)<br>$0 \times \text{buildings}$  | -                             | Storm          | -          | Open Space        | Avoid approaching storm                   |
| Emergency Converging  | Non-controlled (Open air)                              | Non-cooperative UAS (1x)      | -              | -          | Open Space        | Converging situation resolution w. o. UTM |
| Emergency Head on     | Non-controlled (Open air)                              | Non-cooperative UAS (1x)      | -              | -          | Open Space        | Head on situation resolution w. o. UTM    |
| Emergency Multiple    | Non-controlled (Open air)                              | Non-cooperative UAS (3x)      | -              | -          | Open Space        | Multi-collision case resolution w. o. UTM |
| Rule-based Converging | Controlled (Open air)                                  | Cooperative UAS(1x)           | -              | Full       | Follow Rules      | Converging situation resolution with UTM  |
| Rule-based Head on    | Controlled (Open air)                                  | Cooperative UAS(1x)           | -              | Full       | Follow Rules      | Head on situation resolution with UTM     |
| Rule-based Multiple   | Controlled (Open air)                                  | Cooperative UAS(3x)           | -              | Full       | Follow Rules      | Multi-collision case resolution with UTM  |
| Rule-based Overtake   | Controlled (Open air)                                  | Cooperative UAS (1x)          | -              | Full       | Follow Rules      | Overtake by UAS different speed ratio     |

Table 7.1: Test Cases Summary.

### 7.1.3 Performance Evaluation

**Evaluation method:** *Test cases* were evaluated according to performance requirements defined in (sec. ??). The method was tracking critical parameter for *Safety* (sec. ??) (primary) and *Trajectory Tracking* (sec. ??) (secondary) including *Energy Efficiency* (sec. ??).

**Safety Margin Performance Evaluation:** The *safety of UAS* is main concern of *DAA system*. The common concept of *safety margin* is evaluated.

The *threat* is multidimensional; there are often multiple *static obstacles*, *intruders* or *weather constraints*. To reduce the multidimensional threats to one-dimensional value *crash distance* concept is used:

$$crashDistance(t) = distance(UAScenter(t), threat)$$

where *selection the criterion* is:

$$\min \left\{ \begin{array}{l} \left( distance(UAScenter(t), threat) - \dots \right) \\ \dots - threat.SafetyMargin \\ : \forall threat \in KnownWorld(t) \end{array} \right\} \quad (7.1)$$

The *crash distance* (eq. 7.1) for given time is evaluated as shortest distance between UAS center and threat. The threat origins from the known world (sec. ??). The *threat* has safety margin. The distance to safety margin is used as a prioritization criterion in our test cases (tab. 7.1).

The *safety margin* evolution over time (eq. 7.2) is calculated similarly to *crash distance*. The most dangerous threat is selected based on *distance to the safety margin* criterion. The value of *safety margin* property is then used.

$$safetyMargin(t) = threat.SafetyMargin$$

where *the selection criterion* is:

$$\min \left\{ \begin{array}{l} \left( distance(UAScenter(t), threat) - \dots \right) \\ \dots - threat.SafetyMargin \\ : \forall threat \in KnownWorld(t) \end{array} \right\} \quad (7.2)$$

The *distance to safety margin* (eq. 7.3) is calculated as a difference between the *crash distance* (eq. 7.1) and *safety margin* (eq. 7.2). The *acceptance criteria* for safety is the *distance to safety margin*  $\geq 0$ .

$$distanceToSafetyMargin(t) = crashDistance(t) - safetyMargin(t) \geq 0 \quad (7.3)$$

*Note. On Signed Distance:* The most works are using *unsigned distance*. This work considers the *signed distance* with the following intervals:

1. + (away from the safety margin).
2. 0 (touching margin with UAS edge).
3. - (inside margin - crash/collision/broken boundary).

**Distance to Safety Margin** peaks are measured:

1. *Minimal* distance to safety margin indicates if *acceptance criterion* (eq. 7.3 is met).

2. *Maximal distance to safety margin* indicates the future *minimal detection range*. All scenarios were considered as borderline cases.

**Trajectory Tracking Evaluation** is a secondary priority after safety, following parameters were checked:

1. *Waypoint reach* - the *Mission* (??) is considered as completed if and only if  $\forall$  waypoints are reached and in the given order (check the output of ??). Moreover, if there is multiple UAS, each must meet the condition.
2. *Acceptable deviation* - for *tracking problem* (eq. ??) is a trajectory which in addition to *basic obstacle problem* (sec. ??) keeps deviation from the *reference trajectory* under a certain threshold (eq. ??).

*Trajectory tracking deviation threshold* (eq. 7.4) is defined as double of maximal distance between *goal waypoint* and *previous waypoint*.

$$trackingDeviationTreshold = 2 \times distance(goalWaypoint, previousWaypoint) \quad (7.4)$$

*Note.* If *goal waypoint* is first in the *mission*, the *UAS initial condition* is considered as a *previous waypoint*.

**Computation Load:** There is a theoretical definition of *intersection models* for *static obstacles and constraints* (sec. ??), *moving obstacles and constraints* (sec. ??), *avoidance run* (sec. ??), *mission control* (sec. ??) computation complexity.

The practical application requires to measure *computation load* in constrained environment. Let say that *avoidance framework* is running on stand alone embedded computer with 1.2 GHz processor and 1GB of dedicated RAM. This is simulated by *virtual machine*.

The *simulations* were executed in *Matlab/Simulink* environment<sup>1</sup> using: *UTM*<sup>2</sup>, *Navigation loop*<sup>3</sup>, *Avoidance grid*<sup>4</sup> and *Reach set*<sup>5</sup> implementations.

The *decision frame* length is set to 1s which gives *computation load* (eq. 7.5). The *computation load* represents the portion of the *previous decision frame* used to current decision frame calculation.

$$computationLoad = \frac{computationTime(frame)}{decisionFrameDuration} \times 100, \quad [\%; s, s] \quad (7.5)$$

<sup>1</sup>Prototype framework implementation: <https://github.com/logomo/Feature-based-ACAS/>

<sup>2</sup>UTM class: .../UavTrafficManagement/UTMControl.m

<sup>3</sup>Navigation Loop main class: .../MissionControl/MissionControl.m

<sup>4</sup>Avoidance Grid class: .../AvoidanceGrid/AvoidanceGrid.m

<sup>5</sup>Reach set tree class: .../AvoidanceGrid/PredictorNode.m

*Note.* *Computation load* is depending on the actual situation; when the UAS is in *navigation mode*, it should be low, when the UAS is in a *clustered environment* it should be high.

Matlab implementation is quite ineffective; the Python/C++ implementation can give better results.

For *computational feasibility* there is *implicit* acceptance criterion (eq. 7.6): the computation of a feasible path for *this time-frame* must end in the *previous time-frame*.

$$\forall time \in Mission : \quad computationLoad < 100\% \quad (7.6)$$



# Bibliography

- [1] Alojz Gomola, Pavel Klang, and Jan Ludvik. Probabilistic approach in data fusion for obstacle avoidance framework based on reach sets. In *Internal publication collection*, pages 1–93. Honeywell, 2017.