Chapter 4

Problem Statement

A *UAS* equipped with several types of sensors is tasked to fly several types of *Missions* in a 3-dimensional space. There is a *Terrain* map, an *Object map*, and a *Weather* forecast for target region that are known a priori. The map of the *Terrain* may not be up to date, and *Uncharted obstacles* may affect flight safety. The *UAS* has to comply with a set of *Flight Rules* specifying *Flight Constraints*. The performance of the *UAS*, including sensor performance, is affected by the *Weather*.

Several difficulties must be faced. First, the design space of is large and complex. Second, *Trajectory calculation* Third, *Navigation*.

4.1 Basic Definitions

A few definitions are needed.

4.1.1 World

The world of interest consists of:

Space has a Global Reference Frame with three axes X^+, Y^+, Z^+.

$$Space \subseteq \mathbb{R}^3 : \text{main axes:} X^+, Y^+, Z^+, \text{ center}[x_0, y_0, z_0] \tag{4.1}$$

Object (4.2) is a generic subset of *Space* which has *Boundary*.

$$Object \subset Space : \forall point \in Object, point \text{ is solid} \tag{4.2}$$

A nonlinear first order state-space model gives the dynamic model of the UAS:

$$\dot{x} = f(t, x, u) \tag{4.3}$$

Where $x \in \mathbb{R}^{6+n}, n \in \mathbb{N}^{0+}$ is *system state* containing minimal information about UAS position $[x, y, z]$ and orientation $[\theta, \varphi, \rho]$ (roll, pitch yaw), and $u(t)$ is *control signal* belonging to $R^k, k \in \mathbb{N}^+$, bounded by control set $u(t) \in U$.

The map of the terrain is given by *TerrainMap* mapping (x, y) to the terrain elevation

$$TerrainMap : (x, y) \to z \tag{4.4}$$

Weather (4.5) can impact the performance of the *UAS*. *Wind* can decrease flying capabilities and maneuverability, *visibility* can impair the performance of sensors such as LiDAR and cameras, *humidity* can be a serious danger to electronic equipment, and in combination low *temperature* it can cause icing on the UAS body, and *air pressure* can impact ranging sensor and flight performance.

$$Weather : (position \in Space) \times time \rightarrow \begin{bmatrix} \text{wind} & (w_x, w_y, w_z) \\ & [m/s, m/s, m/s] \\ \text{visibility} & v\,[m] \\ \text{humidity} & h\,[0-100\%] \\ \text{air pressure} & a\,[hPa] \\ \text{temperature:} & \tau\,[C] \end{bmatrix} \tag{4.5}$$

4.1.2 Mission

Mission (4.6) which UAS should fly is given as a set of *ordered, feasible regarding UAS dynamic (4.3), waypoints* in a subspace of *Space* (4.1).

$$Mission = \begin{Bmatrix} waypoint_1, waypoint_2, \ldots, waypoint_m : \\ \forall_{i=1\ldots m} waypoint_i \in Space \end{Bmatrix}, \quad m \in \mathbb{N}^+, m \geq 2 \tag{4.6}$$

Waypoint Passing (4.7) function maps system (4.3) trajectory *projection in Space* and *Mission* waypoints (4.6) to a vector of *passing times*.

$$WaypointPassing : TrajectoryProjection \times Mission \rightarrow Time^m \tag{4.7}$$

Note. The *Mission* (4.6) is considered as completed if and only if $\forall$ waypoints are reached and in the given order (check the output of 4.7).

4.1.3 Flight Constraints

Flight constraints arise from aviation rules and ATM/UTM.

There are H sets of hard flight constraints expressed as subsets of *Space*:

$$HFlightConstraints = \{HFlightConstraint_1, \ldots, HFlightConstraint_H\} \tag{4.8}$$

The union of all $HFlightConstraint_i$ is the set $HFlightConstraint$. *Example:* turning radius of UAS, climb rate, etc.

There are S sets of soft flight constraints expressed as subsets of *Space*:

$$SFlightConstraints = \{SFlightConstraint_1, \ldots, SFlightConstraints_S\} \tag{4.9}$$

The union of all $SFlightConstraint_i$ is the set *SFlightConstraint*. *Example:* sharp maneuvers, restricted flight areas etc.

The *Weather* may impose either soft or hard flight constraints.

4.1.4 Partition of Space

In what follows it is convenient at each time t to partition the *Space* into four disjoint subsets *Free(t)*, *Restricted(t)*, *Occupied(t)* and *Uncertain(t)*.

There is a *SpaceClassification* function (4.10)

$$SpaceClassification : y \in Space \mapsto s \in \{Free, Restricted, Occupied, Uncertain\} \tag{4.10}$$

Note. SpaceClassification (4.10) is a *total* function mapping each element of *Space*. Other followup *classifications* are considered as *total* functions.

There is set of *Space* Hard constraints, where one *HardConstraint* (4.11) is given as:

$$HardConstraint = \{point \in Space : SpaceClassification(point) = Occupied\} \tag{4.11}$$

There is set of *Space* Soft constraints, where one *SoftConstraint* (4.12) is given as:

$$SoftConstraint = \{point \in Space : SpaceClassification(point) = Restricted\} \tag{4.12}$$

4.1.5 Information Source

Definition 1. *Information source (4.13) represents a priori information for each* point *in* Space *at time prior mission execution mapping it into one of distinctive categories* Free, Occupied, Restriced, *and* Uncertain.

$$InformationSource = SpaceClassification(PriorTime) \tag{4.13}$$

Definition 2. *Landmark is a partition of* Space *(4.1) which is notable in the context of* society *or* law *given properties. Landmark can be notable buildings, notable natural structures or crucial infrastructure. Landmark is part of terrain map, but it's special status can induce additional properties.*

For obstacle avoidance problem following *Information sources* are available:

1. *Terrain map* - a map of terrain with notable landmarks.
2. *Object map* - a map of notable structures with *protection zones* considered as *occupied* or *restricted* space.
3. *Fly zones restriction map* - a map of restricted flight areas considered as *restricted* constraints.

4.1.6 Single Observation by Single Sensor Classification

The *UAS* is equipped with single *Sensor*, which returns *Space* classification for given *observation position* and observation time.

Observation (4.14) at given UAS *position*, *time*, and for given *sensor* sorts points from *sensor reading* into following distinguish sets (4.14):

1. $Free_O(position, sensor, time)$ - observable space by sensor and considered as $Free$ by sensor reading,

2. $Occupied_O(position, sensor, time)$ - observable space by sensor and considered as $Occupied$ by sensor reading,

3. $Uncertain_O(position, sensor, time)$ - all other points.

$$Observation(position, sensor, time) \rightarrow \begin{cases} Free_O(position, sensor, time) \\ Occupied_O(position, sensor, time) \\ Uncertain_O(position, sensor, time) \end{cases} \tag{4.14}$$

4.1.7 Multiple Observations by Single Sensor Classification

Let Let add pairs of *observation position* and *observation time* in manner $\{(position_1, t_1), (position_2, t_2), \dots (position_k, t_k)\}, k \in \mathbb{N}^+$ that point position is independent and time is ordered in fashion $t_1 < t_2 < \cdots < t_k$.

Free space from multiple sensor *reading* over multiple *positions* is inclusive space because we are obtaining additional information regarding space reachability by sensor reading independent on sensor space and orientation limitation. Therefore the union of single instances of observations is used to represent *Combined free space* $Free_O(sensor)$ (4.15).

$$Free_O(sensor) = \bigcup_{i=\{1,\dots,k\}} Free_O(sensor, position_i, time_i) \tag{4.15}$$

Occupied space (4.16) from multiple sensor *reading* over multiple *positions* is inclusive space, because of increased information; therefore it has similar handling to *Free space*.

$$Occupied_O(sensor) = \bigcup_{i=\{1,\dots,k\}} Occupied_O(sensor, position_i, time_i) \tag{4.16}$$

Uncertain space (4.17) from multiple sensor *reading* over multiple *positions* is exclusive space, because of decrease in uncertainty.

$$Uncertain_O(sensor) = \bigcap_{i=\{1,\dots,k\}} Uncertain_O(sensor, position_i, time_i) \tag{4.17}$$

4.1.8 Sensor Fusion

The observation at fixed time t_{fix} can be made by multiple sensors $sensor_1$, $sensor_2$, ..., $sensor_k$, $k \in \mathbb{N}^k$, for k sensors execute observations $Observation_1$ ($sensor_1$, $position_1$, t_{fix}), $Observation_2$ ($sensor_2$, $position_2$, t_{fix}), ..., $Observation_k$($sensor_k$, $position_k$, t_{fix}).

Sensor Fusion (4.18) function with *data fusion parameters* is introduced which combines *Observations* for each *sensor* and *Weather*, then uniquely maps each point into four distinguish sets: $Free(t_{fix})$, $Occupied(t_{fix})$, $Restricted(t_{fix})$, *and* $Uncertain(t_{fix})$ (special case of (4.10)).

$$SensorFusion: \begin{bmatrix} Observation_1(sensor_1, position_1, t_{fix})\times \\ Observation_2(sensor_2, position_2, t_{fix})\times \\ \times\cdots\times \\ Observation_k(sensor_k, position_k, t_{fix})\times \\ Weather(\dots) \\ \textit{fixed time } t_{fix}\times \\ sensorFusionParameters \end{bmatrix} \rightarrow \begin{cases} Free(t_{fix}) \\ Occupied(t_{fix}) \\ Restricted(t_{fix}) \\ Uncertain(t_{fix}) \end{cases} \tag{4.18}$$

4.1.9 Data Fusion

Multiple *Information Sources*: $InformationSource_1$, $InformationSource_2$, ..., $InformationSource_l$, $l \in \mathbb{N}$, where l is the count of information sources needs to be fused with *Sensor Fusion* function (4.18) outcome at *fixed time* t_{fix}.

Data fusion function (4.19) combines the classification from various *Information Sources* with classification from *Sensor Fusion* function (4.18) for UAS position at *fixed time* t_{fix}, under given *Data Fusion Parameters*.

$$DataFusion: \begin{bmatrix} InformationSource_1\times \\ InformationSource_2\times \\ \times\cdots\times \\ InformationSource_l\times \\ SensorFusion(\dots)\times \\ Weather(\dots) \\ position\times \\ \textit{fixed time } t_{fix}\times \\ dataFusionParameters \end{bmatrix} \rightarrow \begin{cases} Free(t_{fix}) \\ Occupied(t_{fix}) \\ Restricted(t_{fix}) \\ Uncertain(t_{fix}) \end{cases} \tag{4.19}$$

Each *point* in *Space* is uniquely classified into one of sets$Free(t_{fix})$, $Occupied(t_{fix})$, $Restricted(t_{fix})$, $Uncertain(t_{fix})$ (special case (4.10)).

Note. Moreover *Data Fusion* function is covering the case of *Multiple information sources, combined with multiple sensor readings over multiple times*, including *Weather* as *sensor impact factor* and *information source*.

4.1.10 Known World

Known world (4.20) for some *fixed time*t_{fix} is given as the joint set of points which belongs to one of *Data Fusion* (4.19) output sets $Free(t_{fix})$ or $Occupied(t_{fix})$ or $Restricted(t_{fix})$. The *Known World* is compact set with existing boundary.

$$KnownWorld(t_{fix}) = Free(t_{fix}) \cup Occupied(t_{fix}) \cup Restricted(t_{fix}) \tag{4.20}$$

4.1.11 Safety Margin

Let say that *mission* was executed in *time* interval $t \in [missionStart, missionEnd]$ in *Known world* (4.20). For every *position*, extracted from *UAS model state* $x(t)$, keeps distance from any point in $Occupied(t)$ with greater or equal to $safetyMargin$ (s_m) (4.21).

$$\forall t \in [missionStart, missionEnd]: \\ distance(x(t), Occupied(t), t) \geq safetyMargin \tag{4.21}$$

4.2 Basic Obstacle Avoidance Problem

Given:

1. Initial system state x_0, for UAS model (4.3).
2. Mission (4.6) to be executed.
3. Space (4.1), with existing objects (4.2).
4. Sensor system $\{sensor_1, sensor_2, \ldots, sensor_k\}$ with existing sensor fusion function (4.18).
5. Weather information (4.5) during the flight is available.
6. Information sources $informationSource_1$, $informationSource_2$, $\ldots$, $informationSource_l$ containing hard (4.11) and soft space constraints (4.12),with existing data fusion function (4.19).
7. *Hard* (4.8) and *Soft* 4.9 flight constraints given by ATM and rules of the air.

Generate *Control command chain to* complete a mission (4.6) with the satisfaction of following conditions:

1. Waypoint passing function condition (4.7).
2. Set safety margin s_m to *Occupied* space in $KnownWorld(time)$ (4.20) is not breached at any time (4.21).

4.3 Initial Assumptions

Initial assumptions are the following:

Assumption 1. *Filtered sensor readings are available.*
SensorObservation *(4.14) for a given* position, time *returns classification of* Space *which is corresponding with the real situation.*

Assumption 2. *There are no moving obstacles.*
The initial Space Classification Function *(4.10) is static for all observation times* $t \in (-\infty, \infty)$. *Moreover, there are no* intruders *or* adversaries *present.*

Assumption 3. *The movement takes place in the unrestricted airspace.*

Assumption 4. *The mission consists of a set of reachable waypoints.*
For specific UAS system *(4.3) and* Mission *(4.6), there exists a control which satisfies* Waypoint passing *(4.7) criterion and* SafetyMargin *(4.21) condition.*

Assumption 5. *The UAS is moving with constant velocity.*
For given UAS system *(4.3) there is a subset of state* $velocity(t) \subset x(t)$ *which contains velocity parameters. Then there exist transformation function* $LinearVelocity(\circ)$ *which maps* $velocity(t)$ *to* linear velocity $\in \mathbb{R}^1$. *For time* t *in* missionStart *and* missionEnd *in* Mission *(4.6) constraint (4.22) with some* $constantVelocity \in \mathbb{R}^+$ *holds.*

$$\forall t \in \begin{bmatrix} missionStart, \\ missionEnd \end{bmatrix} : LinearVelocity(velocity(t)) = constantVelocity \tag{4.22}$$

Note. Initial assumptions 1., 2., 3., 4, and 5. will be relaxed in *Incremental problem definition.*

4.4 Incremental Problem Definition

This section contains *incremental problem definition* as increments of (sec. 4.2). Each problem contains definition and references to addressed issues.

Problem 1. *Basic Avoidance (sec. 4.2) is to navigate through* KnownWorld *under the assumption that every* waypoint *in* Mission *is reachable. The* KnownWorld *is fed through* SensorFusion *function which is joining* LiDAR *scanning into* Free(t), Occupied(t), *and,* Unknown(t) *sets in* discrete scan times t.

$$\begin{aligned} KnownWorld &:= SensorFusion(t) \forall point \in KnownWorld(t) \\ &= Free(t) \cup Occupied(t) \cup Unknown(t) \\ Mission &:= \forall waypoint \in Mission \text{ are reachable} \\ Sensors &:= \{LiDAR\} \\ SensorFusion &:= \{Clasificaiton\ function\} \\ HFlightConstraints &:= \{vehicle\ dynamic\} \end{aligned} \tag{4.23}$$

Challenges for problem 1. :

1. Navigation Loop Implementation *(sec. ??).*
2. Avoidance Loop Implementation *(sec. ??).*

Problem 2. *Intruder Problem in addition to* Known world evolution *(pr.1) the* ADS-B *sensor is introduced into* Sensors *array, this imposes* HardConstraint *of* Flight corridor *for detected intruders impacting the evolution of* Free*(t), and* Occupied*(t) sets significantly.*

$$\begin{aligned} KnownWorld &:= SensorFusion(t) \forall point \in KnownWorld(t) \\ &= Free(t) \cup Occupied(t) \cup Unknown(t) \\ Mission &:= \forall waypoint \in Mission \text{ are reachable} \\ Sensors &:= \{LiDAR, ADS-B\} \\ SensorFusion &:= \{Advanced\ joint\ sets\} \\ HFlightConstraints &:= \{vehicle\ dynamic\} \\ HardConstraints &:= \{intruder\ corridors\} \end{aligned} \tag{4.24}$$

Challenges for problem 2. :

1. Intruder Intersection Models *(minimal operation requirements achieved):* Linear Intersection Model *(app. ??),* Body-volume intersection *(app. ??),* Maneuverability uncertainty intersection *(app. ??).*
2. Flight Corridors *(sec. ??).*

Relaxed Assumption: *2., the* UAS *encountering cooperative and non-cooperative intruders.*

Problem 3. *Static restrictions, in addition to the* Intruder *problem (pr. 2) the* Information-Sources *are expanded by* static restriction *sources:*

1. ObstacleMap - *a database containing notable landmarks, buildings, structures, with well-defined* protection zones.

2. FlightRestrictions - *a database containing ATM flight restrictions in non-segregated airspace for UAS relevant airspace categories.*

This change impacts DataFusion *by splitting* Free(t) *set into* Free(t) *and* Restricted(t) *disjoint sets. Also* SoftConstraints *are introduced which contain restricted areas from relevant information sources.*

$$\begin{aligned}
KnownWorld &:= DataFusion(t) \forall point \in KnownWorld(t) \\
&= Free(t) \cup Occupied(t) \cup Unknown(t) \cup Restricted(t) \\
Mission &:= \forall waypoint \in Mission \text{ are reachable} \\
Sensors &:= \{LiDAR, ADS-B\} \\
SensorFusion &:= \{Advanced\ joint\ sets\} \\
InformationSources &:= \{TerrainMap, ObstacleMap, FlightRestriction\} \\
DataFusion &:= \{Advanced\ data\ fusion\} \\
HFlightConstraints &:= \{vehicle\ dynamic\} \\
HardConstraints &:= \{intruder\ corridors, terrain, obstacles\} \\
Softconstraints &:= \{protection\ zones\}
\end{aligned} \tag{4.25}$$

Challenges for problem 3. :

1. Obstacle Map *(fig.* **??***).*

2. Visibility Rating Concept *(fig.* **??***).*

3. Static Constraints *(sec.* **??***).*

Relaxed Assumption: *3., the* UAS *is moving in* restricted space *now.*

Problem 4. *Dynamic restrictions in addition to* Static restrictions *(pr. 3), the* Weather *as information source is introduced.* Soft constraints *are extended by medium level dangerous zones from weather map.* Hard constraints *are expanded by protection zones where the* weather *conditions are harsh. Overall* Weather *constraints are dynamic and changing position and shape over mission time. Modern weather systems can provide streamline overview of weather situation.*

$$
\begin{aligned}
KnownWorld &:= DataFusion(t) \forall point \in KnownWorld(t) \\
&= Free(t) \cup Occupied(t) \cup Unknown(t) \cup Restricted(t) \\
Mission &:= \forall waypoint \in Mission \text{ are reachable} \\
Sensors &:= \{LiDAR, ADS-B\} \\
SensorFusion &:= \{Advanced\ joint\ sets\} \\
InformationSources &:= \{TerrainMap, ObstacleDatabase, \\
&\quad FlightRestriction, Weather\} \\
DataFusion &:= \{Advanceddatafusion\} \\
HFlightConstraints &:= \{vehicle\ dynamic\} \\
HardConstraints &:= \{intruder\ corridors, terrain, obstacles,\ protection\ zones\} \\
Softconstraints &:= \{protection\ zones\}
\end{aligned}
\tag{4.26}
$$

Challenges for problem 4. :

1. Moving Constraints including Weather *(def.* **??***).*
2. Weather Avoidance Case *(app.* **??***).*

Problem 5. *Rules of the air, in addition to* Dynamic restrictions *(pr. 4),* Rules of the air *framework introduction, inducing new* SFlightConstraints *including air-spaces and rules of air impact on control mechanism.*

$$\begin{aligned}
KnownWorld :&= DataFusion(t) \forall point \in KnownWorld(t) \\
&= Free(t) \cup Occupied(t) \cup Unknown(t) \cup Restricted(t) \\
Mission :&= \forall waypoint \in Mission \text{ are reachable} \\
Sensors :&= \{LiDAR, ADS-B\} \\
SensorFusion :&= \{\text{Advanced joint sets}\} \\
InformationSources :&= \{TerrainMap, ObstacleDatabase, \\
&\quad FlightRestriction, Weather\} \\
DataFusion :&= \{Advanceddatafusion\} \\
HFlightConstraints :&= \{\text{vehicle dynamic}\} \\
SFlightConstratins :&= \{\text{airspaces, rules of the air}\} \\
HardConstraints :&= \{\text{intruder corridors,terrain,obstacles, protection zones}\} \\
Softconstraints :&= \{\text{protection zones}\}
\end{aligned} \tag{4.27}$$

Challenges for problem 5. :

1. *UTM Implementation (sec.* **??***).*
2. *Rule Engine for UAS (sec.* **??***).*
3. *Rule Implementation (sec.* **??***).*

Relaxed Assumption: *5., the* UAS *is required to move with different velocity during* Overtake maneuver.

Note. The assumptions 1. for *filtered sensor output* and 4. *Reachable waypoints* hold for all *problem increments.*

4.5 Avoidance Requirements

SAA systems have the following conflicting performance criteria:

1. *Energy efficiency* - minimize energy consumption and flight time.
2. *Trajectory tracking* - stick to the proclaimed trajectory in a mission plan.
3. *Safety* - avoid harm sources during the mission execution.

Note. Energy efficiency and *Trajectory Tracking* an optional criteria, while *the safety* is mandatory.

4.5.1 Energy Efficiency

Energy efficiency can be measured by *cost function* (eq. 4.28), consisting from the *cost of flown trajectory* (eq. 4.29) and the *expected reach cost* (eq. 4.30) portions. There are optimalizaiton techniques based on *Reach sets* [1]. The inputs for the *cost function* are:

1. *Time* - current mission time.
2. *Initial state* - UAS state at the beginning of a *mission*.
3. *Applied movements* - list of already executed movements.
4. *Future movements* - list of movements to be applied in future.
5. *Current state* - UAS state at *Time*, with current position and orientation included.
6. *Waypoint* - current goal waypoint.

$$Cost(t, \dots) = costTrajectoryFlown(t, \dots) + expectedReachCost(t, waypoint, \dots) \quad (4.28)$$

Cost of flown trajectory (eq. 4.29) from the *initial state* to the *current state* is calculated as a sum of energy consumed for each movement with the following components:

1. *Direct cost* - a cost of consumed energy to execute the movement.
2. *Horizontal cost* - a portion of the direct cost which was used for horizontal steering multiplied by *horizontal penalization*.
3. *Vertical cost* - a portion of the direct cost which was used for ascending/descending multiplied by *vertical penalization*.

$$costTrajectoryFlown\begin{pmatrix} time, \\ initialState, \\ appliedMovements \end{pmatrix} = \sum_{\substack{movement \in \\ appliedMovements}} \begin{pmatrix} duration.directCost+ \\ horizontal(cost, penalization)+ \\ vertical(cost, penalization) \end{pmatrix} \quad (4.29)$$

Expected reach cost (eq. 4.29) is calculated for a *planned trajectory* portion and a *direct waypoint distance* to the *latest future UAS position*. *Cost of the planned trajectory* is calculated by the same formula as a *cost of flown trajectory* (eq. 4.29), the initial state is replaced with a *current state*, and *executed movements* are replaced with *planned movements*.

$$expectedReachCost\begin{pmatrix} time, \\ currentState, \\ futureMovements, \\ waypoint \end{pmatrix} = \begin{pmatrix} distance(futureState, waypoint)+ \\ costTrajectoryFlown\begin{pmatrix} currectState, \\ futureMovements \end{pmatrix} \end{pmatrix} \quad (4.30)$$

Note. The tuning parameters of cost function are *Horizontal penalization* $\in [0, \infty]$ and *Vertical penalization* $\in [0, \infty]$. Which are used to enhance the outcome of the *cost function*.

Following setup of tuning parameters are used in our simulations:

1. $horizontalPenalization \leq verticalPenalization < \infty$ - in *uncontrolled airspace*, all kind of maneuvers are allowed. Horizontal maneuvers are cheaper for a plane UAS.
2. $horizontalPenalizaiton < vetricalPenalization = \infty$ - in *controlled airspace* any kind of horizontal maneuvering must be allowed by UTM.

The tuning parameters are set up $verticalPenalizaiton \leq horizontalPenalizaiton < \infty$ for *copter* UAS in an *uncontrolled airspace*

4.5.2 Trajectory Tracking

Trajectory Tracking is a crucial parameter for *controlled airspace* and is expected to be important in *upcoming UTM systems*. There is a *mission plan* which is compared with *real-time airspace situation* obtained from UAS *Position notifications*. The optimalization based on *Reach Set* is given in [2].

Motivation: *Situation awareness* for modern DAA systems depends on planned trajectory tracking. The main conflict is between *navigation precision* and *situation evaluation*. If the planned trajectory is defined for the *continuous domain*, it takes much effort to calculate collision points.

The discrete domain of *Movement Automaton* (def. **??**) can be used as a *tool for situation awareness*. The main idea is to use *Movement automaton as a predictor for trajectory intersection* [3, 4].

Movement Automaton trajectory tracking: There is a *reference trajectory* which is used as comparison by *aviation authority* (ATM,UTM) given as:

$$ReferenceTrajectory = \{(point_1, time_1), (point_2, time_2), \ldots, \\ , \ldots (point_n, time_n)\} \quad n \in \mathbb{N}^+, point_k \in \mathbb{R}^3 \tag{4.31}$$

Reference Trajectory (eq. 4.31) is given as set of *points* in *Global Coordinate System* for given *UAS*, *operational time-frame* and other authority depending properties.

The *movement* automaton is executing *Trajectory(initialState,buffer)* where buffer is set of *Movements*. The buffer is changing according to following pattern during *mission* time frame:

$$\begin{aligned}
&\texttt{Mission Start:} \\
&\quad buffer = \{plannedMovements\}, planned = m \\
&\texttt{During Mission:} \\
&\quad buffer = \{executedMovements, plannedMovements\}, \\
&\qquad executed = n, planned = o \\
&\texttt{Mission End:} \\
&\quad buffer = \{executedMovements\}, executed = p; \\
&\texttt{Movement count constraints:} \\
&\quad m, n, o, p \in \mathbb{N}^+, n + o = m, m \leq p
\end{aligned} \tag{4.32}$$

At the beginning of the mission (eq. 4.32) the buffer is filled with m movements, the *Trajectory* generated from this buffer and *initial state* is *predicted*.

During the *mission execution phase*, the buffer contains *executed movements* and *planned movements*. Trajectory created from the *initial state* and this buffer can be split into:

1. *Executed part* - trajectory portion generated and executed from *executed movements*
2. *Predicted part* - trajectory portion generated as a future reference from *planned movements*

After *mission* execution, there is only *executed movements* The trajectory generated from the *initial state* and buffer is *Executed Trajectory*

Note. The part of the trajectory bounded to the past, the part of the trajectory lies in the future. The strong point of *Movement Automaton* is its ability to work as *predictor* and *trajectory memory* at the same time.

By selecting proper time series $t_1 \dots t_n$ one can compare future or past segments of trajectory (eq. 4.32) with reference (eq. 4.31)

Reference Trajectory Deviation for reference trajectory given by (eq. 4.31) and *Trajectory segment* (Executed/Predicted) (4.32) with existing *State projection function* (eq. **??**) for *time series* is given as:

$$Deviation \begin{pmatrix} timeSeries, \\ Trajectory, \\ Reference \end{pmatrix} = \sum_{\substack{time_i \in \\ timeSeries}} \begin{pmatrix} StateProjection(Trajectory, time_i) \\ - \\ Reference(time_i) \end{pmatrix}^2 \tag{4.33}$$

Reference Trajectory Deviation (eq. 4.33) is designed as discrete *Mean Square Error* function, where the *timeSeries* is set of *times* from *reference trajectory* $(point_i, time_i)$ pair. The *state projection.*

Trajectory tracking is defined as *dual minimization problem* where the *primary objective* is depending on the *airspace type*:

1. *Reference trajectory deviation* (eq. 4.33) in *Controlled Airspace.*
2. *Cost of Flown Trajectory* (eq. 4.28) in *Non-controlled Airspace.*

Trajectory tracking can be defined as an optimization problem (eq. 4.34).

$$\begin{array}{lll} \texttt{Minimize:} & costOfTrajectoryFlown & (4.28) \\ \texttt{Minimize:} & referenceTrajectoryDeviation & (4.33) \\ \texttt{Subject to:} & & \\ & UAS\ Dynamics & (\textbf{??}) \\ & & (\textbf{??}) \\ & MovementAutomatonControl & \vdots \\ & & (\textbf{??}) \\ & Mission & (4.6) \\ & KnownWorld(t) & (4.20) \\ & SafetyMargin(t) & (4.21) \\ & HardFlightConstraints & (4.8) \\ & SoftFlightConstraints & (4.9) \\ & HardSpaceConstraints & (4.11) \\ & SoftSpaceConstraints & (4.12) \end{array} \tag{4.34}$$

The *reference trajectory* is given by *mission* set of *waypoints*. The *UAS* dynamics with specific *Movement Automaton* goal is to fly in *Known World* to keep *Safety Margin* form *Obstacle Space*. The *Obstacle space* is a result of *Data fusion* procedure (sec. 4.1.9) combining the *sensor reading*, information sources, and constraints.

Feasible Trajectory for *tracking problem* (eq. 4.34) is a trajectory which in addition to *basic obstacle problem* (sec. 4.2) keeps deviation from the *reference trajectory* under certain threshold:

$$Deviation(timeSeries, Trajectory, Reference) \leq performanceMargin \in \mathbb{R}^{+} \tag{4.35}$$

Feasible trajectory condition (eq. 4.35) is used as *margin* for airworthiness, and *Deviation* is used as a performance indicator further in this work.

4.5.3 Safety

Safety is very broad term there are following incidents which can occur and will be discussed in (app. **??**) *Safety margin* is a broad term describing minimal distance to the center of intruder/adversary, a surface of the obstacle, a boundary of the protected area.

Controlled airspace safety Safety for *controlled airspace* in given *flight level* is given a list of incidents:

1. *Soft constrained zone breach* - UAS fly to *soft constraint body* or *hard constraint protection zone*, these incidents can happen, and have least avoidance priority.
2. *Hard constrained zone breach* - UAS fly to *hard constraint protection zone*, typical geo-fencing, restricted airspace breaches.
3. *Well-clear breach* - UAS fly to *well-clear barrel* without impacting other aircraft, via the wake turbulence or other induced physical phenomenon and vice-versa. This type of breaches are allowed in case of inevitable *near miss situations* or *Clash incidents*
4. *Near miss situation*- UAS fly to *near miss* cone/barrel, inducing wake turbulence, or other kinds of flight disturbance. These incidents are allowed at very low rate (near $1 : 10^6$).
5. *Clash incident* - UAS body impacts another aircraft hull/propulsion/steering systems and components. This kind of incidents are very severe, and they should never happen.

Note. It is assumed that flight level in controlled airspace is free of terrain, static ground obstacles, the climb/descent maneuvers are not covered in this work, and they are topic for multiple dissertation theses. For more information refer to ICAO document 4444.

The *relation* for breach of *safety margin* and *body margin* for each object is given in (tab. 4.1):

Violation of:	Safety Margin	Body Margin
Soft constraint	none	Soft constraint zone breach
Hard constraint	Soft constrained zone breach	Hard constrained zone breach
Intruder	Well clear breach, Near Miss situation	Clash incident

Table 4.1: Controlled airspace margins violations incidents.

Uncontrolled airspace safety Safety for *uncontrolled airspace* is applied in F/G class of airspace, which is given as airspace between the *ground* level and *other airspace prevalence*.

Note. Clarification of controlled/uncontrolled airspace:

1. *Class F* airspace is given as space between the ground level or water surface, and it is constrained up to the 500 feet above ground level.

2. *Class C* airspace or *Controlled airspace* is considered starting at first flight level, which is given by Air traffic control zone starting at least at 300 feet from highest ATC zone ground point. It is measured based on Above Sea Level altitude. *This is not a problem in Portugal, because of terrain diversity, but its a huge problem in the Netherlands.*

3. *Class A* airspace starts at ground level and covers the majority of airport infrastructure - this is not a problem, because it's modeled as hard constraint, which is unbreakable in non controlled airspace.

Safety for *uncontrolled airspace* is given a list of incidents:

1. *Soft constrained zone breach* - UAS fly into the *soft constrained zone* or *hard constraint protection* zone, it is allowed to happen on very a low rate.

2. *Hard constrained zone breach* - UAS fly into the *hard constrained zone*, only airports and critical infrastructure are considered as hard constraints; it is not allowed to happen.

3. *Intruder near miss* - UAS fly into *other aircraft near miss zone*, it is allowed to happen on a very low rate in case of other intermediate threats with higher priority.

4. *Intruder clash* - UAS has contact with other man-made aircraft, it is not allowed to happen.

5. *Adversary clash* - UAS has contact with another flying object which did not intentionally avoided UAS. (*Bird strike, Differential games,etc..* is out of the scope of this thesis).

6. *Structure harm* - UAS fly close to structure, and its propulsion can damage/harm structure.

7. *Structure crash* - UAS fly into a natural/man-made ground structure (building, tree, human).

8. *Ground harm* - UAS fly close to the ground, and its propulsion reflection can impact Ground or UAS.

9. *Ground collision* - UAS collides with the ground.

The *relation* for breach of *safety margin* and *body margin* for each object is given in (tab. 4.2):

Violation of:	Safety Margin	Body Margin
Soft constraint	none	Soft constraint zone breach
Hard constraint	Soft constrained zone breach	Hard constrained zone breach
Intruder	Intruder near miss	Intruder clash
Adversary	none	Adversary clash
Structure	Structure harm	Structure crash
Ground	Ground harm	Ground crash

Table 4.2: Non-controlled airspace margins violations incidents.

4.6 Navigation Requirements

Navigation requirements are not the main part of this work; they are used to show the variability of the approach for *DAA* requirements:

1. *Contextual behavior* - change navigation and decision behavior based on context:
   a. *Airspace type* - Controlled/Uncontrolled,
   b. *Navigation mode* - Cooperative/Emergency.
2. *Determinism* - same result for same dataset in finite time.
3. *Threat prioritization* - threats prioritization based on *context*, (tab. 4.1) and (tab. 4.2).
4. *Rule compliance* - compliance with a given set of rules based on context (focus on rules of the air).

Requirement	Evaluation metrics
Constrained space navigation	Mission scenario does not have a direct path between waypoints, additional borderline cases.
Contextual behavior	Avoidance system changes behavior based on the mission and vehicle context.
Determinism	Multiple runs of same non-borderline scenario returns same avoidance paths.
Rule compliance	Rules applied comply with aviation standardization.

Table 4.3: Navigation requirements evaluation metrics.

Bibliography

[1] Alexander B. Kurzhanski and Pravin Varaiya. Dynamic optimization for reachability problems. *Journal of Optimization Theory and Applications*, 108(2):227–251, 2001.

[2] Pravin Varaiya. Reach set computation using optimal control. In *Verification of Digital and Hybrid Systems*, pages 323–331. Springer, 2000.

[3] Emilio Frazzoli, Munther A Dahleh, and Eric Feron. Trajectory tracking control design for autonomous helicopters using a backstepping algorithm. In *American Control Conference, 2000. Proceedings of the 2000*, volume 6, pages 4102–4107. IEEE, 2000.

[4] Emilio Frazzoli. *Robust hybrid control for autonomous vehicle motion planning*. PhD thesis, Massachusetts Institute of Technology, 2001.