# Appendix G

# Additional UTM functionality

This appendix contains additional UTM functionality description and detailed rule body implementations.

## G.1 Weather Case

**Motivation:** The weather, as defined in (eq. **??**), impacts flight and system dynamics; therefore it impacts the *reach set* is impacted. The *weather impact* can be solved by policy application:

1. *Weather Acceptance* - for bigger *UAS* the normal weather impact does not pose a significant risk. The *segmented movement automaton* (def. **??**) with *Weather situation* as the discrete state is used.

2. *Weather Avoidance* - all *weather* impact zones are considered as hard constraints with protective *soft-constraint* around.

3. *Combined approach* - depending on the type of impact and declared UAS impact resistance the zones are divined into *soft* and *hard* constraints.

*Note.* This work handles small *UAS* avoidance; these are very sensitive to any weather impact; therefore *Weather impacted areas* will be considered as *hard constraints with soft constraint protection zone.*

The original *weather impact zone* is considered as obstacle body and enforces the body margin.

The surroundings of *weather impact zone* up to *safety margin* distance are considered as *soft constraint zone* (implemented as a bloated polygon).

**Purpose:** The *weather case* (tab. G.1) is broadcasted by *Airspace Authority* to *impacted area*, each *UAS* then change their mission according to *their maneuvering capabilities.* Each trajectory must lead away from the *constrained area.* The algorithm used for intersection selected based on [1] the selected algorithm *Shamos-Hoey* [2].

**Constrained Area:** Constrained area can be defined as *static* (sec. **??**) or dynamic constraint (sec. **??**). The *constraint center* is defined on horizontal plane like follow:

$$ConstraintCenter = center \in [latitude, longitude] \tag{G.1}$$

The *Convex Polygon* boundary is defined on horizontal plane, contains at least 3 vertexes:

$$ConvexPolygon = \{point_i : point_i \in [latitude, longitude], i \geq 3\} \tag{G.2}$$

The *Vertical constraint* is defined as *range of barometric altitude* (Above Mean Sea Level):

$$VerticalConstraint = [startAltitude, endAltitude] \tag{G.3}$$

**Additional parameters** : Following additional parameters with additional purpose can be attached to *Weather Constraint.*

1. *Type* - defines required resistance - moisture, temperature, wind.
2. *Severity* - defines the impact for each *aircraft category*, this is used in soft/hard type assessment.
3. *Duration* - start and end of *constraint* validity, if not defined valid for all *UAS mission time.*
4. *Velocity* - velocity and last position assessment time.

*Note.* Our implementation does not consider the *type* or *severity.* All *weather impact* is considered as a *hard constraint.* The velocity differentiates *static* $(= 0)$/*moving* $(> 0)$ *constraints.*

**Avoidance System:** Resolve similar to *Converging/Overtake Maneuver* depending on the *angle of approach.* The *virtual roundabout* is utilized for *static constraints*; the *intruder model* is utilized for *dynamic constraints.*

| **Constrained area** | |
|---|---|
| center position | is given as a geometrical *center point of the boundary.* |
| boundary | is represented as a *convex polygon* on the latitude-longitude plane. |
| start altitude | is lover boundary barometric altitude given at above mean sea level, where given weather factor has a significant impact. |
| end altitude | is upper boundary barometric altitude given at above mean sea level, where given weather factor has a significant impact. |

**Additional parameters**

| | |
|---|---|
| type(s) | lists weather events occurring in the *constrained area.* |
| severity list | is recorded for each plane *category* |
| start | indicates when weather constraint was established. |
| expected end | of weather constraint. |
| velocity | indicates if weather phenomenon is moving. |
| **Miscellaneous** | |
| previous | reference to *weather constraint* decision time-frame data. |
| impacted | list of possibly impacted attendees (planes which obtained divergence order or warning from UTM). |

Table G.1: Static/Dynamic weather constraint for given decision time-frame.

## G.2 Rule: Detect Collision Cases

This rule is activated each *UAS avoidance run. UTM* sent out all related *collision cases* (G.4) based on our *UAS identifier.* Creation of *collision case* is given in (sec. **??**) based on air traffic periodical *position notifications* (sec **??**).

$$UTM \times timeFrame \rightarrow UTMCollisionCases \tag{G.4}$$

If there are available *position notifications* (sec **??**) from surrounding air-traffic, UAS will calculate own *collision cases* (G.5).

$$uasStatus \times positionNotification \times utmTimeFrame \rightarrow UASCollisionCases \tag{G.5}$$

Then UAS merges *own collision cases* with *UTM collision cases*, if there exist following disparities UAS will take action:

1. $distamce(ownCollisionPoint, utmCollisionPoint) \geq threshold$, send UTM notificaiton, use $utmCollisionPoint$
2. $utmMargin \geq ownMargin$, use safety margin from UTM.
3. $utmAvoidanceRole == active, ownAvoidanceRole == inactive$, use UTM avoidance role.
4. $utmCollisionCase == active, ownCollisionCase == uncertain$, use UTM provided collision case, not all *position notifications* are available.
5. $utmCollisionCase == inactive, ownCollisionCase == active$, notify UTM with new collision case, ignore collision case until UTM approves.

*Note. Avoidance role* is classified as *inactive* if and only if UAS has the *right of way*, it is classified as *active* otherwise.

*Safety margin* determined by UTM has priority because not all calculations factors are available for UAS.

*Collision Case* unknown to UTM are ignored, due to safety reasons (false data spoofing), collision case is activated after UTM confirmation. If there is real intruder not confirmed by UTM, it is handled via *non-cooperative* or *emergency* avoidance procedure

The *selection process* of active *collision cases* is based on UAS *avoidance role* in each *collision case.*

1. If the *avoidance roles* are following: *Head On Approach, Converging Maneuver, or Overtake* in all *collision cases* UAS system will stay in cooperative mode.

2. If there exists at least one *collision case* with *own avoidance role* or *intruder avoidance role* set as *avoid-emergency*, the UAS will notify UTM and ask for *diversion order*; meanwhile it sets itself into *Emergency avoidance* mode.

3. If there exist multiple *Overtake avoidance roles* or combination of *Overtake avoidance role* and *Another active role*, the UAS will decrease its cruising speed like follows:

$$UASSpeed = \max \begin{Bmatrix} minimalUASCruisingSpeed, \\ \min\{intruderSpeed\} \quad \forall activeCollisionCases \end{Bmatrix} \tag{G.6}$$

   During *slow-down* UAS switches to *emergency avoidance mode* and asks for *divergence order* from UTM.

The rule is summarized in table G.2.

| | | |
|---|---|---|
| *Invocation:* Every *Decision point* in*UAS main loop*<br><br>*Objective:*<br><br>1. Fetch UTM *Collision cases* for a given decision time frame.<br>2. Create/update *own collision cases* based on received *Position notifications* from surrounding *Intruders*.<br>3. Merge *Collision cases* based on *UTM priority order*.<br>4. Select active *collision cases* based on the following conditions:<br>a. *Active participation* in *collision case* where *avoidance role* $\neq$ *Right of the way*.<br>b. *Collision point* is in the front of UAS.<br>c. *Emergency mode detection* there exists at least one non-cooperative participant.<br>5. Order *collision cases* based on *severity*.<br>6. If there is at least one *active collision case* enforce rule *Resolve collision case* (tab G.3) for each *active collision case*. | | |
| *Context* | *Condition* | *Application* |
| UAS Mission control,<br>Before Avoidance Run,<br>UTM/UAS collision cases | Clean *avoidance grid*,<br>No emergency | Active collision case selection, Prioritization |

Table G.2: Detect collision cases rule definition.

The *ordering of collision cases* starts if and only if the *UAS* is in *cooperative avoidance mode*. The cases are ordered for processing based on severity rating which is calculated based on:

1. *Safety Margin* - the greater safety margins are prioritized.
2. *Intruder vehicle class* - the more dangerous intruders are prioritized.
3. *Collision point distance* - closer collision points are prioritized.
4. *UAS avoidance role* - *Head on Approach* is favored upon *Converging maneuver*, due to direct collision severity.

*Rule engine invocation* for each *active collision case* is then applied on *descending severity sorted* list.

## G.3 Rule: Resolve Collision Case

*Active collision cases* are processed one by one. All collision cases are applied to *Navigation grid*. *Navigation grid* contains all possible *trajectories* in the form of *Reach set*. All *trajectories* are *reachable* at the beginning of the UAS *avoidance frame*. Each application of *collision case resolution* rule disables some subset of feasible *trajectories*. For this reason are *active collision cases* sorted by severity.

It is assumed that UAS is in *cooperative avoidance mode*. If the previous application of this rule forced UAS into *emergency mode*, the rule is not applied to save system resources. *Emergency* mode is invoked if *rule application* disables all *trajectories* in *Navigation grid*. If there is at least one *feasible trajectory* in *avoidance grid* follow-up rule is invoked based on UAS *avoidance role*.
The rule is summarized in table G.3.

*Invocation:* This rule is invoked if exists at least one *active collision case* in given *navigation grid time-frame*; moreover *avoidance grid* must be empty and *cooperative avoidance mode* is enforced.

*Objective:* Based on *active collision case* and *UTM directives* enforce behavior based on *own avoidance role*:

1. *Head on approach* - rule G.5.
2. *Converging maneuver* - rule G.6.
3. *Overtake* - rule G.7.
4. *Emergency mode* - switch from *active avoidance mode* to *emergency mode*.

| *Context* | *Condition* | *Application* |
|---|---|---|
| UAS mission control, Trajectory restriction, Collision cases, | Active merged collision case, Resolution mandate from UTM | Enforce Rules of Air<br>or<br>Enforce emergency |

Table G.3: Resolve collision case rule definition.

## G.4 Rule: Close Collision Cases

*Collection of rule results* detected by rule G.2 and *resolved* by rule G.3 is done via the *context of the rule engine*. For each *time-frame* and each $trajectory \in NavigationGrid$, there exists rule engine *context* query (G.7) which returns *trajectory status* and *list of applied rules on trajectory*.

$$Context(trajectory, timeFrame) \rightarrow \{State : Enabled/Disabled, Rule(s)\} \tag{G.7}$$

*Calculation of possible trajectories* in *navigation grid* is using *collected rule results* (G.7). If the *trajectory state* and linked *rule reason* are sufficient, the *trajectory* is disabled for the the given *time frame*. *Standard navigation algorithm* is used (sec. **??**) to select *feasible trajectory*.

*Rules of the air* and their application in *General Aviation* cases is consistent. Increasing traffic density can impose new layers of rules, which may cause the *soft deadlock* in *maneuverability*. In this case, *Navigation grid* will have all *possible trajectories* exhausted. The following procedure is executed:

1. UAS switch into *Non-cooperative avoidance mode* or *Emergency avoidance mode* depending on situation severity (One conflict can be handled with *vertical separation* of conflicting aircraft).

2. UAS broadcasts *warning message* to all nearby aircraft, and *separation message(s)* to conflicting aircraft. *Separation message* contains an *expected collision point* and *preferred separation type*. Each conflicting aircraft then reacts and sends *action notification* to UTM.

3. If UAS switches into *emergency mode*, non-cooperative avoidance using *avoidance grid* is induced. Each relevant intruder is projected as *timed body volume intruder* (app. **??**), where *safety margin* is used as *body radius*.

*UAS* notifies *UTM* with *course change*, *planned avoidance trajectory*, *avoidance mode*. *UTM* approves planned changes or sends *plan corrections* (out of scope). The rule summary is given in table G.4.

*Invocation:* There exists at least one *active collision case* which had an impact on *Navigation grid.*

*Objective:* Ensure that multiple *avoidance rules* application gives feasible *avoidance strategy*, enter into *emergency avoidance mode* otherwise. Following steps are executed:

1. *Collect rules* applied on *navigation grid* from *active collision cases.*
2. *Calculate possible trajectories for avoidance*; there may be none.
3. If there is no *feasible route*, for each *intruder* from related *collision cases*:
   a. Issue *warning message* containing *expected collision point* and *preferred separation type.*
   b. Create appropriate *intruder object* for *avoidance grid.*
   c. Calculate *evasive maneuver* based on the expected *separation type.*
4. *Notify UTM* with *collision case resolution* for each *active collision case. Notify UTM* with *planned trajectory* and *avoidance mode*

| *Context* | *Condition* | *Application* |
|---|---|---|
| UAS Mission control,<br>After avoidance run,<br>Collision resolutions | At least one trajectory in<br>Navigation grid,<br>Emergency check | Force *Emergency mode*<br>OR<br>Close Collision Case |

Table G.4: Close collision case rule definition.

## G.5 Rule: Head on Approach

Rule (G.5) is invoked based on the *angle of approach* range condition, defined *collision case* section **??**. The handling of *head on* avoidance is given in section **??**.

*Virtual round-abound* for UAS and intruder is created by UTM. The center of virtual round-abound and *corrections for participants margins* are determined based on:

1. *Collision case center* - contributes to the round-abound center median point.
2. *UAS and intruder maneuverability* - determines *attendants avoidance mode* and *maximal avoidance margins.*
3. *Surrounding air-traffic* - contributes to round-abound center median point, determines ideal *ideal avoidance margins* due to *wake turbulence* prevention.

*Invocation:* When *UAS avoidance role* is *Head on* avoidance and *avoidance grid* is empty.

*Objective:* Ensure that the *UAS body* does not enter into *intruder‘s well clear zone.*

1. Prevent *left-side leading* maneuvers (rule G.8).
2. Prevent head on *safety margin* breach(rule G.9).
3. Return to original course, when *navigation grid* is clear.
4. Prevent *wake turbulence* (by safety margin correction).
5. Enforce *Round-about* behavior (by clustering collision cases).

| *Context* | *Condition* | *Application* |
|---|---|---|
| UAS Navigation Grid, Collision Point, Avoidance role | None | Run rules referenced in objective listing. |

Table G.5: Head on Approach rule definition.

The *virtual round-abound center* is calculated as *corrected median* (G.8) taking *cluster of collision cases* and calculates the median of their collision points corrected by *weather* and *wake turbulence* factor.

$$correctedMedian = \sum_{c_i \in collisionCases} \left(c_i.center + correction\right)/count(collisionCases) + \\ + corrrection(Weather) + correction(WakeTurbulence) \tag{G.8}$$

*Corrected margin* needs to be calculated for each *participating aircraft*, because of the *virtual roundabout* center correction (G.8). Each *round-abound participant* is ordered based on importance (lowest maneuverability first). Then for each *round-abound participant* obtains *corrected margin* (G.9) calculated from *collision case safety margin*, corrections based on other *more important vehicles*, *weather*, *wake turbulence.*

$$correctedMargin = \min \begin{bmatrix} caseMargin + correction \begin{pmatrix} ImportantVehicles, \\ Weather, \\ WakeTurbulence \end{pmatrix} \\ maximalAvoidanceMargin \end{bmatrix} \tag{G.9}$$

## G.6 Rule: Converging Maneuver

The rule is invoked based on the *angle of approach* range defined in *collision case calculation* (sec. ??). Behavior enforced to this rule is equal to rule G.5 except the *intruder* stays on his original path. UAS behavior is described in (sec ??). The *rule summary* is given by (tab. G.6).

*Invocation:* When *UAS avoidance role* is *Converging*, and *avoidance grid* is empty.

*Objective:* Ensure that the *UAS body* does not enter into *intruder's well clear zone*.

1. Prevent *left-side leading* maneuvers (rule G.8).
2. Prevent head on *safety margin* breach(rule G.9).
3. Return to original course, when *navigation grid* is clear.
4. Prevent *wake turbulence* encounter (by safety margin correction).

| *Context* | *Condition* | *Application* |
|---|---|---|
| UAS Navigation grid, Collision point, Avoidance role | None | Run rules from objective. |

Table G.6: Converging maneuver rule definition.

## G.7 Rule: Overtake

During overtake maneuver there is our *UAS* and *Intruder* cruising at same *flight level*. The *angle of approach* ($\alpha$) is lesser than 70°. *UAS* absolute velocity is much greater than *overtaken* absolute velocity.

It is assumed that during *overtake* maneuver *overtaken* intruder will keep constant heading and velocity. If this assumption is broken, the *UAS* system will invoke *Emergency avoidance* procedure. *UTM* will calculate such *divergence* and *convergence* waypoints that *overtake safety condition* (G.10) is satisfied.

$$distance(uasPosition, overtakenPosition) \geq utmMargin, \forall t \in manueverTime \tag{G.10}$$

Where *utmMargin* is calculated based on *Collision case* resolution. The *main idea* is to calculate *Safe offset for Overtake maneuver*, let us have:

$$velocityDifference = \|uasVelocity - overtakenVelocity\| \quad [ms^{-1}, ms^{-1}, ms^{-1}] \tag{G.11}$$

*Decision distance* (G.12) is given as distance when *UTM mandate* takes effectiveness, its assumed that *UAS* knows *UTM decision frame* [s]:

$$decisionDistance = velocityDifference \times uasDecisionFrame \quad [m, ms^{-1}, s] \tag{G.12}$$

*Overtake middle distance*(G.13) is a length of the hypotenuse for triangle where *positional difference* and *utm margin* for overtake are cathetuses:

$$overtakeMiddle = \sqrt{\begin{array}{l} \|uasPosition - collisionPoint\|_2 + \\ + safetyMargin^2 \end{array}} \quad [m, \vec{m}, \vec{m}, m] \tag{G.13}$$

*Safe offset* (G.14) is considered as a combination of *overtake middle distance* (G.13), *decision distance* and uas *waypoint reach margin*.

$$safeOffset = \begin{array}{c} overtakeMiddle \\ +decisionDistance \\ +waypointReachMargin \end{array} \quad [m, m, m, m] \tag{G.14}$$

*Note. Waypoint reach margin* [m] is the property of own *UAS navigation algorithm.* It represents the maximal distance of vehicle position and a waypoint at a time when the waypoint is considered reached.

*Local coordinate frame*: UAS and Overtaken are in Local coordinate frame heading in $X^+$ axis direction ($X^+$ front of aircraft, $X^-$ back of vehicles, $Y^-$ right side, $Y^+$ left side, $flightLevel \to Z = 0$), Collision Point is considered as $\vec{0}$,

*Divergence point* (G.15) in local coordinates is given as right offset of *(*UTM margin) and *decision distance*:

$$divergence = \begin{bmatrix} 0 \\ -decisionDistance - utmMargin \\ 0 \end{bmatrix} \quad [\vec{m}, m, m] \tag{G.15}$$

*Convergence point* (G.16) in local coordinates is given frontal *safe offset* (G.14) and right offset of *UTM margin* and *decision distance*:

$$convergence = \begin{bmatrix} safeOffset \\ -decisionDistance - utmMargin \\ 0 \end{bmatrix} \quad [\vec{m}, m, m] \tag{G.16}$$

*Convergence* (G.17) and *Divergence* (G.18) waypoint in global coordinate frame is obtained via transformation function $R_{XYZ}$ as follow:

$$\begin{aligned} divergenceWaypoint =& collisionPoint \\ &+ R_{XYZ}(overtakenOrientation, divergence) \end{aligned} \tag{G.17}$$

$$\begin{aligned} convergenceWaypoint =& collisionPoint \\ &+ R_{XYZ}(overtakenOrientation, convergence) \end{aligned} \tag{G.18}$$

*Overtake rule* is summarized in (tab. G.7).

*Invocation:* Invoked by rule *Collision Case Resolution* (rule G.3)

*Divergence Waypoint* (G.17): waypoint to diverge from original UAS path to ensure Intruder safety, with unchanged intruder velocity and heading.

*Convergence Waypoint* (G.18): waypoint when convergence to original UAS path is enabled, within unchanged intruder velocity and heading.

*Objective:*

1. Calculate *Divergence Waypoint* and *Convergence Waypoint.*
2. Enforce Divergence/Convergence waypoint during avoidance.

| *Context* | *Condition* | *Application* |
|---|---|---|
| UAS Navigation Grid,<br>Collision Point,<br>Avoidance Role | $UASVelocity$<br>$>>$<br>$IntruderVelocity$ | Calculate & Enforce:<br>• Divergence waypoint,<br>•Convergence waypoint |

Table G.7: Overtake rule definition.

## G.8 Rule: Right Plane Heading

There is a need to check if the *trajectory* is heading to the *right-side* from *collision point.* For this purpose, one may need to define a *separation plane in the 3D environment. Separation plane* will be defined according to Samuelson *hyperplane separation theorem* [3].

*Separation plane* (G.19) is defined by three points in *global coordination frame*:

1. *UAS Position* which is fixed to given *time-frame.*
2. *Collision point* which is not equal to *uas position* by definition.
3. *Gravitational acceleration* vector fitted to *UAS position* and orthogonal to vector $(uasPosition \rightarrow collisionPoint)$.

The properties of these three points guarantees that $scale.usasPosition \neq scale.collisionPoint \neq scale.gravitationalAcceleration$ for any linear $scale \neq 0$.

$$SeparationPlane = Plane\begin{pmatrix} uasPosition, collisionPoint, \\ loc2glob(uasPosition, gravitationalAcceleration) \end{pmatrix} \tag{G.19}$$

*Separation plane* (G.19) in *right-hand coordinate frame* where $center = uasPosition$ $X^+$ is given by vector $\vec{x^+}$ (*uasPosition*, *collisionPoint*) and $Z^-$ is given by vector $\vec{z^-}$ (*uasPosition*, *gravitationalAcceleration*). Then *right subspace* can be defined as all points where $y \leq 0$ and *left subspace* as all points where $y > 0$.

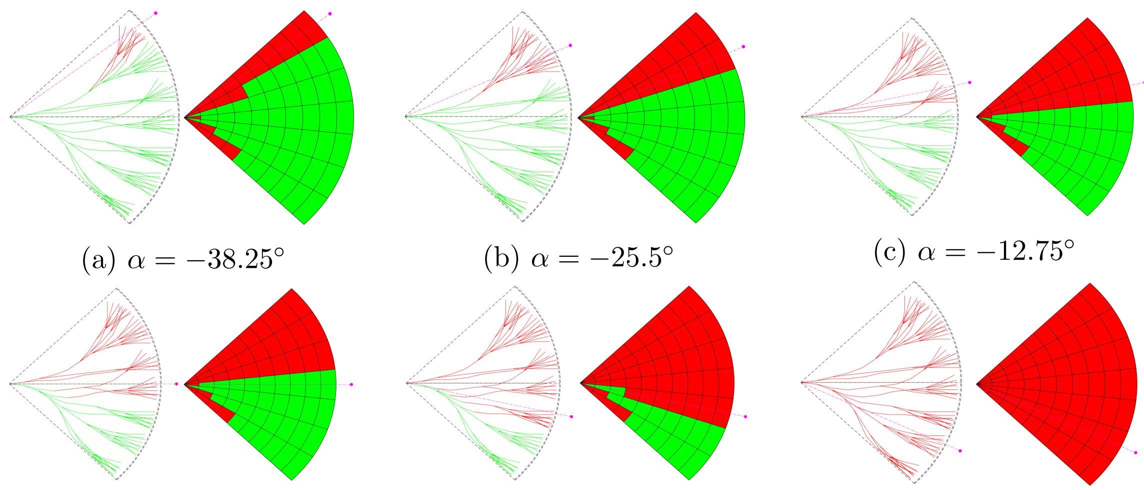
*Reach set* contains *trajectories*, the minimal dataset for trajectory is time-series of *position* and *heading* regardless *underlying nonlinear model*. Let us have *transformation function* which can map UAS *position* and *heading* into *separation plane coordinate frame*.

The *first condition* (G.20) says that each trajectory *point* must lie within the *right space portion*.

$$\forall position \in trajectory, \quad position \in rightSubspace \tag{G.20}$$

The *second condition* (G.21) needs to be applied for each *decision point* when *trajectory* can be re-planned. It must be ensured that in time of reaching *decision point* vehicle is not heading into *left subspace* with given *turning time horizon*. The *minimal information* contains a heading (velocity) vector. Checking if linear projection from *position* point with *heading* in given time-frame $[0, horizon]$ is sufficient.

$$\forall t \in [0, horizon], \quad (position + velocity * t) \in rightSubspace \tag{G.21}$$



(a) $\alpha = -38.25°$ (b) $\alpha = -25.5°$ (c) $\alpha = -12.75°$

(d) $\alpha = 0°$ (e) $\alpha = 12, 75°$ (f) $\alpha = 25.5°$

Figure G.1: Right plane heading rule evaluation for various angles of approach $\alpha$.

Figure G.1. shows *enabled* (green line) and *disabled* (red lines) *trajectories* (left sub-

figure). These trajectories are divided according to the *separation line* (magenta dashed line), given by *vehicle position* and *collision point* (magenta circle). *Space segmentation* (right subfigure) show *reachable* (green fill) *unreachable* (red fill) space. The situation is shown for various *collision point angles of approach* $\alpha$.
The rule for a right plane heading check is summarized in (tab. G.8).

| | | |
|---|---|---|
| *Invocation:* Invoked by other *maneuver rules.*<br><br>*Objective:* Disable all *trajectories* in *Navigation grid's reach set* which are:<br>1. *Heading into collision zone*<br>2. *Leading into collision zone* | | |
| *Context* | *Condition* | *Application* |
| UAS Navigation Grid, Collision point (LOC) | There are feasible trajectories in Navigation Grid. | Disable trajectories in Navigation Grid. |

Table G.8: Right plane heading rule definition.

## G.9 Rule: Enforce safety margin

Rule G.8. checks right plane heading for a single mass point along *trajectories.* The rule needs to account *body mass* of *intruder* and UAS, other factors like safe distance, regulations, etc. All mentioned factors are included in the *safety margin.* The *safety margin* is applied as *radius ball* around *collision point.*

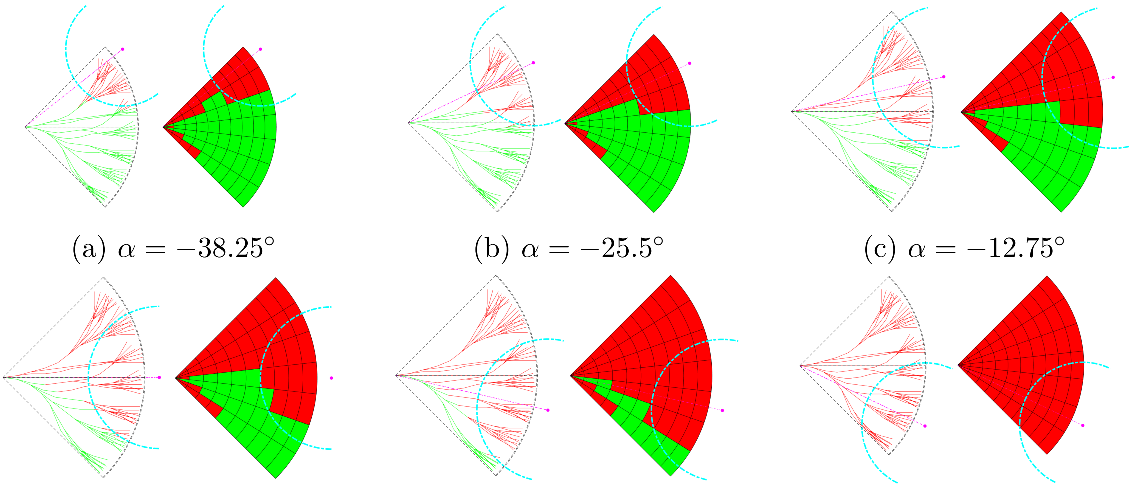
*Collision point* can be mapped from *global coordinate frame* to *reach set coordinate frame*, based on UAS *position and orientation* in a *decision time.* Then a comparison of distance between *collision point* and every *trajectory decision point* is trivial.

*Trajectory feasibility condition* for *non-controlled airspace* (G.22) is given as follow:

$$\forall position \in trajectory, \quad distance(position, collisionpoint) \geq safetyMargin \tag{G.22}$$

*Controlled airspace* must maintain *well clear condition.* To enforce protective barrel around *collision point* one must compare *global coordinates. Trajectory feasibility condition* for *controlled airspace* (G.23) is given as follow:

$$\begin{aligned} \forall position \in & trajectory, \\ & XYdistance(position, collisionPoint) \geq safetyMargin \\ & flightLevelStart \geq Z(position) \geq flightLevelEnd \end{aligned} \tag{G.23}$$



(a) $\alpha = -38.25°$ (b) $\alpha = -25.5°$ (c) $\alpha = -12.75°$

(d) $\alpha = 0°$ (e) $\alpha = 12,75°$ (f) $\alpha = 25.5°$

Figure G.2: Enforce safety margin rule evaluation for various angles of approach $\alpha$.

Figure G.2. shows *enabled* (green line) and *disabled* (red lines) *trajectories* (left subfigure). These trajectories are divided according to the *separation line* (magenta dashed line), given by *vehicle position* and *collision point* (magenta circle). More trajectories are disabled due to *safety margin* (teal dashed line) around the *collision point*. *Space segmentation* (right subfigure) show *reachable* (green fill) *unreachable* (red fill) space. The situation is shown for various *collision point angles of approach* $\alpha$.

The rule for safety margin check is summarized in (tab. G.9).

| | | |
|---|---|---|
| *Invocation:* Invoked by other *maneuver rules*.<br><br>*Objective:* Based on the type of airspace, for the given *collision point* and *safety margin* disable trajectories in:<br>1. Ball radius for *non-controlled* airspace (G.22).<br>2. Well-clear barrel *controlled* airspace (G.23). | | |
| *Context* | *Condition* | *Application* |
| UAS Navigation Grid<br>Collision point<br>Safety Margin | There are feasible trajectories for condition application. | Disable trajectories in Navigation Grid. |

Table G.9: Enforce safety margin rule definition.

# Bibliography


[1] Jon Louis Bentley and Thomas A Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Transactions on computers*, (9):643–647, 1979.

[2] Michael Ian Shamos and Dan Hoey. Geometric intersection problems. In *17th annual symposium on foundations of computer science*, pages 208–215. IEEE, 1976.

[3] Hans Samelson, Robert M Thrall, and Oscar Wesler. A partition theorem for euclidean n-space. *Proceedings of the American Mathematical Society*, 9(5):805–807, 1958.