DAIDALUS v2

César A. Muñoz
(NASA)
cesar.a.munoz@nasa.gov

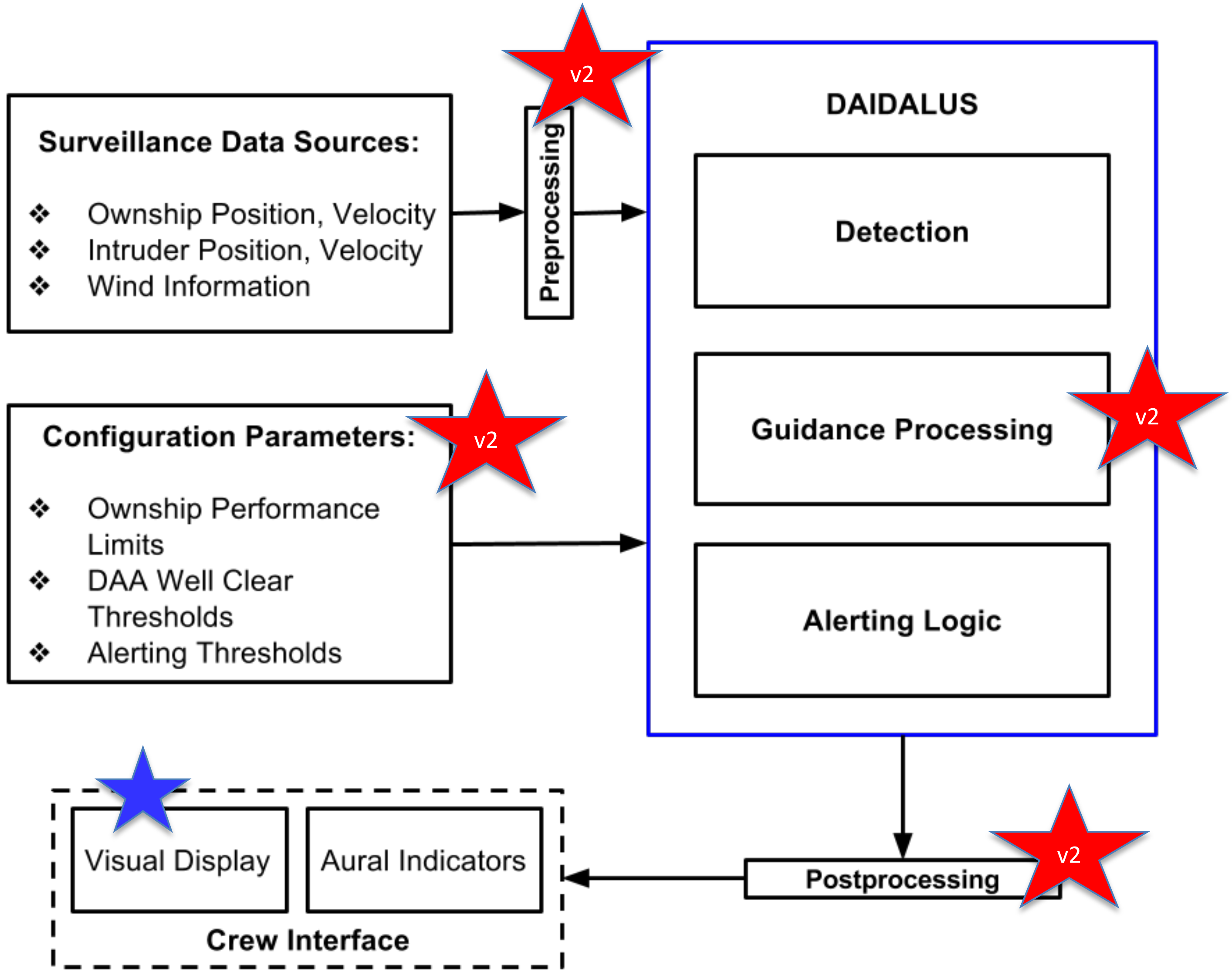
Joint work with:
Aaron Dutle (NASA)
Anthony Narkawicz (formerly NASA)

RTCA SC-228
August 2020

DAIDALUS

- Detect and Avoid Alerting Logic for Unmanned Systems (DAIDALUS) is an open source software library that implements a configurable **detect and avoid** (DAA) concept.
- DAIDALUS core logic relies on **formally verified, ownship-centric, state-less** algorithms that:
  - Determine the current pairwise well-clear status (Detection Logic).
  - Compute maneuver guidance to **maintain or regain** well-clear status (Maneuver Guidance Logic):
    - suggestive guidance (kinematic or instantaneous bands)
    - directive guidance (preferred velocity vector).
  - Determine alert level (Alerting Logic).

DAIDALUS Architecture



DAIDALUS v2 = v1 + …

- Backward compatible configuration files
- Sensor Uncertainty Mitigation (SUM) logic
- Multiple alerters
- Ownship-/Intruder-centric alerting logic
- Hysteresis logic
- DTA logic
- Early Alerting Time Logic
- DAA-Displays*

* P. Masci and C. Muñoz, *A Graphical Toolkit for the Validation of Requirements for Detect and Avoid Systems*, 2020. URL: https://shemesh.larc.nasa.gov/fm/DAA-Displays/

DAIDALUS DAA: Detectors

- A **detector** is a volume parameterized by distance and time thresholds:
  - WCV_TAUMOD(DTHR,ZTHR,TTHR,TCOA)
  - WCV_TCPA(DTHR,ZTHR,TTHR,TCOA)
  - WCV_TEP(DTHR,ZTHR,TTHR,TCOA)
  - CDCylinder(DTHR,ZTHR)
  - TCAS3D(DMOD[7],ZTHR[7],TAUMOD[7],*RA or TA?)*
- DO-365 Phase 1 WC:
  - WCV_TAUMOD(0.66nmi,450ft,35s,0s)

DAIDALUS DAA: Alerters

- An **alert level** is composed of a detector, a band type (region), an alerting time, and an early alerting time.
- An **alerter** is a list of alert levels in increasing order of severity.
- DO-365 Phase 1 Alerter:
  1. (Preventive): WCV_TAUMOD(0.66nmi,700ft,35s,0s), NONE, 55,75.
  2. (Corrective): WCV_TAUMOD(0.66nmi,450ft,35s,0s), MID, 55,75.
  3. (Warning): WCV_TAUMOD(0.66nmi,450ft,35s,0s), NEAR, 25,55.

Sensor Uncertainty Mitigation Logic (SUM)

GOAL: *Give alerts and guidance considering uncertainty estimates in aircraft data.*

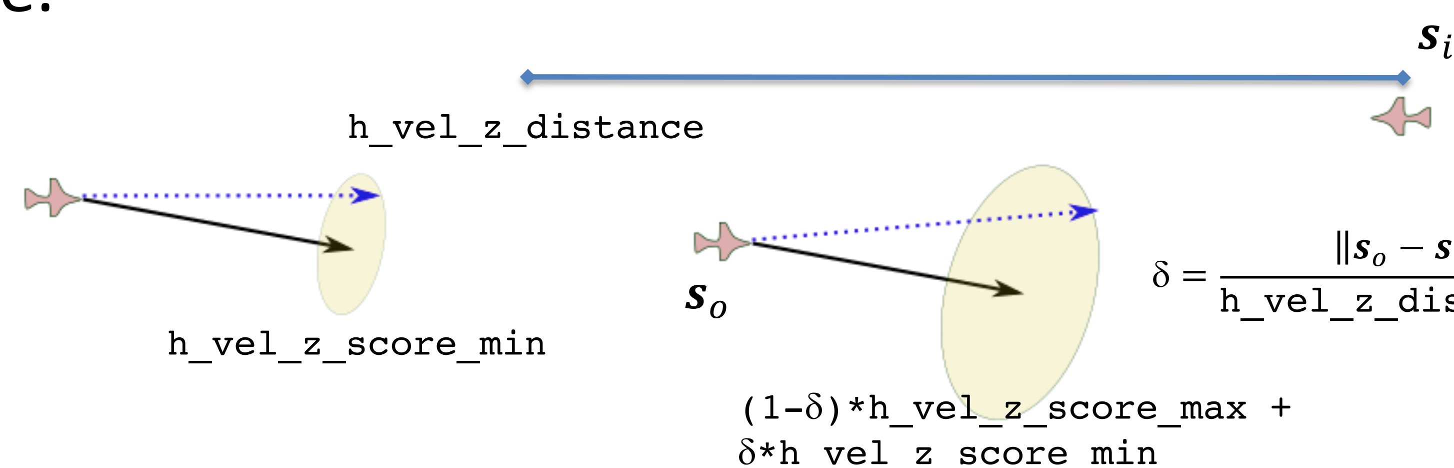
- In v1, SUM was implemented as wrapper on DAIDALUS (work by J. Sturdy, D. Jack, and K. Hoffler at NASA LaRC)
- In v2, SUM is implemented as a detector:
  - This detector uses error ellipses for position and velocities and finds a worst-case position/velocity pair.
  - WCV_TAUMOD_SUM(DTHR,ZTHR,TTHR,TCOA,*position and velocities z-scores*)

SUM Configuration Parameters

- `h_pos_z_score`: Gain on standard deviations of horizontal position uncertainty.
- `h_vel_z_score_min`: Gain on standard deviations of horizontal velocity uncertainty when aircraft are *far apart.*
- `h_vel_z_score_max`: Gain on standard deviations of horizontal velocity uncertainty when aircraft are close together.
- `h_vel_z_distance`: Distance at which to start scaling number of standard deviations for velocity uncertainty.
- `v_pos_z_score`: Gain on standard deviations of vertical position uncertainty.
- `v_vel_z_score`: Gain on standard deviations of vertical velocity uncertainty.
- These parameters are global but can be overwritten per alert level.

Velocity Uncertainty

- A small velocity uncertainty can result in large position distances when propagated over a long time.
- When handling velocity uncertainty the same regardless of distance, optimizing parameters may be dominated by scenarios at *far* distances.
- In DAIDALUS v2, the velocity uncertainty scales with range.



$$\delta = \frac{\|\boldsymbol{s}_o - \boldsymbol{s}_i\|}{\texttt{h\_vel\_z\_distance}}$$

Traffic State SUM Inputs

σ_{px}	std dev of x-coord of position
σ_{py}	std dev of y-coord of position
ϕ_{pxy}	signed co-std dev between x, y coords of position
σ_{vx}	std dev of x-coord of velocity
σ_{vy}	std dev of y-coord of velocity
ϕ_{vxy}	signed co-std dev between x, y coords of velocity
σ_{pz}	std dev of z-coord of position
σ_{vz}	std dev of z-coord of velocity

$\phi_{xy} = sign(c_{xy})\sqrt{|c_{xy}|}$, where c_{xy} is the co-variance of xy

Example: RADAR/H2

DO-365 (WCV_TAUMOD)

DO-365 (WCV_TAUMOD_SUM)


```
h_pos_z_score = 1.5
h_vel_z_score_min = 0.5
h_vel_z_score_max = 1.0
h_vel_z_distance = 5.0 [nmi]
v_pos_z_score = 0.75
v_vel_z_score = 1.5
```

Multiple Alerters

GOAL: *Allow for different well-clear guidance and alerting logic for different aircraft (En-route, Terminal Area, LowSWaP).*

- In DAIDALUS v1, only one alerter can be configured at a time.
- DAIDALUS v2 supports the configuration of an arbitrary number of alerters, e.g.,
  - DO-365 (Phase 1), DO-365A DTA (Phase 2), LowSWaP, etc.
- An alerter is assigned to each aircraft.
- The selection of an alerting schema can be done on-the-fly (external to DAIDALUS)

Ownship- vs. Intruder-Centric Alerting

- Ownship-centric alerting:
  - Enabled when `ownship_centric_alerting = true`
  - At any time, the alerter assigned to the ownship is used for every aircraft.
- Intruder-centric alerting:
  - Enabled when `ownship_centric_alerting = false`
  - At any time, the alerter assigned to each intruder is used for that aircraft.

Hysteresis Logic

GOAL: *Store previous DAIDALUS outputs for use in smoothing of alerts and guidance.*

- DAIDALUS v1 is stateless
  - Previous states are not kept
  - Only configuration values persist from time step to time step
- DAIDALUS v2 is *mostly* stateless
  - Previous states are not kept
  - If hysteresis is enabled, previous alert levels, bands, and resolutions are cached
  - Hysteresis is enabled when `hysteresis_time` is greater than 0
  - If current state is newer than previous state by more than `hysteresis_time`, cached values are reset.

M of N Logic

GOAL: *Reduce the amount of on/off jitter in alerting and guidance*.

- M of N logic returns the highest value that repeats at least M times of the last N values (including the current one)
- This logic is applied to both **alert levels** and **bands regions**
- Configuration parameters:

```
alerting_m = 2
alerting_n = 4
```

Time-Based Persistence Logic

GOAL: *Keep alerts and guidance for an amount of time after they first appear, to allow time for an aural alert.* (Sect 2.2.4.3.5.1-3)

- Time-based persistence logic returns the last value if younger than `persistence_time`
- Time-based persistence is applied *after* M of N to both **alert levels** and **conflict bands**

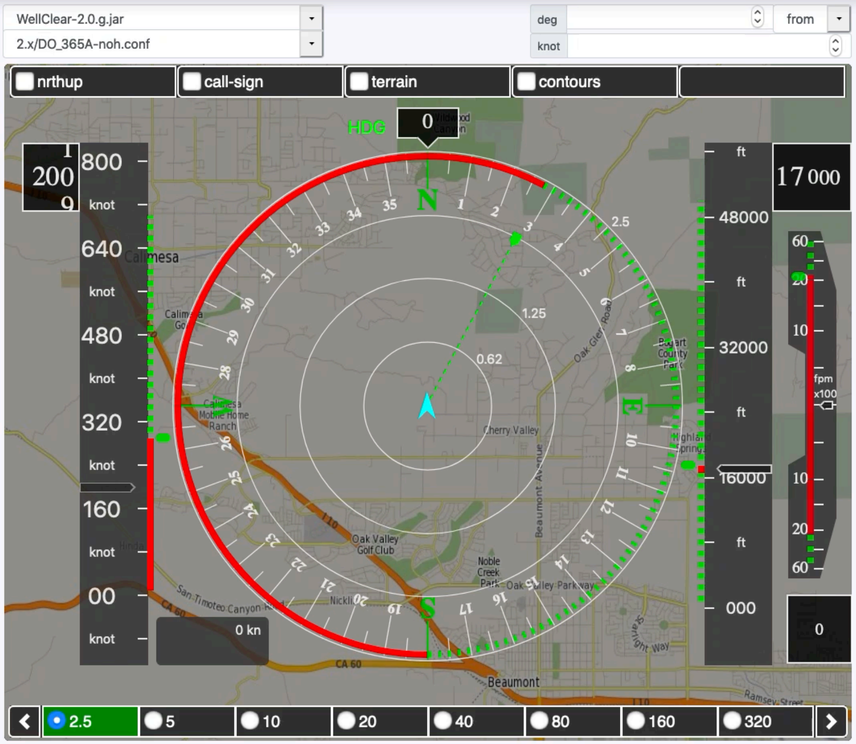
Value-Based Persistence Logic

GOAL: *Reduce reversals and jitter on directional guidance.* (Sect. 2.2.4.4.1.2, shall 261, 265)

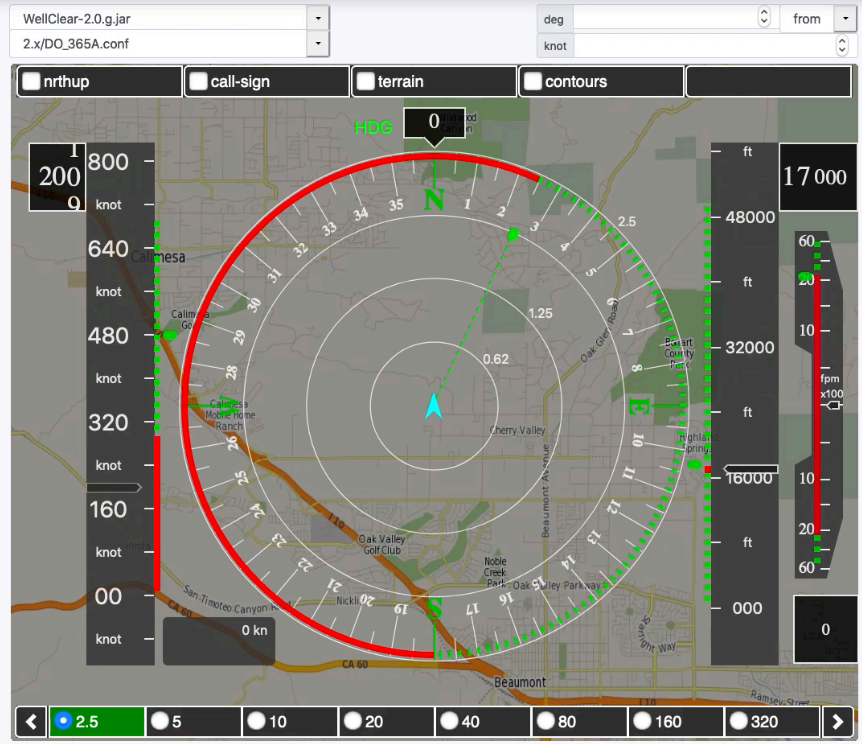
- Value-based persistence logic returns the previous value if still valid and the difference from the new value is less than a configured amount.
- Value-based persistence is applied to **preferred resolution**
- Configuration parameters:
  - `persistence_preferred_hdir`
  - `persistence_preferred_hs`
  - `persistence_preferred_vs`
  - `persistence_preferred_alt`

Example: Truth/H1

DO-365



DO-365 + Hysteresis




```
hysteresis_time = 5.0 [s]
persistence_time = 4.0 [s]
bands_persistence = true
persistence_preferred_hdir = 15.0 [deg]
persistence_preferred_hs = 100.0 [knot]
persistence_preferred_vs = 250.0 [fpm]
persistence_preferred_alt = 250.0 [ft]
alerting_m = 2
alerting_n = 4
```

DAA Terminal Area (DTA) Logic

GOAL: *Support automatic switching to a specified DTA alerter, including special maneuver guidance on warning alert*. (Sect. 2.2.4.3.6.2, 2.2.4.4.2)

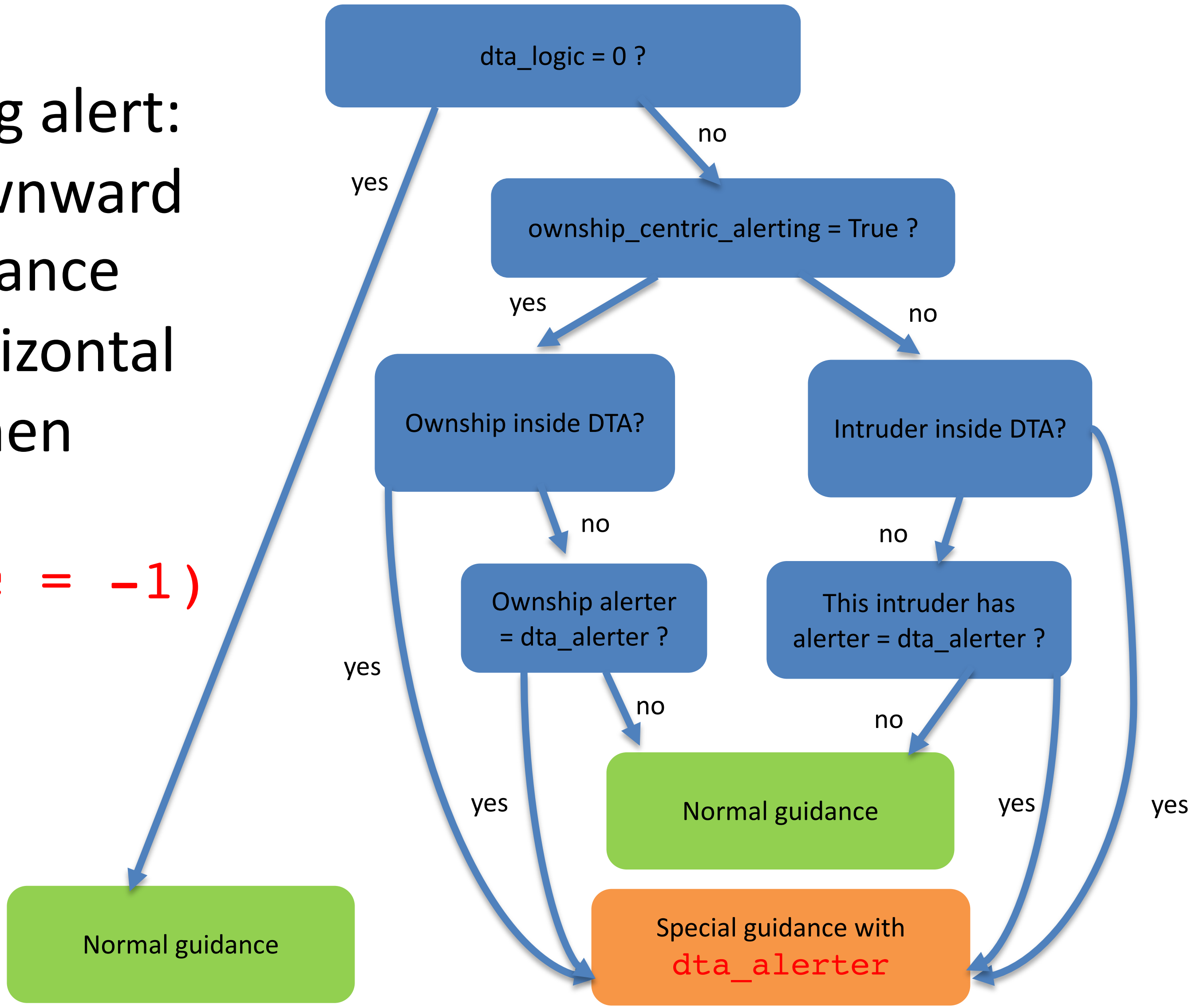
- DTA configuration
  - `dta_logic`: 0 (DTA disabled), 1 (DTA enabled for departing), -1 (DTA enabled for landing)
  - `dta_latitude`: Latitude of DTA center
  - `dta_longitude`: Longitude of DTA center
  - `dta_radius`: Radius of DTA zone
  - `dta_height`: Height of DTA zone
  - `dta_alerter`: Index of Alerter to be used
  - Altitude of DTA floor is assumed to be 0

DTA Special Guidance Logic

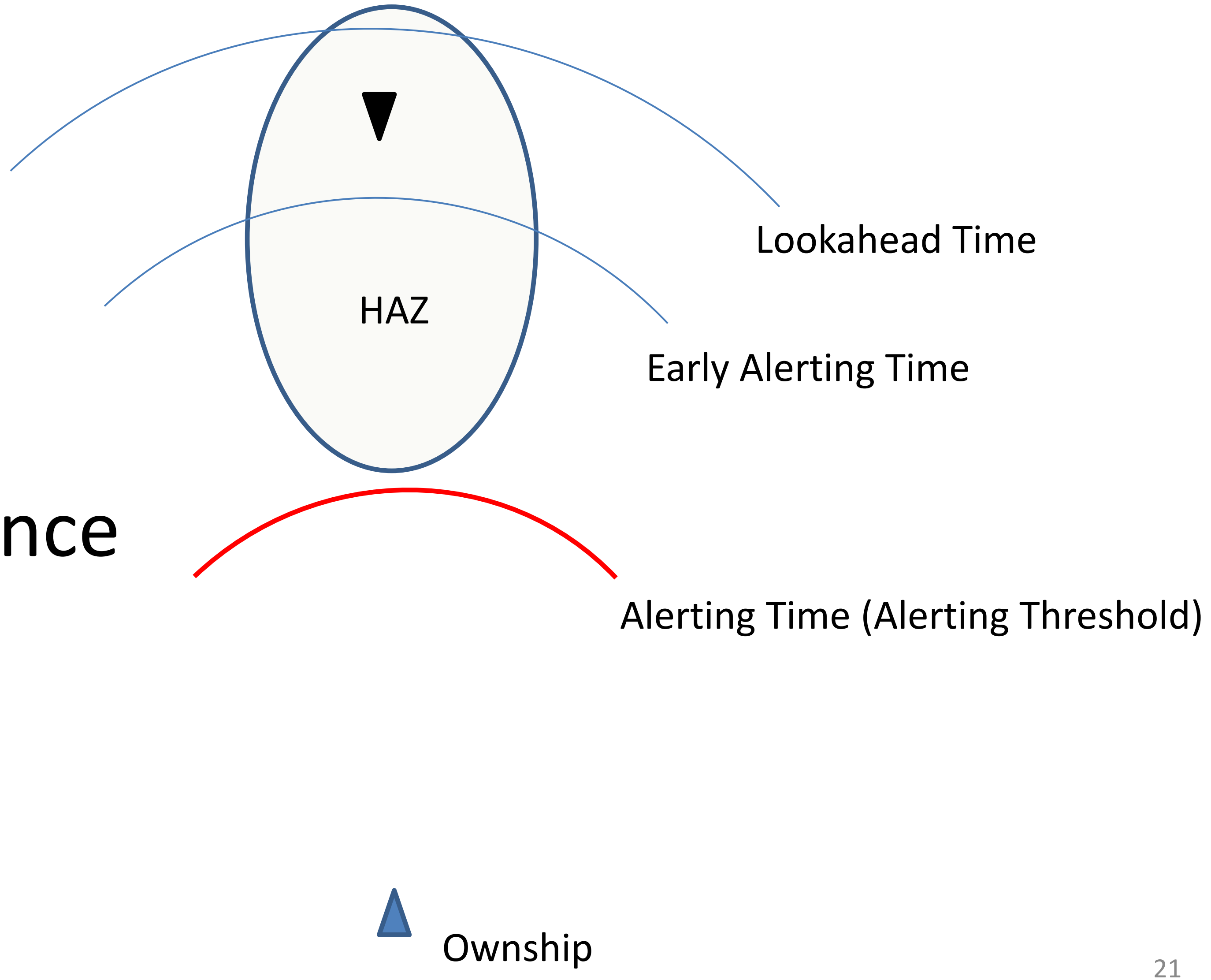
During warning alert:

- Remove downward vertical guidance
- Saturate horizontal guidance when landing

(dta_logic = -1)



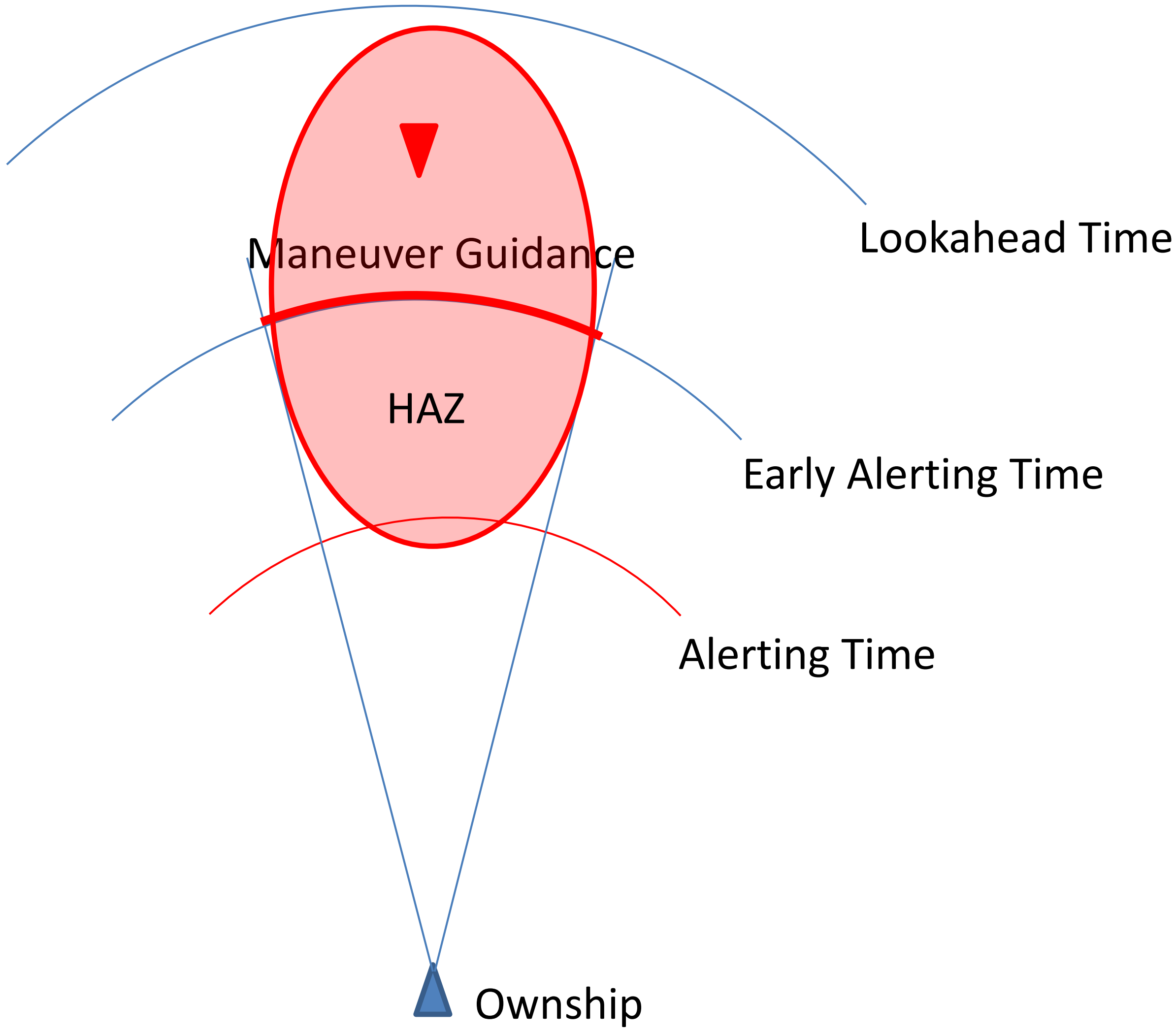
Lookahead Time, Early Alerting Time, and Alerting Time



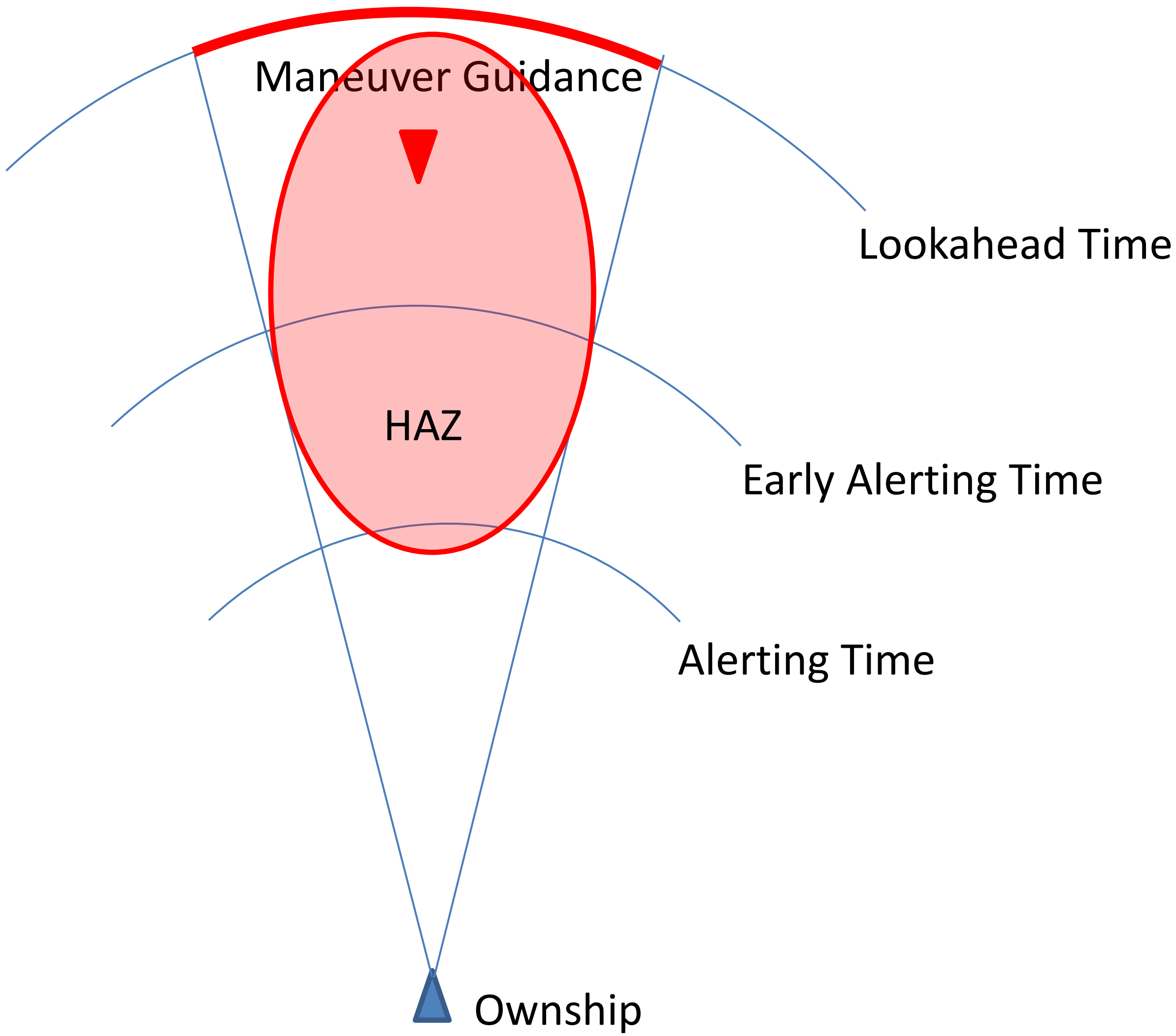
No Alert

No Guidance

Maneuver Guidance in v1

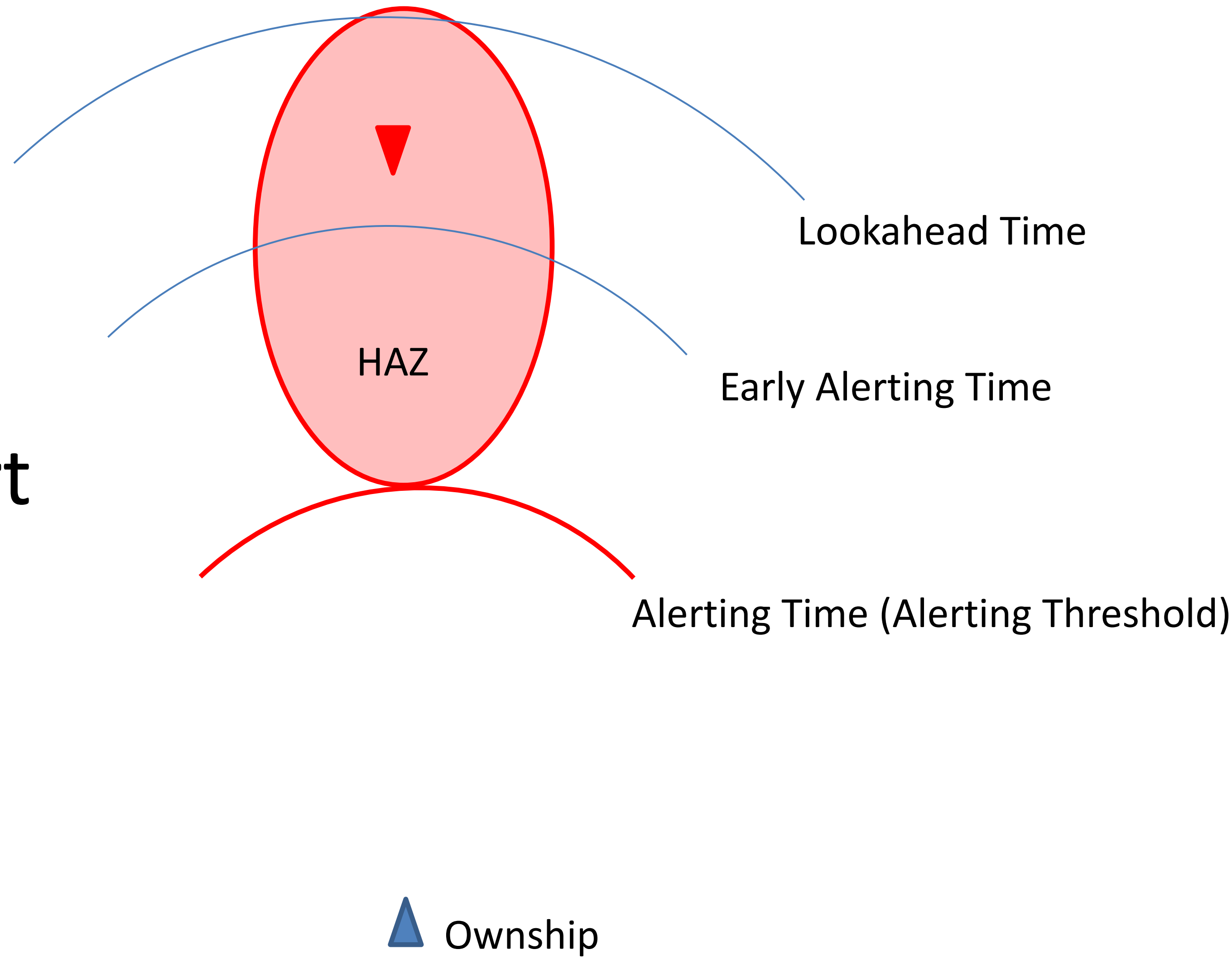


Maneuver Guidance in v2

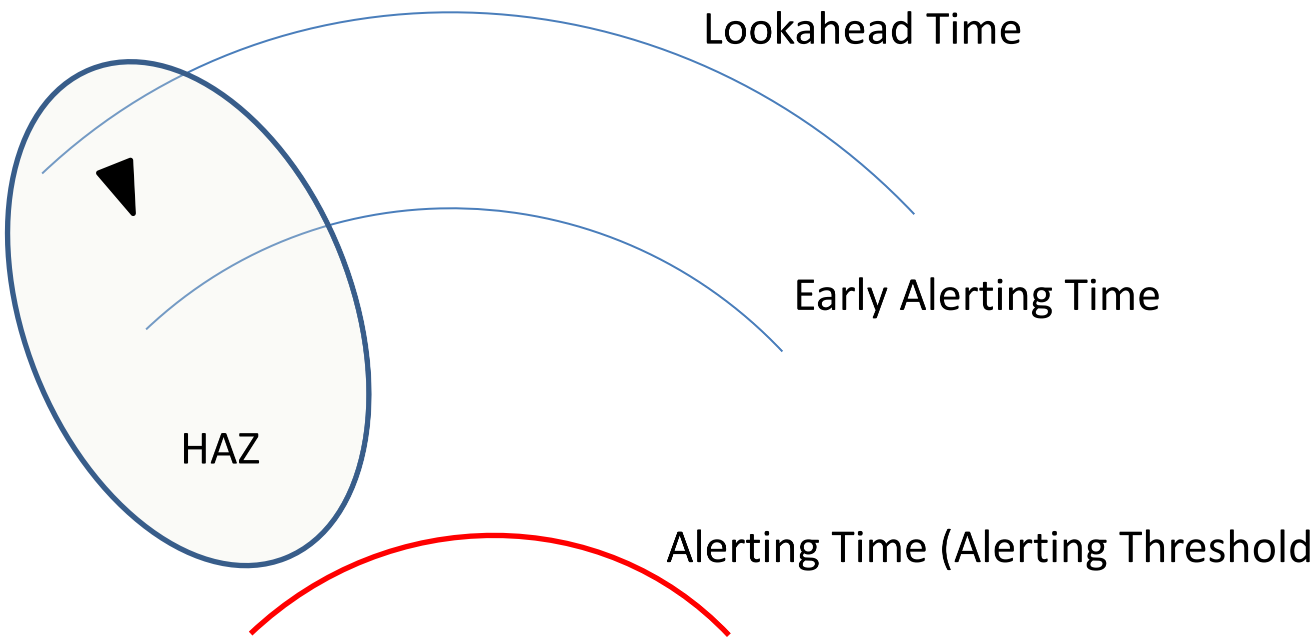


Alerting Logic v1 and v2

First Alert



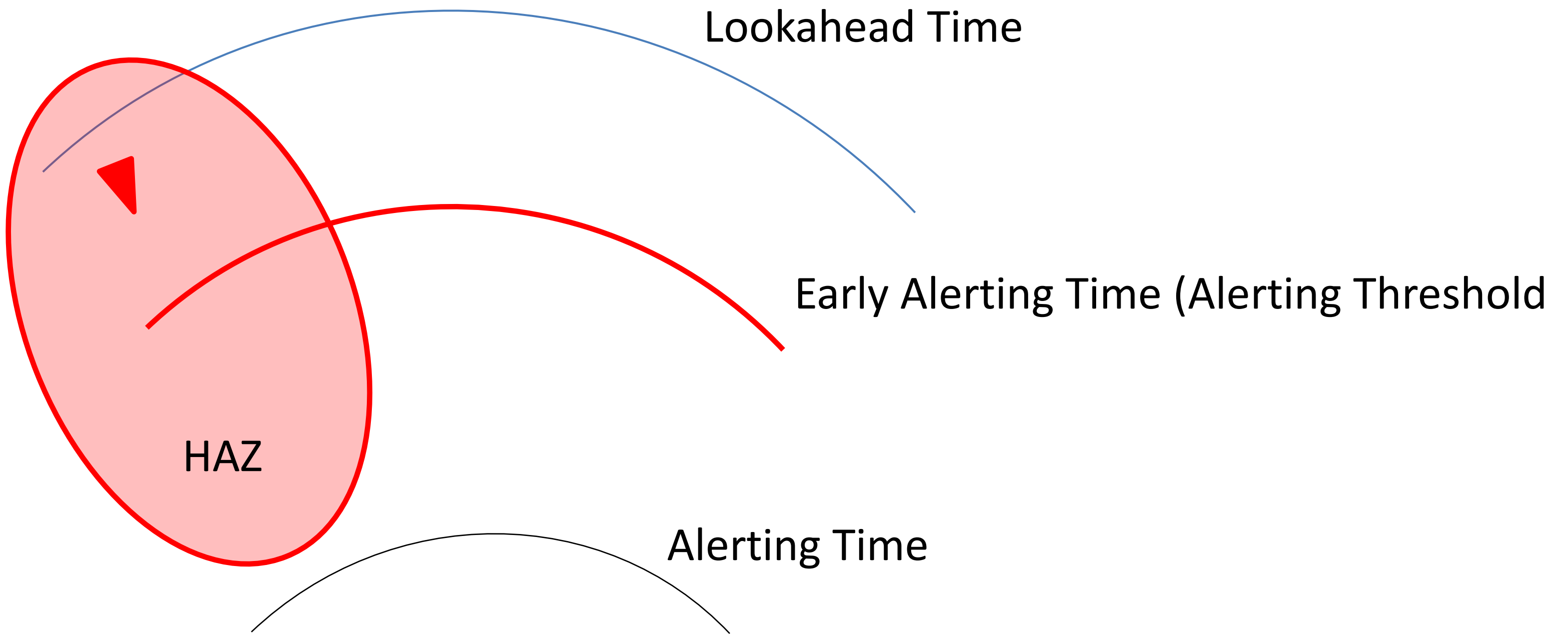
Alerting Logic in v1



If ownship maneuvers after 1st alert:
No Alert

Note: This problem is especially common with winds and slow moving ownship

Alerting Logic in v2*



Ownship maneuvers
after 1st alert:
Alert

* V. Carreño, *Evaluation, Analysis and Results of the DANTi Flight Test Data, the DAIDALUS Detect and Avoid Algorithm, and the DANTi Concept for Detect and Avoid in the Cockpit*, NASA-CR, 2020.

Regain Well-Clear in DAIDALUS

When the aircraft have lost well-clear or when a loss is unavoidable (according to the ownship performance parameters) **regain well-clear logic** computes:

1. Suggestive Guidance (recovery bands)

2. Directive Guidance

3. Preferred Direction

4. Contours

DAIDALUS doesn't artificially saturate recovery bands

Regain Well-Clear Display in the MOPS*

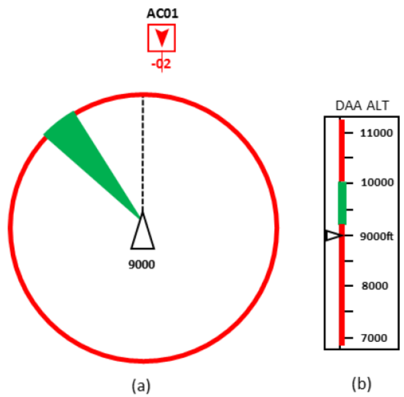
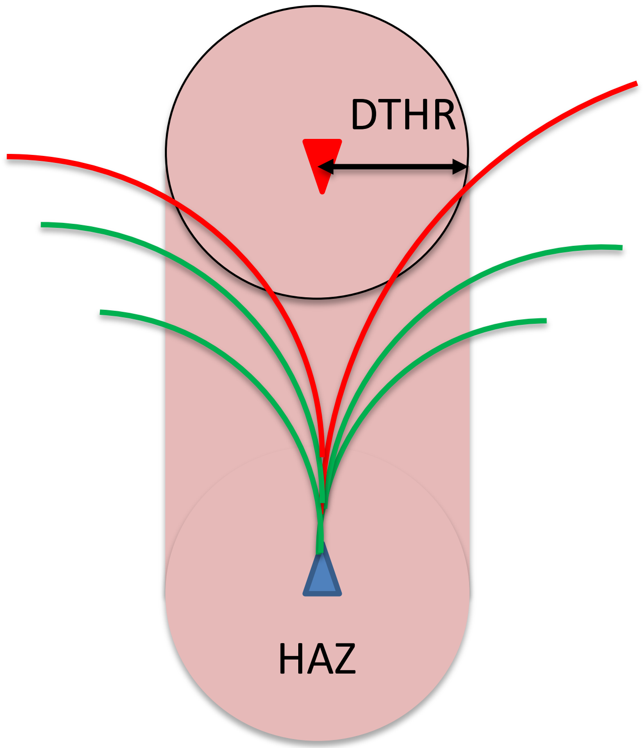


Figure 2-11 Illustration of (a) Horizontal and (b) Vertical DWC

*Sect. 2.2.4.4.1.2, 2.2.4.4.2.2

Well-Clear Recovery Algorithm

1. Compute maneuvers that recover well-clear in minimum time without violating DTHR.
2. If none, reduce DTHR by `ca_factor`
3. If reduced DTHR is greater than NMAC go to 1.
4. Give up.



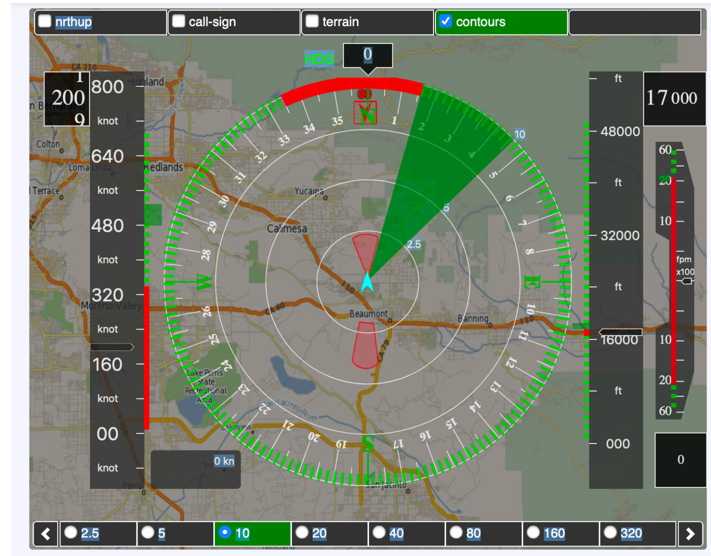
Preferred direction: The one that is closest to current value (with a bias towards right)

Observations on Selecting Single Direction

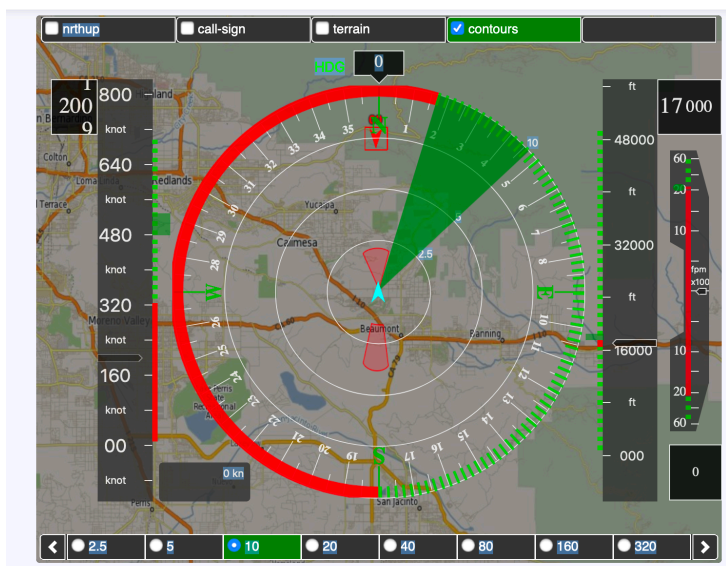
- Algorithmically, it's a difficult problem:
  - Mathematical, physical, operational, and human-factor considerations
  - Too many special cases
  - Inevitable reversals because of secondary conflicts and shifting velocity vectors due to winds and sensor uncertainty
- In case of a reversal, there may be no time to correct and any advantage in pilot reaction time would be lost, even detrimental
- Problem is exacerbated because DO-D365 DWC uses TAUMOD, which forces one direction selection 35s before loss DMOD separation.
- Single direction requirement may not even necessary: Bands often saturate in one direction.

A Head-On Example without Saturation

158s



162s



166s

Concluding Remarks

- DAIDALUS v2:
  - Available at https://github.com/nasa/daidalus
- DAA-Displays:
  - Research tool for visualization and comparative analysis of DAIDALUS configurations
  - Available at https://github.com/nasa/daa-displays
- ICAROUS:
  - An on-board software architecture for autonomus UAS operations.
  - Uses DAIDALUS as DAA system for sUAS
  - Over 400 flight tests!
  - Available at https://github.com/nasa/icarous
- DANTi:
  - DAA in the cockpit (DAIDALUS on an electronic flight bag)
  - Uses DAIDALUS as DAA system on GA manned aircraft
  - 7 flight tests