BHV_OpRegionBounce:
an OpRegion that can bounce you back

Stephanie Kemna

kemna@nurc.nato.int

MOOS-DAWG 2011
MIT, Cambridge, MA, USA

NATO Undersea Research Centre: Underwater acoustics and ASW

1959: SACLANT

NATO maritime and transformational requirements

Seagoing research: Maritime innovation in NATO Nations

- Cooperative Anti-Submarine Warfare
- Autonomous Naval Mine Countermeasures
- Ship and Port Protection
- Marine Mammal Risk Mitigation
- Maritime Situational Awareness
- Environmental Knowledge & Operational Effectiveness

OEX AUVs: Groucho & Harpo

MIT/moos-ivp: BHV_OpRegion

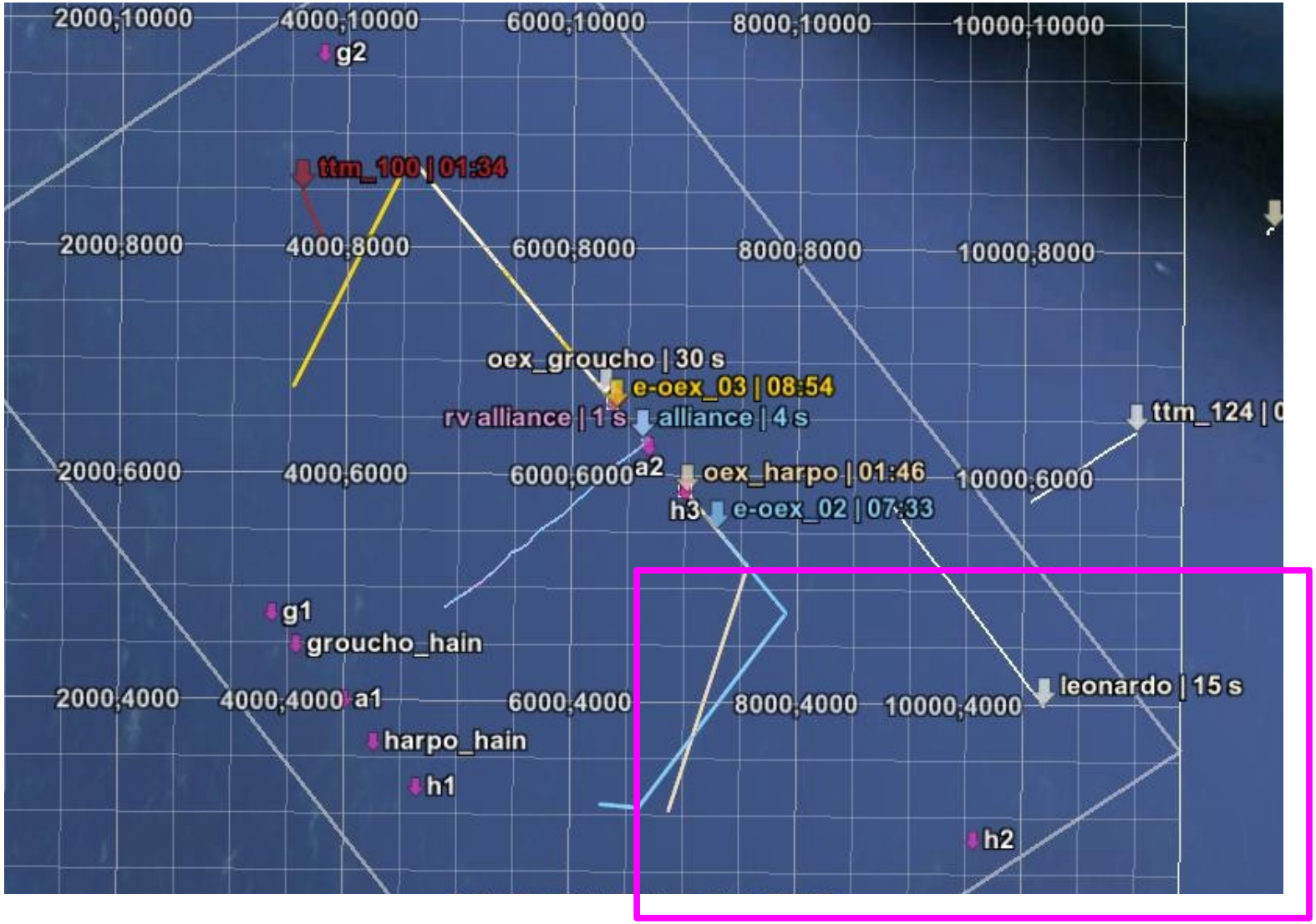
If the vehicle hits the limit for perimeter, depth or altitude:

- BHV_ERROR:

  all DESIRED_* values to zero

Why is this a problem?

March 2011 Engineering Trial



March 2011 Engineering Trial (2)

- Area with high fishing activity
- Vehicle with waypoint close to perimeter
- No communications with vehicle for a long, long time
- Fear of the AUV having hit the perimeter and surfacing (due to positive buoyancy)

Solution: Bounce

In case of OpRegion failure:

- Perimeter: bounce away from perimeter
- Depth/Altimeter: bounce to 'higher' depth

This will keep the vehicle at depth and safe.

Behaviour Design

Starting from BHV_OpRegion

- Add perimeter bounce (for every polygon)
  - orthogonal to perimeter
- Add depth bounce (for depth & altitude)
  - bounce to 'higher' depth
- Both: bounce buffer & no_zone

Bounce buffer & no_zone

buffer ::

activates bounce, utility: linear increase for distance from buffer to limit.

no_zone_factor :: [0 – 1]

buffer*factor = part with highest utility

Perimeter Bounce: calculate course vectors

```
for every polygon
if distToPoly < bounce_buffer
then for every vertex i
    if distToVertex < bounce_buffer
        compute vertex angle a_i (from North)
        a_i += 90     // orthogonal bounce
        w_i = 1- (distToVertex – no_zone)
                  ------------------------
                        bounce_buffer
        w_i *= maxutil
        if w_i > maxutil
            w_i = maxutil
```

weight w_i:
0 to maxutil,
linear increase
with distance

Perimeter Bounce: combine course vectors

```
// rescale all weights so that the max is 100
// scale the weight of the bhv inversely

for every a/w combo
   determine maxWeight

factor = m_maxutil/maxWeight;
w_i *= factor;
m_pwt_course = m_priority_wt/factor;
```

rescale, because bhv weight (pwt) rather than util should reflect bhv importance/ influence

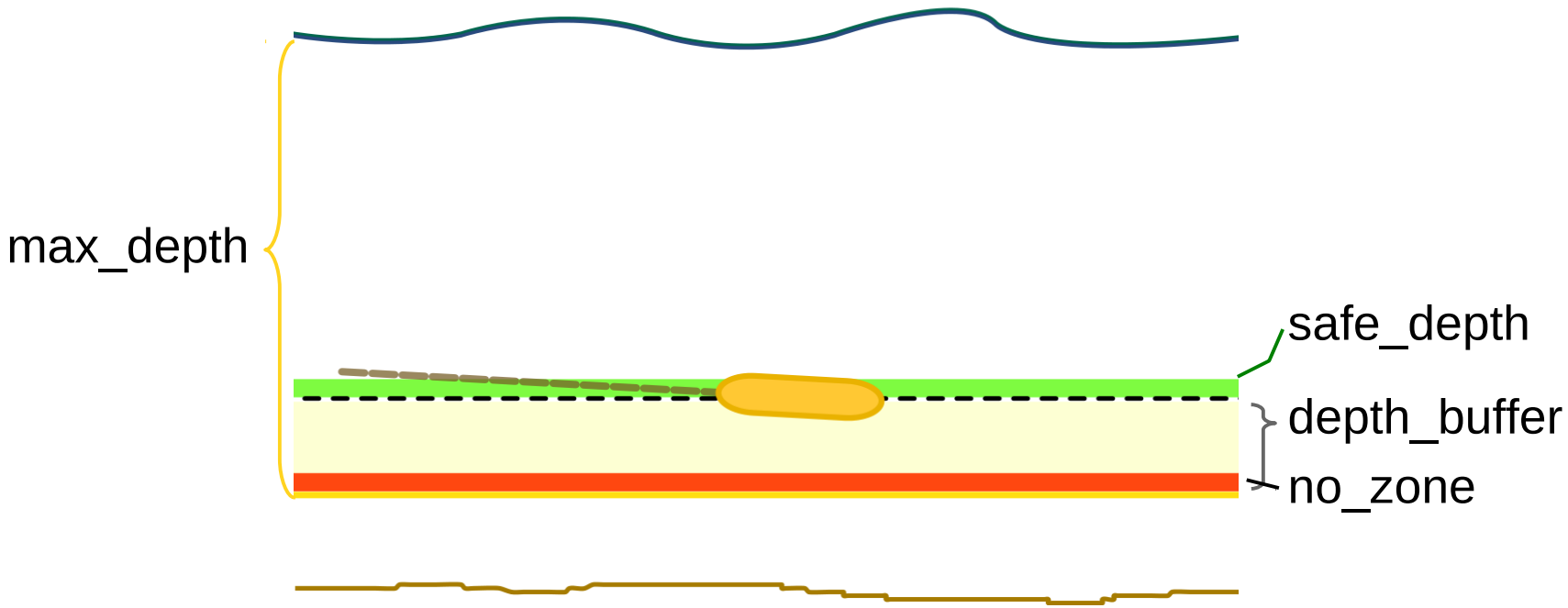
```
for every a/w combo

   create a ZAIC_PEAK component(a_i,w_i)


extract the ZAIC IvP course function
```

Depth Bounce (1): calculate depth vector for Depth

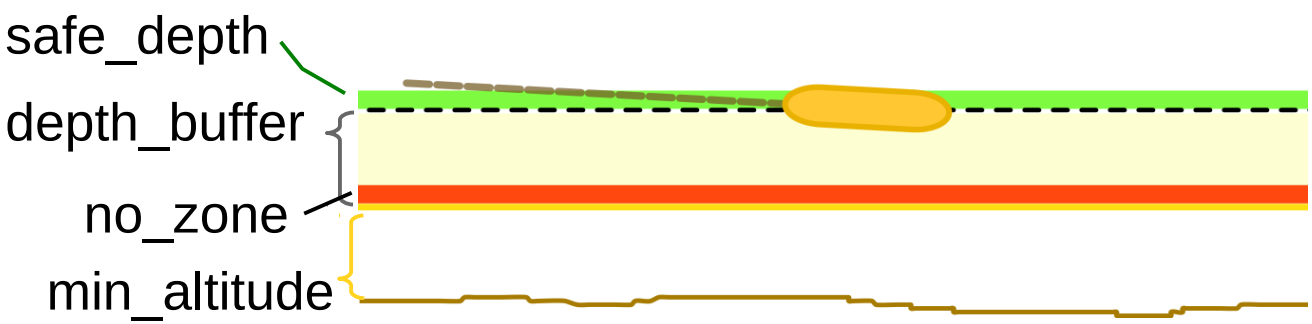
```
if depth > (max_depth - depth_buffer)
    safe_depth0 = max_depth - depth buffer - no_zone
    w0 = 1-(((max_depth-no_zone)-depth)/depth_buffer)
    if w0 > maxutil
        w0 = maxutil
```



Depth Bounce (2): calculate depth vector for Altitude

```
if altitude < (min_altitude + depth_buffer)
    wanted_alt = min_altitude + depth_buffer + no_zone
    diff_altitude = wanted_alt - curr_alt
    safe_depth1 = depth – diff_altitude
    w1 = 1-((curr_alt-(min_alt+no_zone))/depth_buffer)
    if w1 > maxutil
        w1 = maxutil
```

convert from altitude to depth for correct command



Depth Bounce: combine depth vectors (1)

```
if (depth emergency && altitude emergency)

    if (w0 > w1)
        factor = maxutil/w0
        w0 = maxutil
        w1 = w1*factor

    if (w1 > w0)
        factor = maxutil/w1
        w0 = w0*factor
        w1 = maxutil

    if (w0 = w1)
        w0 = maxutil
        w1 = maxutil
```

rescale so that utilities are normalized to [0 - maxutil] given the highest calculated weight

Depth Bounce: combine depth vectors (2)

```
if (m_priority_wt > 100)
   w{0/1} *= m_priority_wt/100
m_pwt_depth = w0 if it was highest, else w1

for safe_depth0 && safe_depth1, if present,
   create a ZAIC_PEAK component(safe_depthi,wi)
extract the ZAIC IvP depth function
```

BHV_OpRegionBounce: couple course and depth functions

```
if depth_emergency && perimeter_emergency
    * rescale lower_pwt function by higher_pwt
    final_pwt = higher_pwt
    ipf = coupler.couple(course ZAIC, depth ZAIC,
                         rel_pwt_course, rel_pwt_depth)
    ipf->setPWT(final_pwt);


else
    output the one, or the other, or neither
```

BHV_OpRegionBounce: coupler issues

BUT coupler seems to not properly process the relative weights, therefore:

- BHV_OpRegionBounce
- BHV_OpRegionBounceDepth

Testing - simulation

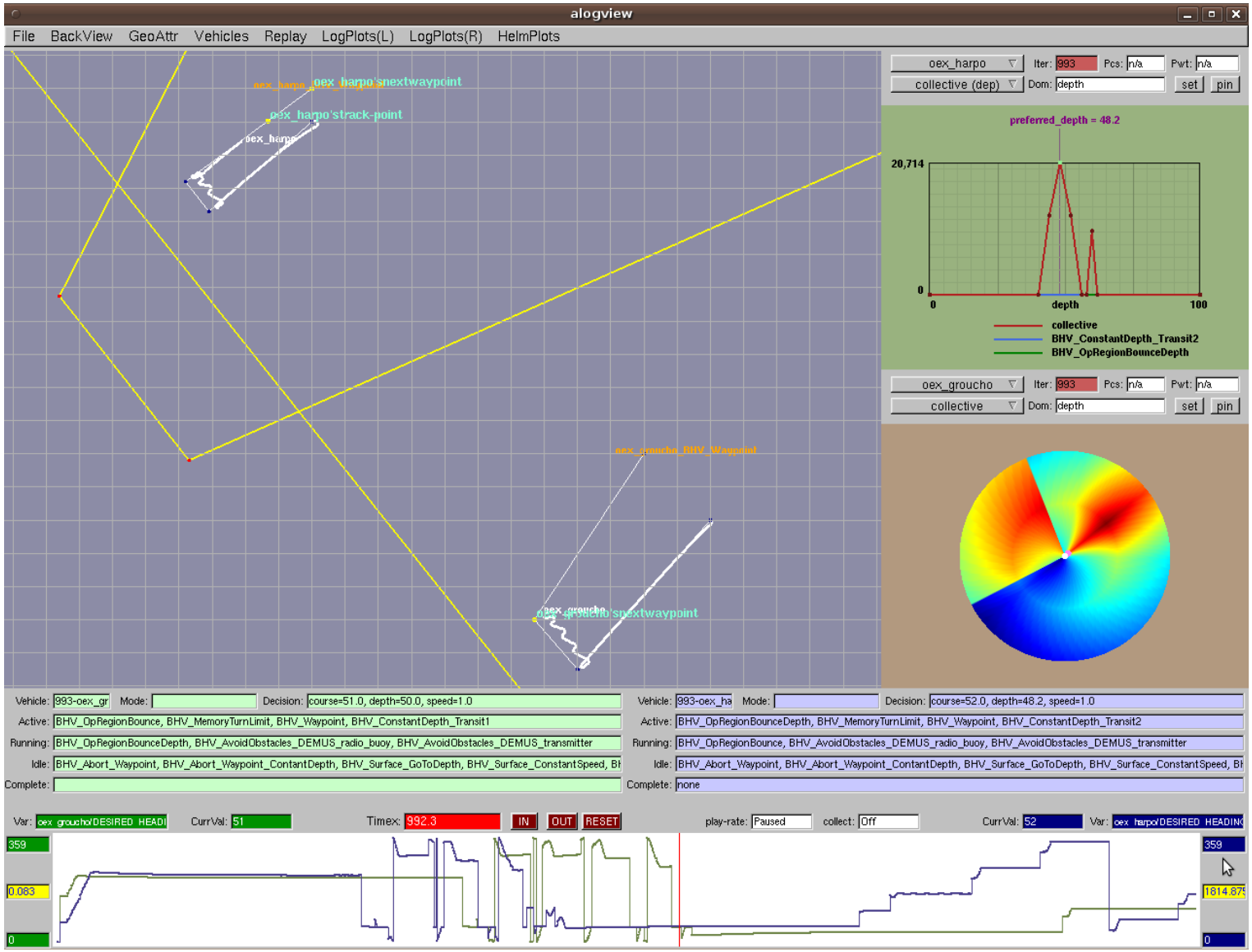
altitude ok

	depth ok	depth ¬ok
perimeter ok		
perimeter ¬ok		
2 polygons ¬ok		

altitude ¬ok

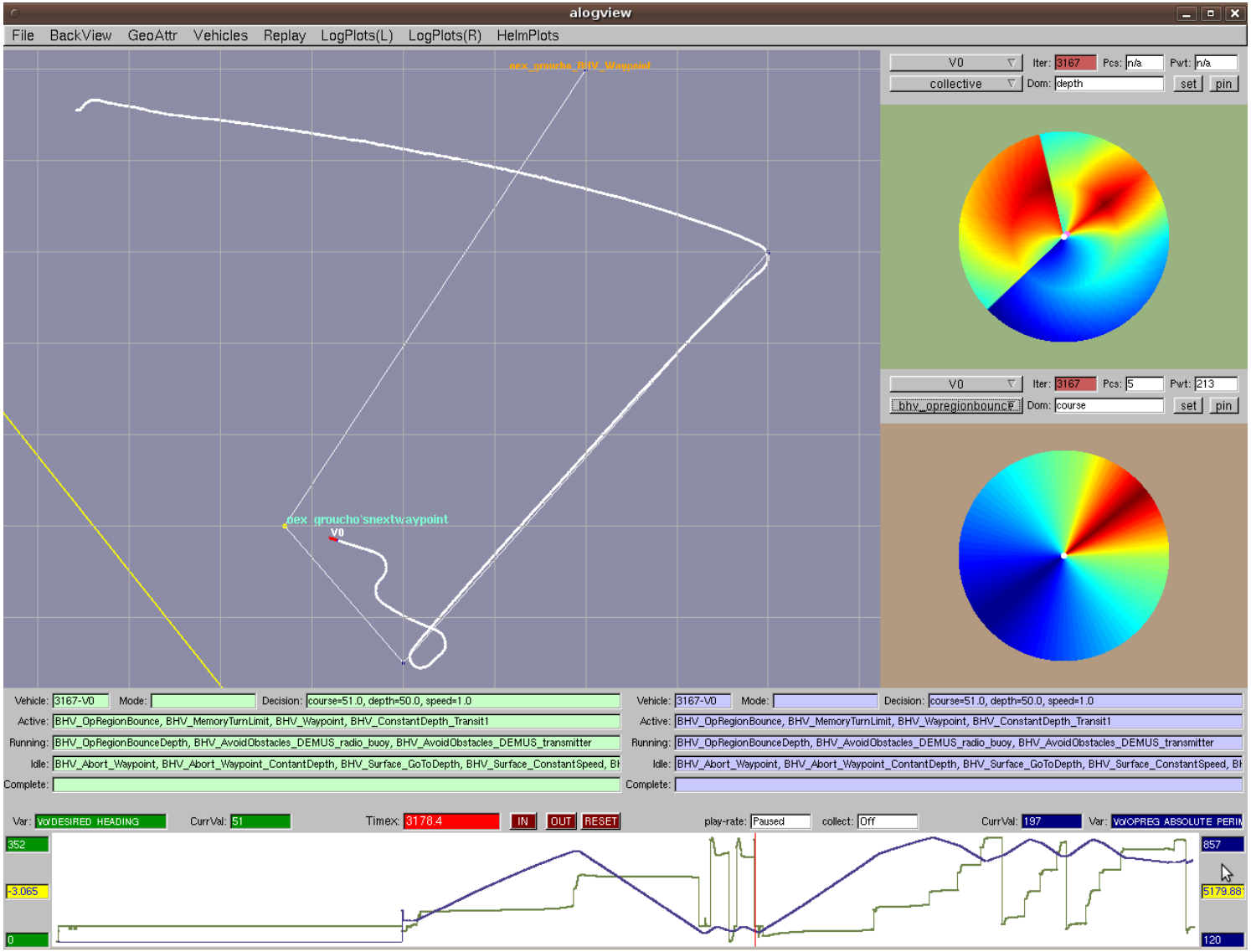
	depth ok	depth ¬ok
perimeter ok		
perimeter ¬ok		
2 polygons ¬ok		

Testing – simulation, results

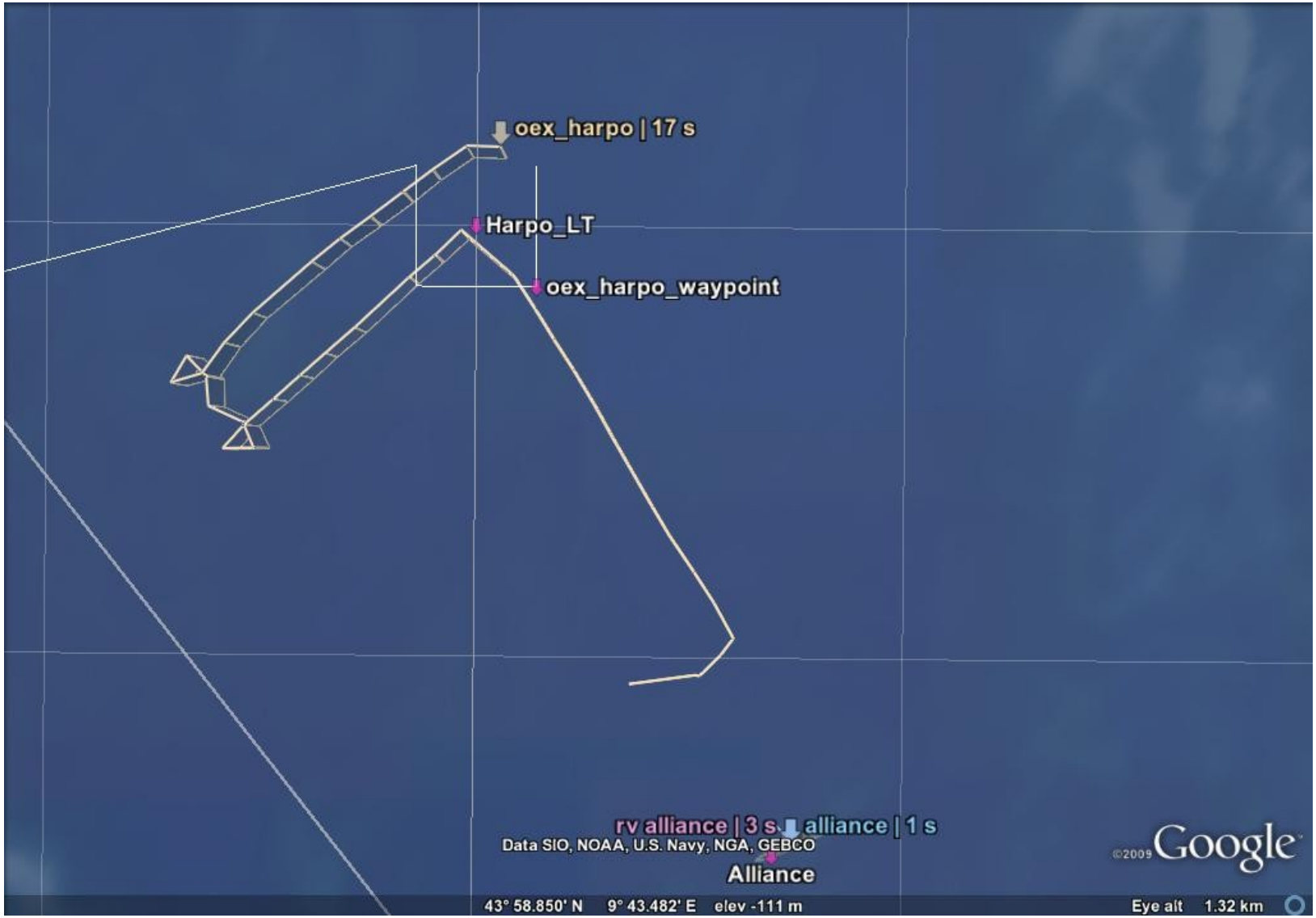


Testing: at sea : Groucho

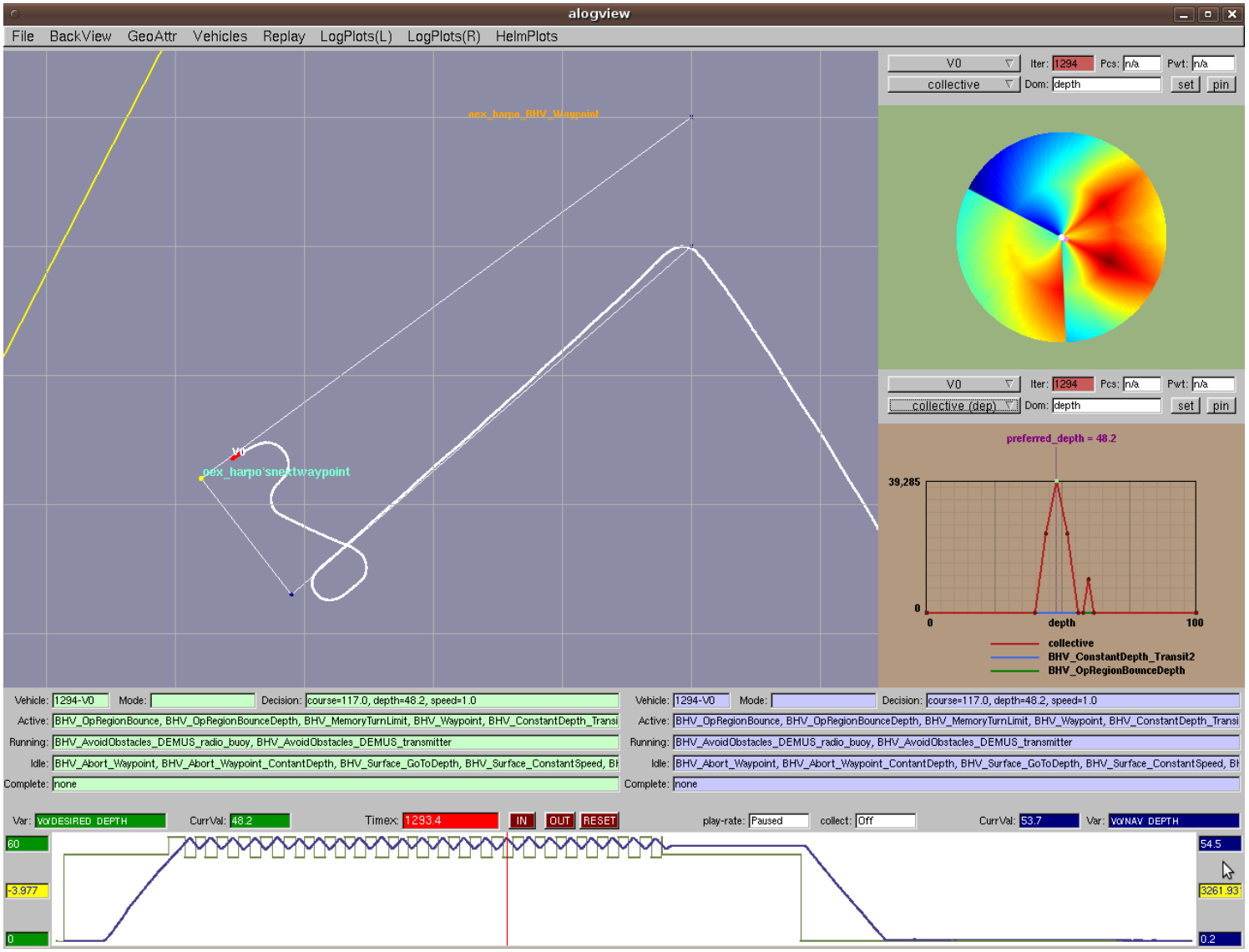
Testing: at sea : results Groucho Perimeter Bounce



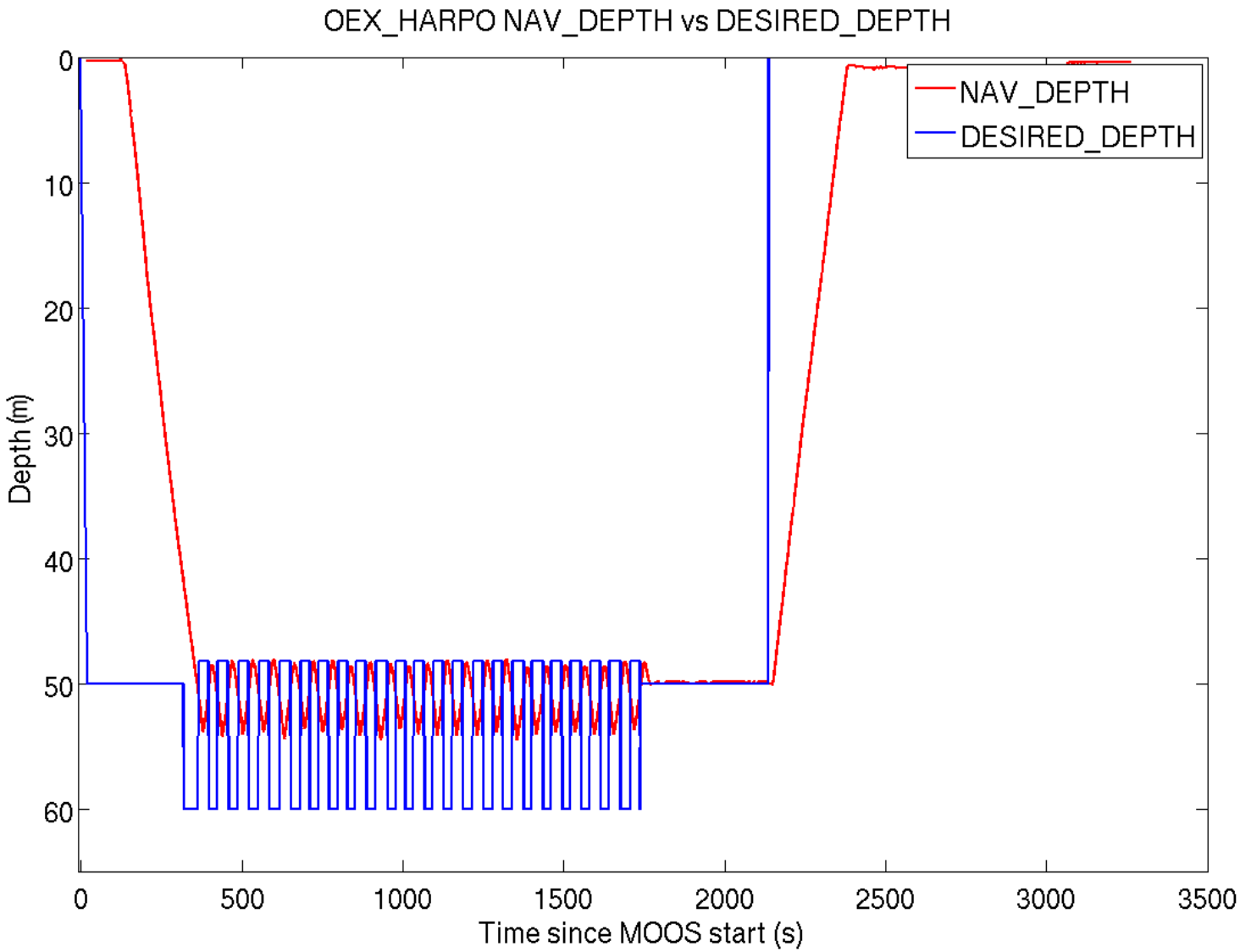
Testing: at sea : Harpo



Testing: at sea : results Harpo Perimeter Bounce



Testing: at sea : results Harpo Depth Bounce



Summary

- BHV_OpRegion's produced error upon OpRegion failure is not (always) desirable for AUVs.
- BHV_OpRegionBounce creates a bounce orthogonal for perimeter, up for depth
- Tested in simulation and at sea

Conclusions / Future

- To be used for our future sea trials.
- Danger: infinite loops

  - careful mission planning
- Distribute / merge into BHV_OpRegion

Questions