

Image Conference Scottsdale, Arizona

14-18 Jul 03



**David Lerman
Senior Systems Analyst
Warfighter Training Research Division
Air Force Research Laboratory**

The research reported in this presentation was sponsored by AFRL under contract # F41624-97-D-5000.

Compensating Vectors for Differences in Earth Representation Between Host and Image Generator



The Warfighter Training Research Division (WTRD) of AFRL is at the former Williams AFB in Mesa, Az, now Williams Gateway Airport. The vision of the WTRD is to provide the world's best training tools, ensuring that warfighters have the skills to win.

The contactor team consists of Lockheed Martin, Boeing and the Link Simulation & Training Division of L3 Communications. L3 pays my salary. However, most people believe that they work for the Lab regardless of the paycheck source.

I work primarily for a WTRD advanced simulation group tasked to develop commercial technologies and innovative software engineering approaches designed to significantly reduce costs and enhance fidelity. This group has produced innovations in COTS processors, re-host of embedded aircraft software, replacement of custom input/output systems with low cost COTS, and addressed network delays, time stamping, and coordinate transformations.

This talk describes problems making a Maverick electro optical or IR simulation hit the target regardless of the Image Generator (IG) earth representation. The problems were fixed by making vector calculations in the host account for differences in earth representation between the host calculations and the IG.



Real World Pre-Launch

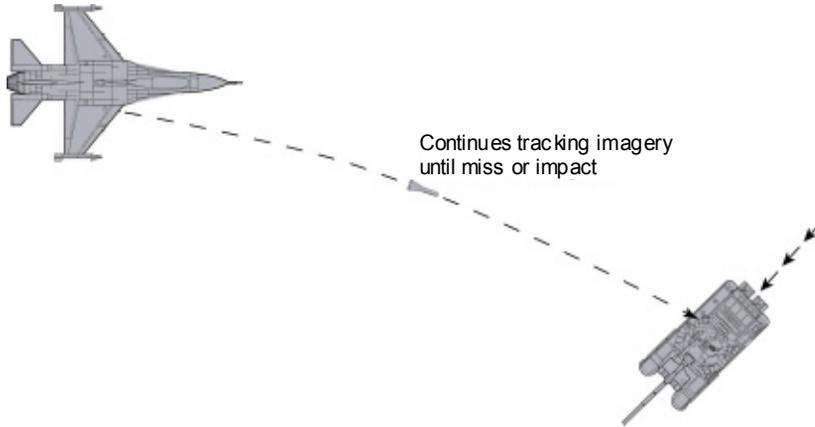


Seeker tracks target imagery





Real World Post-Launch



4

The above slides show the Maverick seeker tracking the target imagery both pre launch and during flyout.



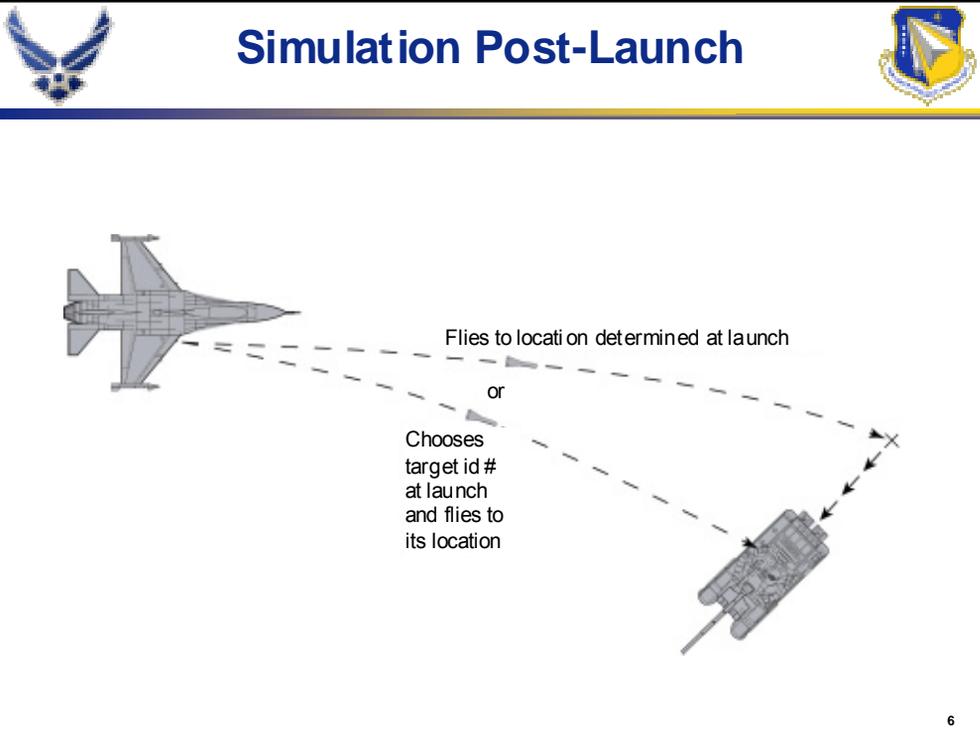
Simulation Pre-Launch



Seeker tracks target imagery

IG uses Laser rangefinder feature to return range and either terrain code or Target ID#





On the simulator we use an IG sensor channel dedicated to the Maverick while it is still on the aircraft. In principle we could keep the sensor channel dedicated to the Maverick after launch, in which case the Maverick would continue to track the imagery in a realistic way until striking or missing the desired target.

Unfortunately, after launch the sensor channel is needed elsewhere. This prevents the Maverick from tracking the target image to impact, so we must be creative (cheat) to complete the fly out.

The above slides show the Maverick seeker tracking the target imagery and using the IG laser rangefinder feature to receive range and either terrain code or target ID# pre-launch. After launch the Maverick maneuvers towards a model or geographic location determined at launch.

The decision whether to follow the position of a model of known ID, or to fly to a fixed location is based upon the algorithm in the next slide.



Simulated Maverick Flight Guidance Decision at Launch



```
if(IG returns a model ID#)
{
  this is ID# of model to follow after launch
}
else
{ /* IG returns range and terrain code */
  use find_tgt_near_mav_gates to find ID of model with LOS closest to seeker
  if(closest LOS is inside tracking gates)
  {
    this is ID# of model to follow after launch
  }
  else
  {
    use get_tgt_posn to find location of end of seeker LOS at launch
    this is location to try to fly to after launch
  }
}
```

7

If the IG returned a model ID #, the missile tries to follow that model position after launch.

Otherwise the host determines the ID # of that target with LOS closest to that of the seeker. If the target LOS is inside the seeker tracking gates the missile tries to fly to the known location of the model, even as it changes.

Otherwise the location of the end of the seeker LOS is calculated from the range and alignment, and the missile tries to fly there.

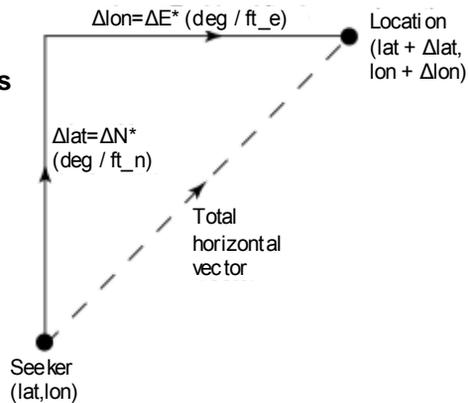


Initial Maverick Vector Work



Use seeker LOS to find ID # of closest model LOS
or find location of end of LOS vector

- Worked in geodetic coordinates
- Used feet/degree N or S at seeker to convert vectors to geodetic increments or vice versa
- Good enough out-the-window
- Miss with Maverick

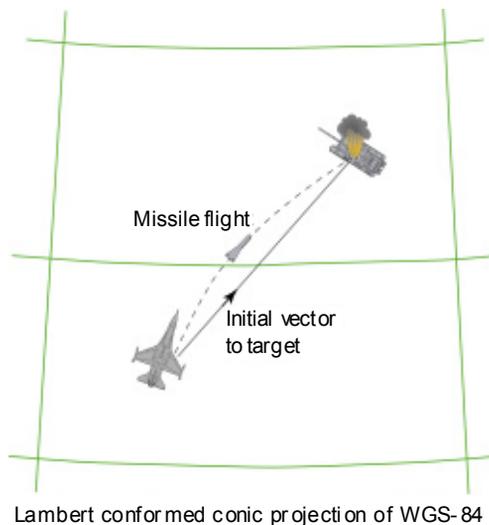


8

The slide shows that initially we used simple feet per degree North and East relationships to convert the sensor LOS vector into latitude and longitude increments or vice versa. In the past these algorithms have been satisfactory for out-the-window work. However, when combined with relatively long range and the effectively high magnification of the narrow FOV seeker, these algorithms gave errors that could cause the missile to miss the target.



Success with F-16 Simulator Integrated with E&S IG Flat Earth



- Do host calculations in same map as IG
- Project seeker location into map
- Project seeker_to_target vector into map, hence target location
- Invert map projection to find target geodetic coordinates

9

The fix for our immediate problem with an F-16 simulator integrated with an E&S IG was to do the host vector calculations in the same map projection as the IG, a Lambert Conformal Conic projection of the WGS-84 spheroid.

We project the sensor geodetic location into the map. With the seeker-to-target vector we can calculate the target location in the map, and convert back into geodetic coordinates. We cannot go wrong!



**Host and IG use WGS-84
Lambert Conformal Conic
Success with many databases
representing different locations**

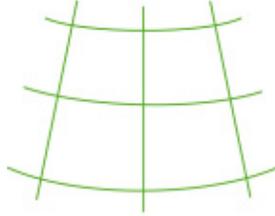
After development on an F-16, the Maverick software was implemented successfully on an A-10 FMT integrated with an E&S IG using a WGS-84 Lambert Conic Projection.

We had success with different databases representing different locations.

Then we integrated with an SE200 IG
and
immediately had problems hitting the target:



Flight Guidance Decision FAILURE TO FIND TARGET when A-10 Integrated with Spherical Earth IG



Host calculations
in Lambert Conic
of WGS-84

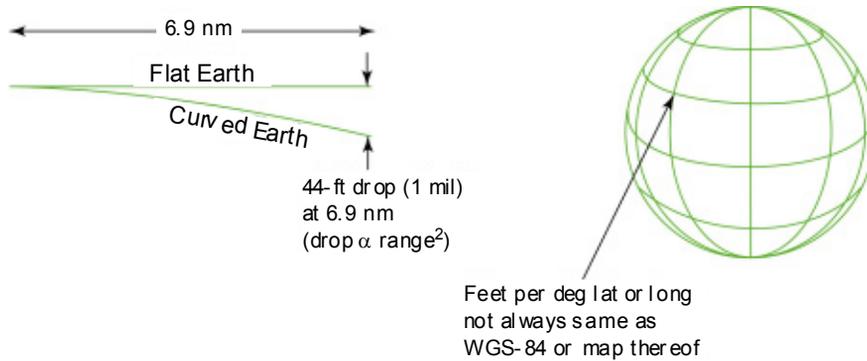


IG display based on
Spherical Earth Model
($R = 20,890,377.0$ ft)

After pacing the floor for a bit, we realized the problems resulted from the IG spherical earth differing from the host flat earth in two important ways.



Discrepancies between IG Sphere and Host Flat Earth



12

The first cause of the problem was that the curved earth MSL drops away from the observer's local horizontal. The drop is about 44 feet at a horizontal range of about 6.9 nm, and is proportional to the square of the horizontal range.

The second cause was that the feet per degree of latitude and longitude can differ significantly from the values on a WGS-84 spheroid or a projection thereof.

To confirm our suspicions, we needed to derive some equations and calculate some values.



Dimensions and Equations for Scale of Spherical Earth Surface Relative to WGS-84



R	spherical earth radius	20,890,377.0 feet
e^2	WGS-84 eccentricity ²	0.00669437999
a	WGS-84 equatorial radius	20925646.33 feet
ϕ	Latitude	

$$n_{scale}_{sphere} = \frac{R}{a} \times \frac{(1 - e^2 \times \sin^2 \phi)^{1.5}}{1 - e^2}$$

$$e_{scale}_{sphere} = \frac{R}{a} \times \sqrt{1 - e^2 \times \sin^2 \phi}$$

The equations and dimensions in the above slide were used to produce the following table.



Tabulation of Northerly and Easterly Scales of Spherical Earth to Surface Relative to WGS-84



Sphere (R = 20,890,370.0 ft) Scales Versus Latitude

Latitude ϕ	$nscale_{sphere}$	$escale_{sphere}$	$nscale/escale$
0 ⁰	1.005043	0.998315	1.0067
15 ⁰	1.004367	0.998091	1.0063
30 ⁰	1.002521	0.997479	1.0051
45 ⁰	1.000001	0.996643	1.0034
60 ⁰	0.997483	0.995806	1.0017
75 ⁰	0.995640	0.995192	1.0005
90 ⁰	0.994968	0.994968	1.0

Maximum bearing error is 3.35 mil, at Equator.

14

The table above shows that the Northerly and Easterly map scales on the sphere differ significantly not only from 1.0, but also from each other.

Compared with a WGS-84 reference spheroid it is clear that there will be a significant difference in the vectors connecting two points of specified geodetic coordinates. The differences in the northerly and easterly components would cause corresponding differences in range and bearing.

At a given latitude, the bearing errors are worst near the inter-cardinal points. The larger the difference between the northerly and easterly scales, the greater the error in bearing. The worst bearing errors would occur at the equator where northerly components are too long by 5.0 parts in 1000 and easterly components too short by 1.7 parts in 1000, causing bearing errors of up to 3.35 mil.

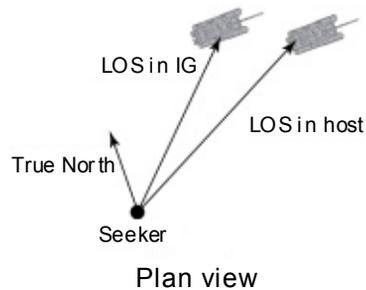
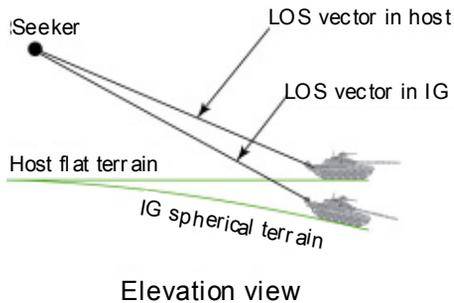
Bearings will be correct near the Poles but ranges will be short by 5 parts in 1000.



Vector Discrepancies Between WGS-84 Flat Earth Host and Spherical IG



Vector to location in IG is not same as vector to corresponding location in host, if host and IG earth representation differ



To use vector from IG in host calculations, it must be compensated for how it would appear in host, and vice versa. Use:

comp_host2ig

comp_ig2host

15

The slide shows plan and elevation views of the seeker-to-target vector in both a flat earth IG and a spherical earth IG. The depression to the target is obviously greater on the spherical earth than on the flat earth. The bearing relative to true North and the range can also differ due to scale differences.

Therefore vectors from the host must be compensated for comparison with vectors in the IG earth representation, and vectors from the IG must be compensated for use in the host earth representation. Use functions *comp_host2ig* and *comp_ig2host*.



We Did Not Use the Exact Solution



**There is an exact solution,
cumbersome for many vectors.
Approximation preferred.**

16

We could convert vectors from their appearance in the host to that in the IG, or vice versa, by going through coordinate transformations or their inverses to find exact solutions. However, compared to the approximations that we derived, this would be time consuming for the LOS to every potential target.

Both the compensating functions require some common parameters that are independent of the vectors being compensated. These common parameters need be calculated only once per frame.



First part of approximations, performed once per frame



IG_earth_curve \approx
1/a_WGS-84, or
1/R for sphere, or
0.0 for flat earth

Obtain IG N & E map scale (N \neq E for sphere)

Calculate host map scale at current location (assume N = E)

Hence:

IGomap_nscale_factor = IG_nscale / host_nscale

IGomap_escale_factor = IG_escale / host_nscale

17

The first part of the approximations is performed once per frame in the host. The output parameters are:

IG earth curvature,

Northerly ratio of the IG map scale to that of the host,

Corresponding Easterly ratio.

Map scale is defined with respect to a WGS-84 spheroid.

The host vector calculations are performed in flat earth map coordinates, so there is no host earth curvature.

Flat earth map projections must be conformal so that at any location East projects perpendicular to true North, and Easterly Scale equals Northerly Scale. Also, the projection must be chosen so that the scale is slow changing and close to 1.0 in the gaming area.



Compensating Host Vector to Appearance in IG



```
comp_host2ig(end_alt, mapned[3], v mapned[3])
{
    if (v is_earth_curve = zero)
    {
        /* because the two map scale factors mutually equal */
        v mapned[0] = mapned[0] * IGomap_nscale_factor;
        v mapned[1] = mapned[1] * IGomap_escale_factor;

        v mapned[2] = mapned[2];
    }
    continued on next slide
}
```

If the IG has zero earth curvature the calculations are simple since there is no earth curvature drop and the map scales do not vary with bearing.



Compensating Host Vector to Appearance in IG (continued)



```
comp_host2ig(continued from previous slide)
    else /* vis_earth_curve ≠ zero */
    {
        altitude_factor = 1.0 + end_alt * IG_earth_curve;
        rotate horizontal comps thru merid conv into true N and E;
        calculate vis horizontal components by
            * (altitude factor * N or E scale factor);
        rotate vis horizontal comps back to get vmapned[0] & [1];
        calculate hor_range_sqd;
        earth_curve_drop = 0.5 * IG_earth_curve * hor_range_sqd;
        vmapned[2] = mapned[2] + earth_curve_drop;
    }
}
```

19

However, with a curved earth IG we must account for the drop due to curvature. Also, because (for a sphere) the scale is not independent of bearing we must rotate the host vector components from map axes into true N and E before applying the scale ratios and then rotating back to get the components in map axes.

Observe that an altitude factor is applied along with the scale ratios. This altitude factor represents the increase in feet per degree with increasing altitude above a curved earth, and is based upon the altitude of the end of the vector, not the altitude of the sensor.



Compensating IG Vector to Appearance in Host



```
comp_ig2host(start_alt, v mapned[3], mapned[3])
{
    if (vis_earth_curve = zero)
    {
        /* because the two map scale factors mutually equal */
        mapned[0] = v mapned[0] / IGomap_nscale_factor;
        mapned[1] = v mapned[1] / IGomap_escale_factor;

        mapned[2] = v mapned[2];
    }
    continued on next slide
}
```

Once more, if the IG has zero earth curvature the calculations are simple since there is no earth curvature drop and the map scales do not vary with bearing.



Compensating IG Vector to Appearance in Host (continued)



```
comp_ig2host(continued from previous slide)
  else /* IG earth is curved */
  {
    calculate hor_range_sqd & earth_curve drop;
    mapped[2] = v mapped[2] - earth_curve_drop;
    end_alt = start_alt - mapped[2];
    altitude_factor = 1.0 + end_alt * IG_earth_curve;
    rotate horizontal comps thru merid conv into true N and E;
    calculate host horizontal components by
      / (altitude factor * N or E scale factor);
    rotate host horizontal comps back to get mapped[0] & [1];
  }
}
```

21

Again, with a curved earth IG we must account for the drop due to curvature and (for a sphere) must rotate the vector components into true N and E before applying the altitude factor and N and E scale ratios, and then rotate back into map coordinates.

The compensating functions were implemented and tested on the A-10 integrated with a spherical earth IG. Tests were run from a known sensor location to a known target location, with and without compensating the vectors.



get_tgt_posn uses *comp_ig2host*

target range about 4.14 nm at bearing about -33.6°

Expected error without compensation:

0.60 mil extra depression, about 2.2 mil az≈
small range error as N stretch and E short cancel

Actual error without compensation:

0.64 mil extra depression, 2.26 mil az
19.4 ft further than known location

Actual error with compensation:

0.035 & 0.044 mil of AZ & EL, (< 1 pixel)
2.4 feet closer than known location

get_tgt_posn uses *comp_ig2host*. Without compensation the alignment and range errors matched expectations. Then function *comp_ig2host* reduced the alignment errors to less than 1 pixel. The target calculated position being 2.4 feet closer than the known location is partly due to the near face of the target being closer than the CG.



find_tgt_near_mav-gates uses *comp_host2ig*

target range about $5\frac{1}{2}$ nm on bearing of about $-33\frac{1}{2}^\circ$

Expected misalignment without compensation = 2.3 mil

Actual misalign without comp = 2.1 to 2.3 mil

Actual misalign with comp = 0.15 to 0.23 mil (3 to 4.6 pixels)

find_tgt_near_mav-gates uses *comp_host2ig*. Without compensation the alignment error matched the expectation. Then function *comp_host2ig* reduced the error to 1/10 of the uncompensated error, although still a little larger than in the preceding test.

It is not known how much the residual errors reflect the approximate nature of the compensation, and how much they reflect noise and other errors in tracking accuracy, sensor window definitions, or digitizing of the video in the video capture board.

The tests were run with the host position and attitude frozen, hence the very accurate seeker alignment. In dynamic tracking conditions the seeker does not track the target so accurately, and the gates are not in the middle of the seeker imagery. [WHAT SIZE ERRORS?] For that reason, functions *find_tgt_near_mav_gates* and *get_tgt_posn* should be modified to account for the gate position in the seeker.

The full equations are not yet implemented in our hosts. The host map scale is set to 1.0, as is the scale for an IG flat earth. However, the tests were run in an area where the scale was exactly 1.0. In general we train in areas where the map scale is not significantly different from 1.0, and provided we use a common map projection between host and IG it does not matter if the scale changes.

However, we are now fielding IGs that use a UTM projection so that the map scale will vary between the IG and the Lambert Conic host as we move around the database. Therefore we should implement the full equations on our system.

With the host and IG using different flat earth conformal projections of a common earth shape, preferably WGS-84, we can compensate the Maverick for scale differences. Out-the-window problems due to scale differences are unlikely to be large enough to notice. However, a lack of visual earth curvature is seen out-the-window. For instance, at high altitude the horizon is insufficiently depressed below the HUD horizon bar.

An IG using a spherical earth probably causes alignment problems between the out-the-window and HUD displays that are ignored or attributed to other causes. Fortunately we are phasing out spherical earth IGs in our programs.

My personal opinion is that we should use the WGS-84 spheroid in our IGs, and use corresponding exact equations in the host. Then the horizon is seen at the correct elevation, the calculations do not need compensation for errors, and we need not spend time analyzing the effects of doing things wrong.

In the meantime, we know how to compensate our Maverick equations for differences in earth representation between host and IG.