Smart Velocity Waypoint Generation

1 Introduction

The purpose of this document is to define a procedure for intelligently computing an entire batch of velocity waypoints given a set of northing and easting waypoints. This method has built-in intelligence ("smart") to insure that acceleration and velocity limits are always met.

2 Inputs

The process of computing velocity waypoints begins with the following set of inputs that are assumed known:

$NWP[\,]$ = N-dimensional vector of northing waypoint coordinates [m]
$EWP[\,]$ = N-dimensional vector of easting waypoint coordinates [m]
v_{start} = initial velocity at first (n=1) waypoint [m/s]
v_{final} = final velocity at last (n=N) waypoint [m/s]
v_{max} = nominal maximum allowable velocity [m/s]
v_{min} = nominal minimum allowable velocity [m/s]
$a_{max,lat}$ = maximum allowable lateral acceleration [m/s/s]
$a_{max,long}$ = maximum allowable longitudinal acceleration [m/s/s]

3 Assumptions

The process for computing velocity waypoints is based on the following assumptions:

- The distance between waypoints is not necessarily constant, but is generally bounded between 0.5m and 20m
- When starting from zero velocity, the first velocity waypoint v_{start} should be chosen to be a small positive value such as 0.5 m/s and should never chosen to be zero
- The final point in the set of waypoints should be a stopping point so that $v_{final} = 0$
- The objective of this algorithm is to maximize the velocity at all waypoints subject to the velocity and acceleration constraints
- This analysis will only be based on the assumption of interpolation of straight line segments between velocity waypoints
- 1-based indexing is used for this derivation

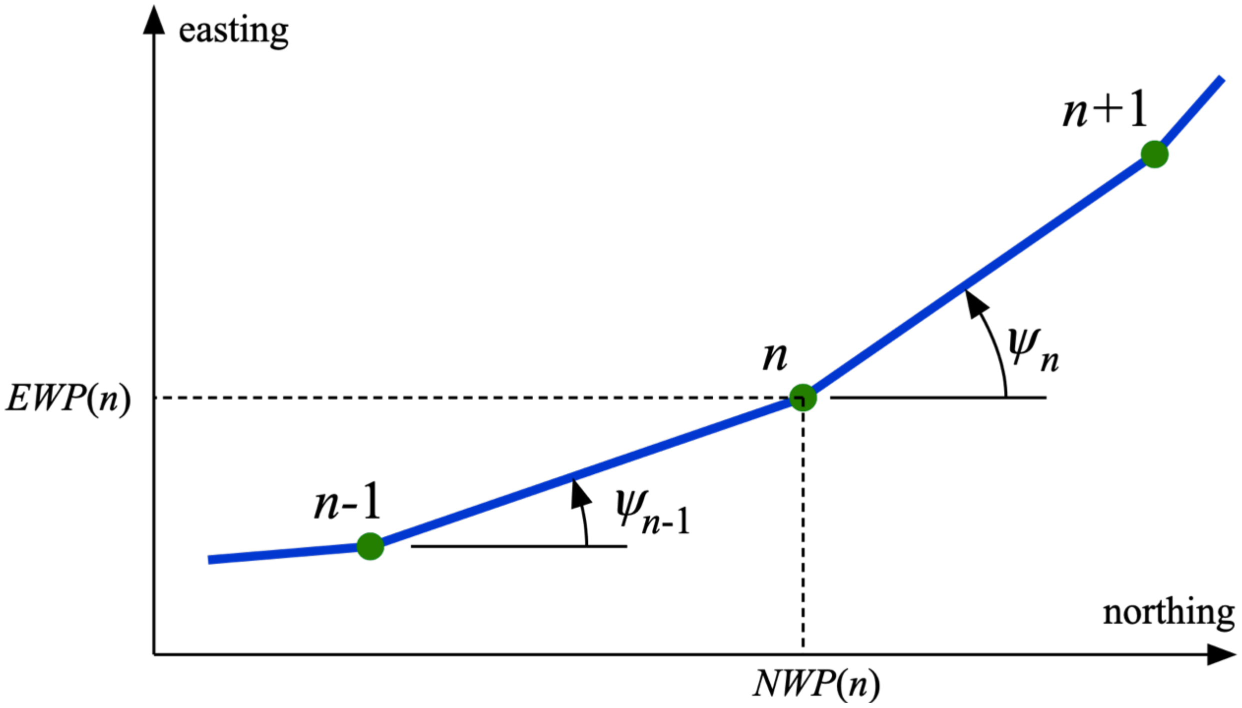
4 Step 1: Estimate Curvature, Heading, and Path Distance

The ***curvature*** κ is defined as the inverse of the radius of curvature, but curvature is also the rate of change of tangent angle with respect to path position:

$$\kappa\left[\frac{rad}{m}\right] = \frac{1}{R[m]} = \frac{d\psi}{ds}\left[\frac{rad}{m}\right]$$

where R is the instantaneous radius of curvature, ψ is the heading angle (i.e. tangent angle), and s is the path position.

Smart Velocity Waypoint Generation



In order to estimate the curvature and heading at the nth waypoint, we first construct two simple quadratic polynomials for each of the sets of northing and easting waypoint coordinates:

$$NWP(u) = p_N(1) \times u^2 + p_N(2) \times u + p_N(3)$$
$$EWP(u) = p_E(1) \times u^2 + p_E(2) \times u + p_E(3)$$

where the parametric variable u is defined to be: $u_1 \le u \le u_3$ such that:

$$\begin{aligned} NWP(u_1) &= NWP(n-1) \qquad & EWP(u_1) &= EWP(n-1) \\ NWP(u_2) &= NWP(n) \qquad & EWP(u_2) &= EWP(n) \\ NWP(u_3) &= NWP(n+1) \qquad & EWP(u_3) &= EWP(n+1) \end{aligned}$$

and:

$$u_1 = -\sqrt{\left(NWP(n) - NWP(n-1)\right)^2 + \left(EWP(n) - EWP(n-1)\right)^2}$$
$$u_2 = \sqrt{\left(NWP(n) - NWP(n)\right)^2 + \left(EWP(n) - EWP(n)\right)^2} = 0$$
$$u_3 = +\sqrt{\left(NWP(n+1) - NWP(n)\right)^2 + \left(EWP(n+1) - EWP(n)\right)^2}$$

The distances between waypoints can be pre-computed and accumulated as:

$$SWP(n) = SWP(n-1) + \sqrt{\left(NWP(n) - NWP(n-1)\right)^2 + \left(EWP(n) - EWP(n-1)\right)^2}$$

Using this definition, the parametric variable limits simplify to:

$$u_1 = SWP(n-1) - SWP(n) < 0$$
$$u_2 = 0$$
$$u_3 = SWP(n+1) - SWP(n) > 0$$

We can now define the general 2nd order polynomial curve fit equations as:

$$\begin{bmatrix} WP(n-1) \\ WP(n) \\ WP(n+1) \end{bmatrix} = \begin{bmatrix} u_1^2 & u_1 & 1 \\ u_2^2 & u_2 & 1 \\ u_3^2 & u_3 & 1 \end{bmatrix} \begin{bmatrix} p(1) \\ p(2) \\ p(3) \end{bmatrix}$$

Note that this equation must be solved using both the northing and the easting coordinates separately. The only unknowns in this linear algebra equation are the polynomial coefficients. Using the symbolic toolbox in Matlab, a closed-form solution can be obtained as:

$$DET = (u_1 - u_2)(u_1 - u_3)(u_2 - u_3)$$

$$INV = \left(\frac{1}{DET}\right) \begin{bmatrix} u_2 - u_3 & u_3 - u_1 & u_1 - u_2 \\ u_3^2 - u_2^2 & u_1^2 - u_3^2 & u_2^2 - u_1^2 \\ u_2 u_3 (u_2 - u_3) & -u_1 u_3 (u_1 - u_3) & u_1 u_2 (u_1 - u_2) \end{bmatrix}$$

The final curve fit solution for the polynomial coefficients is:

$$\begin{bmatrix} p(1) \\ p(2) \\ p(3) \end{bmatrix} = \left(\frac{1}{DET}\right) \begin{bmatrix} u_2 - u_3 & u_3 - u_1 & u_1 - u_2 \\ u_3^2 - u_2^2 & u_1^2 - u_3^2 & u_2^2 - u_1^2 \\ u_2 u_3 (u_2 - u_3) & -u_1 u_3 (u_1 - u_3) & u_1 u_2 (u_1 - u_2) \end{bmatrix} \begin{bmatrix} WP(n-1) \\ WP(n) \\ WP(n+1) \end{bmatrix}$$

To estimate the curvature, we use the parametric curves with the definition from Wikipedia (https://en.wikipedia.org/wiki/Curvature):

$$\kappa(u) = \frac{\left(\frac{dNWP(u)}{du} \times \frac{d^2EWP(u)}{du^2}\right) - \left(\frac{dEWP(u)}{du} \times \frac{d^2NWP(u)}{du^2}\right)}{\left(\left(\frac{dNWP(u)}{du}\right)^2 + \left(\frac{dEWP(u)}{du}\right)^2\right)^{3/2}}$$

Using the parametric polynomial definitions:

$$\frac{dNWP(u)}{du} = 2p_N(1)u + p_N(2)$$
$$\frac{dEWP(u)}{du} = 2p_E(1)u + p_E(2)$$
$$\frac{d^2NWP(u)}{du^2} = 2p_N(1)$$
$$\frac{d^2EWP(u)}{du^2} = 2p_E(1)$$

Evaluating the curvature at the nth waypoint, we have a simplified result since $u = u_2 = 0$

$$\kappa(0) = 2\left(\frac{p_N(2) \times p_E(1) - p_E(2) \times p_N(1)}{\left(\left(p_N(2)\right)^2 + \left(p_E(2)\right)^2\right)^{3/2}}\right)$$

At the very first point and last point in a waypoint data set, we will still compute quadratic polynomial fits for the northing and easting coordinates, but evaluate the curvature at the appropriate end point.

For example, at the very first point in a waypoint data set:

$$\kappa(u_1) = \frac{\left(2p_N(1)u_1 + p_N(2)\right) \times 2p_E(1) - \left(2p_E(1)u_1 + p_E(2)\right) \times 2p_N(1)}{\left(\left(2p_N(1)u_1 + p_N(2)\right)^2 + \left(2p_E(1)u_1 + p_E(2)\right)^2\right)^{3/2}}$$

And at the very last point in a waypoint data set:

$$\kappa(u_3) = \frac{\left(2p_N(1)u_3 + p_N(2)\right) \times 2p_E(1) - \left(2p_E(1)u_3 + p_E(2)\right) \times 2p_N(1)}{\left(\left(2p_N(1)u_3 + p_N(2)\right)^2 + \left(2p_E(1)u_3 + p_E(2)\right)^2\right)^{3/2}}$$

We can also use the quadratic polynomial parametric curves to compute the exact heading angle using a trig identity.

$$tan\left(\psi(u)\right) = \frac{\left(\frac{dEWP(u)}{du}\right)}{\left(\frac{dNWP(u)}{du}\right)} = \frac{2p_E(1)u + p_E(2)}{2p_N(1)u + p_N(2)}$$

At the nth waypoint, the heading angle is:

$$\psi(0) = tan^{-1}\left(\frac{p_E(2)}{p_N(2)}\right)$$

Remember that in the NED (north-east-down) coordinate system which we are using, a heading angle of $\psi = 0$ corresponds to a northing direction, and a positive rotation of the heading corresponds to a rotation from north to east.

Also remember that in the NED coordinate system, since curvature is the rate of change of heading angle with respect to path position, then a positive curvature means we are rotating from northing to easting.

When viewed from space, with north pointed up and east pointed to the right, a positive heading rotation and a positive curvature is in the clockwise direction.

5 Step 2: Compute Maximum Velocity based on Curvature

We now know the curvature at each waypoint from Step 1. We also know that since the curvature is inversely proportional to the radius of curvature, we can easily compute the lateral acceleration, assuming the vehicle has the specified velocity, heading, and path curvature at this waypoint:

$$a_{lat} = \frac{v_{car}^2}{R} = \kappa v_{car}^2$$

The maximum allowable lateral acceleration is specified as $a_{max,lat}$. This acceleration corresponds to a vehicle velocity of:

$$v_{max,lat} = \sqrt{\frac{a_{max,lat}}{\kappa}}$$

For small values of curvature, this result could lead to a velocity that exceeds the specified velocity limit, so velocity waypoints in this step must also be limited by v_{max}:

$$VWP(n) = min\left(v_{max}, \sqrt{\frac{a_{max,lat}}{\kappa(n)}}\right)$$

For the very first and very last waypoints,

$$VWP(1) = min(v_{max}, v_{start})$$
$$VWP(N) = min\left(v_{max}, v_{final}\right)$$

6 Step 3: Rate Limit based on Positive Longitudinal Acceleration

From Step 2, we have a candidate velocity waypoint vector that has the appropriate restrictions to limit velocity and lateral acceleration at the specified levels while maximizing speed. In Step 3, we will further restrict the velocity waypoints to insure that longitudinal acceleration limits are met. This restriction must occur in two separate stages so in this step, we focus only on positive longitudinal accelerations, i.e. $\frac{dv}{dt}>0$. The basic idea is that if we increase the velocity too much between waypoints, we will exceed the longitudinal acceleration limit, so we must limit how much the velocity can increase between waypoints.

Longitudinal acceleration is defined as:

$$a_{long} = \left(\frac{dv}{dt}\right) = \left(\frac{dv}{ds}\right)\left(\frac{ds}{dt}\right) = \left(\frac{dv}{ds}\right)v$$

Invoking the assumption that the velocity profile is interpolated along straight line segments connecting the waypoints, we can approximate the actual longitudinal acceleration as a change in velocity between two waypoints:

$$a_{long}(n) = \left(\frac{VWP(n) - VWP(n-1)}{SWP(n) - SWP(n-1)}\right) VWP(n)$$

If we now substitute $a_{long}(n) = a_{max,long}$, we can solve for the maximum allowable increase in velocity between these two waypoints:

$$\Delta v_{max}(n) = \frac{a_{max,long} \times (SWP(n) - SWP(n-1))}{VWP(n)}$$

Finally, we can apply this maximum change in velocity to the existing VWP data:

```
$if\ \left((VWP(n) - VWP(n-1)) > \Delta v_{max}(n)\right)$
        $VWP(n) = min\ (v_{max},\ VWP(n-1) + \Delta v_{max}(n))$
$end$
```

Note that the conditional test above only affects positive rates of change (i.e. acceleration). In order for this approach to work, the VWP array must be evaluated starting at n=1 and ending at n=N, in sequence.

7 Step 4: Rate Limit based on Negative Longitudinal Acceleration

In this step, we will further restrict the velocity waypoints to insure that negative longitudinal acceleration limits, i.e. deceleration $\frac{dv}{dt}<0$, are met. The basic idea is that if we decrease the velocity too much between waypoints, we will exceed the longitudinal deceleration limit, so we must limit how much the velocity can decrease between waypoints.

In order for this approach to work, the VWP array must be evaluated backward in time starting at n=N and ending at n=1, in sequence. Given this ordering, we must make a slight change to the formula for computing the longitudinal acceleration:

$$a_{long}(n) = \left(\frac{VWP(n) - VWP(n-1)}{SWP(n) - SWP(n-1)}\right) VWP(n-1)$$

If we now substitute $a_{long}(n) = -a_{max,long}$, we can solve for the maximum allowable decrease in velocity between these two waypoints during deceleration:

$$\Delta v_{max}(n) = \frac{-a_{max,long} \times (SWP(n) - SWP(n-1))}{VWP(n-1)}$$

Finally, we can apply this maximum change in velocity to the existing VWP data:

```
$if\ \left(\left(VWP(n) - VWP(n-1)\right) < \Delta v_{max}(n)\right)$
    $VWP(n-1) = max\ (v_{min},\ VWP(n) - \Delta v_{max}(n))$
$end$
```

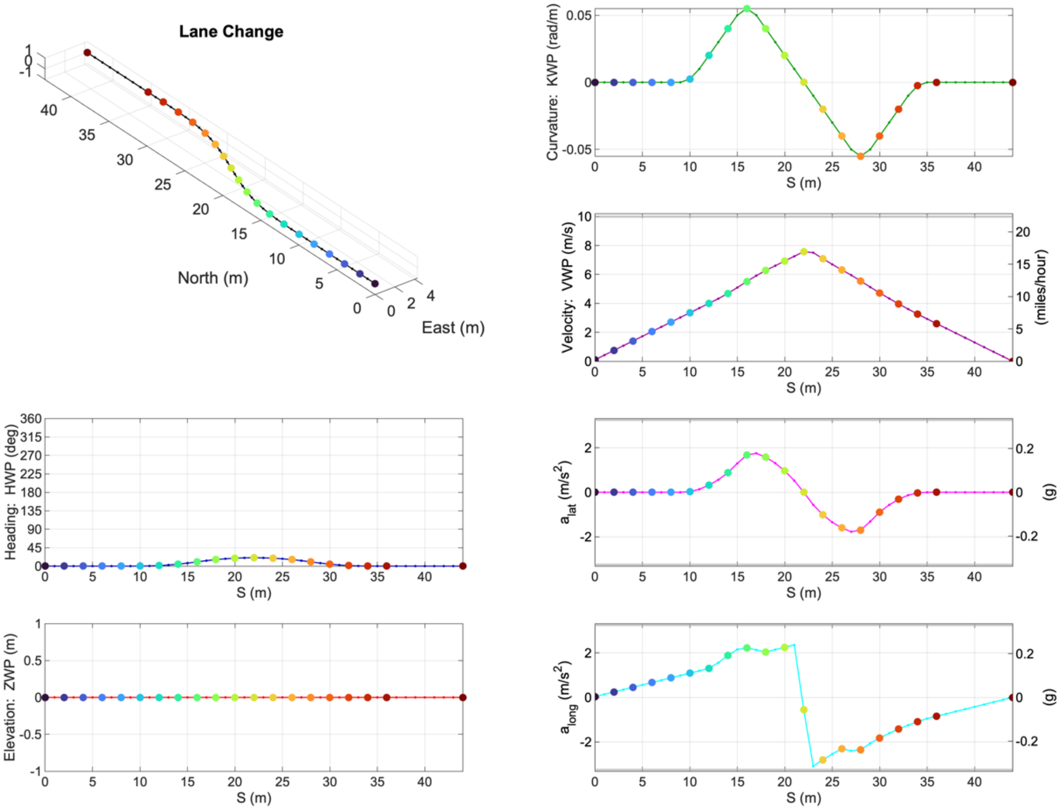
Note that the conditional test above only affects negative rates of change (i.e. deceleration).

8 Validation Results

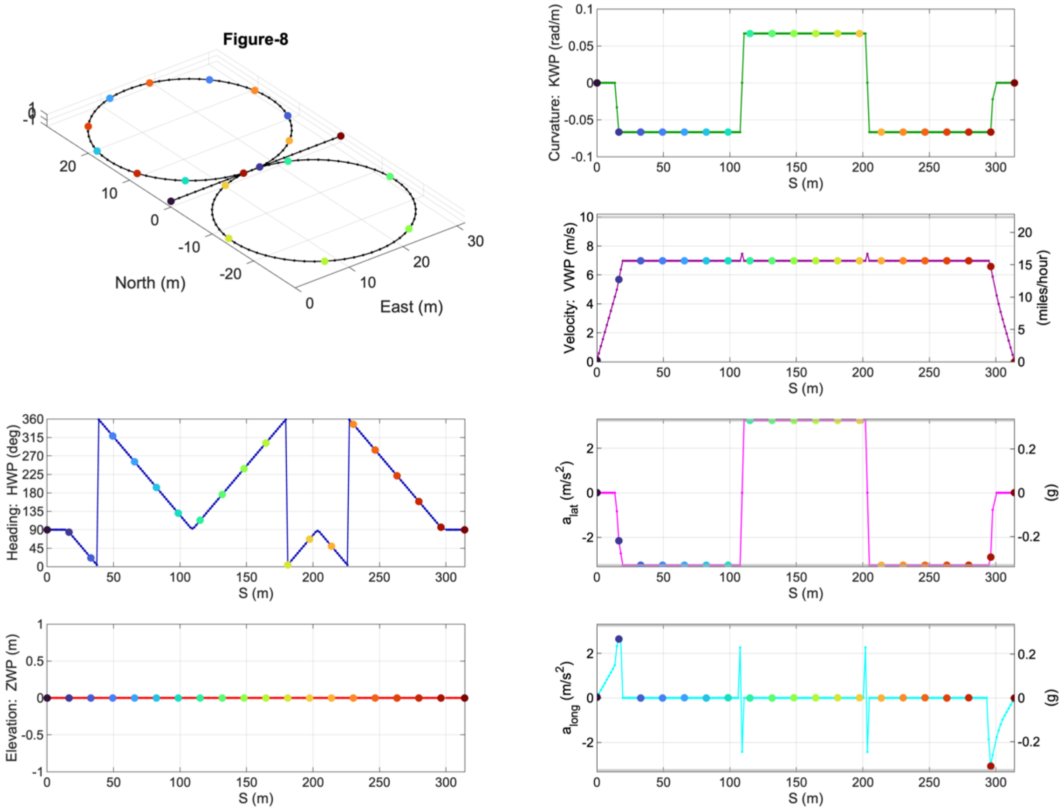
The following plots were generated using sample waypoint data sets used for bench testing an autonomous vehicle controller. In all cases,

$v_{start} = 0.1\left[\frac{m}{s}\right]$ $\quad v_{final} = 0\left[\frac{m}{s}\right]$
$v_{max} = 10\left[\frac{m}{s}\right]$ $\quad v_{min} = 0\left[\frac{m}{s}\right]$
$a_{max,lat} = 3.25\left[\frac{m}{s^2}\right]$ $\quad a_{max,long} = 3.25\left[\frac{m}{s^2}\right]$

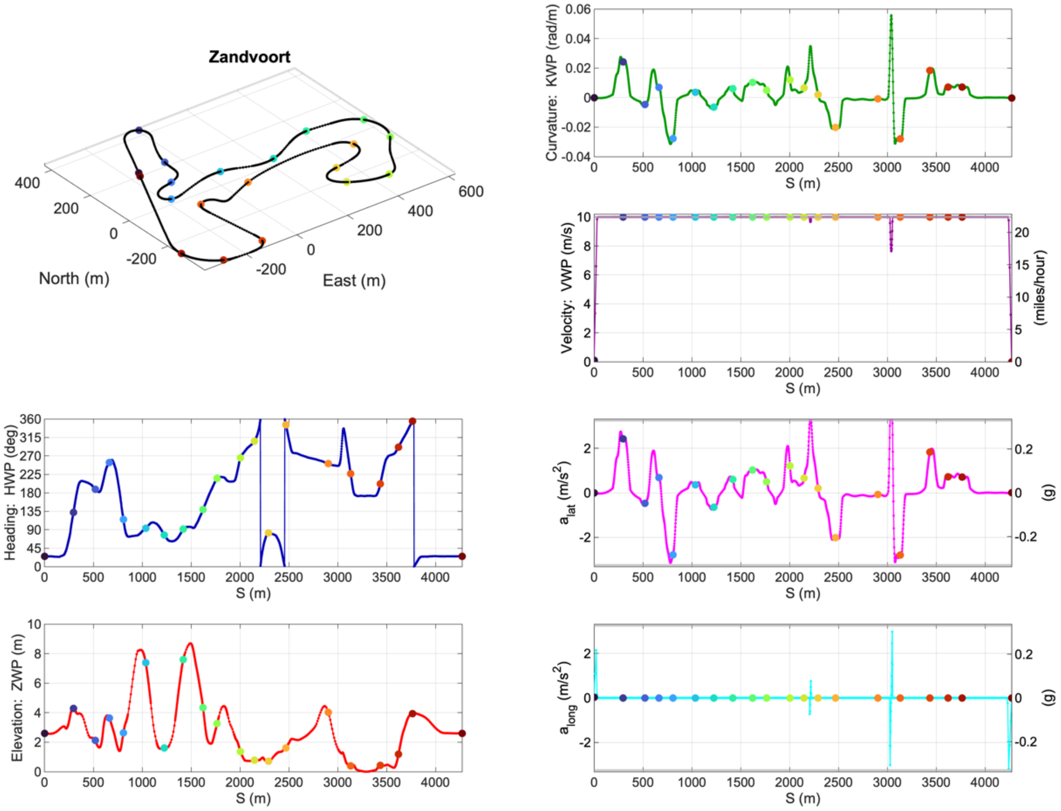
Notice that gray lines are drawn at the velocity and acceleration limits for reference. Also notice that due to the speed and curvature of the data sets, some data sets do not get to the maximum specified velocity.



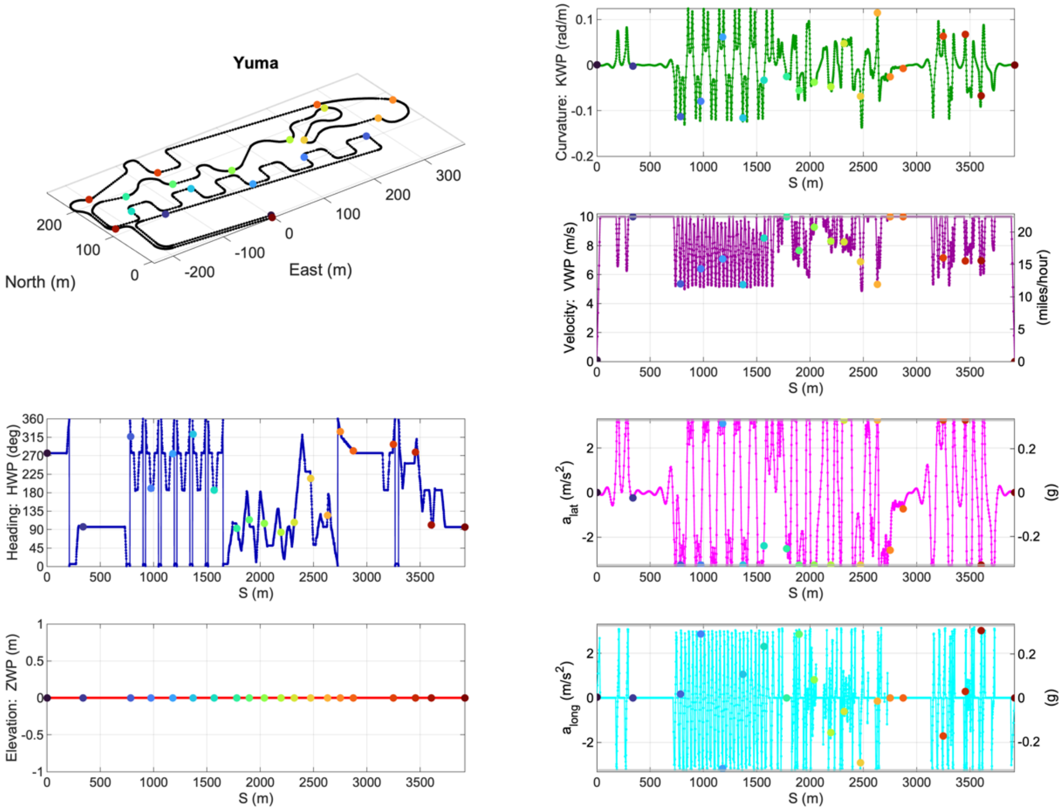
Smart Velocity Waypoint Generation



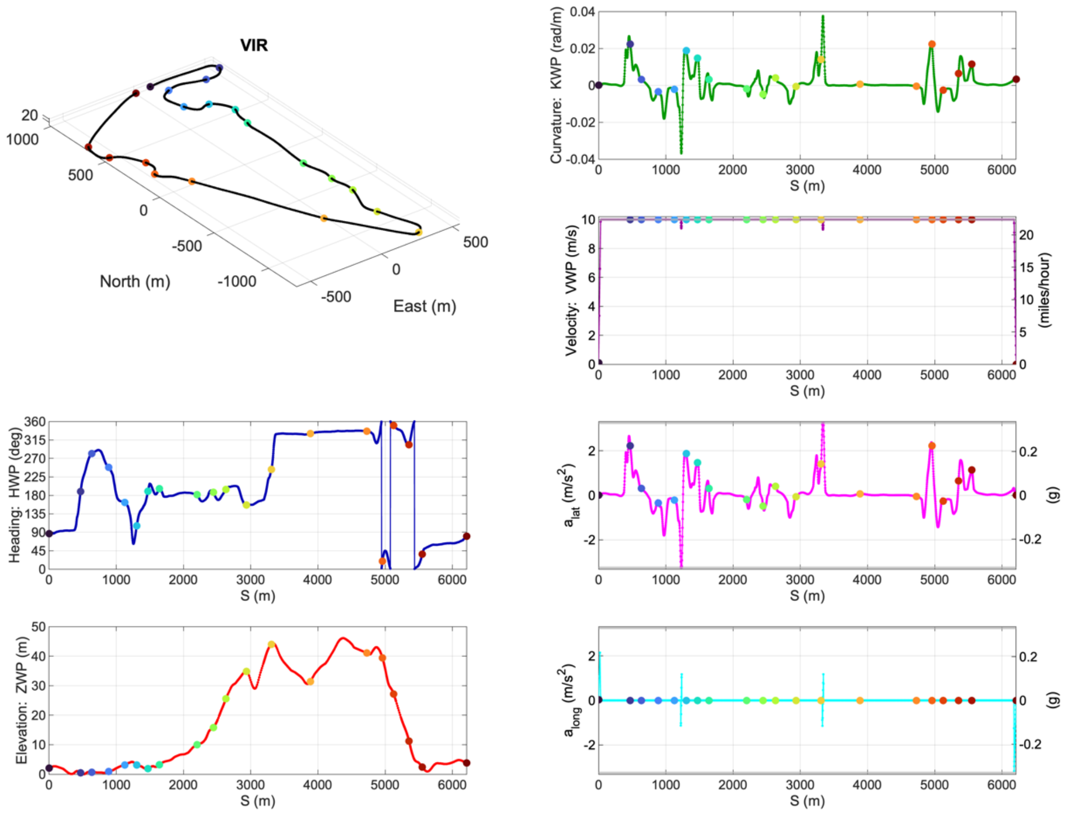
Smart Velocity Waypoint Generation



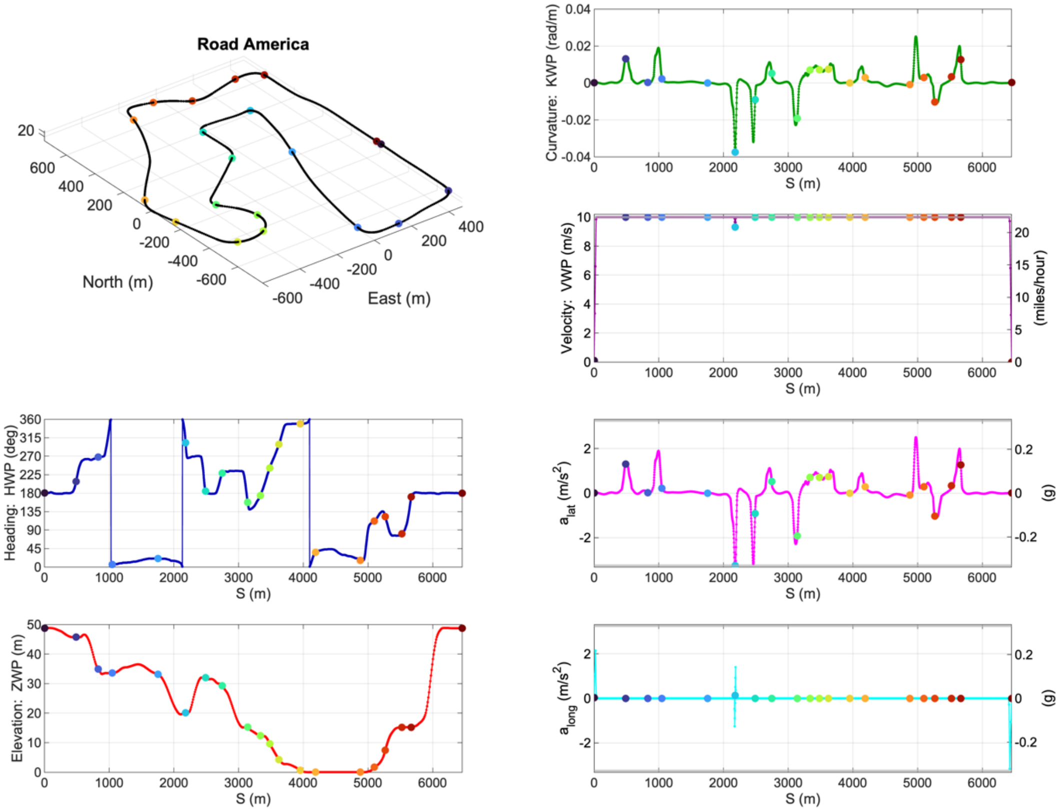
Smart Velocity Waypoint Generation



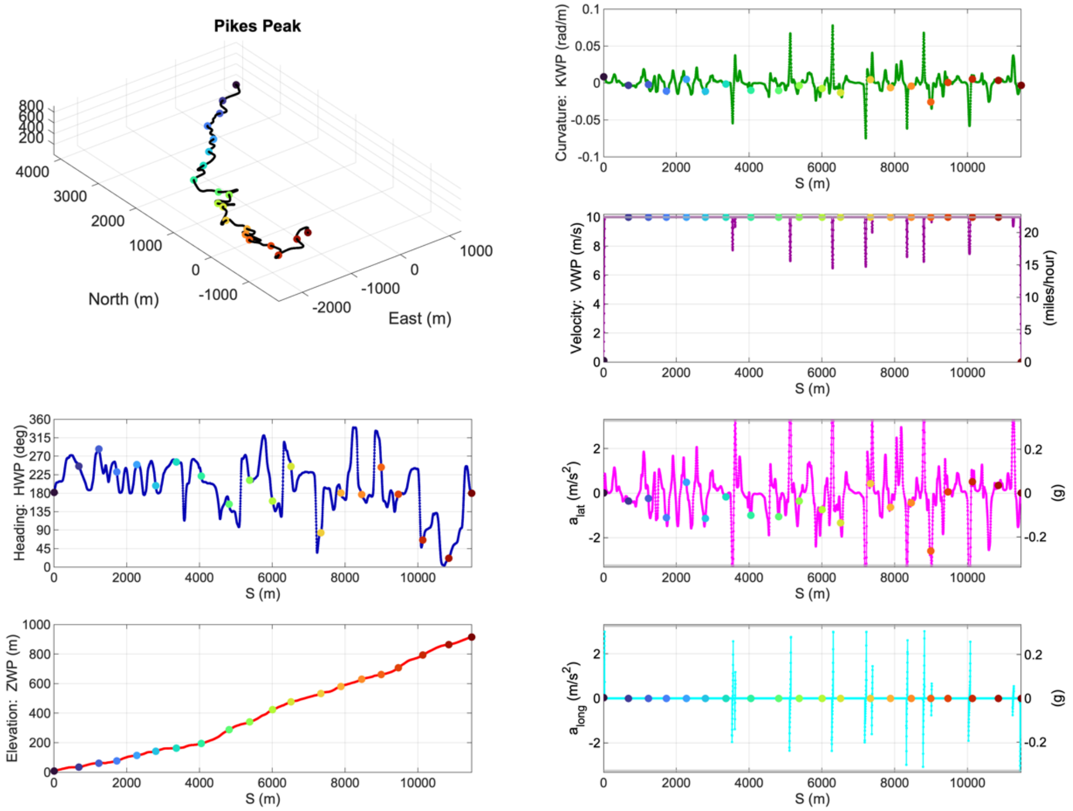
Smart Velocity Waypoint Generation



Smart Velocity Waypoint Generation



Smart Velocity Waypoint Generation


Pikes Peak
North (m)
East (m)
Curvature: KWP (rad/m)
Velocity: VWP (m/s)
(miles/hour)
Heading: HWP (deg)
a_{lat} (m/s^2)
(g)
Elevation: ZWP (m)
a_{long} (m/s^2)
(g)
S (m)