Localization Tools for Benchmarking ADAS Control Systems

M. R. Hafner, K. So Zhao, A. Hsia and Z. Rachlin
Research and Advanced Engineering, Ford Motor Company. Dearborn, MI 48121.

Abstract—We consider the challenge of testing lane centering ADAS control systems using a differential global positioning system localization strategy. Rather than constructing test scenarios using lane line markers or driving test vehicles on public roads, we develop a testing framework that can be used in both simulation and on open test surfaces with no lane demarcation. The primary sensor we leverage is a differential global positioning system (DGPS), which is used to create a localized lane-offset state relative to a map generated offline. The map is generated either via clothoid path primitives, or from known trajectories. We provide simulation results using the vehicle dynamics simulator CarSim, and provide experimental results taken from a test track.

I. INTRODUCTION

The evolving capabilities of sensing technologies has enabled an increase in the sophistication of advanced driver assist systems (ADAS), which can involve control systems that actuate on the vehicle. Monocular cameras for detecting lane lines have been commercially available for some time now [2]. Examples of currently available driver assist features with vehicle actuation include lane keeping assist (LKA) using forward-looking cameras, adaptive cruise control (ACC) using forward-looking radars, trailer backup assist (TBA) using the rear-facing camera, and auto parallel parking using ultra-sonic sensors [1].

Central to the design of these features is a strong interdependency between the sensing system and the control systems. Often, the testing and development involves tuning and calibrating the control systems with development hardware/software. This presents a unique challenge to control system designers, in that they must validate their design and calibration with sensors-in-the-loop. Furthermore, having sensors function in anticipated production use cases involves constructing complex scenarios on test tracks (using lane lines, road camber and other test vehicles), which is logistically challenging, especially for research engineers.

Past research in the area of lane centering control on race tracks use polar polynomial road representations based on fuzzy radius of curvature membership variables [3]. More recent work utilizes third order polynomials to model a predicted course trajectory for developing driver assistance functions [7].

In this paper, we develop a lateral controller testing framework using a differential global positioning system (DGPS) available commercially. We start by developing a formal definition of the vehicle state and road, such that a lane-offset state can be formally constructed. We then create numeric representations for these definitions so as to represent the road and vehicle state in software. We calculate the lane-offset state with an algorithm that can be run on a microcontroller. Last, we verify the methods work in closed-loop by running a lane centering controller in both simulation and on a test vehicle at the test track.

The paper is organized as follows: in Section II we introduce notation and a formal definition of the lane-offset calculation problem; in Section III we introduce a solution with algorithmic implementation; in Section IV we demonstrate our solution in a simulation environment using CarSim to model the vehicle dynamics; in Section V we present experimental results from a test vehicle using a dSPACE MicroAutoBox II; and in Section VI we provide concluding remarks and future research directions.

II. PROBLEM STATEMENT

In this section, we introduce the lane centering problem formally. This starts with notation used throughout this document, then a formal definition of the road and the vehicle state, and a formal definition for the vehicle lane-offset state.

A. Mathematical Notation

We denote the real numbers $\mathbb{R}$, positive real number $\mathbb{R}_+$, natural numbers $\mathbb{N}$ and the n-dimensional vector space $\mathbb{R}^n$. For a vector $x \in \mathbb{R}^n$ we denote the $i^{th}$ element $x_i \in \mathbb{R}_n$. We denote the $n$ times differentiable functions from $A \subset \mathbb{R}^m$ to $B \subset \mathbb{R}^l$ as $C^n(A, B)$, and the piecewise smooth functions $PC(A, B)$. For a set $A$, denote the power set as $P(A)$. For elements $x, y \in \mathbb{R}^n$, denote the distance between them as $||x - y||$, assumed to be the standard Euclidean 2-norm. For element $x \in \mathbb{R}^n$ and set $A \subset \mathbb{R}^n$, denote the distance between the set and element $||x||_A := \inf_{y \in A} ||x - y||$. For the angle $\theta \in [0, 2\pi)$ and vector $x \in \mathbb{R}^2$, denote the canonical rotation matrix $rot(\theta)$ where $rot(\theta)x$ is a counterclockwise rotation of $x$ by $\theta$. We say the curve $\gamma : \mathbb{R} \rightarrow \mathbb{R}^2$ is simple if it is injective, closed if it loops back to itself, and natural if the tangent vector $T(s) := \frac{d}{ds} \gamma(s)$ has unit magnitude $||T(s)|| = 1$ for all $s \in \mathbb{R}$.

B. Vehicle State Definition

We formally define the vehicle state within a planar coordinate system, and thus, omit all higher order dynamics of the vehicle. Define the vehicle state as $\nu(t) \in \mathcal{V} := \mathbb{R}^2 \times [0, 2\pi)$, where $\nu_1(t)$ corresponds to position along easting, $\nu_2(t)$ corresponds to position along northing, and $\nu_3(t)$ corresponds to heading.

The vehicle state can be viewed in Figure 1 within $\nu_1 \times \nu_2$. In practice, vehicle localization information is generated via DGPS, which positions the vehicle within a geodetic coordinate system specified by WGS 1984 [4]. We denote the
received DGPS information as \([l_a(t), l_o(t), h(t)] \in \text{gps} := [-90, 90] \times [-180, 180] \times [-180, 180]\), where the heading is clockwise relative to direct north. The geodetic state representation is not compatible with the planar coordinate system defined by \(V\), thus, we must locally convert the DGPS information to \(V\) via either the local tangent plane method, or the Universal Transverse Mercator (UTM) coordinate system, which is a conformal projection. We remark that the location of the vehicle is assumed to be fixed on the camera, located at the rearview mirror of a vehicle.

For a time interval \(T \subset \mathbb{R}_+\), we denote a vehicle trace as the trajectory
\[
\nu_{1,2}(T) = \bigcup_{t \in T} \nu_{1,2}(t) \subset V_1 \times V_2
\]
The vehicle trace will be useful for constructing road abstractions using previously traveled vehicle trajectories.

To construct the lane-offset state in Section II-D, we construct a set defined by the orthogonal span relative to the vehicle state \(\nu(t) \in V\), which we call the perpendicular projection \(\text{Per}_{\nu(t)} \subset \mathbb{R}^2\) as
\[
\text{Per}_{\nu(t)} := \nu_{1,2}(t) + \text{span} \left\{ \text{rot}(\nu_3(t)) \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\}.
\]
This set can be visualized in Figure 1. We remark this set will be useful for defining what we consider the closest road point. We next define what we mean by a road.

C. Road Definition

Without loss of generality, we define a road as a curve within the plane. We allow the curve to be closed, but do not require this condition.

Definition 1. Let a road be a simple natural curve \(\gamma \in \mathcal{R} := C^1(\mathbb{R}, \mathbb{R}^2) \cap PC(\mathbb{R}, \mathbb{R}^2)\) with arbitrary origin \(\gamma(s_0) = 0\), with the following attributes

1) The heading \(\psi(s)\) of the road at \(s \in \mathbb{R}\) is defined as the angle
\[
\psi(s) := \tan^{-1}\left( \frac{T_2(s)}{T_1(s)} \right);
\]
2) The curvature of the road at \(s \in \mathbb{R}\) is the rate of change of the tangent vector
\[
\kappa(s) := \left\| \frac{d}{ds} T(s) \right\|;
\]
3) The curvature rate of the road at \(s \in \mathbb{R}\) is given by
\[
\dot{\kappa}(s) := \frac{d}{ds} \kappa(s).
\]

Intuitively, these road features are all local properties of the road at the point \(s \in \mathbb{R}\). A road segment can be constructed by simply considering the image of an arc segment \(S = [s_0, s_f] \subset \mathbb{R}\) as \(\bigcup_{s \in S} \gamma(s)\).

We have defined the road constructively as an object that exists a priori, however, we can construct a road a posteriori by fitting the road \(\gamma \in \mathcal{R}\) to a vehicle trace \(\nu_{1,2}(T)\) using numerical tools. For constructive approaches, we will often be interested in constructing roads via curvature parameterization, that is, we assume knowledge of the road curvature \(\kappa(s)\) beforehand. We discuss the method of constructing a road \(\gamma(s)\) when using parametric clothoid curvature primitives in Section III-A1.

D. Lane-Offset State Definition

In this section we define a lane-offset state, consistent with other representations seen in the literature [7]. In particular, we attempt to compute a 3rd order polynomial within a coordinate system originated at the vehicles camera, seen in Figure 1. This emulates the signals produced by commercially available lane centering monocular camera systems [2].

We utilize the orthogonal projected state (2) to determine where the closest road point exists relative to the vehicle state. Define the orthogonal road points \(\text{ORP}_{\nu(t)} \subset \mathbb{R}^2\) as
\[
\text{ORP}_{\nu(t)} := \left\{ s \in \mathbb{R} \mid \gamma(s) \cap \text{Per}_{\nu(t)} \neq \emptyset \right\},
\]
which intuitively corresponds to the road points that are orthogonal to the vehicle state. We remark this set may be empty. The lane-offset state will become non-existent if \(\text{ORP}_{\nu(t)} = \emptyset\). Define the closest orthogonal road point \(s^* \in \mathbb{R}\) as
\[
s^*(t) := \arg\min_{s \in \text{ORP}_{\nu(t)}} ||\gamma(s) - \nu_{1,2}(t)||.
\]

We next define the lane-offset state \(x(t) \in \mathcal{X} := \mathbb{R}^4\) between the vehicle state \(\nu(t) \in V\) and the road \(\gamma \in \mathcal{R}\), which can be used by a lateral controller to steer the vehicle along the lane center. The fundamental ingredient is the closest orthogonal element \(s^*(t)\) defined above.

Definition 2. For a vehicle state \(\nu(t) \in V\), road \(\gamma \in \mathcal{R}\) and non-empty orthogonal projected states \(\text{ORP}_{\nu(t)}\), define the lane-offset state as the vector \(x(t) \in \mathcal{X} := \mathbb{R}^4\) with components

1) The lateral offset \(x_1(t) \in \mathcal{X}_1\) is given by \(x_1(t) := ||\nu_{1,2}(t) - \gamma(s^*(t))||\);
2) The angular offset \( x_2(t) \in X_2 \) is given by \( x_2(t) := v_2(t) - \psi(t) \).
3) The road curvature \( x_3(t) \in X_3 \) is given by the road curvature at the closest point \( x_3(t) := \kappa(s^*(t)) \).
4) The road curvature rate \( x_4(t) \in X_4 \) is given by the road curvature rate at the closest point \( x_4(t) := \dot{\kappa}(s^*(t)) \).

We provide an example of the lane-offset state \( x(t) \) in Figure 1 along with the vehicle state \( \nu(t) \) and road \( \gamma \in \mathcal{R} \).

**E. Problem Statement**

The elements of each map point \( p_i \in \mathcal{E} \) are: location \( \langle p_{i,1}, p_{i,2} \rangle := (x, y) \in \mathbb{R}^2 \); heading \( p_{i,3} := \psi \in [0,2\pi) \); curvature \( p_{i,4} := \kappa \in \mathbb{R} \); curvature rate \( p_{i,5} := \dot{\kappa} \in \mathbb{R} \); arc length from the initial point \( p_{i,6} := s \in \mathbb{R} \) (with the monotonicity condition \( p_{i,6} \leq p_{i+1,6} \)); and index \( p_{i,7} := k \in \mathbb{N} \) (the purpose of this index will become clear in Section III-A3). For two enhanced maps \( \mathcal{E}^a, \mathcal{E}^b \subset \mathcal{M} \), with cardinality \( |\mathcal{E}^a| = n \) and \( |\mathcal{E}^b| = m \), the concatenation \( \mathcal{E} = \mathcal{E}^a \mathcal{E}^b = \{p^a_1, \ldots, p^n_6, \ldots, p^b_{1}, \ldots, p^b_{m}\} \) has cardinality \( |\mathcal{E}| = n+m \) with trivial definition, except for

\[
p_{i,6} = \begin{cases} p^n_{i,6} & \text{if } i \leq n, \\ p^b_{i-n,6} + p^n_{i,6} & \text{if } i > n. \end{cases}
\]

This exception implies that the arc length of a point \( p_{i,6} \) that originated from the second eMap \( \mathcal{E}^b \) has an arc length equal to the original plus the total arc length of the first eMap. We remark that finite cardinality of an enhanced map \( |\mathcal{E}| = n \) implies that the map will intrinsically require a distance resolution \( ds \), which will be consistent with \( p_{i,6} = s = i \times ds \), that is, the arc length at point \( p_i \) will be equal to the index times the distance resolution.

We provide two separate approaches to constructing enhanced maps. The first method is constructive using a curvature parameterization \( \kappa(s) \), utilizing what we call path primitive maneuvers, the second involves filtering experimental traces \( \nu_1,2(T) \). We can visualize the eMap in Figure 3 as the black dots, where the difference between the sparse and enhanced maps will be discussed in Section III-A3 and their use in calculating the lane-offset state \( x(t) \in \mathcal{X} \) in Section III-D. We next define the various ways an enhanced map can be constructed.

1) Path Primitive Construction: We introduce the path primitive, which can be a straight line, clothoid arc, or constant circular arc [9]. Formally, we define a path primitive with the function \( \rho : [0,l] \times \mathbb{R} \to \mathbb{R}^2 \) where \( \rho(s, \kappa_0, \kappa_f) \) takes arc length \( s \in [0,l] \), initial curvature \( \kappa_0 \), and final curvature \( \kappa_f \). We remark that if \( \kappa_0 = \kappa_f \), then the image \( \rho([0,l], \kappa_0, \kappa_0) \) creates a circular arc for non-zero curvatures, and a straight line for zero curvature. For non-equal initial and final curvatures, the image \( \rho([0,l], \kappa_0, \kappa_f) \) represents a clothoid [6], which linearly interpolates the curvature \( \kappa_\rho(s) \) between \( \kappa_0 \) and \( \kappa_f \).
Integrating vector representation of curvature (4) as follows

Fig. 6. Path primitive concatenation between a straight line segment of length $l_1$, a clothoid of length $l_2$ and terminal curvature $\kappa^c_2$, and a constant circular arc of curvature $\kappa^c_3$.

With some abuse of notation, define the concatenation operator on a pair of path primitives as $\rho([0,l],\kappa_0,\kappa_f) = \rho^1([0,l^1],\kappa_0^1,\kappa_f^1) \rightarrow \rho^2([0,l^2],\kappa_0^2,\kappa_f^2)$ as simply a patched together pair of path primitives, where the start of the second primitive occurs at $s = l_1$. We remark that the curvature $\kappa_p(s)$ no longer linearly interpolates the initial $\kappa_0$ and final curvature $\kappa_f$. Therefore, the concatenated path is no longer a path primitive itself, since it is not simply a clothoid, arc or straight line, but rather the concatenation of these objects. We provide a visual depiction in Figure 6.

We next construct an eMap $E \subset M$ from the path primitive. Without loss of generality, for a path primitive $\rho([0,l],\kappa_0,\kappa_f)$, we discretize the arc length space $[0,l]$ by length $ds$, and set each element of the eMap $p_i \in E$ to $p_i = \left[p_i(1,2), p_i(3,4), p_i(5,6), p_i(7,8)\right] = \left[\rho(s,\kappa,\kappa_f), \psi(s), \kappa(s), \frac{\dot{\kappa}(s)}{\kappa}, s_i, \tilde{\kappa}\right]$, where $\psi(s)$ represents the heading of the path primitive at $s \in [0,l]$, which can be found by taking the appropriately evaluated arc tangent of the tangent vector $T(s)$.

2) Experiment Trace Construction: For experimental trajectories, we construct an enhanced map by first fitting the trajectory with cubic smoothing spline, using the matlab function `csaps` parameterized by the fit parameter $p \in [0,1]$, which minimizes the cost functional

$$J = \sum_{n=1}^{n} (Y_i - \mu(x_i))^2 + (1 - p) \int_{x_1}^{x^n} \lambda(t) \rho''(x)^2 \, dx,$$

for data of the form $\{x_i, y_i\}$ [5]. We re-parameterize the data by distance rather than time, giving us $\tilde{x}(s)$ and $\tilde{y}(s)$. We take derivatives of the smoothing spline to find the heading and the curvature of the path at each point $s = i \ast ds$. The heading $\psi(s)$ of the trajectory at arc length $s \geq 0$ is found with $\psi(s) = \tan^{-1}\left(\frac{\tilde{y}(s)}{\tilde{x}(s)}\right)$ and the curvature $\kappa(s)$ with

$$\kappa(s) = \frac{\tilde{x}'(s)\tilde{y}''(s) - \tilde{y}'(s)\tilde{x}''(s)}{(\tilde{x}'(s)^2 + \tilde{y}'(s)^2)^{3/2}}.$$

details of the curvature calculation can be found here [8].

Thus, the enhanced map can be constructed with the elements $p_i \in E$ as $p_i = \left[p_i(1,2), p_i(3,4), p_i(5,6), p_i(7,8)\right]$.

3) Reduction of an Enhanced Map: We next construct an operator that reduces the cardinality of an eMap $E$, formally the reduction operator $R: P(M) \times N \rightarrow P(M)$. This is needed for two reasons, the first being that the map size of every enhanced map $|E| = n$ should be the same regardless of path primitive generated or experimental trace generated. The second reason is the map localizing search algorithm involves two stage search: (a) the initializing search relies on a sparse enhanced map, which we call a sparse enhanced map $E^S \subset M$; (b) the primary search requires a high resolution enhanced map, which we call a dense enhanced map $E \subset M$. For a visual example of the dense and sparse eMap, please see Figure 3.

The reduction operator is defined as follows, for the eMap $E \subset M$ with cardinality $|E| = n$ and reduction index $m < n$, we define the image $E^R = R(E, m)$ as follows: we first set the reduction index $\nu = \text{round}\left(\frac{m}{n}\right) \in \mathbb{N}$, which informally corresponds to the magnitude of size reduction; we then set...
each element \( p^R_j \in \mathcal{E}^R \) equal to the sampled element \( p_j \in \mathcal{E} \) where \( j = \nu * i \) as \( p^R_{k,(1,...,6)} = p_j_{(1,...,6)} \), except for the subtle distinction \( p^R_{k,(1,...,6)} = j \). This implies that every element in the reduced eMap \( \mathcal{E}^R \) has an associated parent element from the enhanced map \( \mathcal{E} \), where now the spacing \( dS^R \) between each element is now enlarged to \( dS^R = vdS \).

B. State Construction in a Localized Coordinate System

We next develop a localized coordinate system to allow for testing on open test surfaces. We assume the availability of DGPS information \([l(t), l(t), h(t)] \in \text{gps}\). Without loss of generality, we assume the availability of a conformal projection into a local rectilinear coordinate system as \([P_E(t), P_N(t), \phi(t)] \in \mathcal{V} \), examples consist of UTM or local tangent plane.

We look to construct what we call the local coordinate vehicle vector \( \nu(t) \in \mathcal{V} \), which is parametrized via the initializing vector \( I = [I_x, I_y, I_\theta] \in \mathcal{V} \). The initializing vector is used to rotate and translate the vector \([P_E(t), P_N(t), \phi(t)]\) via the following affine transformation

\[
\begin{bmatrix}
   \nu_1(t) \\
   \nu_2(t)
\end{bmatrix}
= \begin{bmatrix}
   \cos(I_\theta) & \sin(I_\theta) \\
   -\sin(I_\theta) & \cos(I_\theta)
\end{bmatrix}
\begin{bmatrix}
   P_E(t) \\
   P_N(t)
\end{bmatrix}
\begin{bmatrix}
   1 \\
   0
\end{bmatrix},
\]

with heading \( \nu_2(t) := \phi(t) - I_\theta \).

In practice, we set the initializing vector \( I \in \mathcal{V} \) at a location on the test surface where we would like to start our test maneuver. If using a vehicle trace \( \nu_{1,2}(T) \subset \mathcal{V} \) to construct the eMap \( \mathcal{E} \), the initializing vector \( I \) must be consistent.

C. Closest Orthogonal Map Point

![Fig. 7. State machine for the search algorithm to find the closest orthogonal dense map point \( p^{*}_{\nu}(t) \in \mathcal{E} \). The state machine is initialized in the "Search Sparse Map" state, where it goes through all points \( p^S \in \mathcal{E}^S \) and returns the closest point \( p^S_{\nu}(t) \) if the distance between the closest point and the vehicle state is less than \( T_S \), we then transition into the "Search Dense eMap" state, where we search the dense eMap locally to find \( p^S_{\nu}(t) \). If the distance between the closest point and vehicle state is greater than \( T_D \). With the local state \( \nu(t) \in \mathcal{V} \) calculated, we now look for an algorithm to find the closest orthogonal map point \( p^{*}_{\nu}(t) \in \mathcal{E} \). The search algorithm is constructed with two main states (shown in Figure 7), where the first state searches the entire sparse map, and upon returning a close sparse point, the second state searches the dense map locally near that point. We introduce these modes next.

The first mode runs the function \( \text{searchSparseMap} : P(\mathcal{M}) \times \mathcal{V} \to \mathbb{N} \), denoted as \( i^*(t) = \text{searchSparseMap}(\mathcal{E}^S, \nu(t)) \), which is defined as the solution of the following optimization problem \( i^*(t) := \arg\min_{k \leq m} ||\nu(t) - p^S_{k,1,2}(t)|| \). In software, we solve this optimization problem with a for loop to determine the minimizing index (since no convexity properties hold). If the vehicle state and the closest map point are sufficiently close, given by \( ||p^S_{k,1,2}(t) - \nu_{1,2}(t)|| \leq T_S \), where \( T_S > 0 \) is a configurable distance, then we say the vehicle is close to the sparse map. By design, the index \( i^*(t) := p^S_{k,1,2}(t) \) corresponds to the element \( p_{k,1,2} \in \mathcal{E} \), that is \( p_{k,1,2} = p^S_{k,1,2} \).

Once a close sparse point has been identified, we switch to the second mode, which searches the dense map \( \mathcal{E} \subset \mathcal{M} \) for the closest orthogonal point \( p^{*}_{\nu}(t) \in \mathcal{E} \). The second mode runs the function \( \text{searchDenseMap} : P(\mathcal{M}) \times \mathcal{V} \times \mathbb{N} \to \mathbb{N} \), denoted as

\[
k^*(t) = \text{searchDenseMap}(\mathcal{E}, ORP_{\nu}(t), k^*(t^-), k_e),
\]

which is defined as the solution to the following optimization problem

\[
k^*(t) := \arg\min_{k \in \{k(t^-) \}} ||p_k \cdot 1.2|| ORP_{\nu}(t), \quad (9)
\]

where

\[
\epsilon(k^*(t^-), k_e) := \{k \in \mathbb{N} \mid |k - k^*(t^-)| \leq \epsilon\}
\]

is an epsilon neighborhood around the previous closest index \( k^*(t^-) \). For this optimization problem, we are locally searching around the previous iteration closest point \( k^*(t^-) \) for the path point \( p_{k^*(t^-),1,2} \) that is closest to the orthogonal projection \( ORP_{\nu}(t) \). In software, we implement this algorithm with a for loop. We depict a visualization of the search problem in Figure 3.

We leave the search dense map state if the lane-offset lateral offset \( x_1(t) \) becomes greater than \( T^D \) or the heading offset \( x_2(t) \) becomes greater than \( T^D \). This implies that the vehicle has either strayed sufficiently from the path laterally, or the heading sufficiently away from the path.

D. Constructing the Lane Offset State

Once we know the closest orthogonal point \( p^{*}_{\nu}(t) \in \mathcal{E} \), we can construct the lane-offset state introduced in Section II-D. We compute the lane-offset state \( x(t) \in \mathcal{X} \) with the vehicle state \( \nu(t) \in \mathcal{V} \) and closest point \( p^{*}_{\nu}(t) \in \mathcal{E} \) as

\[
x_1(t) = ||p^{*}_{\nu}(t,1,2) - \nu_{1,2}(t)||, \quad x_2(t) = p^{*}_{\nu}(t,1,2) - \nu_{1,2}(t),
\]

and \( x_1(t) = p^{*}_{\nu}(t,1,2) - \nu_{1,2}(t) \). This can be visualized in Figure 4.

This lane-offset state is provided to a lateral controller to steer the vehicle along the enhanced map. We show a block diagram of how the lane-offset state can be used to test a lateral controller in Figure 2.

IV. SIMULATION

We provide simulation results of the vehicle following the lane-offset state in the vehicle dynamics simulator CarSim. The goal of this simulation is to follow a road section designed within the CarSim road builder. Therefore, the enhanced map creation technique we consider is from a vehicle trace \( \nu_{1,2}(T) \), as described in Section III-A2.

The simulated vehicle is a D-Class passenger sedan, with a rack force input to the steering system. The simulation provides localization for the vehicles center of mass, therefore, we do not implement a local coordinate transformation and instead use the global coordinates supplied by CarSim. This is aligned
with the goal of driving in the center of a defined road rather than a path primitive on an open test course.

![Simulation road](image)

**Fig. 8. Simulation road** $\gamma \in \mathbb{R}$ is shown in green, consisting of: (a) a straight section of 20 m; (b) a 3km clothoid from straight to a 1000 m radius of curvature; (c) a 4.3km constant circular arc with 1000 m radius of curvature. The vehicle trajectory is shown in dashed blue. The upper plot depicts the entire road and trajectory while the bottom plot displays a small section to highlight initial path deviation due non-zero initial condition $x(0)$.

The simulation results are shown as a trajectory in Figure 8, the vehicle is able to stay in lane center even under the presence of the changing road camber.

V. EXPERIMENTAL RESULTS

In this section, we conduct a lane centering experiment utilizing an enhanced map based on an example path primitive (see Section III-A1). The test vehicle is equipped with an OxTS RT3000 DGPS sensing system, the lane-offset state algorithms are implemented in MATLAB Simulink, and run on a dSPACE MicroAutoBox II. The vehicle has a development electronic steering interface to allow torque input to the rack assist motor, allowing the vehicle to be steered by a lateral controller also implemented in the MicroAutoBox II.

We use an enhanced map generated via the path primitive shown in Figure 9. This experiment is conducted on a closed-course test track, with a vehicle velocity of 18 m/s. We use the local tangent plane method to compute the vehicle state. The local coordinate vector is then created using the initializer $I = [214.2, -1228.8, -2.63]$ located at a safe position on the test track.

We provide a snapshot at $t = 10$ in Figure 9 where we also overlay the lane-offset state $x(10)$. Note that the lane-offset state exactly fits the path primitive in the future, since the curvature of the path remains constant, whereas behind the vehicle it does not align with the path, due to the changing curvature prior on the path.

VI. CONCLUSION

In this paper we formally introduced a model of the road, a model for the vehicle state and showed how a path relative state could be constructed for testing ADAS lane centering features. We introduced a specific algorithmic implementation utilizing enhanced maps which can be run on an embedded microcontroller. These algorithms were validated in simulation using the vehicle dynamics simulator CarSim with Simulink, and in experiment on a vehicle using a dSPACE MicroAutoBox II and DGPS.

Future work involves investigating how the camera-based lane center signal could be correlated to the model introduced in this work. We would also like to model virtual vehicles to test ACC use cases that are difficult to exercise in the real-world.

REFERENCES