Trajectory Generation for an On-Road Autonomous Vehicle

John Horst
The National Institute of
Standards and Technology (NIST)
Telephone: 301.975.3430 Email: john.horst@nist.gov

Anthony Barbera
The National Institute of
Standards and Technology (NIST)
Telephone: 301.975.3460 Email: tony.barbera@nist.gov

***Abstract*— We describe an algorithm that generates a smooth trajectory (position, velocity, and acceleration at uniformly sampled instants of time) for a car-like vehicle autonomously navigating within the constraints of lanes in a road. The technique models both vehicle paths and lane segments as straight line segments and circular arcs for mathematical simplicity and elegance, which we contrast with cubic spline approaches. We develop the path in an idealized space, warp the path into real space and compute path length, generate a one-dimensional trajectory along the path length that achieves target speeds and positions, and finally, warp, translate, and rotate the one-dimensional trajectory points onto the path in real space. The algorithm moves a vehicle in lane safely and efficiently within speed and acceleration maximums. The algorithm functions in the context of other autonomous driving functions within a carefully designed vehicle control hierarchy.**

I. Introduction

Automatic control of an on-road vehicle is a remarkably complex task. One must deal with complex intersections, widely varying road surface conditions, radically changing weather conditions, objects moving unpredictably, widely fluctuating types and sizes of objects, dramatically different road types, severely changing lighting conditions, missing or confusing lane markings, etc. In order to manage this complexity, we have defined a control hierarchy with sensible assignment of responsibility for each level in the hierarchy [1]. A carefully designed hierarchy allows design to proceed using a divide-and-conquer strategy that ensures successful component integration. The scope and nature of the vehicle trajectory generator, that is the subject of this work, is heavily dependent on the responsibilities assigned to each level in the hierarchical control system that the trajectory generator supports.

We describe a vehicle trajectory generator that is able to maneuver an on-road autonomous vehicle safely and efficiently with dynamic constraints and in the presence of objects. We include moving objects (*e.g.*, other vehicles) as well as stationary objects (*e.g.*, potholes and parked vehicles). We require the vehicle trajectory generator to allow the vehicle to observe posted speed limits and operate within speed and acceleration (dynamic) maximums based on things such as road surface type and road surface conditions. Our general approach to path planning in the trajectory generator is to separate path generation from trajectory generation. We argue that this is roughly how humans navigate a vehicle in lane. We select a generally safe and efficient path in a lane-centric space ("lane-centric" means that distances are measured tangential and normal to the center of the lane, independent of lane curvature) and then we move along that path (warped into real space) with speed and acceleration values appropriate to a variety of factors, including the type of object the vehicle is maneuvering around, the motion of that object, the tilt in the road, the friction conditions of the tire on the road surface, the curvature of the path, and any posted speed limit.

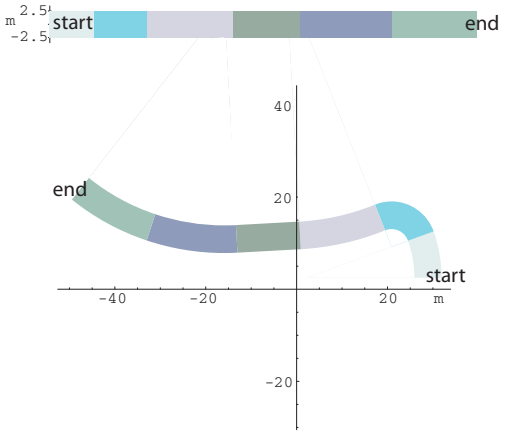

Fig. 1. An example sequence of (constant curvature arc and straight line) lane segments, which have continuous, non-differentiable tangents at the boundaries between adjacent lane segments. Such a sequence is being used to model the path of any lane on any road. The representation on top in this figure is the lane segment sequence in lane space. The bottom representation in this figure is the same sequence in real space.

This simple path modelling approach is chosen, in part, because it turns path planning into simple closed form geometric computations, with no on-line search required. The problem of in-lane vehicle maneuvering around objects and maneuvering on various road surface types and in various weather conditions has been addressed [2].

Splines are commonly used for defining smooth paths for robots [3] [4]. In order to specify robot motion along that path, we must be able to determine equally spaced points on that path, and therefore, we must have an arc length parameterization of that curve. For example, cubic splines do not yield a symbolic expression for arc length and the math required to approximate the arc length parameterization is complex, as well as there being serious real-time computational issues [4]. We offer an approach involving paths consisting of line segments, circular arcs, radially warped line segments, and radially warped circular arcs. Each of these path types yields a symbolic expression for path length. However, for radially warped circular arcs, the

expression is so complex, we also employ an approximated arc length parameterization.

The control hierarchy is designed with a sampling rate fast enough to replan trajectories every 0.1 seconds or less, so that the vehicle is able to respond quickly to a sometimes dramatically and quickly changing environment. Therefore, we only assume simple models of both vehicle motion and predicted object motion. For vehicle motion we assume motion in straight lines, constant curvature arcs, or a radial warp of either lines or arcs. For predicted object motion, we assume either constant velocity or constant acceleration for either linear or circular motion, since we will most likely expect to be replanning the vehicle path for both object motions changes and other reasons.

II. Lane modelling

Our model of the road is chosen to be a two-dimensional representation of the road as sequences of constant curvature arc or line segment lane segments with continuous, non-differentiable tangents at lane segment boundaries. The sequence of lane segments at the bottom of Figure 1 illustrates this road model. Of course, two-dimensional modelling implies that tangential or normal tilt of the road surface is not explicitly represented in the road model. In our control system, tilt of the road surface will affect the choice of target speeds along the curve. Road surface tilt will also affect the choice of a particular maneuver among a set of candidate maneuvers around objects, due to the change in acceleration maximums. Figure 3 shows that these vehicle dynamics maximums are input to the Maneuver Generator module. Road surface tilt may also affect the nature of the planned trajectory. However, tilt will not generally affect the chosen vehicle path, once a maneuver is selected.

III. Path and trajectory modelling

In order to complement the relatively simple model of the road as a sequence of smooth line segments and constant curvature arcs, and to keep path planning mathematically simple, we have chosen to model the path of the vehicle as line segments and constant curvature arcs as well. However, in keeping with our intuition about lane-centric driving, the path model is not in real space, but in a transformation of real space we call "lane space," *i.e.*, in which lane segments are all in a straight line. The transformation of lane segments from real space to lane space is simple: The length of each lane segment in lane space is equal to the length of the curve running through the center of the lane in its companion lane segment in real space as illustrated in Figure 1. Let L_i be the cumulative length of the path along the center of the last i lane segments. The transform makes the path length through the middle of each lane segment to be the length of the new lane segment in lane space. The beginning left-most point in the center of the lane of the i^{th} lane segment in lane space is the point, $(L_{i-1}, 0)$.

To plan paths in lane space is simpler mathematically than planning in real space. Real space has discontinuous curvature, for example, which also makes it harder to keep the planned path within lane boundaries. After planning in lane space, we can warp the path onto the real space of

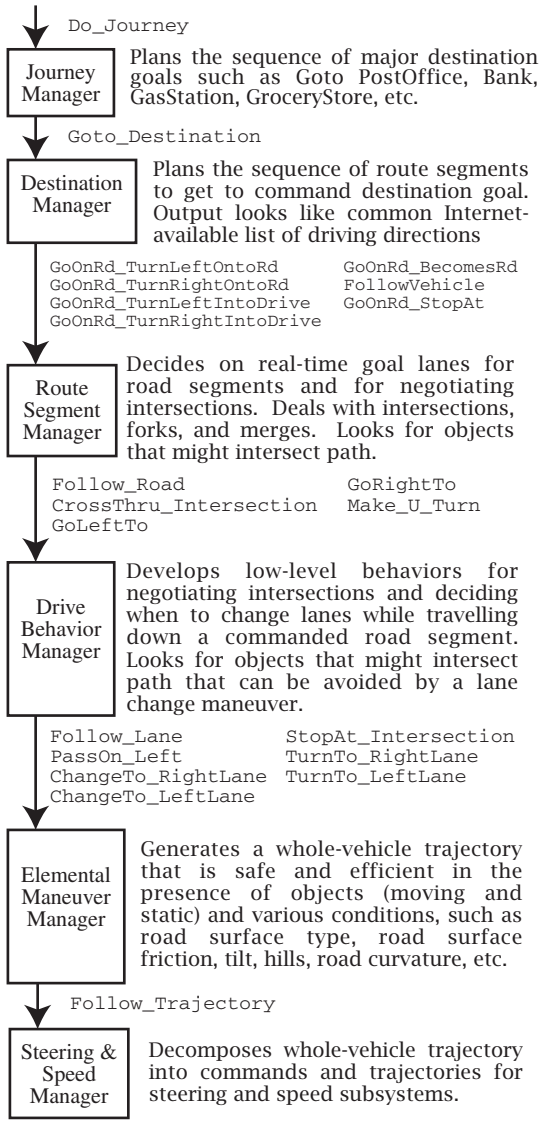

Fig. 2. Hierarchical control nodes for on-road vehicle motion control with commands and responsibilities for each level in the hierarchy

lane segments. The warping complicates the math, but only slightly, and that only when there is an arc path radially warped to reside in an arc lane segment. Such a warping does not yield a simple closed form for the warped path parameterized by path length, requiring multiple on-line numerical integrations to obtain path length.

We generate the full path first in lane space, compute the true path length of that path in real space, compute a one-dimensional trajectory along that path, and warp the trajectory onto the path in real space. This works for planning a vehicle trajectory in the presence of both static and dynamic objects. With static objects, all the trajectory generator needs is a sequence of target positions and velocities in lane space. With dynamic objects, the trajectory generator simply needs the additional information of a time window

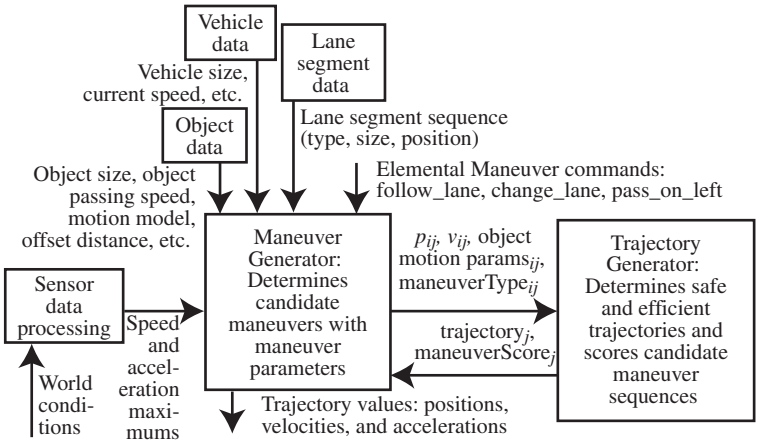

Fig. 3. The key activities of the Elemental Maneuver level in the autonomous vehicle control hierarchy. The Maneuver Generator produces a sequence of candidate positions, velocities, and time windows for the accomplishment of the j^{th} set of maneuvers. Time windows ($t \in [t_{min}, t_{max}]$) are relevant only for moving objects. Maneuver types are only relevant when the change in position and velocity are due to objects, whether dynamic or static. World conditions include road surface type, weather-related road surface conditions, road slope, road horizon distance, and visibility conditions.

when we are predicted to need to achieve that position.

IV. Elemental Maneuver level in the context of the total vehicle control hierarchy

Our particular road model (constant curvature arcs and lane segments) as well as aspects of the trajectory generator are chosen because we employ a hierarchical control approach. The approach allows us to have simpler control models at each hierarchical level with the assurance that local discontinuities at a higher levels will be smoothed out at a lower hierarchical levels [5].

The trajectory generator serves within the elemental maneuver manager (EMM) level of the control hierarchy as described in Figure 2. The EMM is responsible for performing tasks like Follow_Lane, ChangeTo_RightLane, and PassOn_Left. However, the EMM is not responsible for determining the type of object, the cost of collision with a particular object, determining objects of interest, determining whether conditions are good to pass, and other similar responsibilities summarized for the other control modules in Figure 2. On the other hand, the EMM is responsible to make sure that the vehicle accomplishes its assigned tasks safely and efficiently in the presence of objects and dynamic constraints, and furthermore, is responsible for generating those dynamic constraints from sensor input [2].

The EMM level is given a list of objects by the level above (Drive Behavior Manager) with their sizes, safe passing speeds, and safety set-off distances [2]. It may also be given certain target positions and velocities on the road ahead based on knowledge of things like posted speed limits or upcoming road conditions. When encountering an object or objects ahead, the EMM level must choose an appropriate maneuver around the object from a set of all possible maneuvers, including slow down, run over [1], straddle [2], circumvent left, and circumvent right.

Also, thinking hierarchically, we want to have simpler math models and control paradigms at each level for total system perspicuity, which is so critically important to system maintenance, expansion, and improvement.

V. Trajectory generation in the context of EMM level responsibilities

The EMM's task is to be able to successfully perform commands like follow_lane, change_lane, and pass_on_left within speed and acceleration constraints in the presence of objects. To successfully perform these commands, several distinct activities must occur in the EMM, namely, sensory processing, maneuver generation, and trajectory generation, as illustrated in Figure 3.

Sensing and sensor data processing are required to determine speed and acceleration constraints. Sensing and sensor data processing of objects (including motion predictions) as well as computation of necessary lane segment parameters (such as type, size, and position) are the responsibility of the next higher level.

The maneuver generation (MG) module has logic to perform any of the commands required of the EMM level. MG has the responsibility to propose candidate sequences of maneuvers to the trajectory generator (TG) module. Maneuver types include stop, straddle, run over, slow down, circumvent right, and circumvent left.

The TG module receives a set of proposed sequences of target positions, target velocities, ranges of times ($t \in [t_{min}, t_{max}]$) (reserved for moving objects), and the maneuver types. The output of TG to MG is advice on the relative

[1] the "run over" maneuver is to take no obstacle avoidance measures whatsoever.

[2] the "straddle" maneuver means that the vehicle passes the object with the object between left and right side wheels and under the vehicle underbody.

efficiency and safety or outright impossibility of the various proposed sequences of target positions, velocities, and time windows. The MG then outputs the best set of trajectory values for this particular instant in EMM's planning cycle. Planning cycle frequency is dependent on the computational cost of each set of plans. We expect these cycles to occur at frequencies well over 10 Hz, which should be adequate for real-time replanning.

VI. The EMM trajectory generation algorithm

Several options are possible for a design for a trajectory generator for a vehicle in-lane at the EMM hierarchical level. One is to plan both path and trajectory simultaneously in real space. Another is to plan the path in real space, compute a one-dimensional trajectory, and plot that trajectory along the path in real space. A third way, and the one we have chosen, is to plan a path in lane space, warp these paths into real space, compute a safe and efficient one-dimensional trajectory along the path, then warp the trajectory points from lane space to real space.

The path in lane space is simple mathematically: the paths are either straight lines or constant curvature arcs. The use of mathematically simple paths in lane space has the advantage that in most cases we can obtain simple symbolical expressions for the warping functions.

The positives to this approach are that the path is elegant, trajectory is simple, and warping is straightforward. All is smooth, safe, and generally efficient. The negatives are that simplified path planning in lane space plus warping can exacerbate curvature, reducing motion efficiency, but not safety since we can always control the trajectory along the path. Also, numerical integration is required for computation of the path length function (7) for the arc path in an arc lane (also see Section VI-F).

All path planning in real space is challenging due to the fact that the lane segments have curvatures and lengths which can, of course, widely vary. It would be quite a challenge to plan a trajectory without explicit reference to the constraint of lanes, since the trajectory needs to stay in-lane at all points along the trajectory.

We believe that a logical model for trajectory planning is to perform the path planning in a space that is absolutely straight. One way to achieve such a space is to imagine the lane segments as a flexible, non-elastic rope. Then stretch the rope out straight, so that the length of the straightened lane segment is the length of the center line of the unstraightened lane segment. This we call, lane space. An example set of lane segments in real space and transformed into lane space is illustrated in Figure 1.

So, vehicle paths are arcs and straight lines in lane space. This means that steering angles will be sometimes discontinuous, whereas, if we used higher order curves, say a fourth order curves or clothoids, not so. The latter would gain us smooth turns of the steering wheel, but we would have to warp this curve into real space, which would yield substantially complex math. Our chosen method allows us to easily warp line segment paths to line segment lane segments, arc paths to line segment lane segments, or line segment paths to arc lane segments, which are all simple processes

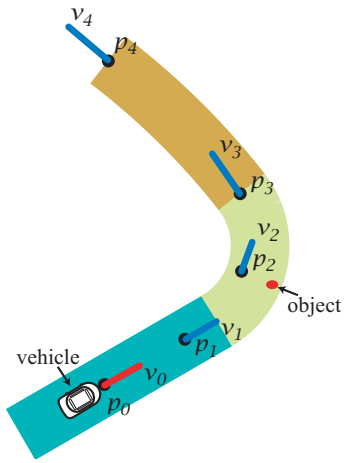

Fig. 4. An example sequence of lane segments with target positions and velocities that are sufficient to guide the generation of both vehicle path and dynamic trajectory.

mathematically. Only warping arc paths to arc lane segments is complex mathematically, requiring numerical integration to obtain an approximate parameterization of the curve by path length.

We need to assign a trajectory for the vehicle to travel along an arbitrary sequence of lane segments. A trajectory consists of target position, velocity, and acceleration points in two dimensions at uniformly spaced instants of time. If there are no objects in the lane, the vehicle will be assigned a safe and efficient trajectory that will require the vehicle to achieve each target speed at its target position. The planned vehicle path will copy the shape of the lane segment. Sometimes, the target speed prior to a lane segment boundary (illustrated in Figure 4) will be governed by amount of curvature in the subsequent lane segment. However, the nature of road surface and tire conditions, and posted speed limits will also govern these target speeds.

What happens when an object is in the lane and the vehicle must maneuver around it in lane to avoid collision? Such a situation is depicted in Figure 4. Since we are modelling roads as sequences of lane segments of either constant curvature arc or straight line segments, it would seem logical to model vehicle object avoidance maneuvering also in terms of constant curvature arcs and line segments. This is what we have done and is described in detail in later sections. Such a sequence of target positions and velocities is shown in Figure 4.

Here is a summary description of the entire trajectory generation algorithm that is the subject of this paper. We will not deal with the maneuvering issues that are upstream from the trajectory generator, which are planned for a subsequent paper.

1) Transform into lane space all current lane segments in vehicle purview
2) Compute or read all target positions, velocities, time windows, and maneuver types in lane space for the current set of lane segments, posted speed limits,

changes in road conditions, and objects of importance
3) Generate the vehicle path in lane space
4) Compute the total path length of the warped 2-dimensional path in lane space into real space
5) Compute a safe and efficient one dimensional trajectory over the path length just computed to achieve the warped target positions and speeds within the time window and within velocity and acceleration maximums, if possible
6) Compute the radius of curvature along the path and if, at any point along the curve, vehicle speed is higher than the radius of curvature and normal acceleration maximums will allow, reduce vehicle speed appropriately and recompute the one-dimensional trajectory
7) Superimpose the one dimensional trajectory on the warped path parameterized by path length and advise the maneuver generator module if any paths are actually impossible to achieve within time and acceleration constraints
8) Loop back to step number 2 for each new target position in the vehicle purview
9) Add to or revise the trajectory as new lane segments, objects, posted speed limits, etc. enter the vehicle's purview

A. Generating the vehicle path in lane space

Given the simplifying nature of the system hierarchy [1], at the hierarchical level of the trajectory generator [2], the problem of moving an autonomous vehicle safely along a road containing other objects (moving and stationary) can be reduced to a problem of going from one velocity at an initial location, to a final velocity at a final location. This means that the vehicle has some initial velocity v_0 at some initial point in two dimensional space, p_0. We want to compute a smooth path from p_0 at v_0 to p_f at velocity v_f[3]. As suggested in Figure 5, we hope to describe the entire path of the vehicle on-road in-lane as a set of smooth and simple paths, described by line segments or circular arcs in lane space, between a series of target positions and velocities.

Choosing circular arcs and line segments only for both lanes and paths allows us, in most cases, to easily compute a unit speed parameterization of the paths warped into real space (unit speed parameterizations are those that are parameterized by path length [6]). After we compute the total path, we can compute a one-dimensional trajectory that is intended to move the robot along the path within acceleration and speed constraints dictated by the curvature of the path and other factors (*e.g.*, road surface conditions, posted speed limit, condition of tires).

A disadvantage of choosing arcs and line segments for lanes and paths is that, in general, the direction of the velocity vector can change substantially at the intersection point of the two arcs. However, since this trajectory is meant to operate within a hierarchical control system, there will be sufficient smoothing of the steering at lower levels.

[3]The path solutions we derive also require that $v_0 \cdot v_f \geq 0$ in lane space, which is an easy constraint to satisfy.

Our path generation algorithm at the highest level is in three parts. First, if the vector between current and target position as well as the two velocity vectors are all collinear, the path (in lane space) is a straight line segment from p_0 to p_f. The second step requires we first form a vector, v, shown in Figure 7, that is the vector at p_f that is tangent to the circle that passes through both p_f and p_0 and is also tangent to v_0. Now if the cross product, $v_f \times v$ equals zero, the planned vehicle path is a single constant curvature arc. We call the sign of this cross product the discriminant. Third, if neither of the first two conditions are true, we form the vehicle path using two circular arcs of (generally) different curvature. This third step will now be detailed.

The path now consists of two arcs in lane space (each having constant curvature) such that the total path has continuous, but not differentiable, velocity. We want to compute the parameters of two circles that have the following constraints: 1) they intersect at one point, 2) they are disjoint (one is not enclosed within the other), 3) the initial circle is tangent to the input vector v_0 at p_0, 4) the final circle is tangent to the input vector v_f at p_0, and 5) the relative lengths of the two radii are scaled by the relative lengths of the input vectors, v_0 and v_f. The scale chosen for radii 1) ensures a single solution and 2) fulfills what is a reasonable requirement, namely, that the greater the speed the lesser the curvature (greater the radius).

If $\|v_0\| = 0$ or $\|v_f\| = 0$, we will fix the initial or final speed to be some non-zero minimum corresponding to the minimum turning radius of the vehicle. Therefore, $\|v_0\| \neq 0 \wedge \|v_f\| \neq 0$ and the algorithm described is governed by these new input values.

There are actually four possible solutions of two intersecting circles to each set of input conditions: 1) The initial tangent circle is on the left side of v_0 (left side while "looking down" v_0) at p_0 and the final circle is on the right side of v_f at p_f ("left/right" solution), 2) the initial tangent circle is on the right side of v_0 at p_0 and the final circle is on the left side of v_f at p_f ("right/left"), 3) the initial tangent circle is on the left side of v_0 at p_0 and the final circle is on the left side of v_f at p_f ("left/left"), or 4) the initial tangent circle is on the right side of v_0 at p_0 and the final circle is on the right side of v_f at p_f ("right/right"). It can be demonstrated that the two latter solutions (left/left and right/right), if such solutions exist, involve circles that are enclosed within one another.

If either of the latter two enclosed circle solutions exist, we find that a straight line (infinite radius for one circle) and a maximum radius circle for the other circle is a possible enclosed solution. In fact, for on-road driving, safe driving is better served by smaller curvatures and this solution (straight line and circle) has the smallest curvature of any reasonable enclosed circle solution. We need to select a left/right and right/left solution if the maximum radius is still either 1) too high a curvature for normal acceleration maximums or 2) less than the turning radius of the vehicle. Otherwise, we always select a legal one circle (plus line) solution for the sake of path efficiency. We will call these solutions one circle solutions. There are four of them: line/circle right, circle/line right, line/circle left, circle/line

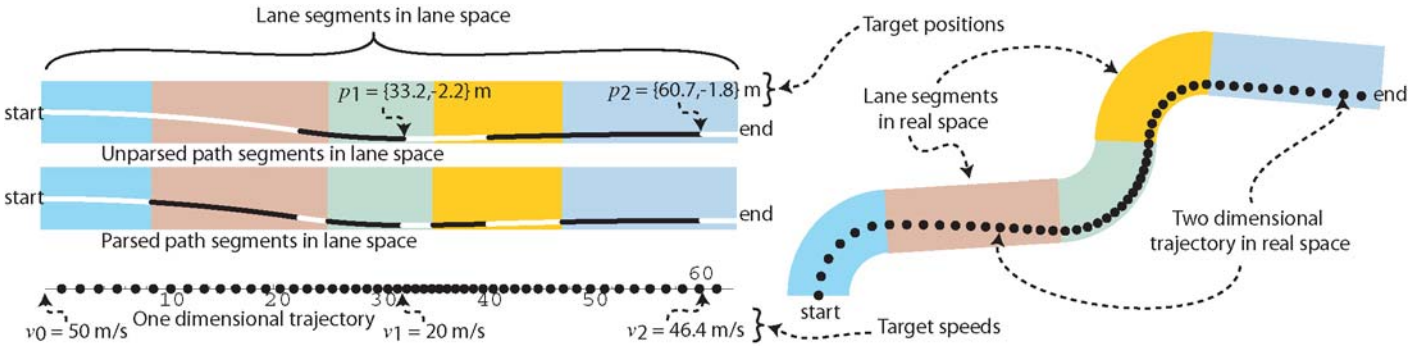

Fig. 5. For a particular sequence of lane segments and input positions and velocities, we have the paths segments in lane space, those same path segments parsed at lane segment boundaries, the one dimensional trajectory with speed changes as required within acceleration maximums, the lane segments in real space, and the one dimensional trajectory superimposed on the path in real space. Achievable speeds are shown in the one dimensional trajectory plot, just where those speeds are achieved. Achievable positions are shown in both the lane space and real space plots. Due to warping of the paths when in an arc type lane segment, the length of the total path in real space will not be the same, in general, as the length of the path in lane space. In the example of this figure the two lengths are nearly equal due to the two high curvature arc lane segments of opposite curvature.

left. We need a slightly more complicated set of discriminants to discern between these four solutions.

In summary, the algorithm for picking the best two circle or one circle solution is first to see if any of the one circle solutions exists and if not, choose the appropriate two circle solution.

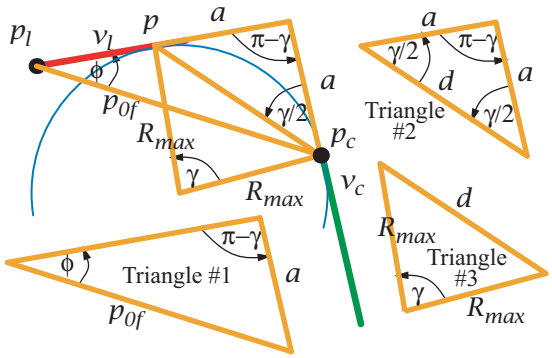

Fig. 6. Triangle definitions necessary to obtain one circle solutions.

1) One circle solution derivations: We examine the values of the following array of four discriminants to determine the best of the four one circle solutions,

$$d_1 = (\text{sgn}(v_f \times v), \text{sgn}(v_0 \times v_f), \\ \text{sgn}(v_0 \times (p_f - p_0)), \text{sgn}(v_f \times (p_f - p_0))).$$

If $d_1 = (-1,-1,-1,1)$, choose the line/circle right solution, if $d_1 = (1,-1,-1,1)$, the choose circle/line right solution, if $d_1 = (-1,1,1,-1)$, the choose line/circle left solution, and if $d_1 = (1,1,1,-1)$, the choose circle/line left solution.

Referring to definitions in Figure 6, we need to find the radius, R_{max}, of the largest circle tangent to v_c at p_c and tangent to the line formed by v_l at p_l. We also need to solve for the unique point p where this circle intersects the line formed by v_c at p_c. Since $\cot(\gamma/2) = \sin\gamma/(1-\cos\gamma)$, where γ is the angular difference in the counterclockwise direction between vectors v_c and v_l and ϕ is the angular difference in the counterclockwise direction between vectors $p_c - p_l$ and v_l. From Triangle #1, we have that $a = p_{0f}\sin\phi/\sin(\pi-\gamma)$. From Triangle #2, $d = 2a\cos(\gamma/2)$. Substituting the expression for a we have $d = 2p_{0f}\csc(\gamma/2)\sin\phi$. We also discover that $d = 2R_{max}\sin(\gamma/2)$ from Triangle #3. Solving for R_{max}, we get

$$R_{max} = \frac{1}{2}p_{0f}\csc(\gamma/2)^2\sin\phi.$$

Finally, we can compute the point p as the point p_c plus the vector of length d in the direction of the side of length d.

$$p = (p_{c_x} + \frac{p_{0f}(v_{c_y} - v_{c_x}\cot(\gamma/2))\sin\phi}{\sqrt{v_{c_x}^2 + v_{c_y}^2}}, \\ p_{c_y} - \frac{p_{0f}(v_{c_x} + v_{c_y}\cot(\gamma/2))\sin\phi}{\sqrt{v_{c_x}^2 + v_{c_y}^2}})$$

Now we have four possible solutions for discriminant conditions that dictate one line and one circle. We only defined one set of equations, because we can get the other three by 1) switching v_0 and p_0 with v_f and p_f, 2) negating v_0 and v_f, or 3) negating v_0 and v_f and switching v_0 and p_0 with v_f and p_f.

2) Two circle solution derivations: Assuming a one circle solution does not exist, recalling that we simply examine the discriminant, $\text{sgn}(v_f \times v)$, to decide the best of the two two-circle solutions, and referring to the definitions in Figure 7, we can compute ϕ_3 and ϕ_4 from input vectors, v_0, v_f, p_0, and p_f. ϕ_3 is the counterclockwise angular difference between two vectors: $p_0 - p_f$ and the vector $\pi/2$ counterclockwise (or clockwise if $v_f \times v = -1$) from v_0. ϕ_4 is the counterclockwise angular difference between two vectors: $p_0 - p_f$ and the vector $\pi/2$ clockwise (or counterclockwise if $v_f \times v = -1$) from v_f. We also set $\alpha = \|v_f\|/\|v_0\|$. From Figure 7, $b + c = R_0(1+\alpha)$ and $d + e = d_{f0}$. Furthermore, the sum of the areas of the four small triangles equals the

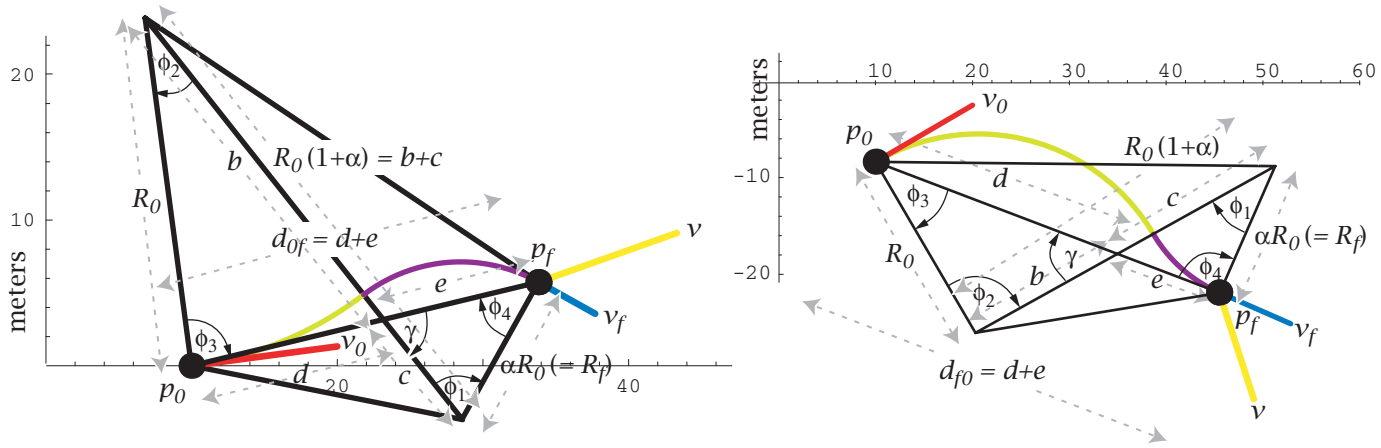

Fig. 7. Here are examples of two types of solutions for when the discriminant, $\text{sgn}(v_f \times v)$, is either 1 or -1. When the discriminant is 1, a "left/right" solution is required, as shown on the left. When the discriminant is -1, a "right/left" solution is required, as shown on the right. For both example solutions, we define a set of triangles and a quadrilateral that are used to generate the smooth path of two circular arcs in lane space.

sum of the areas of one pair of two larger triangles,

$$\frac{1}{2}(be + dc + bd + ec)\sin\gamma = \frac{1}{2}R_0 d_{0f}\sin\phi_3 + \frac{1}{2}\alpha R_0 d_{0f}\sin\phi_4,$$

which gives an expression for $\sin\gamma$,

$$\sin\gamma = \frac{\sin\phi_3 + \alpha\sin\phi_4}{1+\alpha}. \quad (1)$$

We can then compute $\phi_1 = \pi - (\gamma + \phi_4)$ and $\phi_2 = \pi - (\gamma + \phi_3)$.

We can do this again, for the other pair of triangles in the quadrilateral,

$$\frac{1}{2}(be + dc + bd + ec)\sin\gamma = \frac{1}{2}\alpha R_0^2(1+\alpha)\sin\phi_1 + \frac{1}{2}R_0^2(1+\alpha)\sin\phi_2,$$

which implies that

$$\sin\gamma = \frac{R_0(\sin\phi_2 + \alpha\sin\phi_1)}{d_{0f}}. \quad (2)$$

Equating (1) and (2), we have the radius of the initial circle,[4]

$$R_0 = \frac{d_{0f}(\sin\phi_3 + \alpha\sin\phi_4)}{(1+\alpha)(\sin\phi_2 + \alpha\sin\phi_1)}. \quad (3)$$

B. Parsing the path at lane segment boundaries

Now we have a safe and efficient path for the vehicle in lane space and we need to warp (*i.e.*, transform) this path from lane space into real space to compute a one dimensional trajectory, which will then be mapped back onto the warped path. However, even prior to that, we must parse the path at lane segment boundaries. Once this is done, all parsed path segments fall into one of four categories, as can be seen in Figure 5:

1) line path segment in a line lane segment
2) arc path segment in a line lane segment
3) line path segment in a arc lane segment
4) arc path segment in a arc lane segment

C. Warping the paths and computing true path length

In order to generate the one dimensional trajectory, we must first determine the length of all the paths in real space. Transforming any path segment from lane space to real space when the path is within a straight line lane segment does not change the length. However, transforming a path (arc or line) from lane space to real space when the path is within an arc lane segment may change the length.

To compute the path length of both line and arc path segments in an arc type lane segment we first must define a parameterization of the path segment in lane space. Secondly, we must define the radial warping function that takes parameterized points in lane space to real space. With R_p the path radius in lane space, (c_{p_x}, c_{p_y}) the center point of the arc path segment in lane space, a parameterization for points on the arc path in lane space is

$$(q_x(\gamma), q_y(\gamma)) = (c_{p_x} + R_p\cos\gamma, c_{p_y} + R_p\sin\gamma) \quad (4)$$

With (p_{0_x}, p_{0_y}) and (p_{f_x}, p_{f_y}) the initial and final points of the line segment path in lane space, a parameterization for points on the line in lane space is

$$(p_x(s), p_y(s)) = (p_{0_x} + s(p_{f_x} - p_{0_x}), p_{0_y} + s(p_{f_y} - p_{0_y})) \quad (5)$$

To obtain the expression for radial warping, if the center point of the lane segment in lane space is (c_{L_x}, c_{L_y}), c is the path parameter, and $(f_x(c), f_y(c))$ is any parameterized sequence of points in (two dimensional) lane space, the

[4]This solution does not work if the x-component of p_0 is less than the x-component of p_f because the geometry of Figure 7 must be modified, however, this situation holds no interest to our on-road driving application.

radial warp of those points into real space is

$$\alpha(f_x(c), f_y(c)) = (c_{L_x}, c_{L_y}) + (f_y(c) - c_{L_y}) * (-\sin(f_x(c)/c_{L_y}), \cos(f_x(c)/c_{L_y})). \quad (6)$$

Now we have parameterizations for the radial warp of both a line path and an arc path. To find the length of each path, we must compute the path length functions for both path types over the parameter range of the path in lane space. For the arc type path using (4) and (6), with θ_0 and θ_f the start and final angles of the arc path in lane space, the path length in real space is

$$\int_{\theta_0}^{\theta_f} \|\alpha'(q_x(\gamma), q_y(\gamma))\| \, d\gamma = \int_{\theta_0}^{\theta_f} \frac{R_p}{c_{L_y}} (c_{L_y}^2 \cos\gamma^2 + \sin\gamma^2 (c_{p_y} - c_{L_y} + R_p \sin\gamma)^2)^{1/2} d\gamma, \quad (7)$$

which has a symbolic solution. We need to reverse the direction of integration for clockwise arcs ($\theta_0 \rightarrow \theta_f$ for counter-clockwise arcs and $\theta_f \rightarrow \theta_0$ for clockwise arcs). For the line type path, using (5) and (6), the path length in real space is

$$\int_0^1 \|\alpha'(p_x(s), p_y(s))\| ds, \quad (8)$$

which also has a symbolic solution.

Now that we have expressions for path length for all four combinations of path type and lane type, we simply compute the path length for all the parsed paths and sum them up to get total path length. This allows us to consider the trajectory (time-based) planning portion of our problem independent of the lane segment parameters like curvature. However, we *are* interested in the curvature of the planned vehicle path in real space (the warped path), since we may want to adjust the trajectory based on curvature. Happily, the method allows us to change the trajectory along the path after generating the path without changing the shape of the path as will be described in Section VI-E.

D. Generating the Trajectory

Now that computation of vehicle path length in real space for all parsed paths has been defined, it is necessary to compute the trajectory along that path, consisting of the target position, velocity, and acceleration values along that length. This path length is, of course, one-dimensional, so the trajectory can be one-dimensional. The vehicle simply must achieve the sequence of target positions and speeds, as illustrated in Figure 5. The vehicle must achieve these targets within maximum speeds and accelerations. The trajectory planner employs a 2nd order (constant acceleration) motion model to generate the trajectory.

E. Generating target values

The output of the Maneuver Generation (MG) module to the Trajectory Generation (TG) module (see Figure 3) includes target positions and velocities for the vehicle to attain in order to accomplish a particular maneuver (maneuvers such as straddle or circumvent right). MG can detail specific positions and velocities to TG, but only from object safe passing positions and speeds and known posted speed limits.

TG will generate a path and trajectory for these known conditions, but the path and trajectories are not yet planned. Once the path and trajectory are tentatively planned, we must ensure that the speed along the path and the curvature of the path does not excite accelerations normal to the path that exceed a safe normal maximum. This requirement is expressed in Inequality 9. If l is the path length, $a_{N_{max}}(l)$ is the acceleration maximum normal to the path, $R(l)$ is the radius of curvature (inverse of curvature), and $s(l) = \sqrt{a_{N_{max}}(l) R(l)}$, the tangential speed at length l must satisfy the following inequality,

$$v(l) \leq s(l) \quad (9)$$

for all l. The acceleration maximum is going to be specified by various road conditions processed in the sensory processing module at the same level in the hierarchy [2], and $a_{N_{max}}(l)$ is not dependent on the trajectory and only dependent on vehicle position. The radius of curvature, $R(l)$, is also independent of the trajectory.

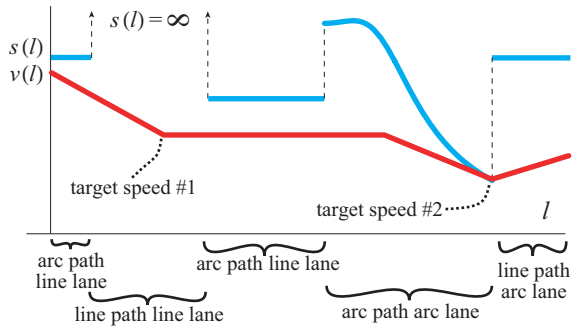

Fig. 8. Here is an example pair of curves, $v(l)$ and $s(l) = \sqrt{a_{N_{max}}(l) R(l)}$ at the conclusion of the recursive algorithm described in Section VI-E. The curve $s(l)$ is in general discontinuous, particularly at lane segment boundaries but also because $a_{N_{max}}(l)$ may also be discontinuous over l.

So, when the curvature of the path exceeds some maximum, TG must determine additional target speed and position values and compute a new trajectory. This does not require a change in the vehicle path, but only a change in the one-dimensional trajectory.

These new target speeds and positions (one-dimensional only) can be computed by a simple recursive algorithm. Whenever the current speed, $v(l)$ is greater than $s(l)$, find the point $s(l_1)$ where the distance between $v(l)$ and $s(l)$ is greatest. Make $s(l_1)$ a new target speed and l_1 a new position and repeat the process until the condition of Inequality 9 is met. The end condition of this algorithm is illustrated in Figure 8.

F. Computing a warped path parameterized by path length

We now have a trajectory that consists of one-dimensional position (path length), speed, and acceleration values at uniformly spaced instances of time. Our goal is to map these trajectory values onto the path in real space. We can use

the same expressions, (4)–(8), we computed in Section VI-C. If a trajectory value lies in a line lane segment, all that is required is to rotate and translate each trajectory value. If a trajectory value lies within an arc lane segment, each trajectory value is first warped then rotated and translated into real space.

In order to map these values onto the path in real space, we need parameterizations by path length for all four lane/path conditions listed in Section VI-B. As described in Section VI-C, we have symbolic closed form solutions for the path length functions for the first three conditions. With these it is not difficult to compute trajectories of position, velocity, and acceleration in real space.

However, for the fourth condition (arc path in an arc lane segment) we cannot find a symbolic closed form solution for the path length function, Equation (7). Therefore, we need to do a numerical approximation of Equation (7). Since our system has a real time requirement and if $\Delta\theta$ is sufficiently small, the following path length function approximation will suffice (we let $\theta_k = k\Delta\theta$ and $k = 0, 1, 2, ..., \text{Floor}(\theta_f/\Delta\theta)$).

$$l(\theta_k) = \sum_{\gamma=\theta_0}^{\theta_0+\theta_k} \frac{R_p}{c_{L_y}} (c_{L_y}^2 \cos\gamma^2 + \sin\gamma^2 (c_{p_y} - c_{L_y} + R_p \sin\gamma)^2)^{1/2} \Delta\theta \quad (10)$$

The integration direction must be reversed for clockwise direction arcs. Since the path length function (10) is a monotonically increasing function, it is invertible, and (using Equations (4) and (6)) $\alpha(q_x(\theta(l_k)), q_y(\theta(l_k)))$ is now parameterized by path length. We then interpolate the one-dimensional trajectory values, l_j, $(j = 0, 1, 2, ...)$ within the sequence l_k and use that interpolation to approximate the warped value, $\alpha(q_x(\theta(l_j)), q_y(\theta(l_j)))$, for all j. Finally, we rotate and translate $\alpha(q_x(\theta(l_j)), q_y(\theta(l_j)))$ to obtain the trajectory position in real space. Velocity and acceleration trajectory values in real space are gotten similarly.

VII. Adding moving objects

The trajectory generator produces target positions and velocities for maneuvering around moving objects. The moving object must have a tangential speed less than that of the vehicle when they are expected to encounter each other, otherwise the vehicle would have no need to maneuver around it, even though the initial speed of the vehicle may or may not be less than the object passing speed. Furthermore, the vehicle may have some non-zero acceleration when the vehicle is commanded to maneuver around the moving object.

Given a simple object motion model (constant velocity or constant acceleration/deceleration to max/min speed) and three types of vehicle motion models (all 2nd order, varying aggressiveness), we have derived closed-form solutions for time- and distance-to-encounter. Such solutions allow us to determine when and where the vehicle is predicted to encounter a moving object, which allows the trajectory generator to determine if a particular maneuver is feasible, given the current state of the environment and vehicle motion constraints. An example time and position to encounter of an autonomous vehicle with a moving object is shown in Figure 9.

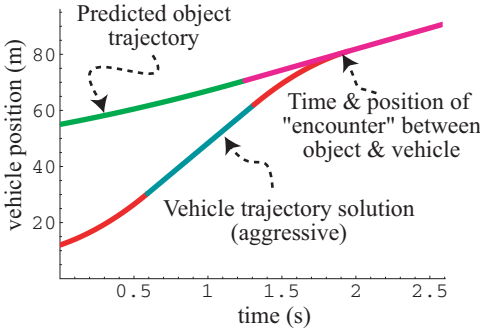

Fig. 9. Example time and position to encounter of an autonomous vehicle with a moving object with simple object motion model.

VIII. Summary

We have presented a trajectory generator for a car-like (non-holonomic) on-road vehicle with simple models for both lane and path, confident that lower levels in the vehicle control hierarchy will smooth any higher order discontinuities in the final path. Because numerical integrations are always required for cubic spline path types and because numerical integrations are only required for radially warped arcs, our trajectory generation method is, in general, less computationally expensive. Due to the radial warping of paths, our approach to vehicle trajectory generation is always safe, but sub-optimal with respect to efficiency, while gaining elegance, simplicity, and some computational efficiency.

References

[1] Anthony Barbera, J. S. Albus, Elena Messina, Craig Schlenoff, and John A. Horst, "How task analysis can be used to derive and organize the knowledge for the control of autonomous vehicles," in *Knowledge Representation and Ontologies for Autonomous Systems: Papers from the AAAI Spring Symposium (Technical Report SS-04-04)*, C. Schlenoff and M. Uschold, Eds. AAAI Press, Menlo Park, CA, 2004.

[2] John A. Horst, Anthony Barbera, Craig Schlenoff, David Aha, "Identifying Sensory Processing Requirements for an On-Road Driving Application of 4D/RCS," Philadelphia, USA, 2004, SPIE.

[3] Andreas Simon and Jan C. Becker, "Arc-length parameterized spline curves for real-time simulation," in *Proceedings of the IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems*, 5-8 October 1999.

[4] Hongling Wang, Joseph Kearney, and Kendall Atkinson, "Arc-length parameterized spline curves for real-time simulation," in *Proceedings of Curve and Surface Design: Saint-Malo 2002*, 2003.

[5] John A. Horst, "Architecture, Design Methodology, and Component-Based Tools for a Real-Time Inspection System," Newport Beach, CA, 2000, 3rd International Symposium on Object-Oriented, Real-Time, Distributed Computing (ISORC).

[6] Alfred Gray, *Differential Geometry of Curves and Surfaces*, CRC Press, second edition, 1998.