Steering Behaviors for Autonomous Vehicles in Virtual Environments

Hongling Wang*
Dept of Computer Science
University of Iowa

Joseph K. Kearney
Dept of Computer Science
University of Iowa

James Cremer
Dept of Computer Science
University of Iowa

Peter Willemsen†
School of Computing
University of Utah

ABSTRACT

This paper presents steering behaviors that control autonomous vehicles populating roadways in virtual urban environments. Behavior programming is facilitated by a set of representations of the environment that use convenient frames of reference in natural coordinate systems. Roadway surfaces are modeled as three-dimensional ribbons that make the local orientation of the road explicit and allow relative distances on the road to be simply computed. Roads and intersections are connected to form a ribbon network. An egocentric representation called a path melds road and intersection segments into a single, continuous ribbon that captures the vehicle's short-term plan of navigation. A topological structure called a route supports wayfinding. We describe how the interrelated ribbon, path, and route representations are used to build multi-component behaviors that plan routes and safely navigate through traffic filled road networks – tracking lanes, shifting lanes to avoid congestion, anticipating lane changes needed to make turns dictated by the route, negotiating intersections, and respecting the rules of the road.

CR Categories: I.6.3 [SIMULATION AND MODELING]: Applications; K.7.m [SIMULATION AND MODELING]: Simulation Support Systems—Environments

Keywords: virtual environments, autonomous agent behavior, steering behavior

1 INTRODUCTION

Synthetically generated traffic plays an important role in high-fidelity virtual urban environments. Simulated vehicles controlled by autonomous behaviors create a backdrop of ambient activity that enlivens the environment and gives a sense of dynamic realism. (See Figure 1.) Programming the behaviors of simulated vehicles with the competence to navigate complex, traffic filled roadway networks remains an enormous challenge.

Driving activities are intimately tied to the structure and geometry of roadways. Roads are contoured to provide good visibility and are designed with smooth curves appropriate to the speed limit. They are divided into lanes that channel traffic into parallel streams. These lanes provide a critical frame of reference for guidance and for determining the spatial relationships with nearby objects. Traffic control signals regulate safe access to shared space at intersections where roads cross.

The way in which the roadway environment is represented has enormous impact on the ease with which driving behaviors can be programmed. For example, while scene graph representations containing textured polygons are handy for rendering, they are awful for tracking roadways and finding routes. Good representations simplify control by making important attributes of environment salient. For example, the frame of reference (egocentric vs. exocentric) and the coordinate system (Cartesian vs. road-based curvilinear) make a tremendous difference on how simple it is to express driving actions.

Figure 1: A child negotiating an intersection with on-coming traffic in our virtual environment Hank.

In [22], we presented a representation of navigable surfaces based on mathematical ribbons – smooth surface strips that twist and turn in space. These ribbons are linked through surface patches (i.e. intersections) which are themselves covered with ribbons that guide traffic from incoming to outgoing lanes. Roads and intersections are annotated with information about the rules of the road that govern right of way and constrain behaviors. In addition, we introduced an egocentric representation called a path that melds road and intersection segments into a single, continuous ribbon. A path is essentially a one lane ribbon overlaid on the road network that represents the immediate plan for movement in a natural frame of reference.

In this paper we extend our representational framework with a representation adapted to the needs of wayfinding called a route. A route is a topological description of a sequence of choice points much like the directions produced by route finding services such as MapQuest. We describe how the interrelated ribbon, path, and route representations are used to build multi-component behaviors that plan routes and safely navigate through traffic filled road networks – tracking lanes, shifting lanes to avoid congestion, anticipating lane changes needed to make turns dictated by the route, negotiating intersections, and respecting the rules of the road.

2 RELATED WORK

Our work draws on a broad cross-section of research in behavior modeling for autonomous agents, control of autonomous vehicles (both real and simulated), and studies of human driving behavior.

Reynolds' [14] landmark work on flocking behaviors of ani-

*e-mail: {howang,kearney,cremer}@cs.uiowa.edu
†e-mail: willemsn@cs.utah.edu

IEEE Virtual Reality 2005
March 12-16, Bonn, Germany
0-7803-8929-8/05/$20 ©2005 IEEE

Proceedings of the IEEE Virtual Reality 2005 (VR'05)
1087-8270/05 $20.00 © 2005 IEEE

mated agents called boids demonstrated the power of decomposing complex control into independent behavior modules each taking responsibility for maintaining some property of global behavior. In later work [15], he distinguished three layers of motion control for a general model of vehicles covering walkers, crawlers, wheeled vehicles, and flying objects. The three layers are: action selection, steering, locomotion. Action selection focuses on strategic planning or goal identification. Steering behaviors have responsibility for immediate path determination allowing agents to "navigate around their world in a life-like and improvisational manner." The details of how motions are generated are the focus of the locomotion layer. While we don't explicitly stratify behaviors into layers, it is natural to categorize our vehicle behaviors as primarily responsible for planning, steering, or locomotion. In addition, the three fundamental representations we use (routes, paths, and ribbons) clearly serve the needs of planning, steering, and locomotion.

A number of groups have used hierarchical schemes for organizing complex control [2, 10, 3, 7]. In [7], we presented a model for autonomous driving behavior based on Hierarchical Concurrent State Machines (HCSM). A *most conservative rule* was introduced to resolve competing behaviors. In [8], the HCSM model is used to generate scenarios from ambient traffic composed of microscopically simulated autonomous vehicles.

Sukthankar[17] introduced methods for controlling the tactical behavior of vehicles in highway traffic. The emphasis of this work is on tactical situation awareness and action selection.

Pursuit point tracking is commonly used for geometric path tracking for both real and simulated vehicles [17, 5, 24]. The pure pursuit approach, which is believed to be the most robust and reliable path tracking method, geometrically determines the curvature that will drive a vehicle to a chosen goal point. This goal point is a point on the path that is one lookahead distance from the current vehicle position. Our method is slightly different from Coulter's in that we make use of the vehicle's orientation when formulating constraints to construct an arc that joins the current point and the goal point.

A substantial body of literature [1, 11, 16] on transportation research has examined driving behaviors of real drivers. Some of the concepts we use in our work such as discretionary lane changing and mandatory lane changing are motivated by this body of research.

3 Background

The research in this paper extends previous results[23, 22] in which we concentrated on creating real-time databases for networks of intersecting roads and walkways in urban virtual environments. The roads in a virtual environment are represented as ribbons interconnected at intersections. A ribbon defines the geometry of an oriented navigable surface. The spine of the ribbon is represented as a 3-dimensional space curve which is approximately arc-length parameterized [18, 19]. A surface normal is defined at each point on the curve allowing the ribbon to twist about its spine. The ribbon establishes a curvilinear coordinate system in which 3-dimensional points are expressed in coordinates of distance along the spine, D, offset on the ribbon surface from the spine, O, and loft (displacement above or below the ribbon), L. Efficient algorithms are designed to compute the mapping between local ribbon coordinates (D,O,L) and global Cartesian coordinates (X,Y,Z) [18, 20].

To provide transition between roads, we define corridors on intersections to connect incoming lanes and outgoing lanes. Corridors are represented as one-lane ribbons.

Although agents can rely on the ribbon structure to determine where they are, where they are going, and what is around them, the transitions at intersection boundaries can make it cumbersome to extract and interpret this information. To facilitate behavior programming, we created a data object called a path. A path is a one-lane ribbon overlaid on the road network. It consists of a composition of lanes on roads and intersection corridors that define a single, continuous coordinate system.

4 Ribbons as a Basis for Tracking

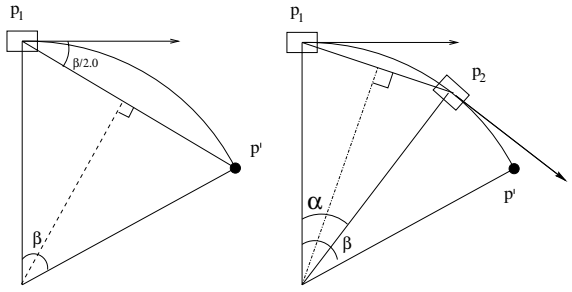
In our virtual environment software, Hank, steering behaviors control the motion of a virtual vehicle through two parameters: acceleration and steering angle. Because vehicles roll on wheels, they are subject to non-holonomic constraints that restrict their allowable motions. Thus, vehicles with wheels that roll on fixed axles cannot move directly sideways (as much as we might like when parallel parking.) We model these constraints by moving a vehicle on a circular arc tangent to both the front and back wheels.

Road tracking is accomplished by aiming at a succession of pursuit points on the lane center line. At each time step, the steering angle that intercepts the pursuit point is computed by calculating the circle that passes through the vehicle's current position, is tangent to vehicle's current orientation, and intersects the pursuit point.

The vehicle's path provides a convenient representation for determining a pursuit point on the lane center line a distance Δd ahead of the vehicle. Let the ribbon coordinates of the vehicle on its path be $(D,O,0)$. This places the vehicle on the center line of a lane or intersection corridor in contact with the surface. The path coordinates of the pursuit point are computed as $(D+\Delta d, O+\Delta O, 0)$, where Δd is the look ahead distance and ΔO is the lateral distance between the vehicle and its pursuit point on the ribbon. If the path is not planar, then the pursuit point may not lie in the tangent plane of the ribbon surface at the current position of the vehicle. In order to calculate the circle trajectory that will intercept the pursuit, we first project the pursuit point onto the tangent plane of the ribbon at the vehicle's position.

Let the current position of a vehicle be p_1, the pursuit point be p, and the rotation matrix of the vehicle be (p_1^X, p_1^Y, p_1^Z) where p_1^X is the facing direction, p_1^Y points to the left side, and p_1^Z points upward. The projection of the pursuit point is computed as

$$p' = p_1 + (\overrightarrow{p_1 p} \cdot p_1^X) * p_1^X + (\overrightarrow{p_1 p} \cdot p_1^Y) * p_1^Y. \quad (1)$$



(a) A circular track computed from a pursuit point

(b) New position and orientation on the circular track

Figure 2: Pursuit point tracking.

In each time step, a virtual vehicle is assumed to follow a planar circular track uniquely determined by the current position, the current orientation and the projected pursuit point of the vehicle. The circular track is shown in figure 2(a), where p_1 is the current position and p' is the projection of the pursuit point. The curvature κ of

Proceedings of the IEEE Virtual Reality 2005 (VR'05)
1087-8270/05 $20.00 © 2005 IEEE

the circular track is computed and used directly to control the moving direction of a vehicle. The arc β between the current position p_1 and the projected pursuit point p' is computed as

$$\beta = 2 * acos(\frac{p_1^X \cdot \overrightarrow{p_1 p'}}{|p_1^X| * |\overrightarrow{p_1 p'}|}). \quad (2)$$

The curvature of the circular track is

$$\kappa = \pm sin(\frac{\beta}{2.0}) / \frac{|\overrightarrow{p_1 p'}|}{2.0}, \quad (3)$$

where κ is positive if p' is located on the left side of the facing direction and negative if p' is located on the right side of the facing direction.

Let p_2 be the new position of the vehicle on the circular track after a time step of length Δt. The new position p_2 is shown in figure 2(b), where p_1 is the position of the vehicle at the start of a time step of length Δt and p' is projection of the pursuit point on the tangent plane of the ribbon surface at p_1. With the current speed v and the current acceleration a, we compute the new position of the vehicle after a time step of length Δt. The arc length between the old position p_1 and the new position p_2 along the circular track is

$$l = v * \Delta t + 0.5 * a * \Delta t^2. \quad (4)$$

The angle between p_1 and p_2 is

$$\alpha = l * \kappa. \quad (5)$$

The chord length between p_1 and p_2 is

$$c = 2 * \frac{sin(\alpha/2.0)}{\kappa}. \quad (6)$$

The vector $\overrightarrow{p_1 p_2}$ is

$$\overrightarrow{p_1 p_2} = c * (cos(\frac{\alpha}{2.0}), sin(\frac{\alpha}{2.0}), 0) \times (p_1^X, p_1^Y, p_1^Z). \quad (7)$$

The new position p_2 is computed as $p_1 + \overrightarrow{p_1 p_2}$. The facing direction of the vehicle at p_2 on the tangent plane of the ribbon surface at p_1 is

$$p_2^X = (cos(\alpha), sin(\alpha), 0) \times (p_1^X, p_1^Y, p_1^Z). \quad (8)$$

After the vehicle is moved along the circular track for a time step, we should locate the vehicle on the ribbon. It is possible that the new position p_2 is above or below the ribbon surface if the ribbon is not a plane. The location of the vehicle on the ribbon can be determined by perpendicularly projecting p_2 to the ribbon surface. We map p_2 from Cartesian coordinates to the ribbon coordinates $(D_{p_2}, O_{p_2}, L_{p_2})$. The ribbon coordinates of the projected point p_2' is $(D_{p_2}, O_{p_2}, 0)$. The Cartesian coordinates of p_2' is computed by mapping its ribbon coordinates into Cartesian coordinates. After p_2' is determined, we compute the rotation matrix of the vehicle on p_2', $(p_2'^X, p_2'^Y, p_2'^Z)$. The upward direction $p_2'^Z$ is just the ribbon surface normal at p_2'. The left direction $p_2'^Y$ is $p_2'^Z X p_2^X$. The facing direction $p_2'^X$ is $p_2'^Y X p_2'^Z$.

5 Steering Behaviors Based on the Path of a Virtual Vehicle

Vehicle motion is controlled by a collection of independent behaviors each responsible for some aspect of driving and each proposing a single value influencing one of the control parameters (steering angle or acceleration).

One set of behaviors focuses on steering direction (i.e. on where to go). They indirectly control steering angle by selecting a pursuit point towards which the vehicle aims. In Section 6.1 we describe how lane changes are accomplished by smoothly shifting the pursuit point from one lane to another.

Another set of behaviors focuses on controlling vehicle speed (i.e. when to go and how fast to get there). On each iteration of the simulation, these behaviors compute an acceleration that would best satisfy the goal they are charged with attaining. In this section we describe three behaviors that vie for control of acceleration. We then describe a simple process to synthesize their independent computations into a coordinated whole behavior that respects all of the constraints on speed and produces a smooth and consistent overall pattern of behavior.

5.1 Cruising Behavior

The basic goal of the cruising behavior is to run the vehicle at its desired speed. We use a simple proportional controller for cruising,

$$a_c = \begin{cases} min(a_p, k_p^c * (v_d - v)) & \text{if } v < v_d \\ 0 & \text{if } v = v_d \\ max(a_n, k_p^c * (v_d - v)) & \text{if } v > v_d \end{cases}, \quad (9)$$

where a_c is the acceleration for cruising behavior, v is the current speed, v_d is the desired speed, a_p is the maximum positive acceleration, a_n is the maximum deceleration, k_p^c is a proportional parameter. If a_c is applied to vehicle control, the vehicle will slow down if the current speed is higher than the desired speed or speed up if the current speed is lower than the desired speed. Over time, the cruising controller converges smoothly to the desired speed.

5.2 Following Behavior

Following behavior is responsible for controlling a vehicle to maintain a safe distance between it and the vehicle ahead of it on the lane [12]. Usually, an agent looks ahead a range of distance to search for its leader on its path. If there is no vehicle on its path within this range of distance, the vehicle is treated as having no leader. If some vehicles are on its path within this range of distance, the lead vehicle which is closest to the vehicle is defined to be the leader of the vehicle. If a vehicle has a leader, a proportional-derivative controller [6] for following behavior computes an acceleration for the vehicle,

$$a_f = max(a_n, k_p^f * \triangle s - k_v^f * \triangle v), \quad (10)$$

where a_f is the computed acceleration, $\triangle s$ is equal to the actual following distance minus the desired following distance, $\triangle v$ is equal to the follower's speed minus its leader's speed, k_p^f is a proportional parameter, k_v^f is a derivative parameter, and k_v^f is equal to $2.0 * \sqrt{k_p^f}$ for critical damping [6]. To compute the actual following distance of the follower vehicle, we project the leader vehicle into the ribbon coordinate system of the path of the follower vehicle. The difference between the distance coordinates of the follower vehicle and the leader vehicle is computed as the actual following distance.

Following prescriptions of driving manuals, the desired following distance is set to be proportional to the vehicle speed,

$$s_d = max(s_1, v * k_d), \quad (11)$$

where s_d is the desired following distance, v is the current speed, k_d is a constant, and s_1 is a minimum following distance that assures reasonable separation when the leader is stopped. Similarly, the range of distance over which a vehicle looks for a leader is proportional to vehicle speed to ensure that vehicles react in sufficient time as they approach a leader,

$$s_l = max(s_0, v * k_l), \quad (12)$$

Proceedings of the IEEE Virtual Reality 2005 (VR'05)
1087-8270/05 $20.00 © 2005 IEEE

where s_l is the range of distance, v is the current speed, k_l is a constant and s_0 is the least range of distance.

The value of a_f computed in formula 10 might be positive, 0, or negative. Since the goal of the following behavior is to prevent collisions, positive accelerations are ignored.

5.3 Intersection Behavior

Intersection behavior regulates access to intersections in conformance with conventional rules of the road. It acts as a gate to the intersection, preventing vehicles from entering the intersection until it is their turn to cross. This includes adherence to traffic lights and yielding to other vehicles in or approaching the intersection that have right of way. If the vehicle is denied access to the intersection, the behavior will slow the vehicle to a stop and keep it stopped until it is safe to cross. Once the vehicle enters the intersection, it must rely on other behaviors to navigate through the intersection.

5.3.1 To Enter or Wait?

As the vehicle approaches an intersection, the intersection behavior attends to the traffic control signal regulating the vehicle's path through the intersection. Each corridor is regulated by a traffic control signal and labeled with right-of-way information. When the time to arrival at the intersection is sufficiently small, the intersection behavior considers the state of the light and the rights of way to determine whether or not it is legal and safe to enter the intersection.

If the vehicle is approaching an intersection under a red light, the intersection behavior computes a deceleration based on a control law that will bring the vehicle to a halt at the stop line that marks the end of the lane. If the traffic light is green, the driver assesses traffic conditions to determine if it can safely cross the intersection, respecting right-of-way rules. If the traffic light is yellow, the driver determines whether it should stop or go based on its speed and proximity to the intersection and the state of surrounding traffic.

The state of the traffic control signal and the movements of traffic are highly dynamic. The action selection mechanism must constantly reassess the changing circumstances to determine what the appropriate response is at the moment.

In addition to respecting traffic signals, drivers must obey right-of-way rules that apply to the path they plan to take through an intersection. The corridors that cross an intersection are labeled with right-of-way information that describes the priorities of traffic crossing this corridor with respect to traffic crossing other corridors. The intersection behavior estimates the time t_1 when the vehicle will enter the intersection and the time t_2 when the vehicle will leave the intersection based on its current position and speed. If the time between t_1 and t_2 represents an interval of time during which no vehicle having the right of way will traverse the corridor, then the vehicle is permitted to proceed through the intersection. Otherwise, it will stop and yield the right of way to other vehicles. For each approaching vehicle that has right of way over the yielding vehicle, we estimate the time t_1^i when it will enter the intersection and the time t_2^i when it will leave the intersection. If the time window $[t_1, t_2]$ overlaps with any time window $[t_1^i, t_2^i]$, the yielding vehicle has no gap and its decision is to stop. Otherwise, the yielding vehicle has found a gap and its decision is to go forward.

5.3.2 Stopping the Vehicle

If the vehicle is required to stop, either because the traffic light is red or yellow, or because it must yield right of way, the intersection behavior will compute a deceleration that will bring the vehicle to a stop at the stop line. To accomplish this, we calculate the constant deceleration that will stop the vehicle at the stop line [4],

$$a_i = -\frac{v_c^2}{2 * s}, \tag{13}$$

where a_i is negative acceleration, v_c is the current speed and s is the distance between the current position and the desired stopping position. If a_i is continuously applied to vehicle control, a vehicle will just stop at the desired stopping position when its speed declines linearly to 0. We define a desired range $[a_i^l, a_i^u]$ for negative acceleration. The lower bound a_i^l represents the maximum braking capacity of the vehicle. The upper bound a_i^u represents the minimum negative acceleration the agent would like to apply for stopping purpose. The value of a_i tells us whether a vehicle is too close to an intersection to be able to stop without exceeding the maximum braking power and whether it is so distant from the intersection that it is too early to be concerned about decelerating.

- If $a_i \in [a_i^l, a_i^u]$, a successful stopping action can be applied to vehicle control.
- If a_i is out of the desired range $[a_i^l, a_i^u]$ in positive direction, i.e., $a_i > a_i^u$, then we believe the desired stopping position is still far away. It is too early to respond to the traffic control signals.
- If a_i is out of the desired range in negative direction, i.e., $a_i < a_i^l$, it means that the vehicle is too close to the desired stopping position so that it is impossible to stop the vehicle at the desired stopping position.

Traffic engineering standards establish guidelines for the timing of yellow lights to allow vehicles sufficient time to cross intersections before the light turns to red. With properly timed lights, the third condition should only occur during the yellow interval and the vehicle should have time to pass through the intersection before it turns to red.

If the decision of action selection is to stop and a_i computed in formula 13 is within $[a_i^l, a_i^u]$, we get an acceleration contribution a_i from intersection behavior. Otherwise, the intersection behavior has no acceleration contribution and a_i computed in formula 13 is disregarded.

5.4 Combining Acceleration Behaviors

The cruising, following, and intersection behaviors all produce acceleration as output. Employing a most conservative approach, we choose to apply the minimum acceleration of three accelerations,

$$a = min(a_c, a_f, a_i), \tag{14}$$

where a_c is the acceleration contribution of cruising behavior, a_f is the acceleration contribution of following behavior, a_i is the acceleration contribution of intersection behavior, and a is the acceleration after combination of component behaviors. If the following or intersection behaviors are inactive, we remove their terms from formula 14.

6 Route

A path provides a moment-to-moment guide for steering a virtual vehicle. It is the short-term plan of action for the driving agent. However, it is poorly suited for route planning to get from one place to another on a map. As vehicles navigate through the environment they need both the local geometric information provided by a path and more abstract topological directions about the sequence of

Proceedings of the IEEE Virtual Reality 2005 (VR'05)
1087-8270/05 $20.00 © 2005 IEEE

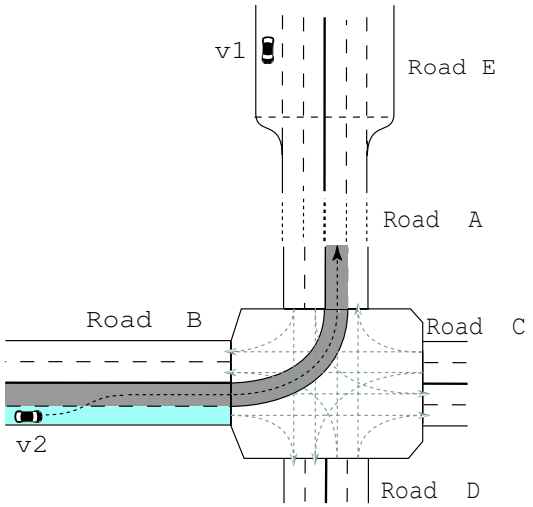


Figure 3: Two situations requiring Mandatory Lane Changes

roads and turns that a good navigator provides. For this purpose we introduce a new representation called a route.

A route is defined as a connected sequence of roads and intersections that an agent has traversed, is traversing, or will traverse in the future. A route encodes traveler directions similar to what is generated by route finding services such as MapQuest. In the route of an agent, we designate roads and intersections but not lanes on roads or corridors on intersections. Thus, a virtual vehicle can drive on different lanes on the same road or different corridors on the same intersection while it follows the route.

We see the route of a virtual vehicle as a strategic goal of the driving agent. Tactical behaviors make use of the path of the virtual vehicle to achieve the strategic goal of the driving agent. The route of a virtual vehicle is extended in advance of the path of the vehicle. The path is selected to conform to the requirements of the route.

6.1 Lane Changing Behavior

On multi-lane roads, drivers have a choice of lane. Drivers are often free to choose which lane suits them best based on preference for speed, density, and/or visibility. Sometimes, this choice is restricted by transit requirements, for example, because the current lane soon terminates. When a lane change is motivated by preference, for example, because a driver wants to increase its speed, we call it a discretionary lane change (DLC). When a lane change is motivated by transit requirements, for example, to position the vehicle to turn at the next intersection, we call it a mandatory lane change (MLC) [1].

A driving agent makes a decision to consider a DLC when it is not satisfied with the driving conditions on the current lane. The desire for a DLC is to seek better driving conditions on another reachable lane. The driving agent compares the driving conditions on the current lane and the other lanes. A goal lane must have better driving conditions than the current lane. The route of the driving agent forms a constraint on the choice of the goal lane. A candidate lane should not be a goal lane if the driving agent is not able to reach the next road in its route from this lane.

A driving agent makes a decision to consider an MLC when it must leave the current lane in order to follow its route. Vehicle v_1 on road E and vehicle v_2 on road B in figure 3 show two representative cases of MLC. In the first case, vehicle v_1 has to leave the current lane because the current lane will soon terminate. In the second case, where the route of vehicle v_2 includes road B, the intersection, and road A, vehicle v_2 has to leave the current lane to position itself for a turn at the upcoming intersection so that the vehicle is able to enter road A after leaving the intersection. In both cases, the vehicle must leave the current lane before it runs to the end of the lane. The goal lane for an MLC is determined by the route of the virtual vehicle.

A driving agent may choose to cross 1 or more lanes to get to the goal lane. However, a vehicle is presumed to change one lane at a time. After the vehicle changes one lane toward the goal lane, the lane changing decision and the goal lane are reevaluated. The adjacent lane to which the vehicle changes is called the target lane. The target lane should be on the same side with and is uniquely determined by the goal lane. Therefore, we have a long-term goal and a short-term target. In this way the control of lane changing becomes easier because the control is based on a shorter process. This allows the driving agents to continuously adapt to the changing dynamics of the surrounding traffic.

After the target lane is determined, the next step is to wait for an acceptable gap on the target lane so that the vehicle can move into the target lane safely. If the required gap is available, the vehicle starts a lane changing action. Although a lane change can be canceled before a lane changing action is started, it must be completed once the lane changing action is started.

6.1.1 Lane Changing Decision Making

The first step of lane changing is making the decision to change lanes. Driving agents continuously assess road conditions to determine whether or not it is desired or needed to change lane.

Let's consider DLC decisions first. In our behavior model, a driving agent uses the speed of the traffic on the current lane to evaluate the driving condition on the current lane. Traffic speed on a lane is limited by the slowest moving vehicles. To determine the speed of the traffic on the current lane, a driving agent looks ahead a range of distance and checks the speed of all the vehicles on the current lane within this range. The range of distance is given by,

$$s_r = max(s_2, v * k_r), \tag{15}$$

where k_r is a constant, v is the current speed of the vehicle, and s_2 is the minimum range of distance. The speed of the traffic on the current lane is defined to be the speed of the vehicle which runs slowest among the vehicles within this range of distance. If the speed of the traffic on the current lane is slower than the desired speed of a driving agent, the driving agent is motivated to seek a DLC. Otherwise, the driving agent seeks no DLC.

The MLC decisions of our driving agents are motivated by the desire of driving agents to follow their routes. To determine if an MLC is necessary, a driving agent checks to see if the vehicle must leave the current lane before the current lane ends (see the cases in figure 3). If it must leave the current lane, the driving agent estimates the length of time $\triangle t$ that it will take for the vehicle to travel from the current position to the end of the lane,

$$\triangle t = \frac{\triangle s}{v},$$

where $\triangle s$ is the distance between the current position of the vehicle and the lane end and v is the current speed of the vehicle. The driving agent then estimates the time needed to successfully execute the n lane changes needed to move from the current lane to the goal lane,

$$\triangle t > n * t_0, \tag{16}$$

where t_0 is a liberal estimate of the time needed to successfully change from one lane to an adjacent lane and the current lane is n lanes apart from the goal lane. If the condition in formula 16 is true,

Proceedings of the IEEE Virtual Reality 2005 (VR'05)
1087-8270/05 $20.00 © 2005 **IEEE**

the vehicle has more than enough time to transfer from the current lane to the goal lane. Therefore, it is not necessary to start an MLC yet. If the condition in formula 16 is not true, the driving agent starts an MLC.

An MLC decision and a DLC decision may conflict. For example, the target lane for an MLC decision may be the left lane while the target lane for a DLC decision may be the right lane. To resolve this conflict, we fuse the DLC and MLC decisions together into a single lane changing decision. We define a decision to change lane has preemptive priority over a decision to not change lane, and an MLC has a preemptive priority over a DLC. With this two rules, we arrive at the below result,

$$\begin{array}{lllll} \text{no MLC,} & \text{no DLC} & => & \text{no Lane changing} & \\ \text{MLC,} & \text{no DLC} & => & \text{MLC} & . \\ \text{MLC,} & \text{DLC} & => & \text{MLC} & \end{array} \quad (17)$$

What should the result be when the vehicle wants to make a DLC and has no need to make a MLC? At first glance, it appears to be a DLC. However, this conclusion is problematic if we look a step deeper. Let's consider the below situation,

$$(n+2)*t_0 > \triangle t > n*t_0, \quad (18)$$

where the variables n, t_0, and $\triangle t$ have the same meaning as in formula 16. The condition 18 satisfies the condition 16, therefore an MLC is still not necessary. Consider what happens if the driving agent decides to perform a DLC that takes it one lane further away from the goal lane of the MLC. We need time t_0 to complete one DLC plus time $(n+1)*t_0$ to complete $(n+1)$ MLCs. In total, we need time $(n+2)*t_0$. The DLC leaves the vehicle short of time to perform the required $(n+1)$ lane changes needed to reach the goal lane of the MLC. Therefore, the fusion of no MLC and a DLC is divided into two cases,

$$\text{no MLC, DLC} => \begin{cases} \text{DLC} & \text{if } \triangle t \geq (n+2*m)*t_0 \\ m\text{-limited DLC} & \text{if } \triangle t < (n+2*m)*t_0 \end{cases} . \quad (19)$$

By an m-limited DLC, we mean that the DLC can increase the distance to the goal lane of the MLC by at most m. Therefore, the vehicle will still have sufficient time to complete the required $(n+m)$ MLCs after it has completed the m-limited DLC. Combining formula 17 and formula 19, the lane changing decision can be one of four possibilities: an MLC, an m-limited DLC, a DLC, or no lane change.

6.1.2 From Lane Changing Decision to Lane Changing Action

After a decision of changing lane is made, the goal lane must be determined before lane changing action is started. For a DLC, the goal lane can be chosen from all the lanes on which the traffic flow is in the same direction as that on the current lane. The speed of the traffic on the goal lane should be higher than that on the current lane and highest among all the other candidate lanes. For an MLC, the driving agent picks a candidate lane which satisfies route requirements. Often, there is only one choice. It is the nearest lane to the current lane that satisfies the vehicle's route requirements. For an m-limited DLC, the choice of the goal lane combines information about the lane structure of the current road, the route, and the road traffic. The candidate lanes are all the reachable lanes that are at most $m+n$ lanes away from the goal lane of the MLC. From the candidate lanes, we select the lane for which the speed of the traffic is higher than that on the current lane and highest among all the candidate lanes as the goal lane for an m-limited DLC.

Once the goal lane for a lane change is determined, the target lane is also determined. For an MLC, the vehicle must get to the target lane before the current lane ends in order to follow its route. However, it is possible that the vehicle cannot complete the MLC and get to the target lane in the current speed before the current lane ends, for example, because the target lane continuously has heavy traffic. If the vehicle cannot stop on the road to wait for the traffic on the target lane to pass by (for example, vehicle v_2 in figure 3), the agent has to give up its current route and generates a new route. Otherwise, the vehicle must slow down and finally stop to wait for the traffic on the target lane to pass by before it starts a lane changing action. This is achieved by intersection behavior. For example, intersection behavior will control vehicle v_1 in figure 3 to slow down and stop before the zero-area intersection between road A and road E until the traffic on the target lane has cleared.

A lane changing vehicle needs a free space ahead of the vehicle and a free space behind the vehicle on the target lane so that the lane changing vehicle can move to the target lane safely. The space between the vehicle and the lead vehicle on the target lane is called a lead gap. The space between the vehicle and the lag vehicle on the target lane is called a lag gap. After the target lane is determined, the driving agent waits for an acceptable lead gap and an acceptable lag gap [1] on the target lane before it starts a lane changing action. It is reasonable to estimate the minimum acceptable lead and lag gaps with the speed of the vehicle multiplied by a time constant,

$$\begin{cases} g_{lead} = v*k_{lead} \\ g_{lag} = v*k_{lag} \end{cases} , \quad (20)$$

where g_{lead} is the length of the minimum acceptable lead gap, g_{lag} is the length of the minimum acceptable lag gap, v is the current speed of the vehicle, k_{lead} and k_{lag} are time constants. If the lead and lag gaps are larger than the minimum acceptable lead and lag gaps respectively, the vehicle starts to move over to the target lane gradually.

6.1.3 The Action of Lane Changing

When a vehicle is tracking its current lane on road, the pursuit point is located on the centerline of its path. If we gradually shift the pursuit point off the centerline of its path and control it to gradually approach the centerline of the target lane, a vehicle will deviate from the centerline of the current lane and approach the centerline of the target lane [17].

We use the path as a frame of reference to control the trajectory of the vehicle during the lane changing process. The pursuit point is computed in ribbon coordinates on its path. Before the lane changing action starts, the offset coordinate of the pursuit point is set to be 0 so that the vehicle will track the centerline of the current lane. After the lane changing action ends, the pursuit point is moved to the centerline of the target lane. We use a proportional-derivative controller to control the lateral motion of the pursuit point. In each time step, we compute an acceleration o'' for moving the pursuit point in the lateral direction,

$$o'' = k_p^{LC} * (o_t - o) - k_v^{LC} * o', \quad (21)$$

where k_p^{LC} is a proportional parameter, k_v^{LC} is a derivative parameter, o_t is the distance between the centerlines of the original lane and the target lane, o is the current offset coordinate of the pursuit point, and o' is the current speed of the pursuit point in the lateral direction. The pursuit point has no lateral motion before the lane changing action starts. Therefore o' starts from 0. For critical damping, we have $k_v^{LC} = 2.0 * \sqrt{k_p^{LC}}$. The offset coordinate of the pursuit point will change to

$$o = o + \int_{t_0}^{t_1} o' dt, \quad (22)$$

at the end of the current time step, where t_0 and t_1 are the start and the end of the current time step respectively. The speed of the

Proceedings of the IEEE Virtual Reality 2005 (VR'05)
1087-8270/05 $20.00 © 2005 IEEE

pursuit point in the lateral direction is increased to

$$o^{'} = o^{'} + \int_{t_0}^{t_1} o^{''} dt, \tag{23}$$

at the end of the current time step.

In figure 4 we show an example of the trajectory of a vehicle during a lane change and the trajectory of the pursuit point of the vehicle. The offset coordinate of the pursuit point changes from 0 to the target offset under control of a *PD* controller. We can see the vehicle follows the pursuit point gracefully in an "s" curve from the centerline of the current lane to the centerline of the target changing lane. This is consistent with the result reported by Salvucci[16] from driving simulator data generated with real drivers.

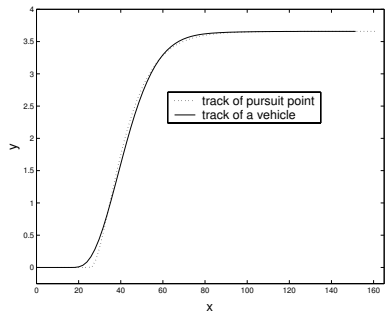


Figure 4: The trajectories of a virtual vehicle and its pursuit point during a lane change on a straight road

6.2 Combining Orientation Behaviors

In the section 5.4, we describe how the three behaviors that control acceleration are combined by selecting the minimum acceleration produced by the active behaviors. This conservative strategy works because the underlying constraints can be treated as inequality constraints (i.e. vehicles should drive less than the speed limit, follow no closer than the following distance, and stop before the intersection). By selecting the minimum acceleration we simultaneously satisfy all of the implicit inequality constraints.

The choice of steering direction cannot be framed in the same way. The driver must make an unequivocal commitment to either track the current lane or steer to an adjacent lane. Thus, we need a decision rule to select one action or another. The preemption rules in 17 and 19 give the basis for deciding whether or not to change lanes. Once this decision is made and the action is initiated, lane tracking is disabled and the lane change controller is engaged. When the vehicle has completed the lane change and is prepared to track it's new path, control reverts back to lane tracking.

6.3 Interactions Among Steering Behaviors

During initial testing of the lane changing behavior we discovered unanticipated interactions between the lane changing and following behaviors. Further analysis revealed two sources of trouble. The first problem was that the following behavior unnecessarily inhibited acceleration once a lane change action had been initiated. The problem was most apparent when a vehicle approached a slow moving or stopped vehicle and had been unable to change lanes because of the density of traffic in adjacent lanes. Once a follower has reached its desired following distance and had matched the speed of the leader, the following controller in formula 10 prohibited acceleration. As a consequence, when the opportunity to pass did present itself, the follower could not overtake the leader until it had moved out of the leader's lane. This led to sluggish lane changing behavior that looked unnatural. Human drivers appear to avoid this trap by either leaving additional space between themselves and slow moving drivers so that they have more maneuvering room or by relaxing the following distance and approaching the leader as they turn into the adjacent lane. We've implemented the second approach by creating a link between the lane change and following behaviors. When a lane change is initiated, the desired following distance is reduced permitting the vehicle accelerate and overtake the follower as it turns into the adjacent lane. In addition, we disable the following behavior when the follower has a clear trajectory to the adjacent lane. In [18], we present a method to compute the visibility of the leader. As the follower steers around the leader, we compute obstacles in a beam representing the forward trajectory of the follower's vehicle. When the leader is no longer in this beam, we suspend the following behavior for the duration of the lane change action.

An undesirable side effect of reducing the following distance is that there is an increased potential for a collision should the leader abruptly decelerate. To address this danger, we also heighten the responsiveness of the controller by increasing k_p^f in formula 10. This stiffens the controller and leads to quicker responses should the leader unexpectedly decelerate.

The second way in which the lane changing and following behaviors interacted was related to the switch in leaders as the follower departed one lane and entered another. In our initial implementation, the lane changing vehicle was assigned to a single lane based on the position of its center of mass. This meant that at any instant, it had a single leader. The consequence for lane changing was that there was a sudden change in leader as the vehicle crossed the threshold between the lanes. This sometimes resulted in abrupt deceleration when the following behavior discovered an unveiled threat in the new lane. To address dual constraints of leaders in the old and new lanes, we now activate a second following behavior in the new lane when the lane change action is initiated. The two following behaviors execute in parallel until the follower has satisfied to visibility constraint presented above.

The interaction between the lane changing and following behaviors is important because it demonstrates the interrelatedness of component behaviors. For the most part, behaviors run as independent processes. However, there are times when actions require coordination between behaviors. Based on our experience to date, we conjecture that these interactions are most likely to occur during periods of transition such as in lane changing when a vehicle leaves one stream of traffic and enters another.

7 Results and Discussion

This paper presents five steering behaviors for controlling the autonomous traffic on simulated roadways. Three of the behaviors (cruising, following, and intersection) influence the acceleration of the vehicle. The other two (tracking and lane changing) determine the steering angle. The five behaviors run independently with no explicit supervisory control and with just a few simple mechanisms to coordinate the interactions among the behaviors. Steering behaviors rely on prior planning to make choices about which roads to take represented in the vehicle's route and a kinematic model to move the vehicle consistent with the control parameters.

We've tested the aggregate behavior in a wide variety of traffic conditions on simulated city and highway road networks. The code is exceptionally robust. In hundreds of hours of tests, we've had no failures or aberrant behavior. In most respects, the traffic distributions and patterns are plausible as compared to real roadways. We comment on a few exceptions below.

We attribute much of our success in behavior modeling to having a good underlying foundation of roadway representations. The ribbon structure of roads and intersection corridors provides a natural

Proceedings of the IEEE Virtual Reality 2005 (VR'05)
1087-8270/05 $20.00 © 2005 **IEEE**

frame of reference for driving behaviors. Local orientation is explicitly represented, relative distances along the road are simple to compute, and the cross-sectional lane structure maps onto the lateral offset dimension making it easy to move between lanes and to determine relative positions of vehicles on neighboring lanes. The ribbon network consisting of interconnected roads and intersection corridors provides a natural basis to construct the egocentric representation that simplifies steering behaviors by blending roads and intersections into a single, uniform path. Map directions are easily constructed from the road network and used to guide immediate choices of lane or corridor.

One of the challenges in representing roadways and programming behaviors to drive on them is dealing with the many different ways intersections are formed. One approach to coping with the variety has been to create specialized representations for different intersection types and code a separate behavior to handle each configuration [9]. In contrast, we have aimed to develop generic representations of road network and general purpose behaviors to drive on them. To test this approach, we've tried our behaviors on 4-way intersections, T-junctures, Y-split intersections, single lane merges, and single lane exits.

Highway merges and exits presented a modeling and behavior challenge because they share properties of both roads and intersections. From the perspective of vehicles on the through lanes, the road continues through the merge area. However, from the perspective of the merging vehicle, the change in road is similar to an intersection. Because of the change in topology of the road network, we needed to introduce an intersection to connect road segments together. Our solution was to treat the merge area as a road (including a merge lane) spliced into the highway. The intersections that connect the merge segment to the rest of the highway are lines with no width. These zero-area intersections provide the connective structure while preserving the road-like properties of the highway. With this design, merging and exiting are well handled as mandatory lane changes. We believe the flexibility to model many different kinds of intersections is a strength of our approach.

While individual vehicles demonstrate plausible driving behaviors, we discovered that strict adherence to the rules of the road can lead to unusual traffic patterns. The cautiousness of our intersection behaviors sometimes caused turning vehicles to wait through many light cycles for a safe gap when traffic was dense. By including shoulder periods in the light cycle when all sides see red, we were able to loosen gap acceptance rules so that at least one car in each lane crossed the intersection during each green light.

One of the consistent observations made about demonstrations of our traffic is how well-behaved our drivers are. As is often the case with computer generated graphics, the idealized world of synthetic drivers is too perfect. We believe the cluster of behaviors we present in this behavior provide ample opportunity to create plausibly diverse behaviors by varying control values (for example by adding noise to the output of controllers) and by varying controller parameters such as controller gains and preference values for speed, following distance, lane change decisions, acceptable gaps, etc. In future work, we plan to investigate techniques to exploit the many degrees of freedom in our behaviors to generate a more natural appearance characteristic of real traffic.

8 Acknowledgements

This material is based on work supported through National Science Foundation grants CDA-9623614, INT-9724746, EIA-0130864, IIS-0428856, and IS-0002535 and National Center for Injury Prevention and Control/Centers for Disease Control and Prevention grant R49/CCR721682.

References

[1] K. I. Ahmed. *Modeling Driver's Acceleration and Lane Changing Behavior*. PhD thesis, Massachusetts Institute of Technology, 1999.

[2] B. Blumberg amd T. Galyean. Multi-level direction of autonomous creatures for real-time virtual environment. In *ACM Siggraph*, pages 47–54, Aug 1995.

[3] N. Badler, B. Webber, W. Becket, C. Geib, M. Moore, C. Pelachaud, B. Reich, and M. Stone. Planning and parallel transition networks: Animation's new frontiers. *Computer Graphics and Applications: Proc. Pacific Graphics'95*, pages 101–117, 1995.

[4] E. Boer, N. Kuge, and T. Yamamura. Affording realistic stopping behavior: A cardinal challenge for driving simulators. In *Proceedings of 1st Human-Centered Transportation Simulation Conference*, November 2001.

[5] R. C. Coulter. Implementation of the pure pursuit path tracking algorithm. Technical report, The Robotics Institute, Carnegie Mellon University, 1992.

[6] J. J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley, 1989.

[7] J. Cremer, J. K. Kearney, and Y. Papelis. A framework for behavior and scenario control in virtual environments. *ACM Transactions on Modeling and Computer Animation*, (3), 7 1995.

[8] J. Cremer, J. K. Kearney, and P. Willemsen. Directable behavior models for virtual driving scenarios. *Transactions of the society for computer simulation international*, (2), 6 1997.

[9] S. Donikian. Vuems: a virtual urban environment modeling system. *Computer Graphics International*, pages 84–92, June 1997.

[10] S. Donikian and E. Rutten. Reactivity, concurrency, data-flow and hierarchical preemption for behavior animation. *Programming Paradigms in Graphics'95, Eurographics Collection*, 1995.

[11] X. Fang, H. Pham, and M. Kobayashi. Pd controller for car-following models based on real data. In *Proceedings of 1st Human-Centered Transportation Simulation Conference*, November 2001.

[12] M. Lemessi. An slx-based microsimulation model for a two-lane road section. In *Proceedings of the 2001 Winter Simulation Conference*, March 2000.

[13] J. M. Plumert, J. K. Kearney, and J. F. Cremer. Children's perception of gap affordances: Bicycling across traffic-filled intersections in an immersive virtual environment. *Child Development*, 75(4):1243–1253, 2004.

[14] C. W. Reynolds. A distributed behavior model. *Computer Graphics*, 21:25–34, 1987.

[15] C. W. Reynolds. Steering behaviors for autonomous characters. In *proceedings of Game Developers Conference*, 1999.

[16] D. D. Salvucci and A. Liu. Time course of a lane change: Driver control and eye-movement behavior. *Transportation Research*, March 2002.

[17] R. Sukthankar. *Situation Awareness for Tactical Driving*. PhD thesis, Robotics Institute, Carnegie Mellon University, 1997.

[18] H. Wang. *Efficient Roadway Modeling and Behavior Control for Real-time Simulation*. PhD thesis, The University of Iowa, 2004.

[19] H. Wang, J. K. Kearney, and K. Atkinson. Arc-length parameterized spline curve for real-time simulation. In *Proceedings of 5th international conference on Curves and Surfaces*, pages 387–396, 2002.

[20] H. Wang, J. K. Kearney, and K. Atkinson. Robust and efficient computation of the closest point on a spline curve. In *Proceedings of 5th international conference on Curves and Surfaces*, pages 397–406, 2002.

[21] H. Wang, J. K. Kearney, J. Cremer, and P. Willemsen. Steering autonomous driving agents through intersections in virtual urban environments. In *Proceedings of 2004 International Conference on Modeling, Simulation and Visualization Methods*, pages 10–16, June 2004.

[22] P. Willemsen, J. K. Kearney, and H. Wang. Ribbon networks for modeling navigable paths of autonomous agents in virtual urban environments. In *Proceedings of IEEE Virtual Reality Conference*, pages 79–86, March 2003.

[23] P. J. Willemsen. *Behavior and Scenario Modeling For Real-Time Virtual Environment*. PhD thesis, The University of Iowa, 2000.

[24] J. Wit, C. D. Crane, and D. Armstrong. Autonomous ground vehicle path tracking. *Journal of Robotic Systems*, (8):439–449, July 2004.