In Proceedings of IEEE Virtual Reality 2003, p. 79–86, Los Angeles, March 2003

Ribbon Networks for Modeling Navigable Paths of Autonomous Agents in Virtual Urban Environments

Peter Willemsen
School of Computing
University of Utah
willemsn@cs.utah.edu

Joseph K. Kearney
Dept. of Computer Science
University of Iowa
kearney@cs.uiowa.edu

Hongling Wang
Dept. of Computer Science
University of Iowa
howang@cs.uiowa.edu

Abstract

This paper presents a real-time database modeling complex networks of intersecting roads and walkways in urban virtual environments. The database represents information about the layout of streets and sidewalks, the rules that govern behavior on roads and walkways, and the locations of agents with respect to road and sidewalk structures. This information is used by programs that control the behavior of autonomous vehicles and pedestrians populating the virtual urban environment. Roads and sidewalks are modeled as ribbons in space. The ribbon structure provides a natural coordinate frame for defining the local geometry of navigable surfaces. This geometry is important for way finding and also forms the geometric basis on which spatial relationships among agents are defined. The database includes a powerful run-time interface supported by robust and efficient code for locating objects on the ribbon network, for mapping between Cartesian and ribbon coordinates, and for determining behavioral constraints imposed by the environment.

Figure 1: Virtual urban environment.

1. Introduction

Building dynamic, active content for use in virtual environments is a difficult and time consuming process. Quite often, the information required for programming the behaviors that might populate such environments, such as vehicle or pedestrian navigation, is not easily computed or inferred from sets of polygons into a usable form. Such behavior codes are generally complex and require significant spatial, logical, and socio-cultural information about the environment. The research addressed in this paper involves creating a run-time environment database to facilitate behavior programming in virtual environments. This work bridges a gap between the visual, polygonal model and the needs of autonomous agents to gain spatial awareness.

In the real world, much of our way finding in urban environments occurs on ribbon-like pathways such as roads, sidewalks, and alleys. These pathways wind through cities meeting and crossing other pathways. They structure the movements of travelers and provide a frame of reference for determining local spatial relationships. Social conventions for organizing interactions on roadways and walkways rely on having a shared understanding of this frame of reference. For example, in many societies we walk and drive on the right side of a sidewalk or road. In order to traverse pathways in a virtual environment, autonomous agents must understand the spatial structure and local geometry of navigable routes. To travel safely, they must be aware of nearby agents and obstacles.

This paper presents a scheme for representing networks of ribbon-like pathways to support behavior and scenario programming in virtual environments. The scheme encodes (1) geometric information about the shape of pathways, (2) topological information about interconnections among pathways, (3) logical information encoding rules governing behavior on the pathways, and (4) occupancy information giving the locations of nearby objects on the pathway.

Because we focus on real-time, interactive simulation, a high premium is placed on robustness and efficiency. A database employing the scheme has been implemented in the Hank virtual environment software and rigorously tested in psychological studies investigating the behavior of children and adults riding a virtual bike on roadways populated with simulated vehicles.

2. Related Work

Our work builds on two related bodies of research: urban modeling aimed at animating humans in virtual environments and road modeling for driving simulation. In the spirit of Farenc et al's notion of informed environments, we embed information in the representation of the roads and sidewalks to guide agent behaviors [11]. In their scheme, behavior content is connected to the hierarchical scene graph used for rendering. In contrast, our representations are independent (but correlated with) the scene graph, relatively flat, and emphasize the 3D ribbon-like structure of pathways. This simplifies motion planning and allows us to build very fast, robust computations for accessing local geometry, the positions of nearby objects, and recovering embedded content that guides agent decision making and actions.

Our research is closely tied to research in modeling urban roadway environments [2, 3, 5, 14, 19]. Civil engineering guidelines specify design standards for road construction based on three planar shapes: straight, arc of a circle, and spiral [1, 10]. Modern roads are composed of a sequence of straight and curved sections with a transition spiral interposed between them to smoothly blend from one curvature to another curvature. A desirable property of roads built to this standard is that curvature varies smoothly along the road contour. One problem with this approach is that many real roads do not conform to the design standards (especially roads in older urban areas). The work presented here uses a spline based representation, but provides code to translate standard road specifications into accurate spline approximations.

Donikian et al. developed a comprehensive system for modeling complex urban networks of streets, sidewalks, and tramways called the Virtual Urban Environment Modeling System (VUEMS) [9, 15]. As in our work, the underlying geometry of pathways in VUEMS is based with interconnected ribbon-like surface segments. Our work extends this approach from 2D to 3D, addresses computational difficulties in the parameterization and coordinate transforms, and introduces overlay ribbons to simplify steering and collision avoidance behaviors.

3. Representing Ways as Ribbons in Space

Our representation of urban streets and sidewalks is based on a network of interconnected ribbons. Each ribbon represents a segment of a ”way” (a roadway or walkway). The ribbon defines the geometry of a navigable surface and gives a local orientation to the way. The ribbon channels pedestrian and vehicle traffic into parallel streams by defining two preferred directions of travel (along the two opposing tangents of the central axis of the ribbon). It is important to emphasize that the representation places no restrictions on agent behaviors. Autonomous vehicles and pedestrians can choose to move across a ribbon, for example, to change lanes.

In addition to representing geometric information for navigation and route planning, the ribbon provides a frame of reference for defining spatial relations among occupants of the way. Thus, an oncoming vehicle heading straight at us in an adjacent lane on a curved road poses no threat because we expect it to remain in its lane and pass by us.

We represent a ribbon by an annotated 3-dimensional space curve. This curve acts as a central axis or spine for the way. 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. Figure 2 illustrates the ribbon based frame of reference.

Some simulation computations are most naturally expressed in Cartesian coordinates. For example, the dynamics code that computes object motions from control parameters set by object behaviors is most simply computed in Cartesian coordinates. Other computations, such as behavior code that tracks roads and avoids obstacles, require that object locations be expressed in ribbon coordinates. Because these computations are performed at a very high frequency, it is essential to have efficient and robust code to map from ribbon coordinates to global Cartesian coordinates and to compute the inverse mapping from global Cartesian coordinates to local ribbon coordinates.

To avoid self-intersections, the width of a way is restricted to be less than the radius of curvature of the spine. As a consequence, there is a single nearest point on the spine for all points on the surface of a way. Thus, the mapping from Cartesian coordinates to way coordinates is unique.

3.1. The Guts of Ribbon Computations

The choice of a mathematical representation for the ribbon axis is critical to the efficiency and effectiveness of database computations. Parametric cubic splines are commonly used to represent space curves in computer graphics. They are flexible, smooth, differentiable, and simple to evaluate. In addition to these properties, it is important that the ribbon axis be parameterized by arc-length so that distances can be easily computed from ribbon coordinates.

Parametric spline curves are not, in general, parameterized by arc length. Most approaches to compute arc length or to reparameterize a curve by arc length are too inefficient to be used in real-time applications. In [16] we present a simple and efficient technique to generate approximately arc-length parametrized spline curves that closely match the shapes characteristic of road and sidewalk contours. A desirable byproduct of the approach is that the component cubic segments of the spline have equal arc length. This makes indexing of segments very efficient.

Following notation conventions used in robotics [6], we identify the coordinate system in which a point is expressed with a preceding superscript. Thus, ${}^Cp = (X_p, Y_p, Z_p)$ represents the point p in Cartesian coordinates and ${}^Rp = (D_p, O_p, L_p)$ represents the point p in ribbon coordinates.

The mapping from ribbon coordinates to Cartesian coordinates is computed by a simple three step process. Given ribbon coordinates for a point ${}^Rp = (D_p, O_p, L_p)$:

1. Compute a point, ${}^Cp_1 = (X_{p_1}, Y_{p_1}, Z_{p_1})$, on the ribbon axis, $Q(s)$, at distance D_p by evaluating the spline curve at $Q(D_p)$.

2. Compute a point, ${}^Cp_2 = (X_{p_2}, Y_{p_2}, Z_{p_2})$, displaced from p_1 in a direction perpendicular to both the ribbon axis and the ribbon normal at p_1 by offset O_p.

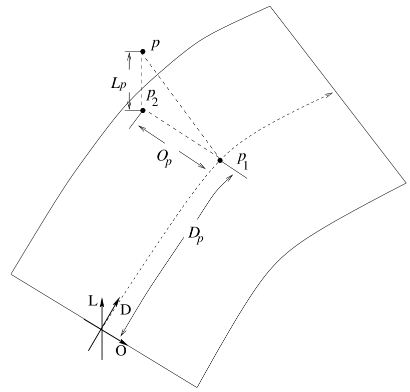

Figure 2: Curvilinear coordinate system based on a ribbon structure.

3. Compute a point, ${}^Cp = (X_p, Y_p, Z_p)$, displaced from p_2 in the direction of the ribbon normal at p_2.

The point ${}^Cp$ is the Cartesian representation of the point ${}^Rp = (D_p, O_p, L_p)$ in ribbon coordinates.

The inverse mapping (from Cartesian to ribbon coordinates) is usually a serious computational bottleneck in driving simulators. The steps to compute the inverse mapping mirror those for the forward mapping. Given Cartesian coordinates ${}^Cp = (X_p, X_p, Z_p)$:

1. Compute the closest point on the ribbon axis, $Q(D_{p_1}) = {}^Cp_1 = (X_{p_1}, Y_{p_1}, Z_{p_1})$. This point has ribbon coordinates: ${}^Rp_1 = (D_{p_1}, 0, 0)$. Note that $D_{p_1} = D_p$.

2. Project the Cartesian point, ${}^Cp$, onto a line perpendicular to both the ribbon axis and the ribbon normal at $Q(D_p)$ to get the nearest point on the ribbon surface, ${}^Cp_2 = (X_{p_2}, Y_{p_2}, Z_{p_2})$. The distance between ${}^Cp_1$ and ${}^Cp_2$ gives the offset O_p.

3. Compute the distance from ${}^Cp$ to the point ${}^Cp_2$ on the ribbon surface. This gives the loft L_p.

The key component in this mapping is the computation of the closest point on the central axis of the ribbon from Cartesian coordinates (step 1 above). Conventional optimization techniques such as Newton's method or quadratic minimization work well most of the time. However, we found that the standard techniques consistently fail (converge very slowly or diverge) at a small number of points on many ordinary ribbons. Because of the frequency with which the mappings are performed (i.e.

thousands of times a second for a modestly complex simulation) even these rare problematic instances are likely to occur with regularity. This leads to unacceptable computational delays and can halt a simulation if the optimization procedure is not terminated.

To address weaknesses with standard optimization techniques, we developed a two stage technique that combines quadratic minimization and Newton's method. This method finds the closest point on the spine of the ribbon with very high reliability in a small number of iterations [17].

To facilitate modeling of roads from standard engineering specifications, we provide interfaces to construct spline segments from straight, arc of circle, and spiral parameters. Adjacent segments can be concatenated together to form multi-piece ribbons. Database management software allows these composite ribbons to be treated as a single, uniform ribbon.

3.2. Coping with Cracks

Small numeric errors in modeling can cause cracks and overlaps to appear along the borders where ribbons connect to other ribbons or intersections. A point very near the boundaries of two logically connected ribbons may map onto one, both, or neither ribbons. Numeric error by itself is not a significant problem; for the most part, virtual environments do not demand that object positions be known with very high precision. (The exception is high fidelity dynamics computations used to drive haptics devices and motion platforms.) Cracks pose the greatest challenge. If a Cartesian point maps onto neither of the adjoining ribbons it disappears from the logical pathway network. When the point represents the location of an agent, this disappearing act can disrupt the behavior of this agent as well as other nearby agents.

We solve this problem by giving an object an imperceptible nudge whenever it lands in a crack. The nudge is sufficiently minute that it creates no difficulties with controllers and is not noticeable to viewers. This simple adjustment eliminates a thorny problem.

3.3. Ribbon Structure and Attributes

The ribbon structure provides a framework in which to embed logical information about the properties of ways. This information is important to inform behaviors about the characteristics of the pathways they traverse.

The cross-section of a ribbon is decomposed into lanes that serve to channel traffic flow. Lanes carry attributes that indicate their width and typical function, e.g. vehicle lane, parking lane, sidewalk, or boulevard. In the current implementation, ways have a constant cross-sectional profile. The model can be extended to permit varying lane widths by adding supplemental functions to compute lane widths and offsets at a distance D_p along the way.

Longitudinal attributes encode markings and features that govern the rules of the roads. This includes speed limits, passing zones, stop lines, and the location of traffic control objects such as flag men. Wayside features are parameterized by ribbon coordinates and provide ribbon-relative information to behavior programs. Tying the data to the ribbon structure affords agents convenient access to the attributes regulating appropriate behavior on a road or walkway.

4. Intersections

Ways connect to other ways through intersections. In contrast to roadways and walkways, an intersection has no central axis and hence imposes no local orientation. An intersection defines a surface area with a well-defined boundary along which incident ways connect to it.

The ribbons that bundle road and sidewalk lanes together terminate at the boundary of an intersection. Agents entering the intersection must choose an appropriate lane to exit the intersection and then plot a path across the intersection from the point of entry to the point of departure. To guide agents across an intersection, we overlay the intersection with corridors (essentially invisible one-lane roads) that splice together the lanes of incoming and outgoing ways. Agents track corridors through intersections to reach outgoing ways.

Intersections provide queries to determine how incoming and outgoing lanes interconnect. Agents use this information to plan routes through the way network.

Intersections pose a difficult challenge for agents [4]. Because corridors cross and merge with one another, agents must be alert for potential collisions with other agents traversing the intersection. Formal right-of-way rules and informal social conventions are critical to the safe and orderly flow of traffic through intersections. Right-of-way rules prioritize movement of vehicles through intersections on the basis of arrival time, incoming and outgoing lanes, signage, and the state of signals that regulate traffic flow. Generally accepted social conventions help to avoid problems where the formal rules are ambiguous or incomplete.

To assist agents in determining priorities, we annotate corridors with right-of-way information. This includes information about signs and traffic control devices that regulate passage on corridors and information about the relationships between corridors. Traffic control devices and signage define constraints on corridor traversal. For example, a stop sign indicates that approaching vehicles should stop at the entrance to the corridor and yield to traffic on unrestricted corridors that cross or merge with the corridor to be taken.

The traffic control state tells agents what rule currently applies to a corridor (stop, yield, go, protected go, etc.), The internal structure of an intersection sets a context for interpreting what a rule means. For example, in a four-way stop example, the agent must know what corridors cross or merge with the corridor to be taken. The stopped agent must wait until there are gaps on all these corridors sufficiently large to permit the agent to safely transit its corridor. To compute corridor gaps, the agent must inspect the corridors and the incident lanes that feed the intersection corridor. To assist agents in applying right-of-way rules, corridors are annotated with information about dependency relations between corridors. An agent waiting at a stop sign examines the dependency relations on the corridor to be taken to find the crossing and merging corridors that have right of way. Object behaviors are responsible for selecting actions that respect traffic control state consistent with the relationships encoded in the corridor.

4.1. Connecting Ways to Intersections

Ways connect to intersections at attachment points on the boundaries of intersections. The beginning or ending point of the ribbon axis (either end will do) must be coincident with a fixed connection point on the edge of the intersection. Surrounding the fixed point of attachment on the intersection boundary are floating "junctures" to which lanes of the incident ribbon connect. The number and type of junctures on an edge is predetermined in the intersection definition. Only roads with profiles that match the junctures in number and type can be connected to the intersection at this attachment point. By saying that the junctures are floating, we mean that the order of the junctures is set, but the junctures can slide along the edge to match roads with different lane widths.

Corridors start and end at junctures. By fixing the number and order of junctures along the boundary of the intersection, we can specify the interconnection topology

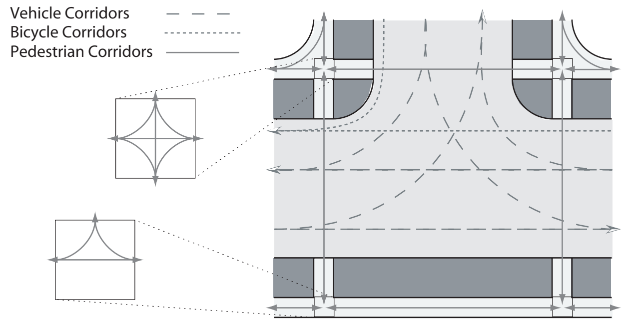

Figure 3: Example hierarchical intersection.

(how junctures are linked by corridors) and dependency relations among corridors without knowing what specific roads will be connected to the intersection. This greatly simplifies modeling because many intersections come in standard configurations. These can be duplicated, transformed, and easily modified to adapt to new configurations. For many intersections, we can compute corridor geometry automatically using Hermite cubic spline curves constructed from the attachment points and tangent directions of the connecting ribbons. This curve is then sampled and converted into an arc-length parameterized spline. Complicated corridors must be explicitly modeled by specifying a series of interpolation points in the intersection through which the corridor must pass.

In addition to modeling road crossings, we use intersections to add or delete lanes on a continuous stretch of pavement and to close a loop by connecting two ends of a road together. To add or delete a lane, we join two separate roads with n and $n+1$ lanes. We've found it convenient to build the intersection so that the endpoints of connecting lanes are coincident. Thus, the intersection becomes a line with lanes attached to junctures on both sides and, consequently, there are no corridors or dependency relations. The $n+1$st lane terminates on the intersection boundary. If the lane is deleted, vehicles must change from the terminating lane to an adjacent lane before reaching the intersection. Vehicles can choose to change to an added lane after they cross the intersection. Zero-length intersections provide a convenient means to specify the topological relationship between connecting roads with differing numbers of lanes while maintaining a simple and consistent interface to the database.

Intersections are hierarchically structured. Thus, an intersection can contain other intersections. The corridors of the parent intersection link to attachment points on the sub-intersection. We've found the hierarchical structure

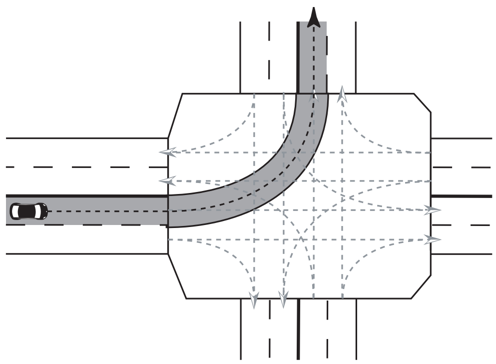

Figure 4: Path structure overlayed onto the ribbon network.

most useful in modeling sidewalk structures. Figure 3 illustrates intersection topology.

5. Paths – Overlay Ribbons

Agents rely on the ribbon structure of a way to determine where they are, where they're going, and what's around them. The transitions at intersection boundaries can make it cumbersome to extract and interpret this information. The change in coordinate systems that happens as attention shifts from a way to an intersection (on entering an intersection) and from an intersection to a way (on exiting the intersection) causes behavior code to be cluttered with bookkeeping chores. For example, to cross an intersection, a vehicle must track a multi-segment contour composed of a lane of a way, an intersection corridor to which it connects, and an outgoing lane of another way. In addition, collision avoidance behaviors must monitor other objects following the same route ahead of the vehicle. Adding additional complication, the coordinate systems of the connected ribbons may be oriented in opposing directions (i.e. linked head to head or tail to tail) making distance computations tricky and error prone.

From the egocentric view of an object traversing the road network, queries such as "*Where am I?*", "*Where am I going?*", "*What's around me?*", and "*What rules apply to me?*" are most naturally understood with respect to the near-term route the object plans to take. From the object's point of view, this route forms a natural frame of reference for navigation and sets a context for interpreting spatial relations and right of way rules.

To facilitate behavior programming, we created a data object called a path that represents the short-term, intended route of an object. A path is a one-lane ribbon overlayed on the road network. Paths consist of a composition of lanes on ways and intersection corridors that define a single, continuous coordinate system. As an example, in Figure 4, the path for a vehicle making a left turn is shown with shading.

A path provides a local, egocentric coordinate system for database queries. Steering behaviors control motion trajectories by aiming for a succession of look-ahead points located on the path some distance ahead of the object's current location. By formulating the look-ahead queries with respect to the composite path, we avoid awkward bookkeeping to handle transitions from road to intersections and intersections to roads. The database maps path queries into the corresponding way or intersection queries and returns the results. As a consequence, the steering code is enormously simplified.

Similarly, path-based occupancy queries return information about relative positions of other objects on the path. One commonly used query determines the next object in front of the reference object (called the leader of the object.) As with geometric queries, the path-based occupancy queries eliminate the need to cope with road and intersection boundaries throughout the behavior code. The path provides a smooth, continuous frame of reference.

In contrast to the permanence of roads and intersections, paths are temporary constructions. They are frequently modified as an agent moves through the environment and reformulates its plans based on traffic conditions, impulses, and instructions from directors that orchestrate scenarios [8]. The database provides operations to assist in path construction and modification.

5.1. Occupancy

An important advantage of the ribbon framework is that it naturally defines spatial relationships among occupants. This is essential for the virtual environment since dynamic objects must know where they are and where other nearby objects are located. For example, the object in immediately ahead of another object on a path is important for following behavior.

There are two general forms of occupancy queries. The first form returns the first object on a way between point a and point b. The second form returns an ordered list of all objects on a way between point a and b. The way can be a road, an intersection corridor, or a path. The queries are oriented from point a to point b and can be reversed by swapping the order of points a and b. Moreover, queries

Figure 5: An adult riding through virtual “Biketown”.

can be restricted to a single lane of a multi-lane road.

Queries are made efficient by exploiting the spatial structure of ways. As objects enter the virtual environment, they are added to an occupancy list for a way. On each iteration of the simulation, occupancy for each way is updated based on new object positions.

6. A Modeling Language for Way Networks

A language that describes way networks is implemented and is used with the Hank virtual environment system. The language preserves the structure of way networks and includes the ability to create intersections and road forms which can be instanced and transformed so that they fit together. The language attempts to mimic traditional instancing and templating so that ways, intersections, and lane definitions can be reused within the model space. Complete syntactic and semantic details for the language are presented in [18].

7. Results

The ribbon network presented in the preceding sections forms the basis of the urban environment model in our real-time simulation software, Hank. In addition to database modeling of city streets and sidewalks, Hank includes modules for real-time process scheduling, for simulating the dynamics of scene entities, for rendering images on large projection screens, and a substrate for modeling reactive and intentional behaviors through Hierarchical Concurrent State Machines (HCSM) [7]. The Hank software drives our immersive virtual bicycle riding environment shown in Figure 5.

Our research broadly focuses on the use of virtual environments as laboratories for the study of human behavior. We are currently using the bicycle simulator to conduct experiments investigating the ability of children and adults to negotiate traffic-filled roadways.

Synthetic traffic is generated by populating the roads with vehicles controlled by independent, autonomous driving behaviors. Vehicle behaviors query the database to plan paths through the road network, to steer along roads and corridors, to obey speed limits, to detect vehicles ahead on their path and follow at a safe distance, to observe the state of traffic lights and stop accordingly, and to yield right of way as they cross intersections. Intersection crossing behaviors use corridor dependency relations to determine priorities. When intersection navigation requires crossing or merging with a higher priority corridor, vehicles wait for gaps in the traffic sufficient to permit safe traversal.

In other work, we are examining methods to control the behavior of autonomous walkers traveling alone or in small groups on networks of ribbon walkways [12, 13].

In addition to behavior programming, the database is also important in configuring scenarios to meet the needs of experimenters. In order to compare results across subjects, it is important that the essential aspects of the simulation be replicated from trial to trial. Autonomous vehicles are dynamically created and removed from the environment during the simulation by director objects. These disembodied agents are responsible for making sure the right things happen at the right place and time. The ribbon structure provides a natural way to specify experimental requirements for object placement and the sites for critical actions.

The database software has undergone rigorous testing under the stringent demands of psychological studies. In six months of running an average of about 20 hours a week we have had no failures in database computations. The most costly computations are the functions that map from Cartesian to ribbon coordinates. We’ve found our implementation sufficiently fast to run dozens of vehicles in real time with very high accuracy.

While performance is very important, the most critical test of the approach is the ease of use in coding complex behaviors. Our experience has been enormously positive. Ribbon based coordinate frames provide a simple and natural means to situate objects in their environment.

8. Conclusion

Engaging virtual urban environments need street life – cars that drive and pedestrians that stroll. Creating animated, autonomous agents that realistically locomote on foot or on wheel and that plausibly interact with each other and with human participants remains an enormously difficult challenge. A good conceptual model of the urban landscape forms a key building block on which to develop behavioral "street sense".

To adeptly maneuver through cities, autonomous agents need to understand their whereabouts. To gain this situational awareness, agents must be able to ascertain: "*What routes are accessible to me?*", "*Where other objects are in relationship to my intended route?*", and "*What constraints does the environment impose on my interactions with other objects?*".

We believe interconnected ribbon coordinate systems provide a natural and efficient means to situate objects in their environment. They provide a convenient way to represent the spatial layout of navigable pathways. Our implementation provides a robust and efficient means to access information about the surroundings in natural coordinate frames. By embedding logical constraints in the environment model, we provide ready access to the rules of the road that govern object interactions.

9. Acknowledgments

The authors wish to thank Ken Atkinson for his help on the numerical methods employed in ribbon computations and Jim Cremer for his significant contributions to the Hank simulator and many valuable insights on behavior and scenario control. We also wish to thank Joan Severson, Shayne Gelo, and Kate Kearney for creating our visual databases. This material is based on work supported through National Science Foundation Grants CDA-96-23614, INT-9724746, EIA-0130864, and IIS-0002535.

References

[1] B.E. Artz. An analytical road segment terrain database for driving simulation. In *Driving Simulation Conference*, pages 274–284, Sophia Antipolis, France, September 1995.

[2] A.C. Bailey, A.H. Jamson, Parkes A.M., and Wright S. Recent and future development of the Leeds driving simulator. In *Driving Simulation Conference*, July 1999.

[3] S. Bayarri, M. Fernandez, and M. Perez. Virtual reality for driving simulation. *Communications of the ACM*, 39(5):72–76, May 1996.

[4] E. Bonakdarian. *Generation and Management of Ambient Traffic in Real-Time Driving Simulation*. PhD thesis, University of Iowa, May 2001.

[5] O. Carles and S. Espie. Database generation system for road applications. In *Driving Simulation Conference*, pages 87–103, 1999.

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

[7] J. Cremer, J. Kearney, and Y. Papelis. HCSM: A framework for behavior and scenario control in virtual environments. *ACM Transactions on Modeling and Computer Simulation*, 5(3):242–267, July 1995.

[8] J. Cremer, J.K. Kearney, and P. Willemsen. Directable behavior models for virtual driving scenarios. *Transactions of the Society for Computer Simulation*, 14(2), 1997. Special Issue on Multiagent Systems.

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

[10] D.F. Evans. Ground vehicle database modeling. In *Real Time Systems '94*, Paris, France, 1994.

[11] N. Farenc, R. Boulic, and D. Thalman. An informed environment dedicated to the simulation of virtual humans in urban context. *EUROPGRAPHICS*, 18(3), 1999.

[12] T.R. Hostetler. *Controlling Steering Behavior for Small Groups of Pedestrians in Virtual Urban Environments*. PhD thesis, University of Iowa, August 2002.

[13] T.R. Hostetler and J.K. Kearney. Strolling down the avenue with a few close friends. In *Third Irish Workshop on Computer Graphics*, pages 7–14, Dublin, Ireland, March 2002.

[14] G. Reymond, O. Munier, and A. Kemeny. A road description using reference points. In *Proceedings of the first seminar on traffic generation*, pages 15–21, 1994.

[15] G. Thomas and S. Donikian. Modelling virtual cities dedicated to behavioural animation. *EUROPGRAPHICS*, 19(3), 2000.

[16] H. Wang, J. Kearney, and K. Atkinson. Arc-length parameterized spline curves for real-time simulation. *5th International Conference on Curves and Surfaces*, June 2002.

[17] H. Wang, J. Kearney, and K. Atkinson. Distance calculation between space points and cubic curves. *5th International Conference on Curves and Surfaces*, June 2002.

[18] P. Willemsen. *Behavior and Scenario Modeling for Real-Time Virtual Environments*. PhD thesis, University of Iowa, May 2000.

[19] Papelis Y. and S. Bahauddin. Logical modeling of roadway environment to support real-time simulation of autonomous traffic. In *SIVE95: The First Workshop on Simulation and Interaction in Virtual Environments*, pages 62–71, Iowa City, IA, March 1995.