IMPLEMENTATION OF THE 4D/RCS ARCHITECTURE WITHIN THE SOUTHWEST SAFE TRANSPORT INITIATIVE

George McWilliams
Michael Brown
Southwest Research Institute
San Antonio, TX

ABSTRACT

Over time, the National Institute of Standards and Technology (NIST) has refined the 4Dimension / Real-time Control System (4D/RCS) architecture for use in Unmanned Ground Vehicles (UGVs). This architecture, when applied to a fully autonomous vehicle designed for missions in urban environments, can greatly assist in the process of saving time and lives by creating a more intelligent vehicle that acts in a safer and more efficient manner. Southwest Research Institute (SwRI®) has undertaken the Southwest Safe Transport Initiative (SSTI) aimed at investigating the development and commercialization of vehicle autonomy as well as vehicle-based telemetry systems to improve active safety systems and autonomy. This paper will discuss the implementation of the 4D/RCS architecture to the SSTI autonomous vehicle, a 2006 Ford Explorer.

INTRODUCTION

The Southwest Safe Transport Initiative (SSTI) was started in 2006 and aimed at investigating the development and commercialization of vehicle autonomy as well as vehicle-based telemetry systems to improve safety and facilitate traffic flow. The first phase of the program focused on understanding the state-of-the-art. It was during this phase that the 4D/RCS Architecture was selected as the software architecture for the SSTI vehicle. The following sections give a brief background on the 4D/RCS Architecture and the SSTI program.

4D/RCS Architecture

4D/RCS is a reference model architecture for conceptualizing, designing, engineering, integrating, and testing intelligent control systems software for cognitive systems in real-world environments. It is especially useful in the development of vehicle systems with any degree of autonomy, from manually operated to fully autonomous. The 4D/RCS architecture consists of a multi-resolution hierarchy of feedback control loops between sensing and acting that integrate reactive behavior with perception, cognition, world modeling, decision-making, and planning, and forming a hybrid deliberative/reactive system [1].

A 4D/RCS based architecture contains a hierarchy of intelligent control nodes where each has a well defined role. Each node operates on a specific time horizon that is appropriate to the level, or echelon, in which the node exists. Within the vehicle, there are 4 echelons that exist: Vehicle, Subsystem, Primitive, and Servo. These levels are shown in Figure 1.

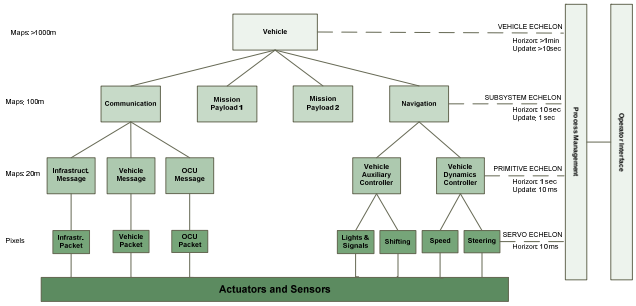

Figure 1. 4D/RCS Echelons

Each node acquires sensory input from nodes at a lower level in the architecture, or directly from hardware interfaces, and performs Sensory Processing (SP) on that data. As a result, the SP module updates the World Model (WM) for the node, which along with the Knowledge Database (KD) and Value Judgment (VJ), are used by the Behavior Generation (BG) module to create appropriate actions. The actions generated by the BG module are dependent on the location of the node in the hierarchy. For example, the actions created at the Servo level of the hierarchy will likely correspond to motor commands. The actions created at the Navigation level will likely correspond to a route or path generation. The internal structure of a generic 4D/RCS node [2] is shown in Figure 2.

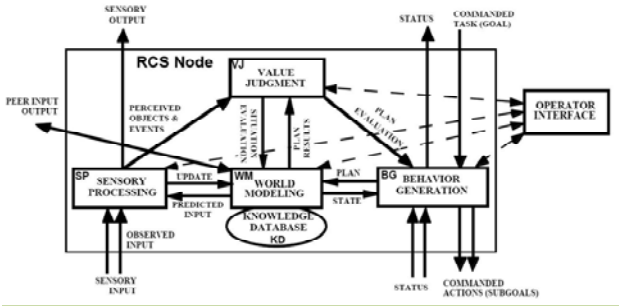

Figure 2. Internal Structure of a 4D/RCS Node

Each node provides processed sensory data and also status information to the nodes above it in the hierarchy. Each node also provides the generated behaviors and commanded actions to the nodes below it in the hierarchy. Also note that each node shares information with other nodes at the same level of the hierarchy. This creates a system of nodes with well-defined interfaces that when combined, can provide both a reactive and a deliberative control loop.

The Southwest Safe Transport Initiative

The SSTI program is a Southwest Research Institute (SwRI) internally-funded research project focusing on advancing active safety and autonomous vehicle technologies. The goal of SSTI is to integrate commercially available sensors, software, and algorithms into an autonomous vehicle. The program has integrated commercially available equipment into a 2006 Ford Explorer shown in Figure 3.

Figure 3. The SSTI 2006 Ford Explorer

The following components have been integrated into the base vehicle platform to date:

- Ibeo Laser Scanners and Fusion System
- Oxford RT3052 Global Positioning System (GPS) and Inertial Navigation System (INS)
- Electronic Mobility Controls (EMC) AEVIT Drive-by-Wire System
- High Resolution Prosilica EC1350C Camera
- Intel Core 2 Duo Blade Cluster – ExtremeNode EN-8740 by PCW MicroSystems
- Technocom 5.9 GHz Dedicated Short Range Communication (DSRC) radio
- dSpace Autobox with MatLab Simulink/CarSim

The SSTI vehicle can follow predefined routes autonomously while performing static and dynamic obstacle recognition and avoidance. The vehicle's route planning algorithms utilize Route Network Definition File (RNDF) files similar to the ones used in the Defense Advanced Research Projects Agency (DARPA) Urban Challenge. An operator can simply provide a destination within the RNDF and the vehicle will safely travel there autonomously.

The vehicle can also communicate with an infrastructure and other vehicles using a DSRC based Extra-Vehicle Communications System (EVCS). The EVCS components are able to communicate using either the Society of Automotive Engineers (SAE) J2735 and or the SAE AS-4 Joint Architecture for Unmanned Systems (JAUS) messages. This allows the vehicle to utilize information from external systems as well as its own sensors to be more knowledgeable of the surrounding environment.

IMPLEMENTATION OF 4D/RCS ARCHITECTURE

While development on the SSTI vehicle is still in progress, the development team has implemented a preliminary version of the Servo, Primitive, Subsystem, and Vehicle echelons of the 4D/RCS architecture. These echelons are divided even further into nodes like the ones described above. Each node accepts sensor data, commands from higher-level nodes, and status from lower-level nodes. The responsibility of each node is to enact a certain behavior that is dependent upon this information that it receives. The following sections describe the application of the 4D/RCS architecture on the SSTI vehicle, including the various echelons, the nodes contained in each echelon, and the behaviors they enact.

Description of the Architecture Hierarchy

The Servo echelon is provided by the EMC drive-by-wire system. This echelon performs the closed-loop control of the servo motors and other actuators. It accepts commands

to control the position of the steering motor, the position of the throttle/brake motor, the position of the transmission (i.e. Park, Reverse, Neutral, Drive), and states of other auxiliary components like the windshield wipers and turn signals. Since this system was purchased off-the-shelf, it is treated like a black box by the rest of the system.

The Primitive echelon is implemented using a combination of hardware and software to interface with the GPS sensor and the drive-by-wire system. A dSpace Autobox serves as the heart of the Primitive echelon and provides the low-level command and control interface to send steering and throttle/brake position commands to the drive-by-wire system. It accepts local path and speed commands from the higher echelons and ensures the vehicle maneuvers on the path at the desired speed. The dSpace operating system provides a real-time, deterministic platform to monitor vehicle dynamics and provide immediate stability control feedback.

The Subsystem Echelon is implemented on a blade server cluster using software components developed using Real Time Multisensor Advanced Prototyping Software (RTMaps). These custom components, along with RTMaps integrated components, provide an environment in which algorithms can more easily be developed and explored to optimize sensory processing and higher level vehicle commands. Custom components acquire and process data from the various sensors to develop a world model. This world model utilizes a situational awareness system to fuse the sensor data and generate vehicle trajectories. These trajectories are realized in near-real-time by sending lower level path segment commands to the dSpace Autobox to perform.

The Vehicle Echelon, also implemented on the blade cluster in the RTMaps environment, provides an interface for the user to choose a destination on a map or provide the vehicle with a mission by choosing an MDF file. It uses a priori map data through an RNDF file in addition to the user input to calculate a route from the vehicle's current position to the goal position. If the route is detected to be blocked by the on-board sensors (from construction or otherwise), this information can be passed up to the Vehicle echelon so it can dynamically recompute a new route. This route information, including any additional relevant map data (e.g. passing lanes, intersections, etc.), is passed to the lower echelons.

Hardware/Software Interfaces

The sensor processing modules perform preliminary processing on the acquired data to determine if immediate behaviors need to be implemented (e.g. for reflex actions). This pre-processed data is then provided to higher level situational awareness modules that further process the data to build the vehicle's World Model. The World Model is used by the behavior generation components to implement the vehicle's behaviors (e.g. route planning and following, obstacle avoidance). The primitive level of the behavior generation software interfaces with the EMC drive-by-wire system to control the vehicle while monitoring vehicle dynamics and maintaining the safety of the system. A diagram of the hardware and software interfaces of the system components is shown in Figure 4.

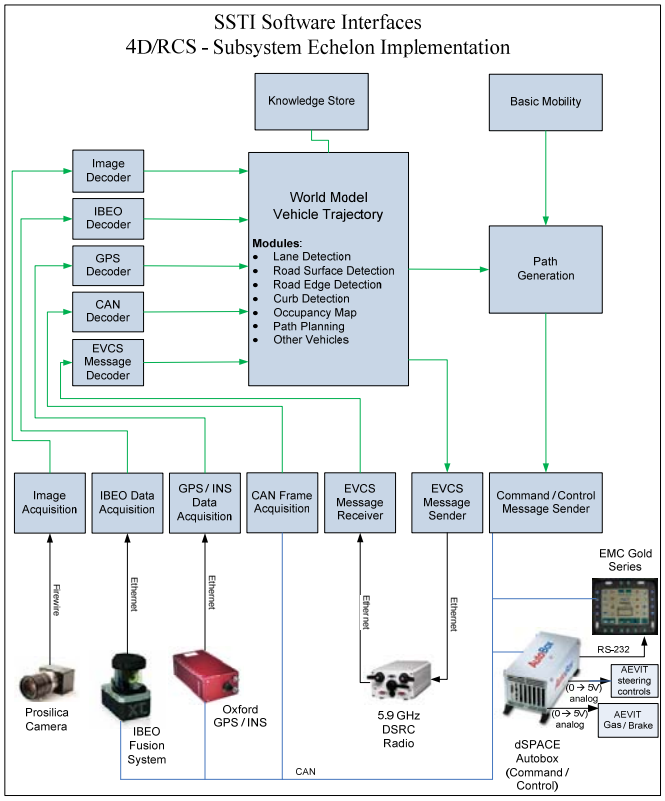

Figure 4. SSTI 4D/RCS Subsystem Echelon Implementation

Behavior Generation

The SSTI software architecture implements nodes that operate within the Vehicle, Subsystem, and Primitive 4D/RCS echelons. Each node generates specific behaviors according to its level within the behavior generation hierarchy. This behavior hierarchy, along with the corresponding 4D/RCS echelons and nodes are shown in Figure 5. The following section describes each one of the nodes in the software architecture. The section after that describes each one of the behaviors in detail.

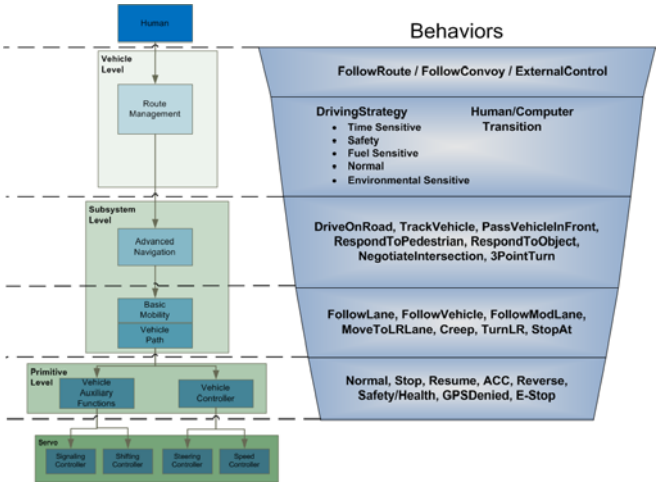

Figure 5. SSTI 4D/RCS Echelons, Nodes, and Behaviors

DESCRIPTION OF ARCHITECTURE NODES

Each echelon within the SSTI software architecture contains one or more nodes perform functions for autonomous navigation. These nodes include Route Management, Advanced Navigation, Basic Mobility, Vehicle Path, Vehicle Controller, and Vehicle Auxiliary.

Route Management

The Route Management Node is found at the Vehicle level of the software architecture. Its primary responsibility is to provide an interface to the user, allowing the user to choose a configuration and a destination for the vehicle. The user can be someone inside the vehicle or someone viewing the interface remotely. The configuration means that the vehicle could be a member of a convoy of vehicles. In the normal operating configuration, this node will calculate the entire route based on priori information (data from an RNDF file, an online mapping program, or an on-board navigation database). When the user selects the convoying configuration, this node will determine its position in the convoy and create the route based on the track of the vehicle directly ahead of it in the convoy. This node also receives user commands to switch from one configuration to another. Once the route data is determined, it will then be sent to the Advanced Navigation Node.

Advanced Navigation

The Advanced Navigation Node is found at the Subsystem level of the software architecture. It receives commands (the route and configuration) from the Route Management Node. It uses this route data along with high-level object data to generate high-level behaviors that will instruct the vehicle to safely continue along the route while responding to vehicles, pedestrians, and other objects. When in the convoying configuration, it receives formation commands such as desired following distance and lateral offsets from the path. This node also ensures that the route will be continuous when switching between the convoying configuration and normal configuration. The commands generated by this node will be sent to the Basic Mobility Node.

Basic Mobility

The Basic Mobility Node is found at the Subsystem level of the software architecture. It receives commands from the Advanced Navigation Node, which includes information about the route. It uses data about the route along with sensor information specific to vehicles, objects, and the road around it. It generates behaviors that will allow the vehicle to drive in a specific lane, change lanes, make turns, go around objects, follow other vehicles, and stop at stop signs. The commands generated from these behaviors will be sent to the Vehicle Path Node.

Vehicle Path

The Vehicle Path Node is found at the Subsystem level of the software architecture. It receives commands from the Basic Mobility Node, which includes information about the near-term desired route and speed the vehicle should travel. It creates the path segments based on this information. It sends these path segment commands to the Vehicle Controller Node. It also generates desired states of the vehicle's auxiliary functions such as the turn signals and shifter. It sends these commands to the Vehicle Auxiliary Functions Node.

Vehicle Controller

The Vehicle Controller Node is found at the Primitive level of the software architecture. It receives commands from the Vehicle Path Node, which contains information about the desired near-term path and speed profile of the vehicle. It uses this information to generate desired steering and speed commands at each point along the path. This node also uses vehicle and sensor state information to perform safety checks and generate safety behaviors. It sends the steering information to the Steering Controller and the speed information to the Speed Controller.

Vehicle Auxiliary

The Vehicle Auxiliary Node is found at the Primitive level of the software architecture. It receives commands from the Vehicle Path Node, which contains information about the desired state of the auxiliary functions of the vehicle such as the turn signals and the shifter. It then sends commands to the Shifting Controller and Signal Controller.

DESCRIPTION OF BEHAVIORS

Each node described above receives information at various levels of the World Model and generates a set of behaviors to enact based on that information. Behaviors at the higher levels contain behavior subsets. For example, when the TrackVehicle behavior is generated at the Advanced Navigation node, then its behavior subset (or behaviors that can be generated by the Basic Mobility node) include FollowVehicle and FollowModLane. Behaviors that include lower level behaviors list those specific behaviors in the *Behavior Subset* sections.

FollowRoute

The The FollowRoute behavior is generated at the Route Management Node of the software architecture. The Route Management Node will switch to this behavior when the user selects a destination and commands the node to generate a route to that destination. When the FollowRoute behavior is selected, a command will be sent to the Advanced Navigation Node that contains the route information.

Behavior Subset: DriveOnRoad, PassVehicleInFront, RespondToPedestrian, RespondToObject, NegotiateIntersection, 3PointTurn

FollowConvoy

The FollowConvoy behavior is generated at the Route Management Node of the software architecture. The Route Management Node will switch to this behavior when the user places the vehicle in convoy mode. The node then decides the vehicle's placement in the convoy and which vehicle it needs to track. When the FollowConvoy behavior is selected, a command will be sent to the Advanced Navigation Node that contains the route information based on the tracked vehicle.

Behavior Subset: TrackVehicle, RespondToPedestrian, RespondToObject, NegotiateIntersection

DriveOnRoad

The DriveOnRoad behavior is generated at the Advanced Navigation Node of the software architecture. The Advanced Navigation Node will initially switch to this behavior when it receives a FollowRoute command in normal configuration from the Route Management Node. It will also switch to this behavior when there are no objects or pedestrians in the vehicle's route, when the vehicle is not trying to pass another vehicle, and when the vehicle is not at an intersection. Essentially, this behavior acts as the default behavior at the Advanced Navigation Node. When the DriveOnRoad behavior is selected, a command will be sent to the Basic Mobility Node that contains the route information.

Behavior Subset: FollowLane, FollowVehicle, MoveToRightLane, MoveToLeftLane

TrackVehicle

The TrackVehicle behavior is generated at the Advanced Navigation Node of the software architecture. The Advanced Navigation Node will only switch to this behavior when it receives a FollowConvoy command from the Route Management Node. In this behavior, the route will be generated based on the track of the vehicle ahead of it in the convoy. When the TrackVehicle behavior is selected, a command will be sent to the Basic Mobility Node that contains the route of the vehicle ahead of it.

Behavior Subset: FollowVehicle, FollowModLane

PassVehicleInFront

The PassVehicleInFront behavior is generated at the Advanced Navigation Node of the software architecture. This behavior is selected when another vehicle is detected in front, a passing lane exists to the left (or the right), the passing lane is clear, and the Advanced Navigation Node determines the difference in speeds warrants a pass. When the PassVehicleInFront behavior is selected, a command will be sent to the Basic Mobility Node that contains information about the route, the lanes, and the vehicle to pass.

Behavior Subset: FollowLane, FollowVehicle, MoveToRightLane, MoveToLeftLane

RespondToPedestrian

The RespondToPedestrian behavior is generated at the Advanced Navigation Node of the software architecture. This behavior is selected when a pedestrian is detected that is interfering with the vehicle route or may interfere with the route in the future. When the RespondToPedestrian behavior is selected, a command will be sent to the Basic Mobility Node that contains information about the route and the pedestrian(s) that may cause interference.

Behavior Subset: FollowLane, FollowModLane, StopAt

RespondToObject

The RespondToObject behavior is generated at the Advanced Navigation Node of the software architecture. This behavior is selected when an object is detected that is interfering with the vehicle route or may interfere with the route in the future. When the RespondToObject behavior is selected, a command will be sent to the Basic Mobility Node that contains information about the route and the object(s) that may cause interference. This behavior differs from the RespondToPedestrian behavior because there may be cases where it is acceptable to swerve around or even hit the object.

Behavior Subset: FollowLane, FollowModLane, StopAt

NegotiateIntersection

The NegotiateIntersection behavior is generated at the Advanced Navigation Node of the software architecture. This behavior is selected when the vehicle is approaching an intersection. When the NegotiateIntersection behavior is selected, a command will be sent to the Basic Mobility Node that contains information about the route and the intersection type.

Behavior Subset: FollowLane, FollowModLane, StopAt, FollowVehicle, TurnLeft, TurnRight

3PointTurn

The 3PointTurn behavior is generated at the Advanced Navigation Node of the software architecture. This behavior is selected when the road is blocked and the only way to reach the destination is to turn around and ahead the opposite direction on the same road. The vehicle will try to perform a U-Turn and will revert to a 3PointTurn when there is not enough space. When the 3PointTurn behavior is selected, a command will be sent to the Basic Mobility Node that contains information about the turn.

Behavior Subset: FollowModLane, StopAt

FollowLane

The FollowLane behavior is generated at the Basic Mobility Node of the software architecture. This behavior is selected for a variety of reasons depending on the state of the Advanced Navigation Node. In general, this behavior is selected when there are no vehicles, pedestrians, or other objects in front of the vehicle, and the vehicle is not at an intersection or changing lanes. Essentially, this behavior acts as the default behavior at the Basic Mobility Node. When the FollowLane behavior is selected, a command will be sent to the Vehicle Path Node that contains information about the route and the lane.

Behavior Subset: Normal, Reverse

FollowVehicle

The FollowVehicle behavior is generated at the Basic Mobility Node of the software architecture. This behavior is selected when another vehicle is traveling on the same route at a speed lower than the desired speed. When the FollowVehicle behavior is selected, a command will be sent to the Vehicle Path Node that contains information about the route, the lane, and the lead vehicle.

Behavior Subset: ACC

FollowModLane

The FollowModLane behavior is generated at the Basic Mobility Node of the software architecture. This behavior is selected when there is a reason to continue on the route, but not follow a lane precisely. Potential reasons include pedestrians or other objects in the vehicle's lane. When the FollowModLane behavior is selected, a command will be sent to the Vehicle Path Node that contains information about the route and the modified lane.

Behavior Subset: Normal, Reverse

MoveToLRLane

The MoveToLRLane behavior is generated at the Basic Mobility Node of the software architecture. This behavior is selected when the route requires a lane change, or when a vehicle is being passed and a lane change is needed. The lane is required to be clear. When the MoveToLRLane behavior is selected, a command will be sent to the Vehicle Path Node that contains information about the route and the lane to move to.

Behavior Subset: Normal

TurnLR

The TurnLR behavior is generated at the Basic Mobility Node of the software architecture. This behavior is selected when a left or right turn is being made at an intersection. When the TurnLR behavior is selected, a command will be sent to the Vehicle Path Node that contains information about the route, and a command will be sent to the Vehicle Auxiliary Node that contains information about the turn signals.

Behavior Subset: Normal, Stop, Resume

StopAt

The StopAt behavior is generated at the Basic Mobility Node of the software architecture. This behavior is selected for a variety of reasons depending on the state of the Advanced Navigation Node. If the vehicle is negotiating an intersection, then StopAt is selected when the vehicle is at a stop sign or stop light or the vehicle is making a left turn and another vehicle is in the way. If the vehicle is responding to a pedestrian or object, then StopAt is selected when the vehicle needs to stop and wait for the pedestrian or object to move from the route. When the StopAt behavior is selected, a command will be sent to the Vehicle Path Node that contains information about the route, the lane, the speed, and the stopping point.

Behavior Subset: Normal, Stop

BEHAVIOR TRANSITIONS

The SSTI vehicle implements complex behaviors by combining lower level behaviors to implement mission objectives. For example, following a route combines behaviors that involve avoiding static and dynamic obstacles, negotiating intersections, and parking, all while obeying traffic laws and implementing the specific driving strategy of the vehicle. A diagram illustrating the transition between Advanced Navigation behaviors to implement the FollowRoute behavior is shown in Figure 6.

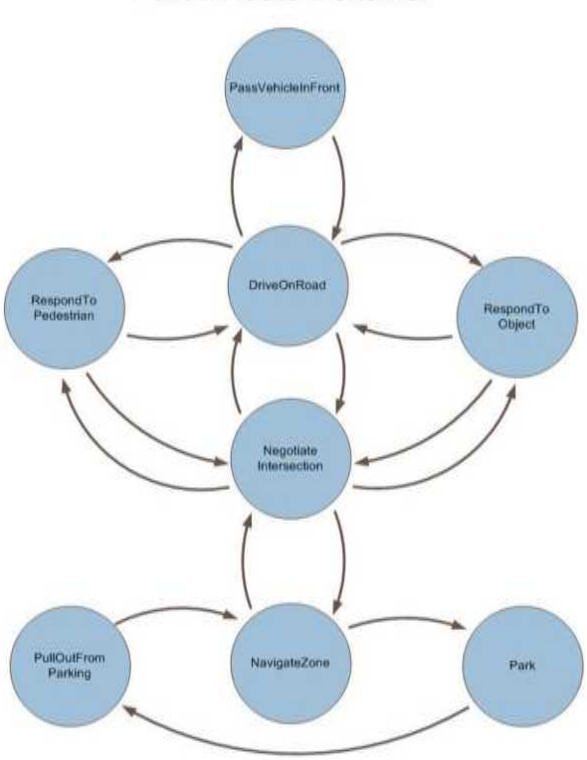

Figure 6. FollowRoute State Transition Diagram

OPERATIONAL CONCEPT

Figure 7 shows an operational concept of the SSTI vehicle in a convoying operation. This operational concept highlights some of the behavior switching that occurs in a UGV mission. In this operational concept, the UGV (the SSTI vehicle) starts out in the FollowConvoy behavior and is the last vehicle in a small three-vehicle convoy. While convoying, the SSTI vehicle is instructed to modify its formation by adjusting its target following distance. When approaching an urban area, an incident occurs that separates one of the vehicles from the convoy. After the incident, the SSTI vehicle is instructed to take the lead through the urban environment because it is deemed hazardous. The SSTI vehicle enters the FollowRoute behavior and leads the now two-vehicle convoy through the urban area, negotiating intersections and stopping for any pedestrians or animals crossing the street. The separated vehicle rejoins the UGV-led convoy after the urban environment has been successful navigated. When the convoy enters a safe area, the SSTI vehicle is instructed to take a follower position and slows down so a manned vehicle can re-take the lead. The SSTI vehicle then enters back into FollowConvoy mode for the rest of the mission.

Figure 7. Operational Concept of the Behavior Switching of the SSTI Vehicle

LESSONS LEARNED

There still remains a considerable amount of development and testing of the SSTI vehicle software, but the development effort has already shed light on a number of considerations that are important in the design and implementation of a 4D/RCS based software architecture. These include:

1. It was important to have well-defined interfaces between the RCS nodes to allow parallel algorithm development within separate echelons.

2. The nodes should be as loosely coupled as possible. This was sometimes challenging in a 4D/RCS based architecture as each node receives status input from subordinate nodes as well as sends commands to them.

3. It was good to error on the side of passing too much information between the nodes initially and tune for performance later.

4. A graphical development environment such as RTMaps and MATLAB/Simulink helped when implementing a 4D/RCS based architecture as the nodes and echelons can be visually represented.

CONCLUSION

In conclusion, the use of proven, real-time control system architectures such as 4D/RCS helps in the process of creating a vehicle that can act deliberatively as well as reactively to environmental conditions. The hierarchical

nature of 4D/RCS provides a system of nodes in which each node operates within a time horizon appropriate for the echelon in which it exists. This system of nodes creates a vehicle system that can perform complex behaviors as well as safety-critical behaviors in a timely fashion. Thus, creating a vehicle that can act, or assist a human in acting, in a safer and more efficient manner may save time and lives.

ACKNOWLEDGMENTS

The authors wish to acknowledge the technical contributions from the following SSTI project team members: Steve Dellenback, Ryan Lamm, Roger Lopez, Dan Pomerening, Joe Steiber, Paul Avery, Kevin Alley, Chris Mentzer, Kris Kozak, Bapi Surampudi, and Josh Curtis. This research was funded under Southwest Research Institute IR&D project 10-R9648.

REFERENCES

[1] R. Madhavan E. Messina, and J. Albus, "Intelligent Vehicle Systems: A 4D/RCS Approach", Nova Science Publishers, January 15, 2007.

[2] J. Albus, "4D/RCS: A Reference Model Architecture for Intelligent Ground Vehicles", Proceedings of the SPIE 16th Annual International Symposium on Aerospace/Defense Sensing Simulation, and Controls, Orlando, FL, April 1-5, 2002.