CDS 270-2: Lecture 1-2
Case Study: Alice

Richard M. Murray

29 March 2006

Goals:

- Provide detailed overview of a a model networked control system
- Introduce NCS features to be addressed in upcoming lectures

Reading:

- "Alice: An Information-Rich Autonomous Vehicle for High-Speed Desert Navigation", Cremean et al. *Journal of Field Robotics*, 2005 (submitted)
- http://gc.caltech.edu/wiki - online documentation for Alice
  - Alice Documentation - primary page for documentation links
  - 2005 SURF - project reports for individual components

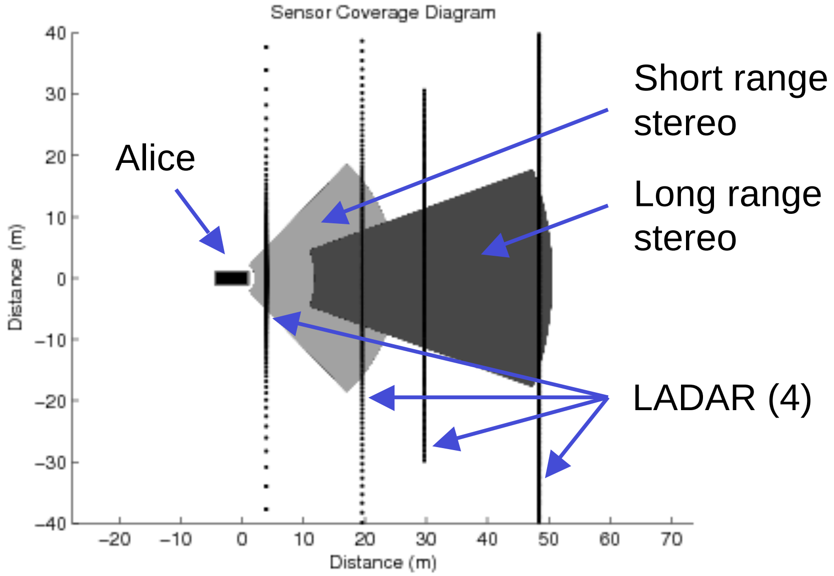
Alice Overview

Team Caltech

- 50 students worked on Alice over 1 year
- Course credit through CS/EE/ME 75
- Summer team: 20 SURF students + 6 graduated seniors + 4 work study + 4 grads + 2 faculty + 6 volunteers (= ~40)

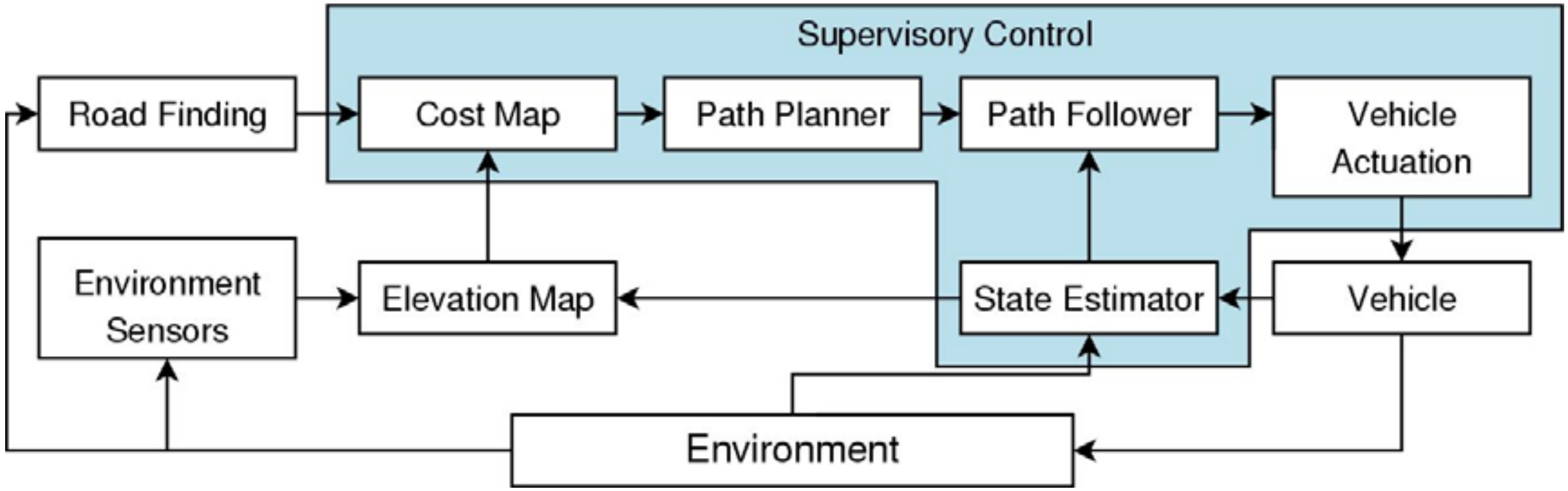
Alice

- 2005 Ford E-350 Van
- Sportsmobile 4x4 offroad package
- 5 cameras: 2 stereo pairs + roadfinding
- 5 LADARs: long, med*2, short, bumper
- 2 GPS units + 1 IMU (LN 200)



- 4 seats w/ computer workstations

Alice's Architecture



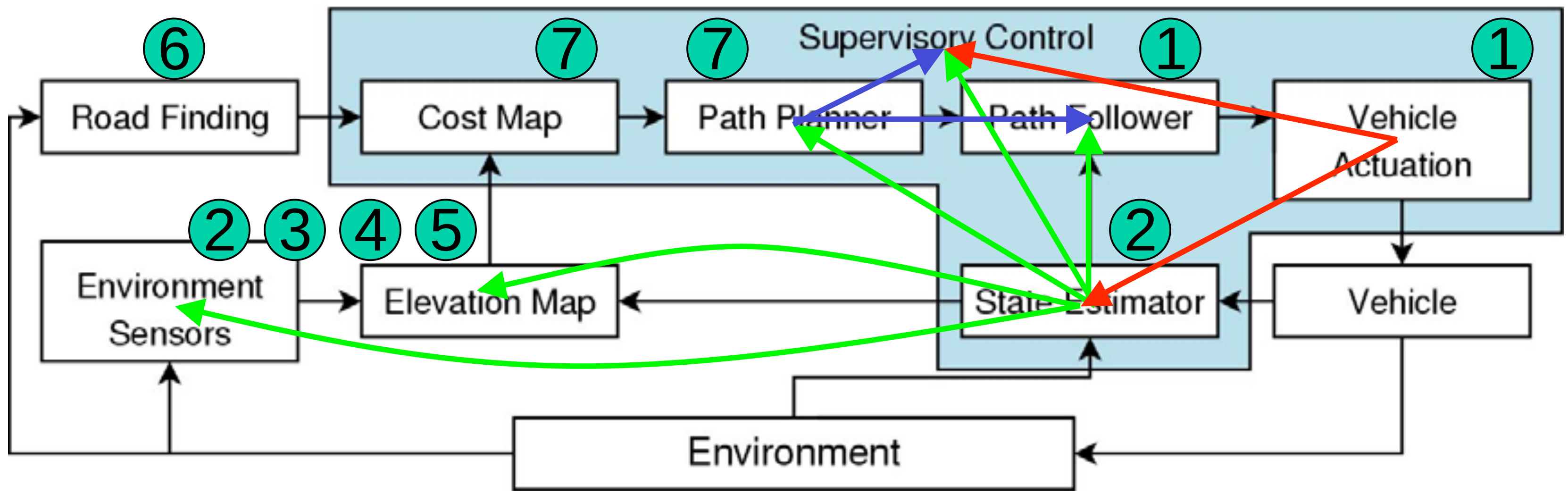
Computing

- 6 Dell 750 PowerEdge Servers (P4, 3GHz)
- 1 IBM Quad Core AMD64 (fast!)
- 1 Gb/s switched ethernet

Software

- 15 individual programs with ~50 threads of execution
- FusionMapper: integrate all sensor data into a speed map for planning
- PlannerModule: optimization-based planning over a 10-20 second horizon

Communication Management: Spread



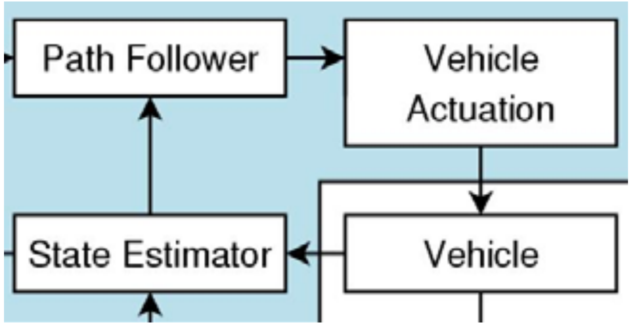
Modular architecture

- Each block represents one or more processes (programs) communicating via network (packets)
- Processes linked to specific hardware run on dedicated computers; otherwise can run on any computer
- Each process can have multiple threads of execution (multi-tasking)

Communication Groups

- Modules subscribe to "groups"; receive all messages to that group
- Multiple levels of reliability/causality: unreliable, guaranteed, causal
- Use individual "keys" to allow multiple users to avoid conflicts (especially useful for simuilations)
- Graphical user interface (GUI) subscribes to all messages

Path Follower/Actuation



Vehicle Actuation: adrive

- Accept actuation commands from control algorithm; command actuators
- Check proper vehicle operation; pause vehicle on error (and signal superCon)
- Broadcast actuator state

Trajectory Tracking: pathFollower

- Accept desired trajectory from planner
- Read vehicle state via broadcast
- PID controller to generate actuation commands
- Modes: normal, pause, reverse

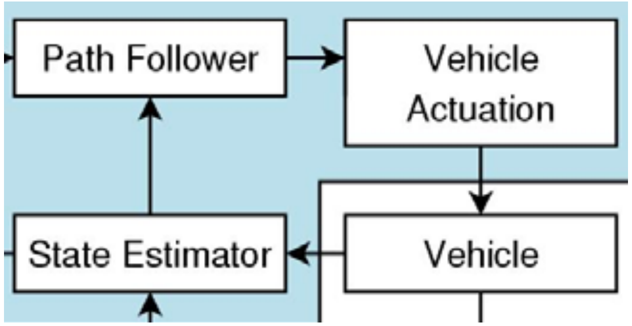
Adrive

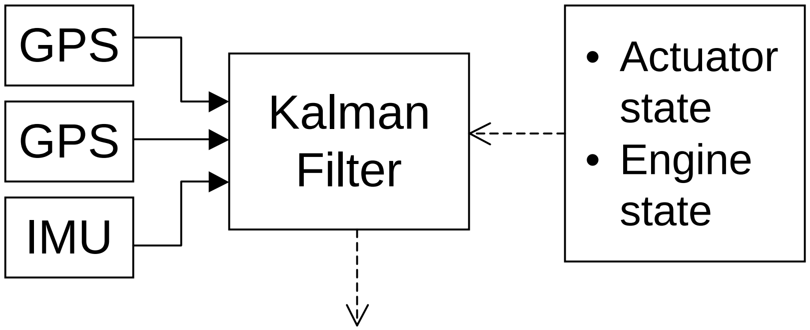
- HW: steering, throttle, brake, ignition, transmission, engine diagnostics - serial port interfaces
- In: normalized actuation commands, engine diagnostics (OBD II)
- Out: actuator values and engine state
- Independent threads for each actuator
- “Interlock” logic to ensure safety

PathFollower

- HW: none
- In: desired trajectory, mode (fwd/rev)
- Out: actuation commands
- PID controller, with trajectory storage and “reverse” capability

State Estimation





Vehicle position, orientation, velocities, accelerations

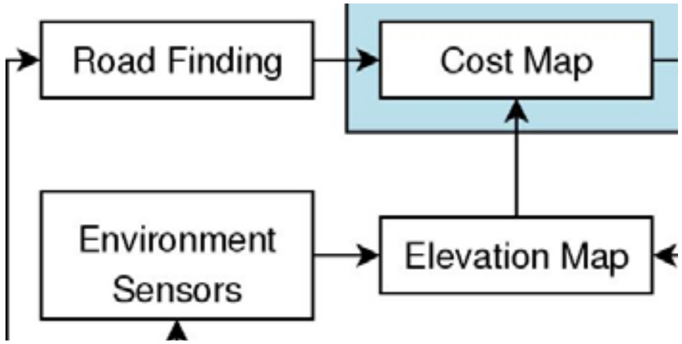
State estimation: astate

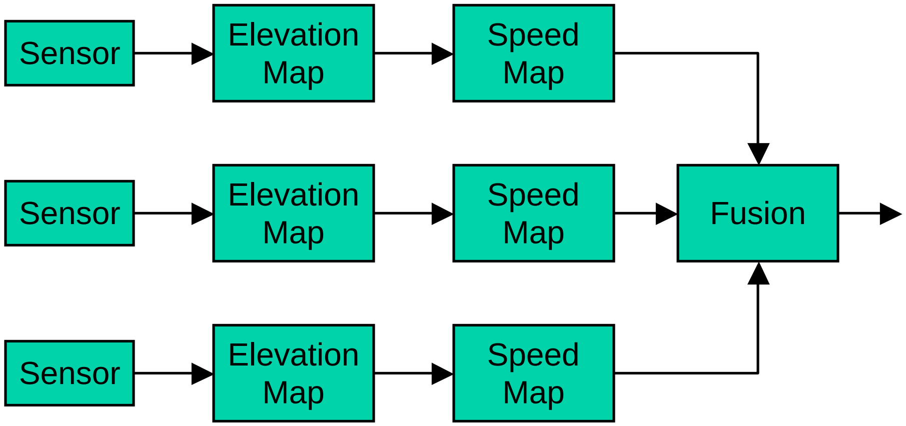
- Broadcast current vehicle state to all modules that require it (many)
- Timing of state signal is critical - use to calibrate sensor readings
- Quality of state estimate is critical: use to place terrain features in global map
- Issue: GPS jumps
  - Can get 20-100 cm jumps as satellites change positions
  - Maintain continuity of state at same time as insuring best accuracy

Astate

- HW: 2 GPS units (2-10 Hz update), 1 inertial measurement unit (gryo, accel @ 400 Hz)
- In: actuator commands, actuator values, engine state
- Out: time-tagged position, orientation, velocities, accelerations
- Use vehicle wheel speed + brake command/position to check if at rest

Terrain Estimation





Sensor processing

- Construct local elevation based on measurements and state estimate
- Compute speed based on gradients

Sensor fusion

- Combine individual speed maps
- Process "missing data" cells

Road finding

- Identify regions with road features
- Increase allowable speed along roads

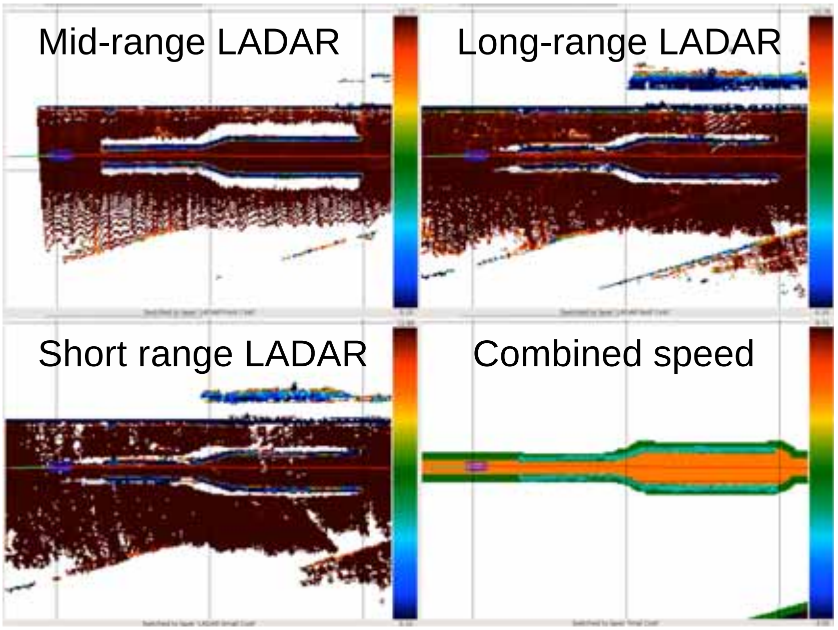
LadarFeeder, StereoFeeder

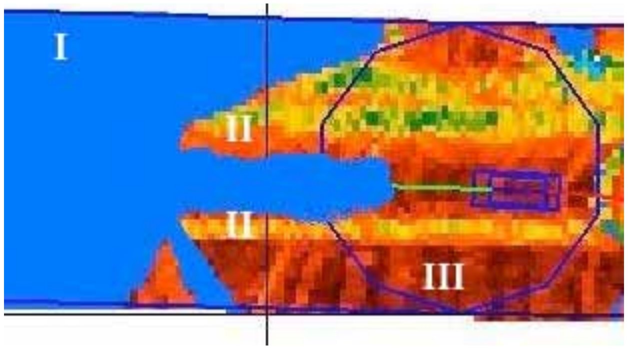
- HW: LADAR (serial), stereo (firewire)
- In: Vehicle state
- Out: Speed map (deltas)
- Multiple computers to maintain speed

FusionMapper

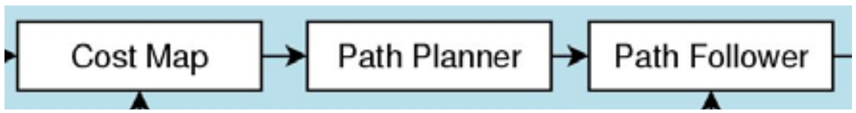
- In: Sensor speed maps (deltas)
- Output: fused speed map
- Run on quadcore AMD64

Sensor Fusion and Cost Map Processing





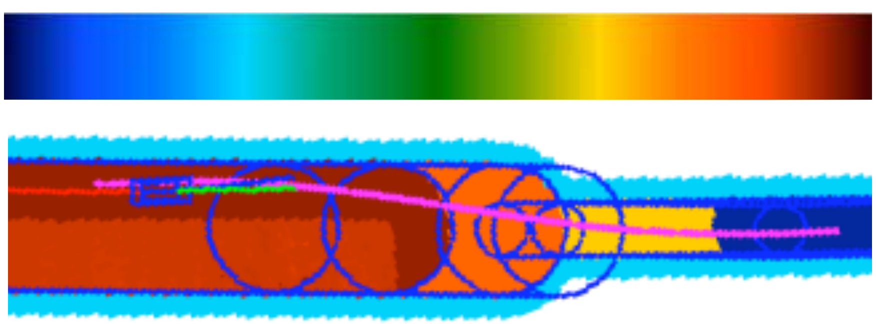
Path Planner



$$\arg\min \int_t^{t+T} L(x,u)d\tau + V(x(T))$$

$$\begin{aligned} \dot{x} &= f(x,u) \\ g(x,u) &\le 0 \end{aligned}$$

$$\begin{aligned} \dot{N} &= v\cos\theta \\ \dot{E} &= v\sin\theta \\ \dot{\theta} &= \frac{v}{L}\tan\phi \\ \dot{\phi} &= \omega = u_1 \\ \dot{v} &= a = u_2 \end{aligned} \qquad s.t. \quad \begin{aligned} \phi &\in [\phi_{min}, \phi_{max}] \\ \omega &\in [\omega_{min}, \omega_{max}] \\ v &\in (0, v_{max}] \\ a &\in [a_{min}, a_{max}] \end{aligned}$$



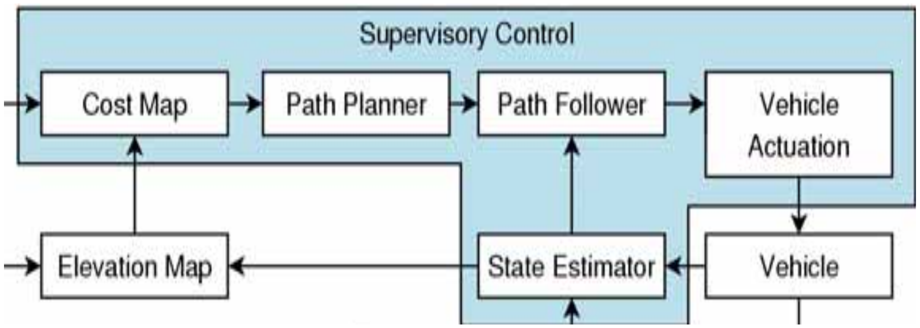
Trajectory Generation: plannerModule

- Use speed map to plan trajectory that maximizes distance traveled
- Two phase planner: first stage uses simple grid to seed optimization
- Exploit differential flatness for speed

PlannerModule

- HW: none
- In: speed maps, vehicle state
- Out: desired trajectory
- Algorithm runs on quadcore AMD64 at approx. 5 Hz

Supervisory Control

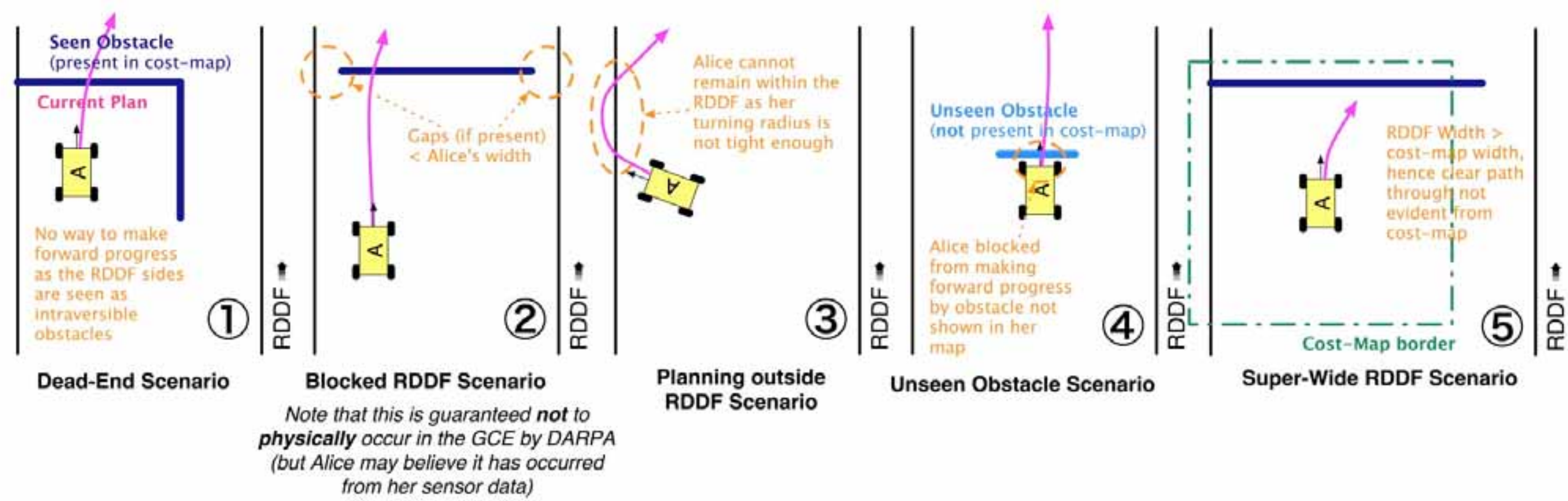


SuperCon

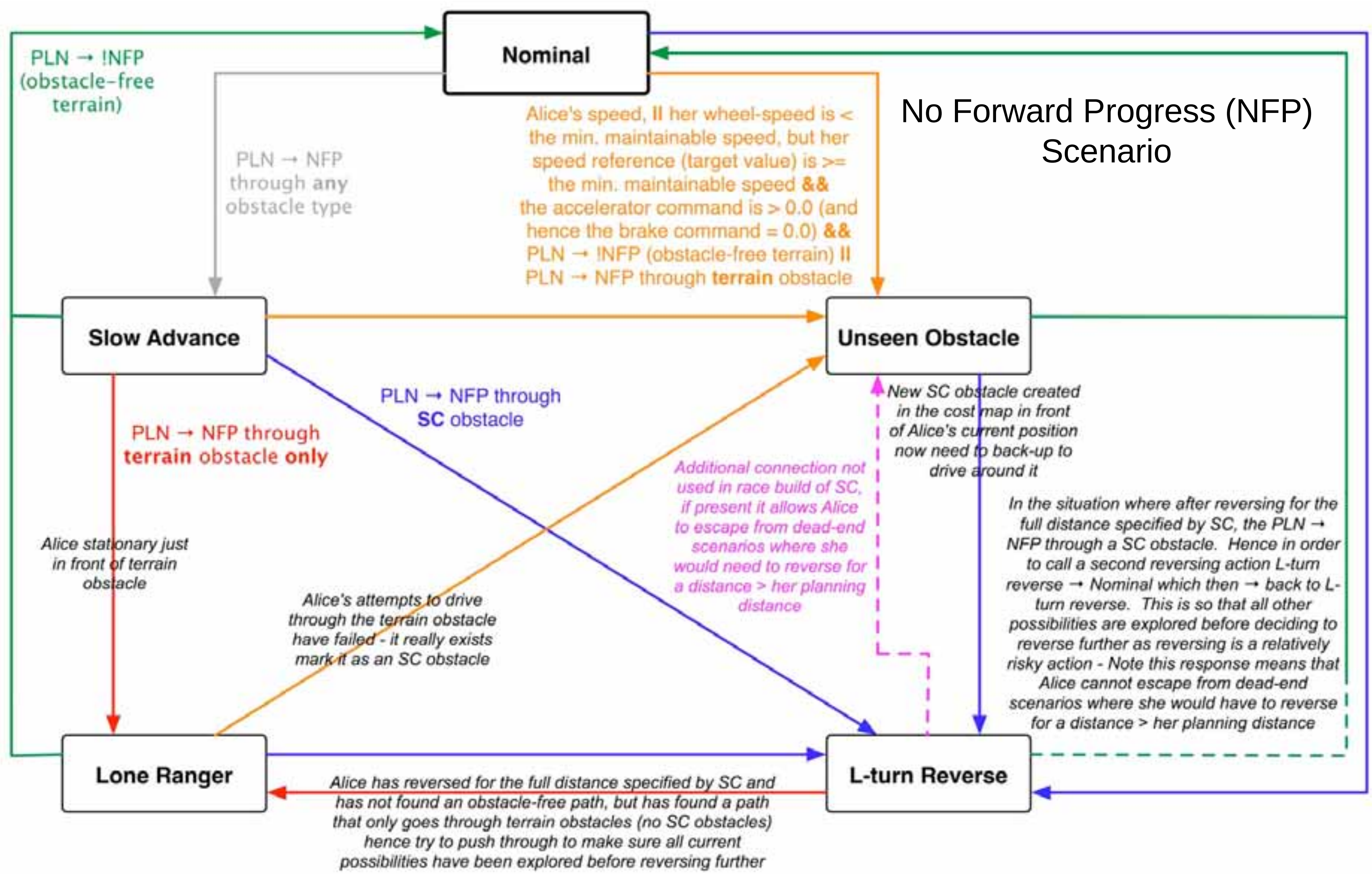
- Input: read all published information
- Output: targetted mode messages
- Reason about different situations and control operation of other modules based on current strategy
- Make heavy use of networked architecture, especially communication groups

Supervisory Control

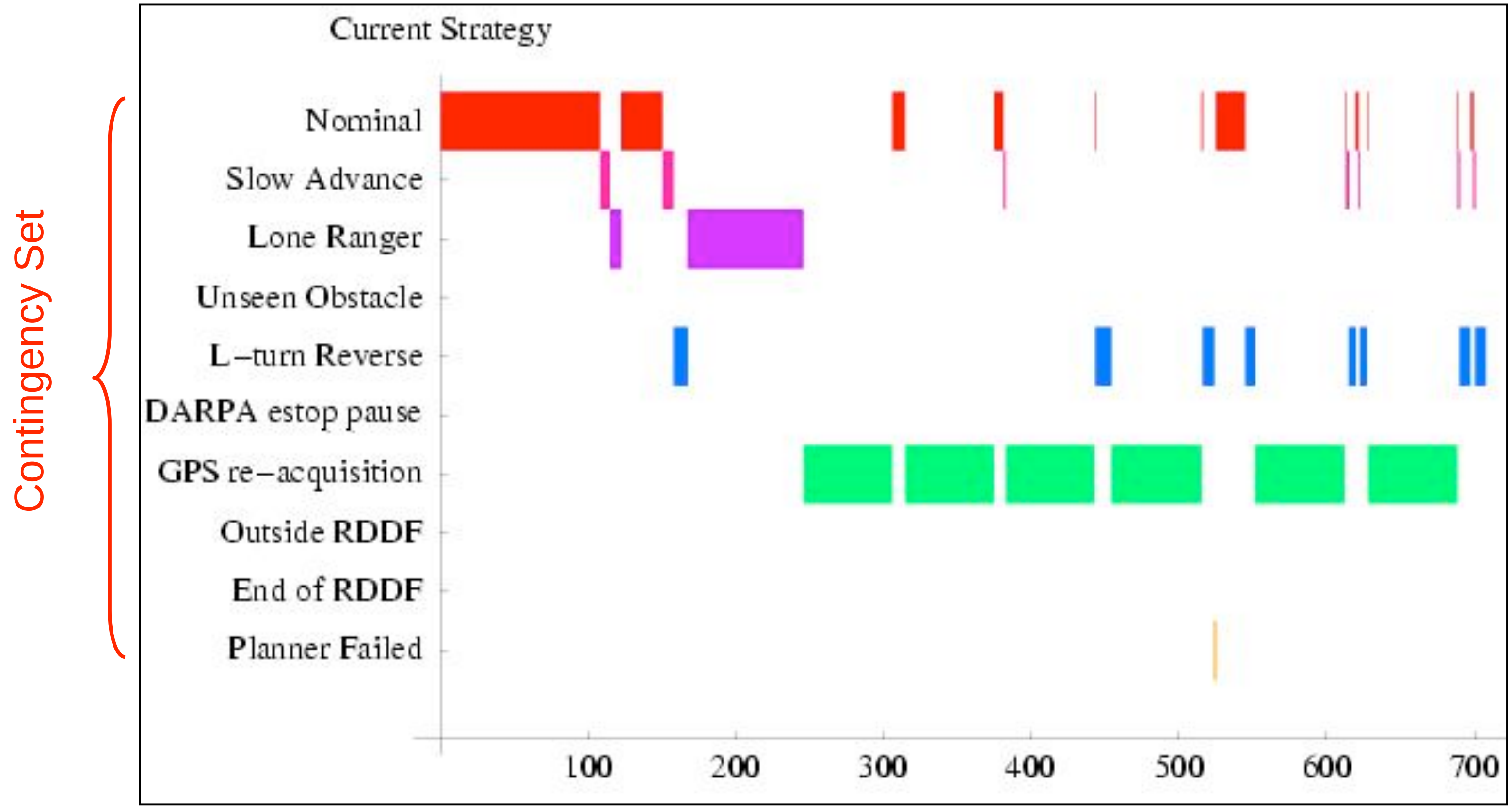
- Control operation of other modules
- Always maintain forward progress



SuperCon Logic



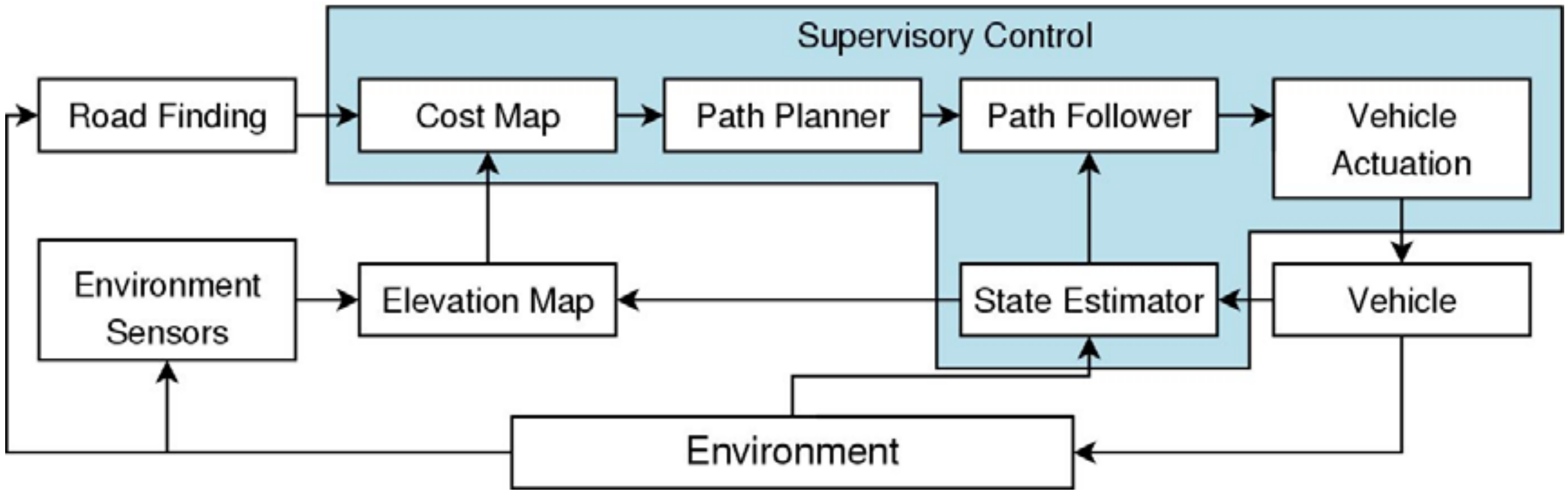
SuperCon Usage (NQE Run 1)



Heavy usage of superCon modes during “typical” operations

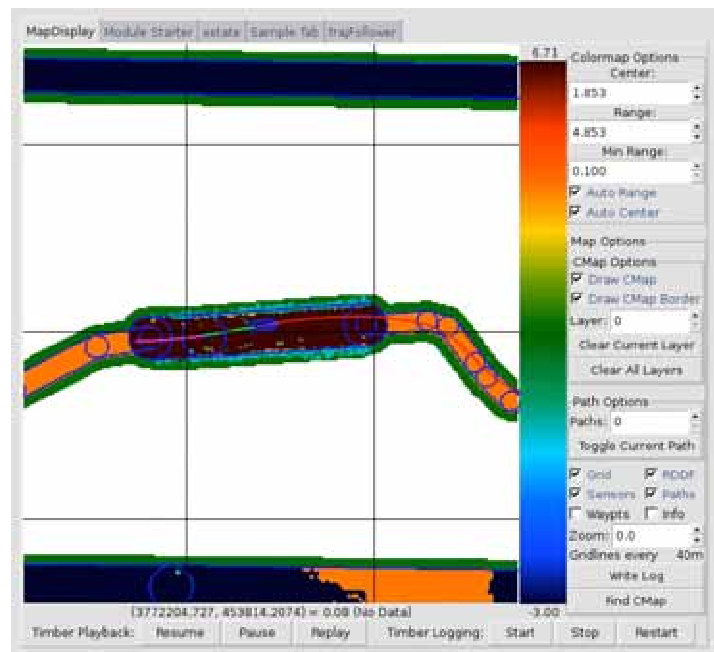
- Vehicle must be able to operate in “degraded” mode

Architecture Summary

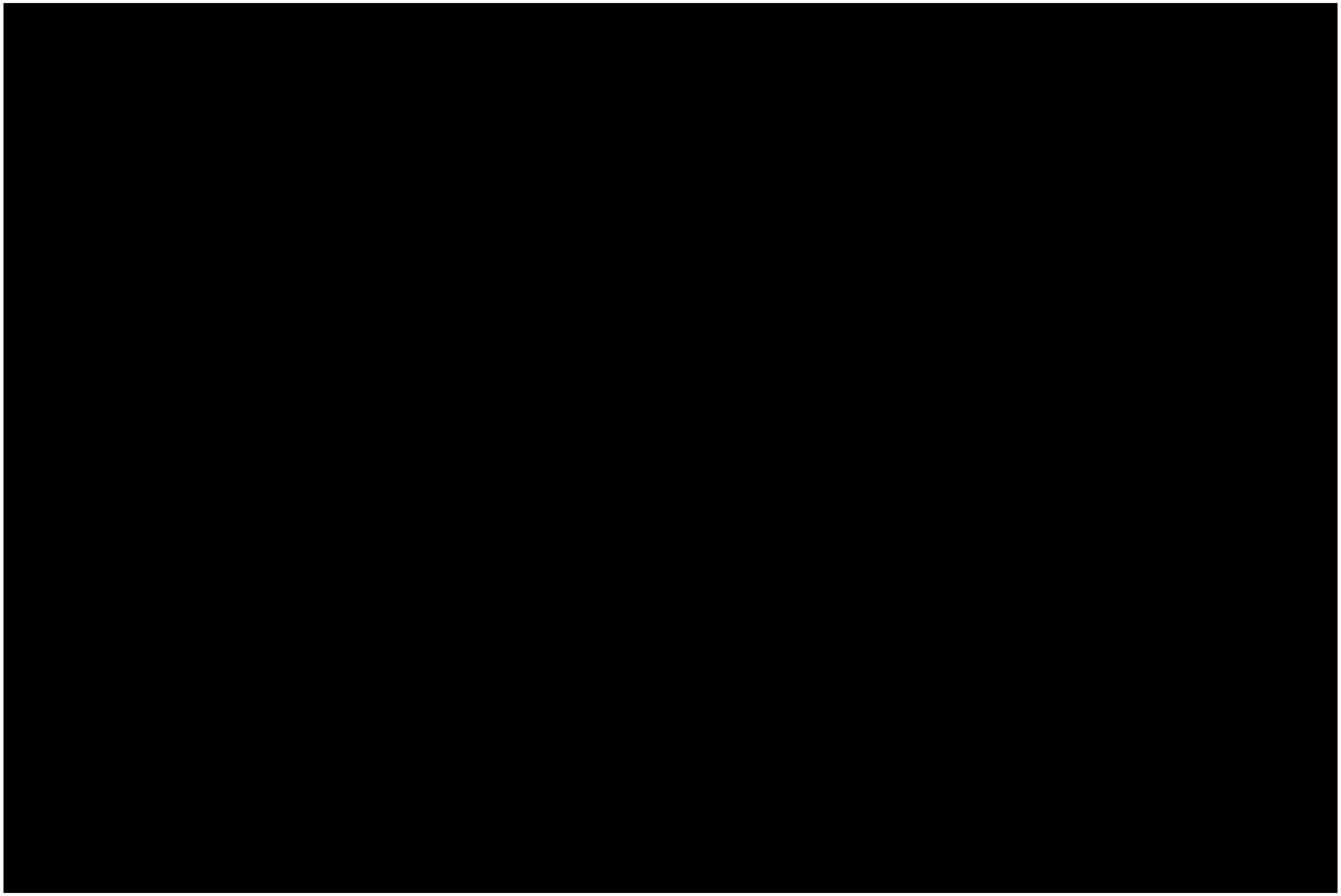


Additional modules/features

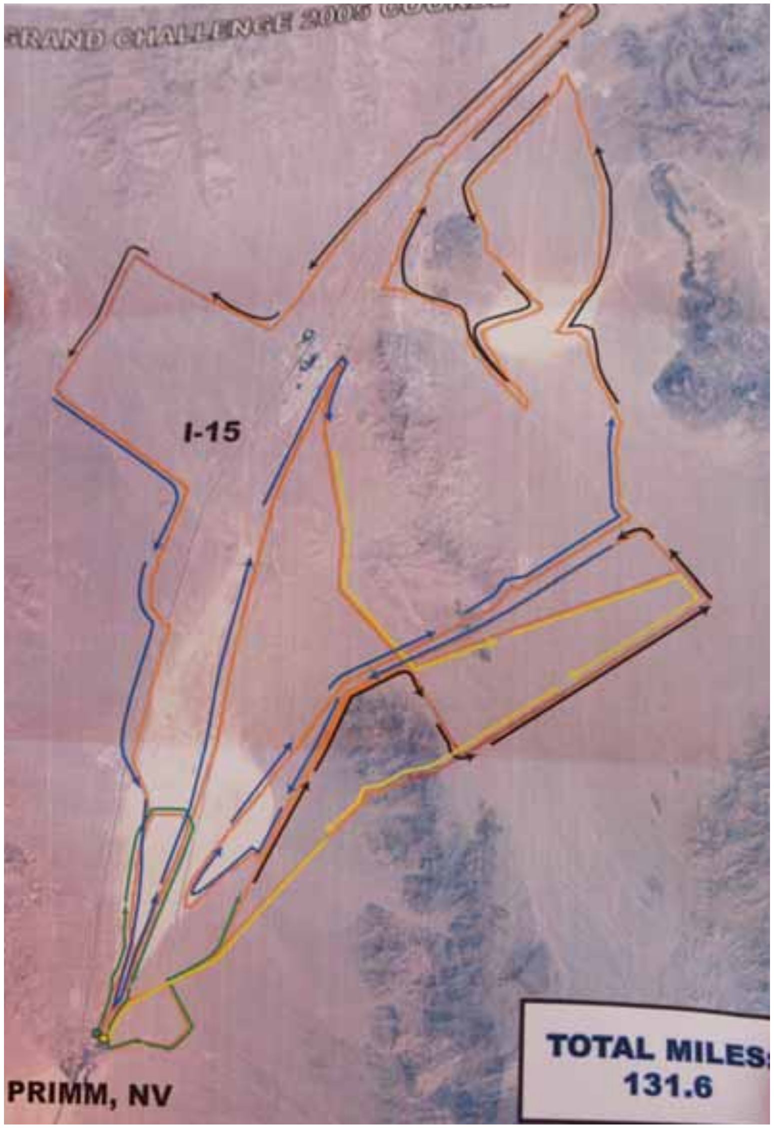
- GUI: show system states in real-time
- Sensor logging ("timber"): log and playback raw sensor data
- Network logging ("author, logplayer"): capture and playback all network traffic
- Simulator: read actuation commands and generate (simulated) state data
- Runlevels: automatically restart crashed modules



NQE Performance



Race Results



- WP1: 9:03a - begin vehicle motion
- WP 23: second intersection
- WP 58: small and roof have cut out
- WP 74, RDDF intersection (fork to right)
- WP 147, RDDF has narrowed to road width
- WP 156, cross intersection with future section of RDDF
- WP 171, begin approach to straight section

GUI View

Cremean et al
JFR 05 (s)

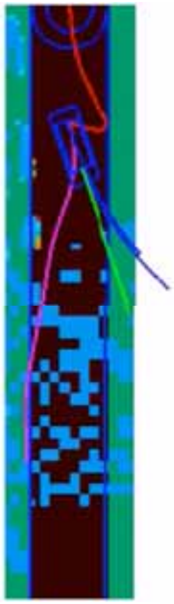
What Happened

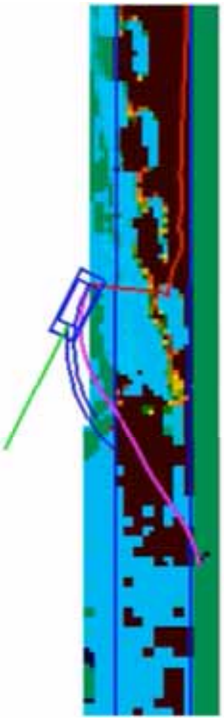
GPS signal lost under power lines

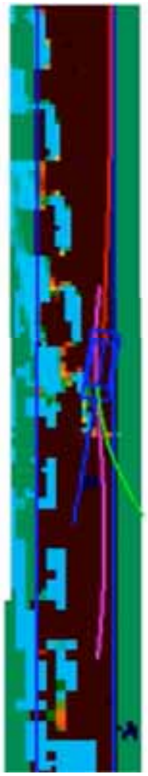
- Software recognized condition and stopped vehicle to allow position estimate to converge
- GPS receiver reacquired the signal, but with very high error estimates ⇒ slow convergence of state estimate
- Software confused slow convergence with convergence and began to move
- Alice headed down “corridor” that was lined up with barriers

Other factors

- Midrange LADAR units stopped working ⇒ relied on long (35m) and short (3m) units







Alice Off the Road

Networked Control Systems