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

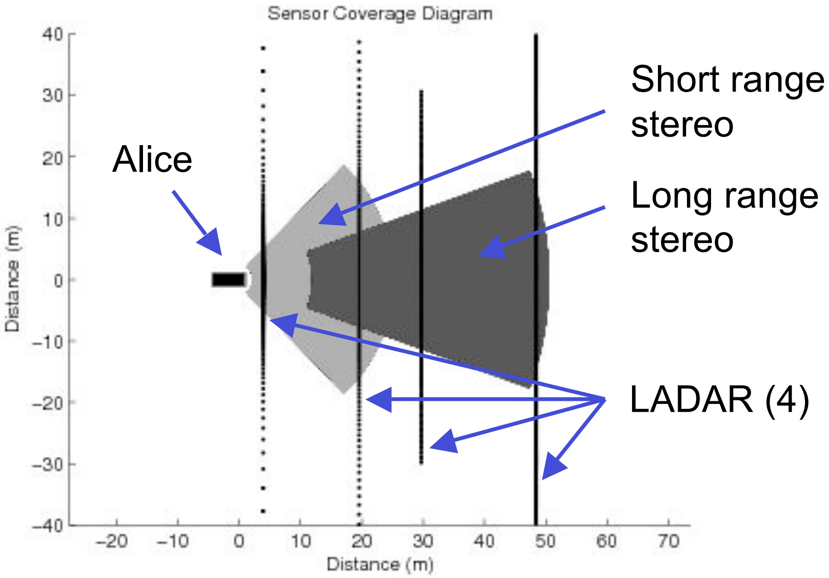
Alice Overview

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



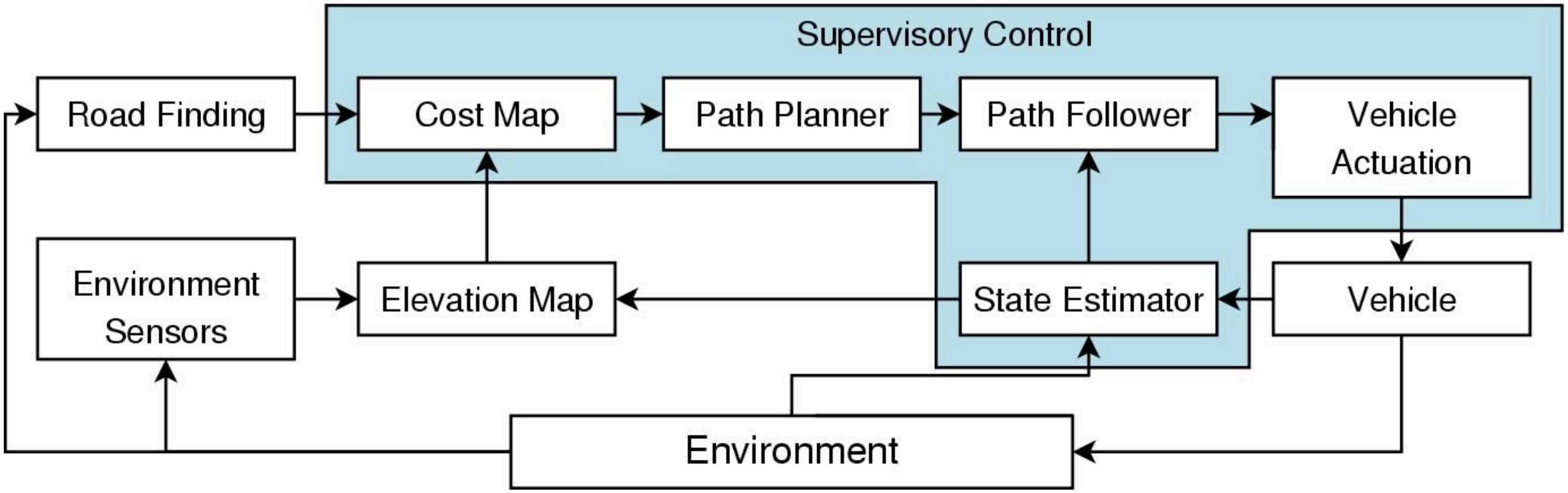
Alice

Short range stereo

Long range stereo

LADAR (4)

Alice’s Architecture



Supervisory Control

Road Finding

Cost Map

Path Planner

Path Follower

Vehicle Actuation

Environment Sensors

Elevation Map

State Estimator

Vehicle

Environment

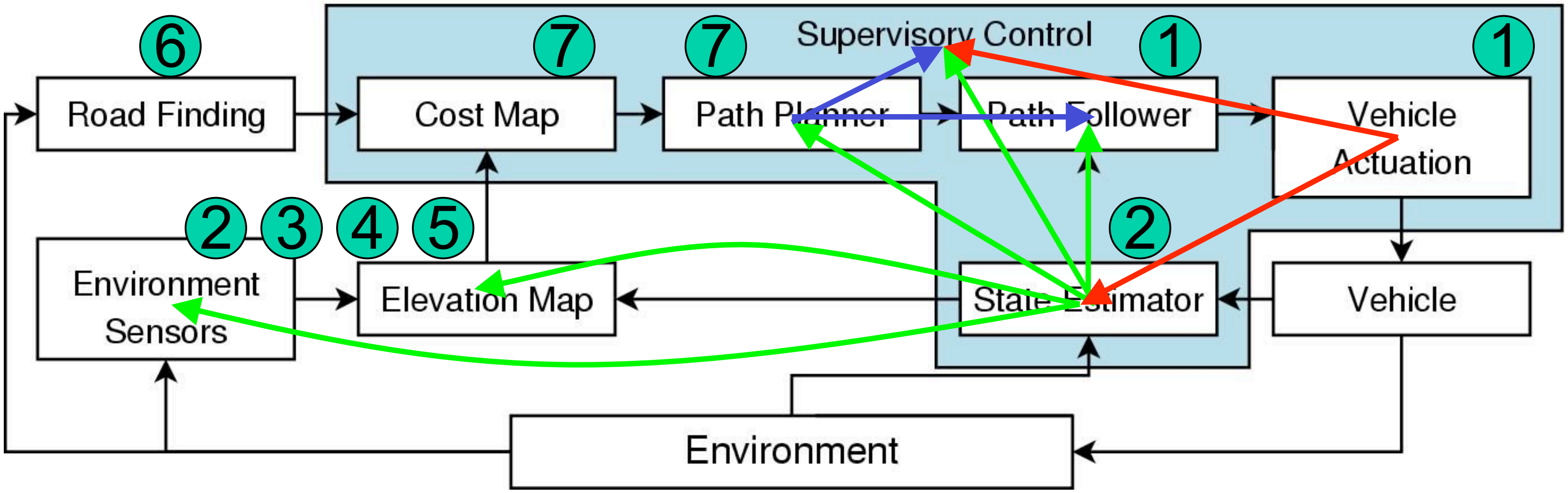
Computing

- 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



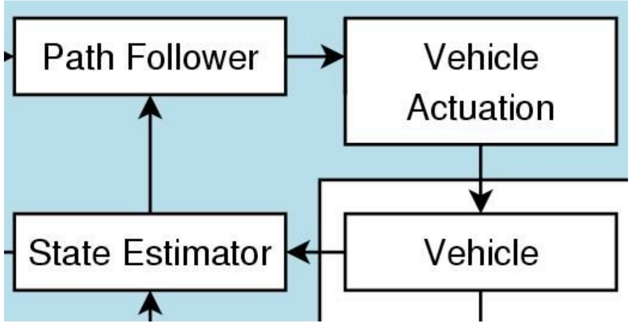
Modular architecture

- 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

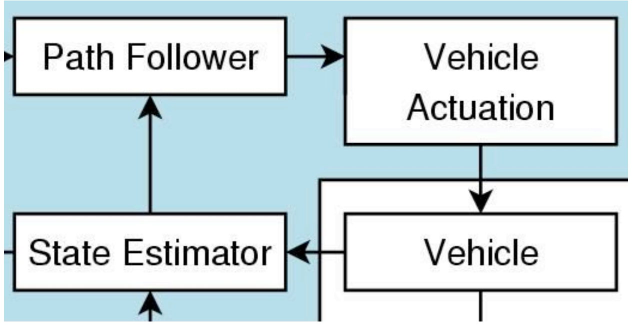
Adrive

- 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

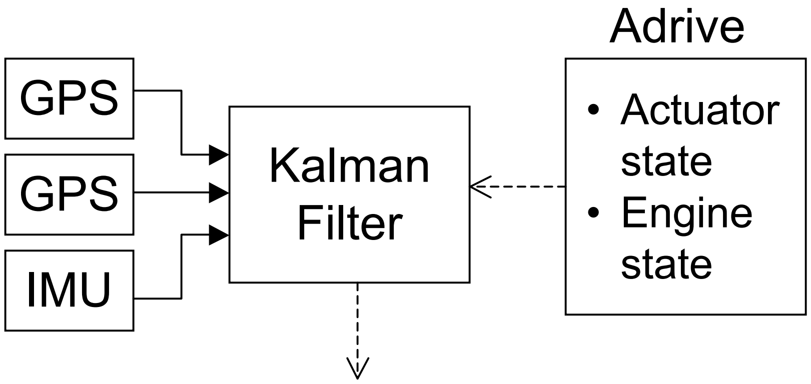
- HW: none
- In: desired trajectory, mode (fwd/rev)
- Out: actuation commands
- PID controller, with trajectory storage and "reverse" capability

State Estimation



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

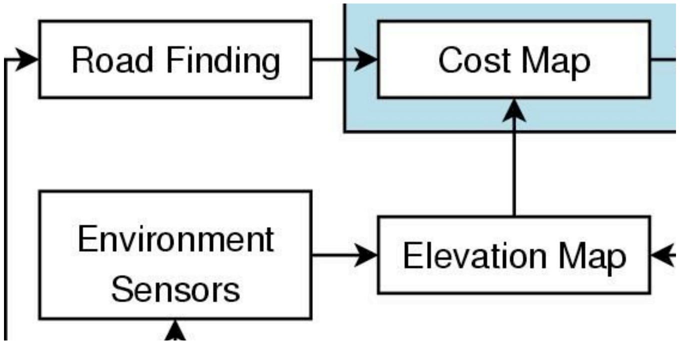


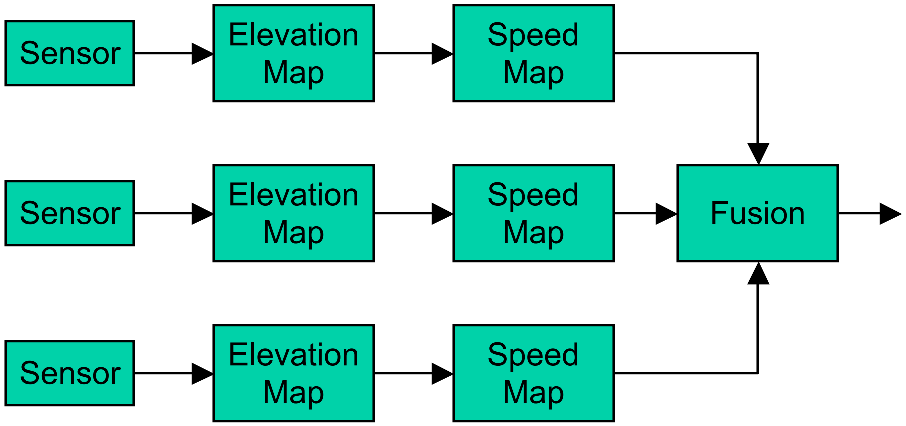
Vehicle position, orientation, velocities, accelerations

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

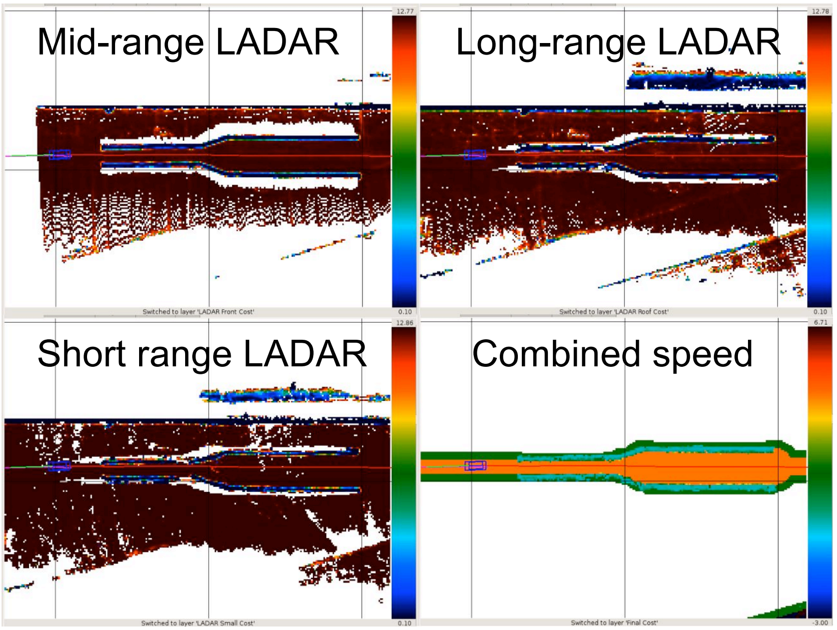
LadarFeeder, StereoFeeder

- 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

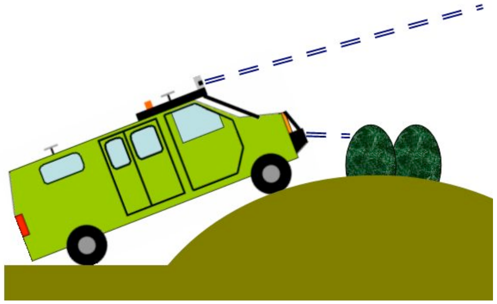

Mid-range LADAR

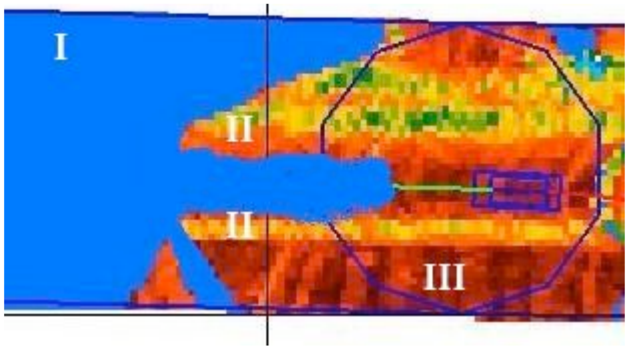
Long-range LADAR

Short range LADAR

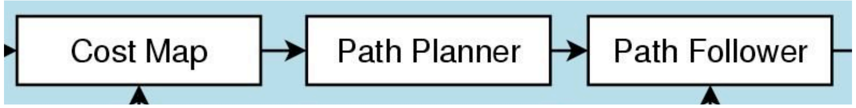
Combined speed

12.77
0.10
Switched to layer 'LADAR Front Cost'
12.78
0.10
Switched to layer 'LADAR Roof Cost'
12.86
0.10
Switched to layer 'LADAR Small Cost'
6.71
-3.00
Switched to layer 'Final Cost'





Path Planner



Cost Map

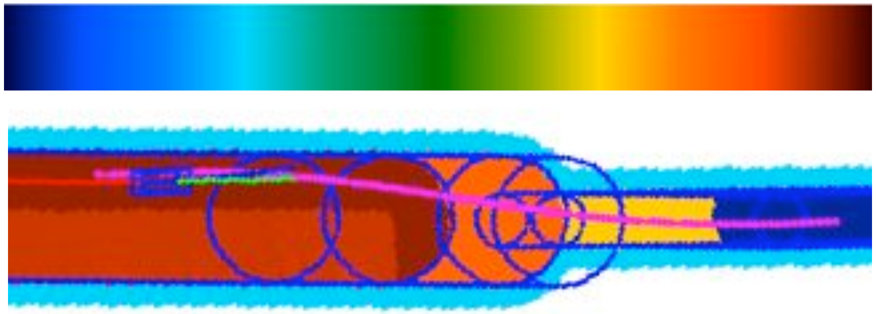
Path Planner

Path Follower

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

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

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



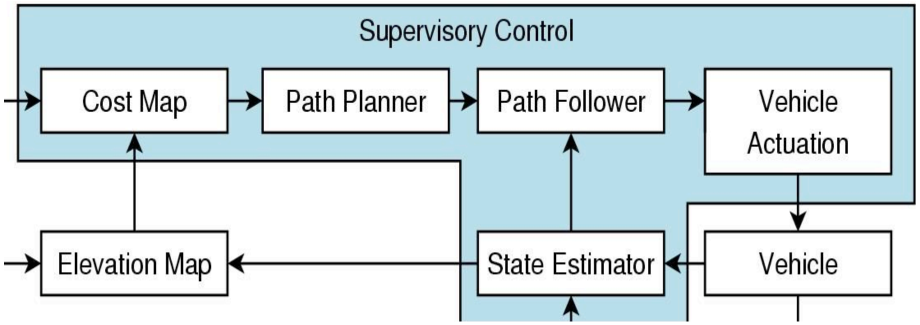
Trajectory Generation: plannerModule

- 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

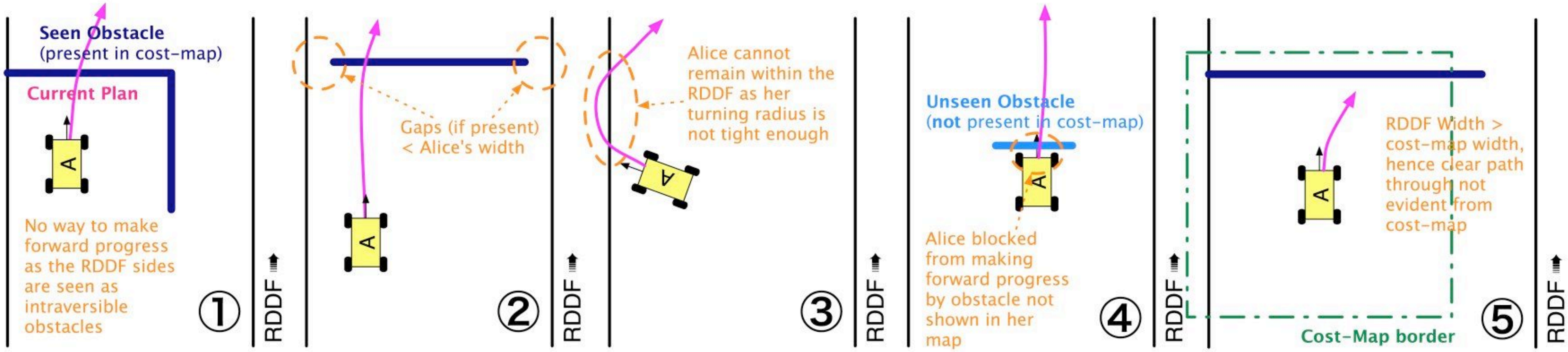


Supervisory Control

- Control operation of other modules
- Always maintain forward progress

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


Seen Obstacle
(present in cost-map)
Current Plan
A
No way to make forward progress as the RDDF sides are seen as intraversible obstacles
①
RDDF
Gaps (if present) < Alice's width
A
②
RDDF
Alice cannot remain within the RDDF as her turning radius is not tight enough
A
③
RDDF
Unseen Obstacle
(not present in cost-map)
A
Alice blocked from making forward progress by obstacle not shown in her map
④
RDDF
RDDF Width > cost-map width, hence clear path through not evident from cost-map
A
Cost-Map border
⑤
RDDF

Dead-End Scenario

Blocked RDDF Scenario

*Note that this is guaranteed **not** to **physically** occur in the GCE by DARPA (but Alice may believe it has occurred from her sensor data)*

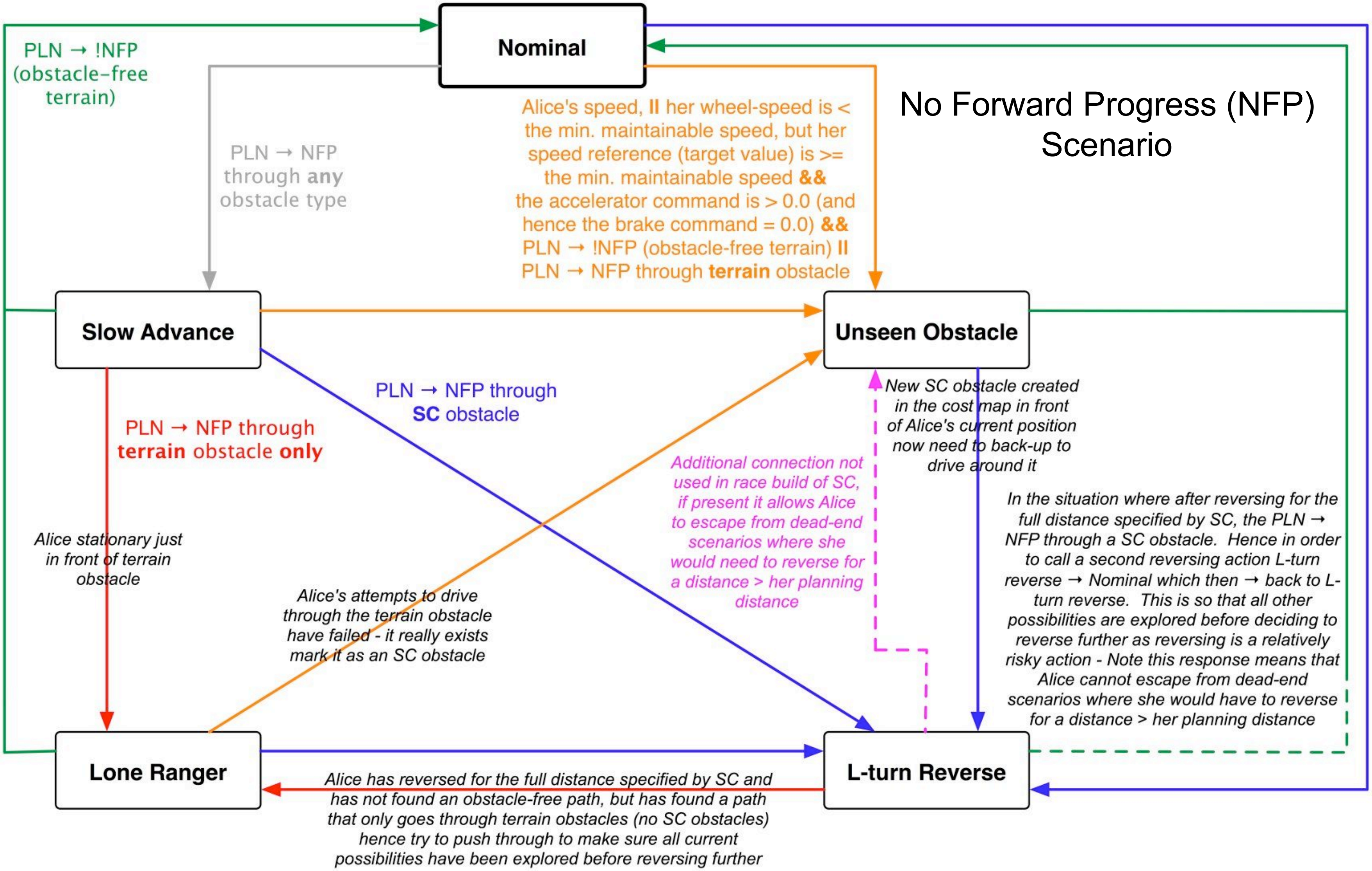
Planning outside RDDF Scenario

Unseen Obstacle Scenario

Super-Wide RDDF Scenario

SuperCon Logic

No Forward Progress (NFP) Scenario



Nominal

Slow Advance

Unseen Obstacle

Lone Ranger

L-turn Reverse

PLN → !NFP (obstacle-free terrain)

PLN → NFP through **any** obstacle type

Alice's speed, **II** her wheel-speed is < the min. maintainable speed, but her speed reference (target value) is >= the min. maintainable speed **&&** the accelerator command is > 0.0 (and hence the brake command = 0.0) **&&** PLN → !NFP (obstacle-free terrain) **II** PLN → NFP through **terrain** obstacle

PLN → NFP through **SC** obstacle

PLN → NFP through **terrain** obstacle **only**

New SC obstacle created in the cost map in front of Alice's current position now need to back-up to drive around it

Additional connection not used in race build of SC, if present it allows Alice to escape from dead-end scenarios where she would need to reverse for a distance > her planning distance

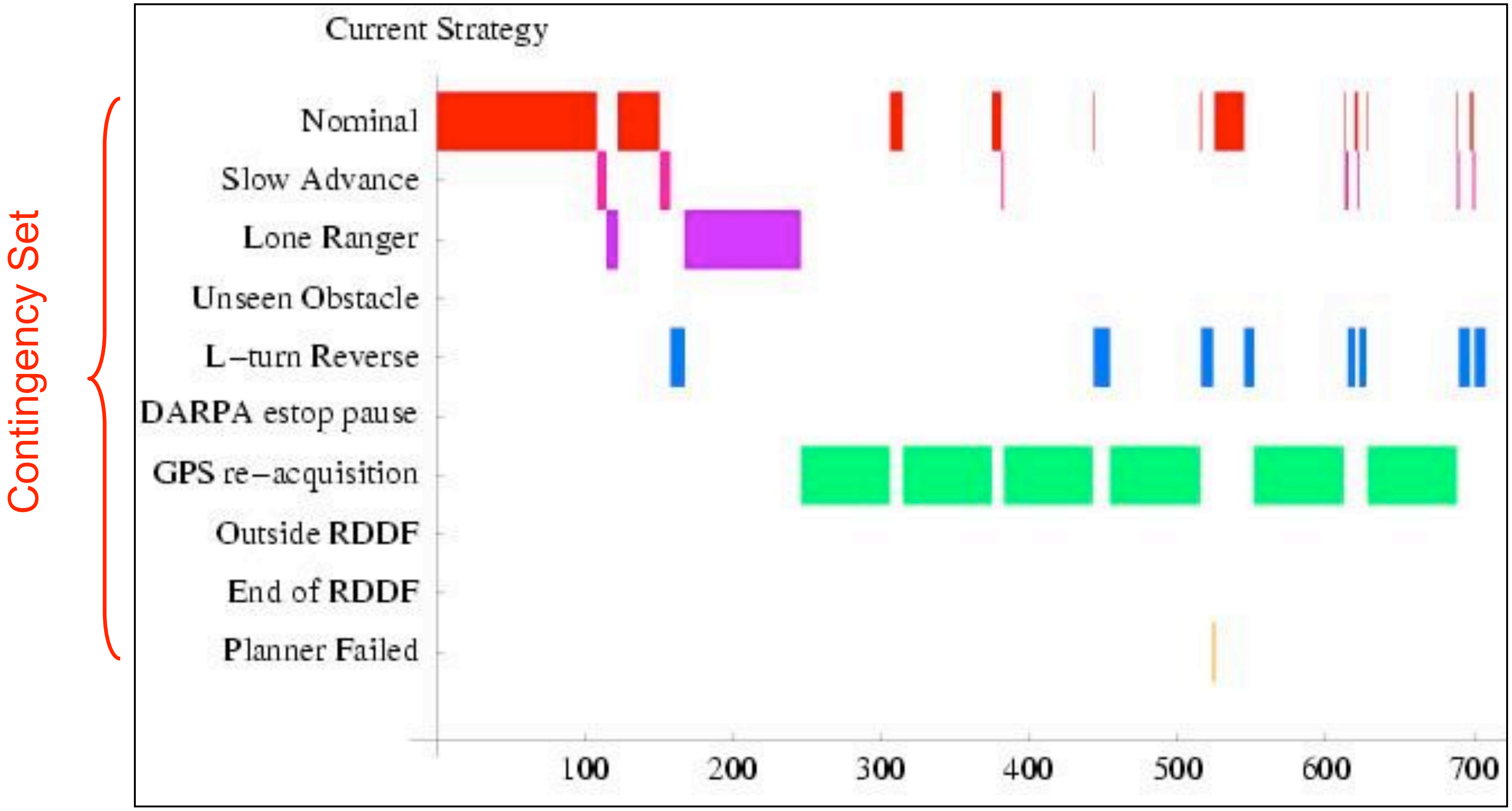
In the situation where after reversing for the full distance specified by SC, the PLN → NFP through a SC obstacle. Hence in order to call a second reversing action L-turn reverse → Nominal which then → back to L-turn reverse. This is so that all other possibilities are explored before deciding to reverse further as reversing is a relatively risky action - Note this response means that Alice cannot escape from dead-end scenarios where she would have to reverse for a distance > her planning distance

Alice stationary just in front of terrain obstacle

Alice's attempts to drive through the terrain obstacle have failed - it really exists mark it as an SC obstacle

Alice has reversed for the full distance specified by SC and has not found an obstacle-free path, but has found a path that only goes through terrain obstacles (no SC obstacles) hence try to push through to make sure all current possibilities have been explored before reversing further

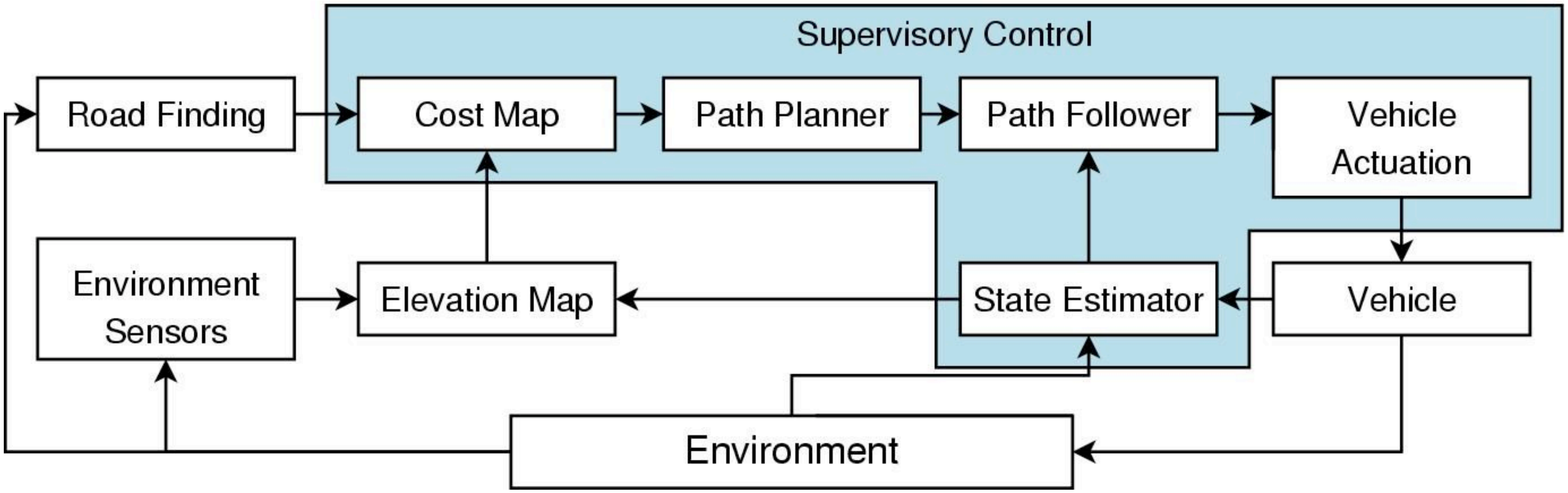
SuperCon Usage (NQE Run 1)



Heavy usage of superCon modes during “typical” operations

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

Architecture Summary



Supervisory Control

Road Finding

Cost Map

Path Planner

Path Follower

Vehicle Actuation

Environment Sensors

Elevation Map

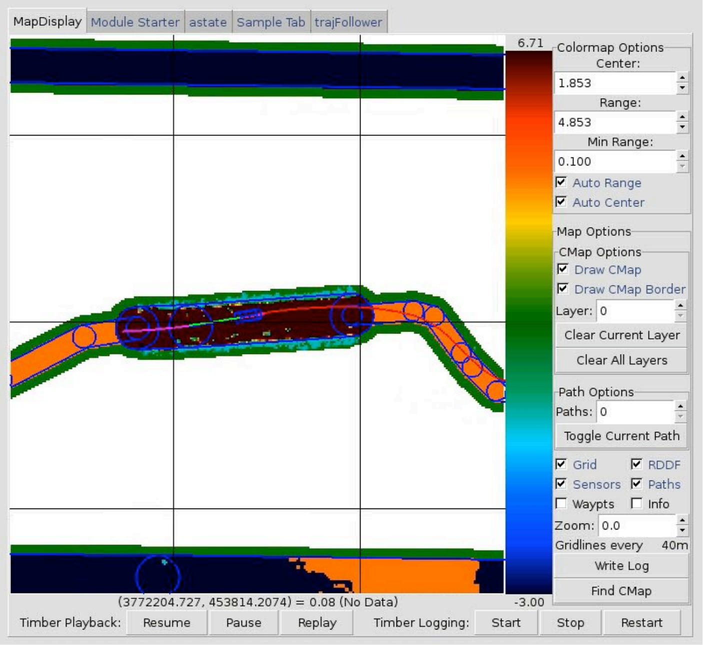
State Estimator

Vehicle

Environment

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



Colormap Options

Center: 1.853

Range: 4.853

Min Range: 0.100

☑ Auto Range

☑ Auto Center

Map Options

CMap Options

☑ Draw CMap

☑ Draw CMap Border

Layer: 0

Path Options

Paths: 0

☑ Grid ☑ RDDF

☑ Sensors ☑ Paths

☐ Waypts ☐ Info

Zoom: 0.0

Gridlines every 40m

(3772204.727, 453814.2074) = 0.08 (No Data)