CDS 270-2: Lecture 3-3
Alice Path Planning

Richard M. Murray

14 April 2006

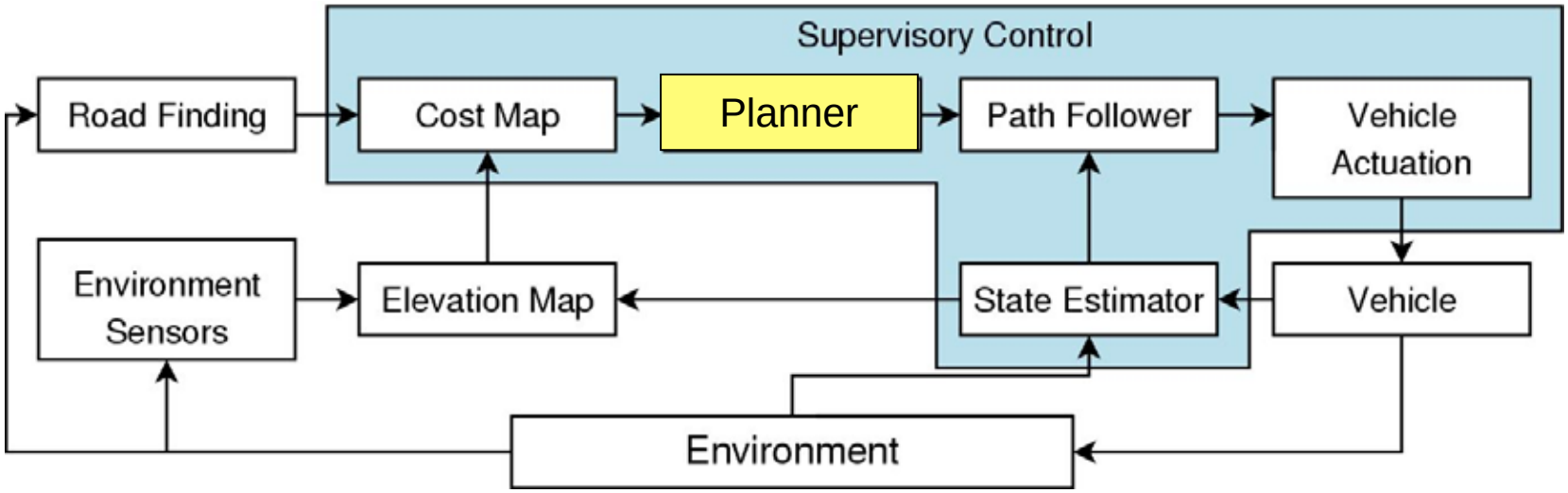
Goals:

- Describe how real-time trajectory generation and receding horizon control are implemented on Alice

Reading:

- "Real-time Path Planning Via Nonlinear Optimization Methods", Dmitriy Kogan and Richard Murray. To be submitted, *IEEE T. Robotics*, 2006.

Path Planner Specification



Planner Specification

- 15 cm high obstacles at ~7.5 m/s
- 30 cm high/deep obstacles at ~1 m/s
- Detect and avoid situations that are worse than this

Method:

- Two stage, optimization-based planner
- Cost function given by velocity maps

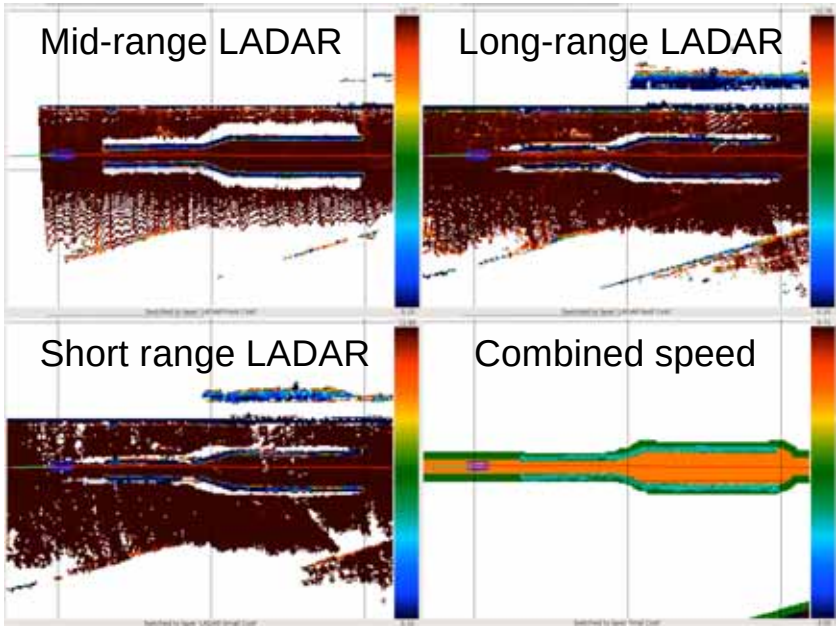
Inputs

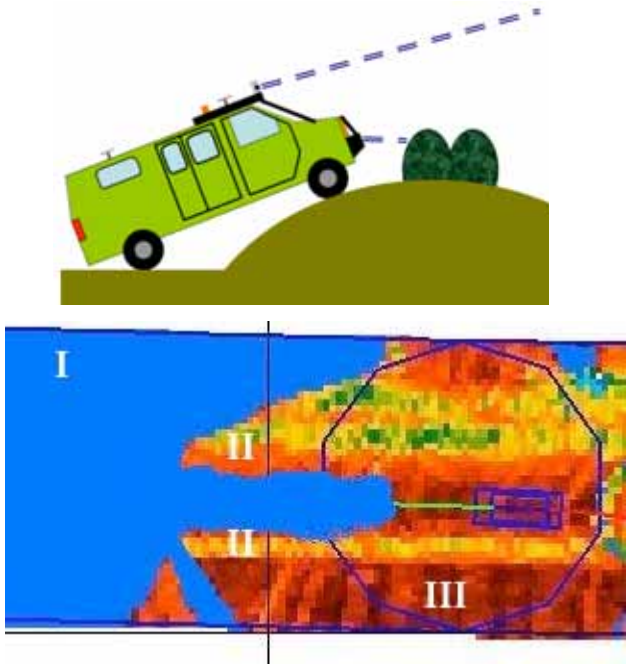
- Speed maps from fusionMapper (sent as deltas)
- Current position of vehicle (state)
- Supervisory commands

Outputs

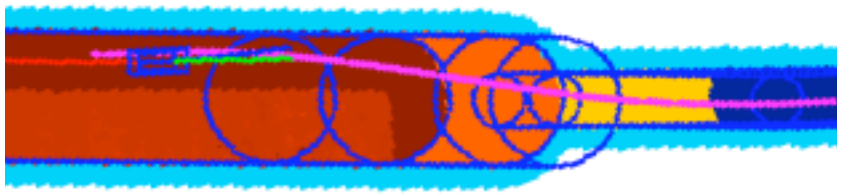
- Planned trajectory for next 20-40 m

Sensor Fusion and Cost Map Processing

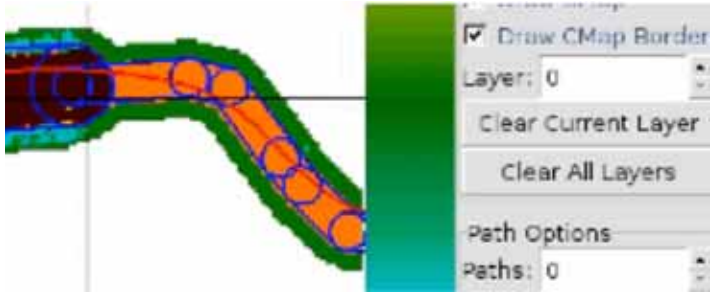




Simple cost map:



Realistic cost map:



Planner Approach

Underlying strategy

- Nonlinear programming optimization with SNOPT
- Traversal time optimized over space of trajectories
- Dynamic feasibility as constraints
- Obstacle avoidance as constraint/cost
- Receding horizon

Vehicle dynamics

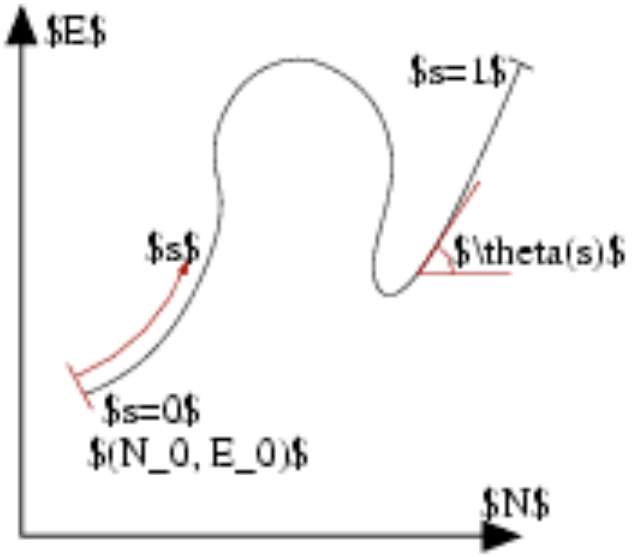
$$\begin{aligned} \dot{N} &= v\cos\theta \\ \dot{E} &= v\sin\theta \\ \dot{\theta} &= \frac{v}{L}\tan\phi \\ \dot{\phi} &= \omega = u_1 \\ \dot{v} &= a = u_2 \end{aligned} \qquad s.t. \quad \begin{aligned} \phi &\in [\phi_{min}, \phi_{max}] \\ \omega &\in [\omega_{min}, \omega_{max}] \\ v &\in (0, v_{max}]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

Reparameterization

- Choose θ, v as indep variables; parameterize by quadratic splines
- Integrate from initial pos to get N, E
- Impose speed limit, accel limit, steering mag/rate limits, rollover constraints
- Minimize traversal time

$$T = S_f \int_0^1 \frac{1}{v(N(s), E(s))} ds$$

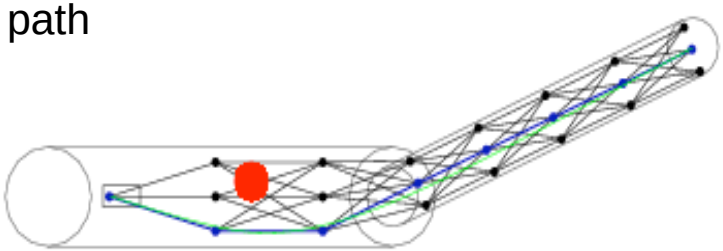
plus steering, acceleration “effort”



Additional Modifications

Seed path (stage 1 planner)

- Provides initial conditions for optimization based planner
- Minimize time based on velocity along path



Obstacle convexification

- Flat obstacles create problems for optimization algorithm
- Solution: create gradient on obstacles to allow optimizer convergence

Growing obstacles

- Need to grow obstacles to take into account vehicle width
- Use orientation of obstacles in deciding how to grow

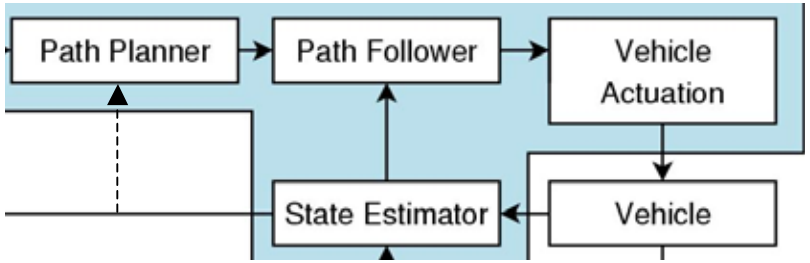
No data regions

- Encode no data regions as “negative” velocity in maps
- No data regions inside vehicle stopping distance: use low speed (3 m/s)
- No data regions further than stopping distance: use 2X current speed
- Allows vehicle to keep moving at high speed until slowing down is required

Receding Horizon Implementation

Problems with RHC implementation

- Original plan: replan trajectory from current path
- Issue: gave very oscillatory operation
- Cause: interaction between planner and trajectory tracker - update times too close

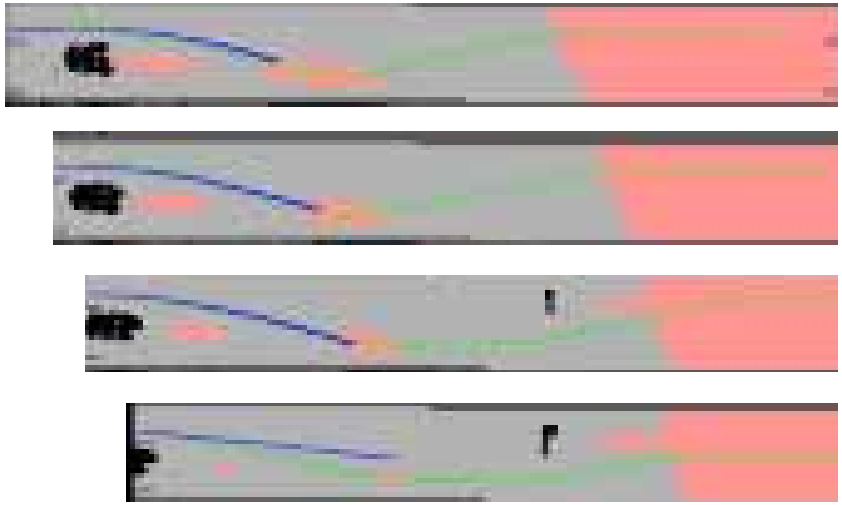


Solution: plan from *previous* path

- Use planner to mitigate uncertainty from new obstacles in obstacle map
- Use follower to mitigate uncertainty in vehicle dynamics, path changes

Example: NQE run

- Seed path in green; planner path in blue
- Dark areas = obstacles
- Pink area = no data regions



Notes

- Path changes as obstacles appear
- No data regions beyond stopping distance treated favorably

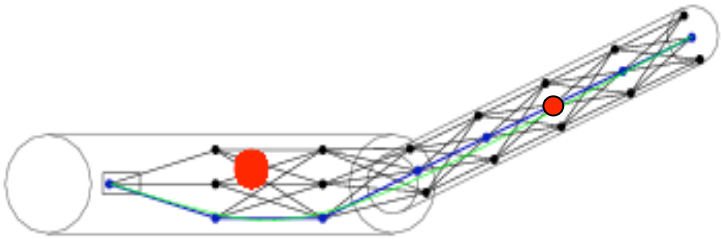
Interface Issues

Planning horizon

- Allow planning to increase as speed increases
- Min horizon = 25 m, max = 65 m
- Use lag to avoid rapid changes

Stage 1 "indecisiveness"

- If obstacle is directly in front of vehicle, stage 1 can switch sides each iteration
- Solution: penalize distance from path of previous iteration

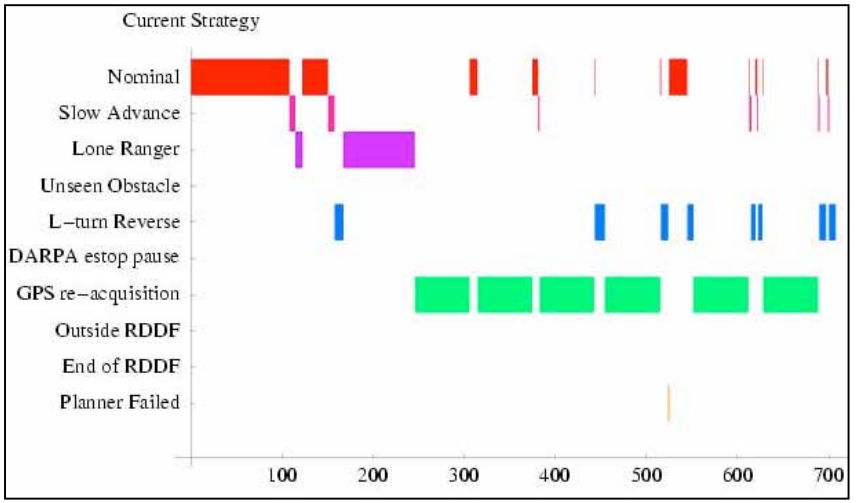


Vehicle width

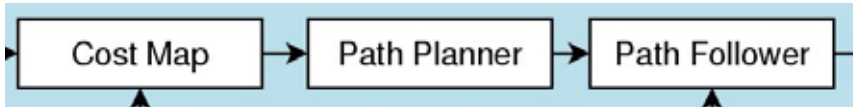
- Increase width of vehicle at high speed

Supervisory control

- Trajectories can go through obstacles or outside "corridor"
  - Obstacles appear after plan is made
  - Optimizer fails to converge
- Supervisory control catches these cases and brings vehicle to stop
- Supervisory controller can then "force" vehicle along last planned path ("lone ranger"



Summary: Path Planner

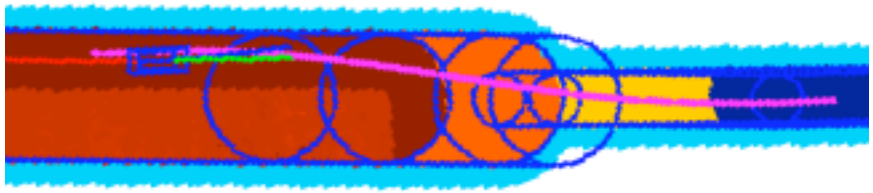


$$\arg\min \int_t^{t+T} L(x,u)d\tau + V(x(T))$$

$$\dot{x} = f(x,u)$$

$$g(x,u) \leq 0$$

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



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