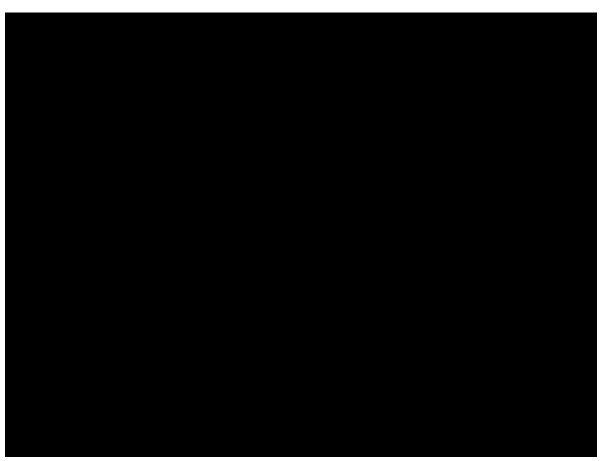
TBA

Examples from 1) Likhachev & Ferguson 2) C. Urmson *et al.*, Tartan Team

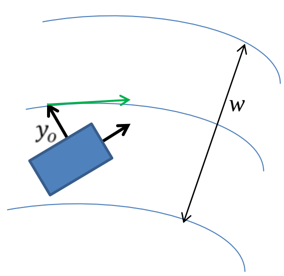


Focus on two things:
Sampling for perception
Planning

Perception: Static

- Multiple algorithms from LIDAR fused
  - Spatially: Filter out spurious detections
  - Temporally: Filter out moving objects
- Specialized detectors for curbs, road boundaries

Perception: Roads (application of sampling)



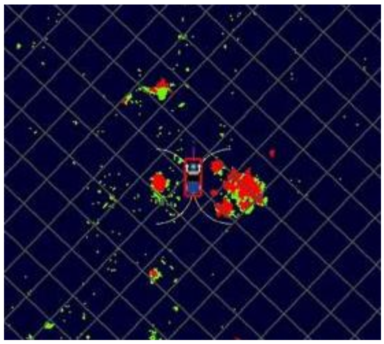
$$y(x) = y_0 + tan(\phi)x + C_0\frac{x^2}{2} + C_1\frac{x^3}{6}$$

$$X = [y_o\ \phi\ C_o\ C_1\ w]$$

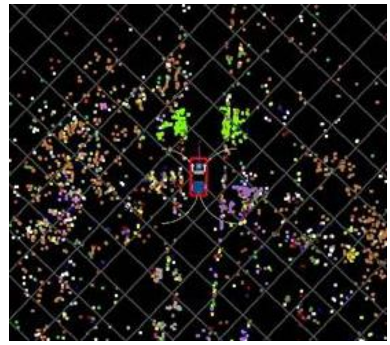
- Observation Z from sensor data
- Find X: $\max_X p(X|Z)$
- Problem: Cannot represent distribution explicitly
- Solution: sampling

Peterson et al., IROS2009

Input from sensing: Z

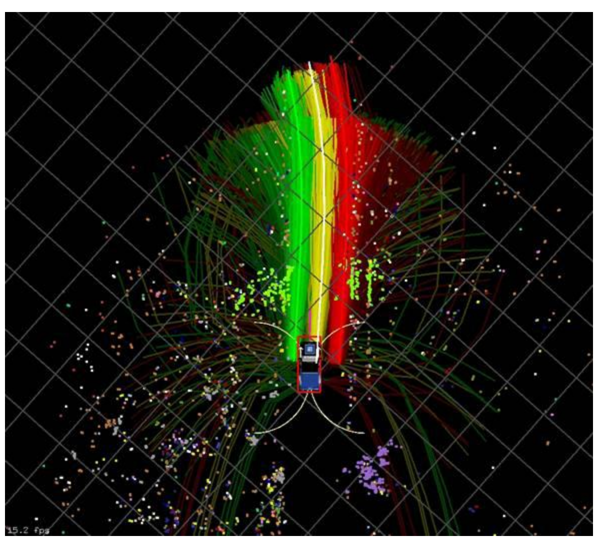


Obstacle map



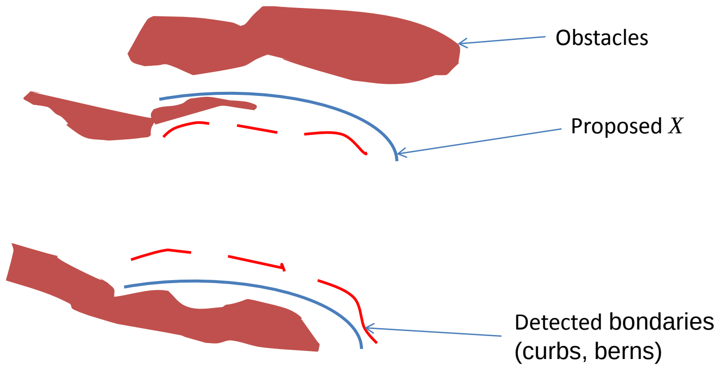
Curb detection (potential boundaries)

Roughness/ edge density



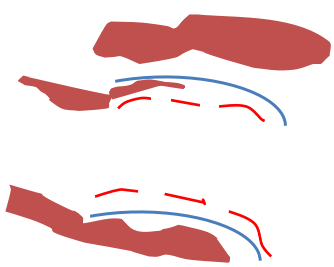
- Particle filters/sampling: Good way to handle hard-to-model noisy perception
- Very fast (keeps up with driving speed)

SIR Sampling: Define likelihood for which to sample



- Minimize number of obstacles within X
- Minimize roughness of X
- Minimize distance between boundaries of X and predicted boundaries

SIR Sampling: Define likelihood for which to sample



$$P(X|Z) = e^{-\theta^T E(X)}$$

- $E_o(X)$ = Number of obstacles cells within X
- $E_r(X)$ = Number of obstacles cells within X (roughness)
- $E_{br}(X)$ = Sum of distances between right boundary of X and detected boundaries
- $E_{bl}(X)$ = Sum of distances between right boundary of X and detected boundaries
- $E(X) = [E_o(X)\ E_r(X)\ E_{br}(X)\ E_{bl}(X)]$

Reminder: Importance sampling

- Evaluating expectation from p not normalized so instead:

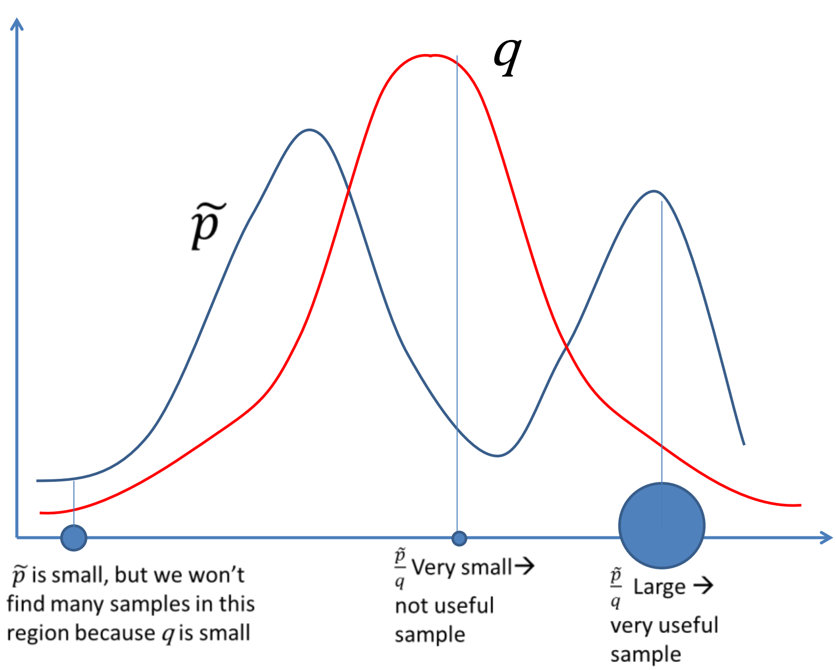
$$E_P(f) = \frac{1}{Z}\int f(x)\,\frac{\tilde{p}(x)}{q(x)}q(x) \approx \frac{1}{Z}\frac{1}{N}\sum_i f(x_i)\frac{\tilde{p}(x_i)}{q(x_i)}$$

For f = 1: $\frac{1}{Z}\frac{1}{N}\sum_i \frac{\tilde{p}(x_i)}{q(x_i)} = 1$

- $E_P(f) \approx \sum_i f(x_i) w_i \quad w_i = \frac{\tilde{p}(x_i)}{q(x_i)} / (\sum_l \tilde{p}(x_l)/q(x_l))$

If x_i are sampled from q

"Importance" of x_i



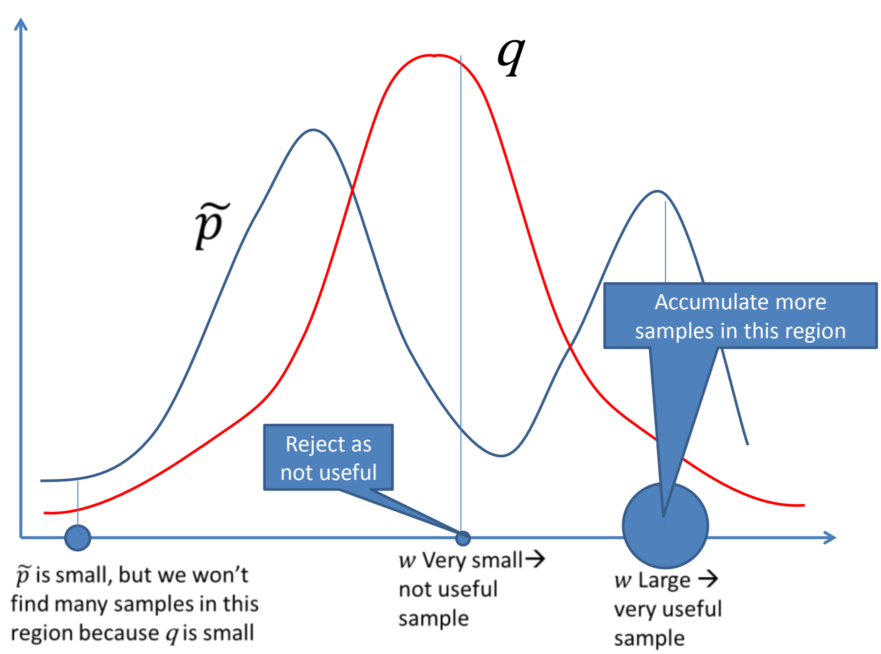
Compromise: SIR

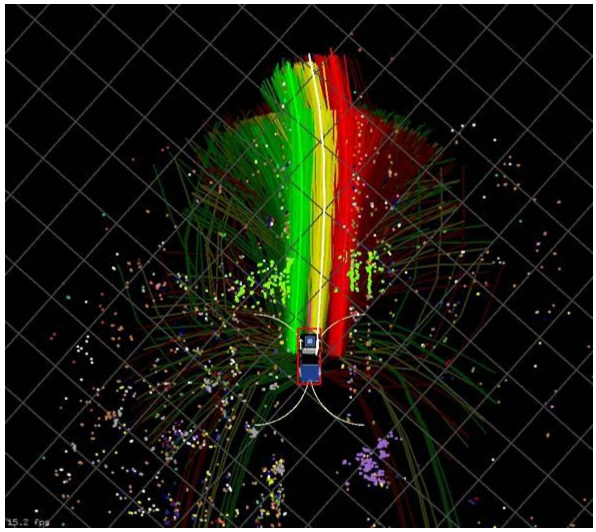
- Fine to evaluate expectation but we may want to draw actual samples

- Draw N samples x_i, w_i (with normalized w_i)
- Draw again N samples from $(x_1, .., x_N)$

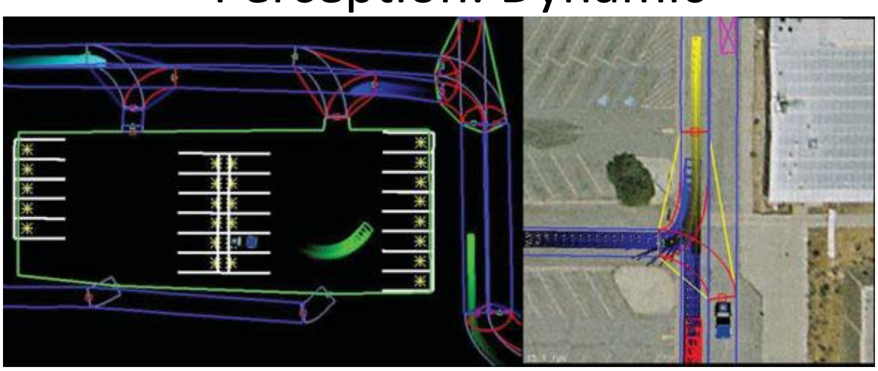
using distribution $(w_1, .., w_N)$

- Basically: Smart way of reject samples with low weight
- Guaranteed to converge to p when N→ ∞



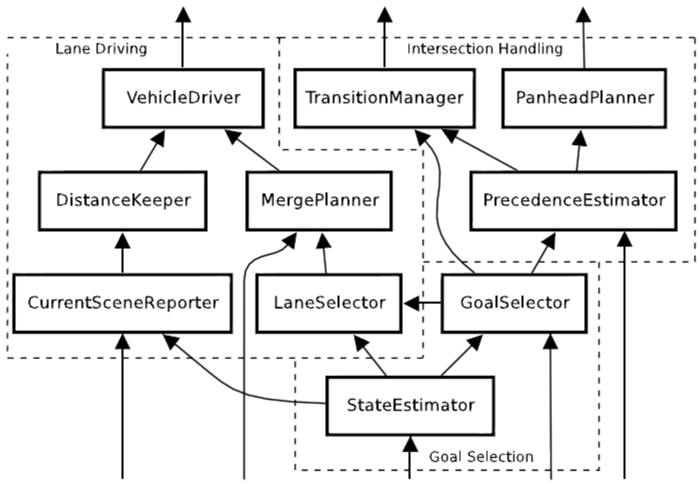


Perception: Dynamic



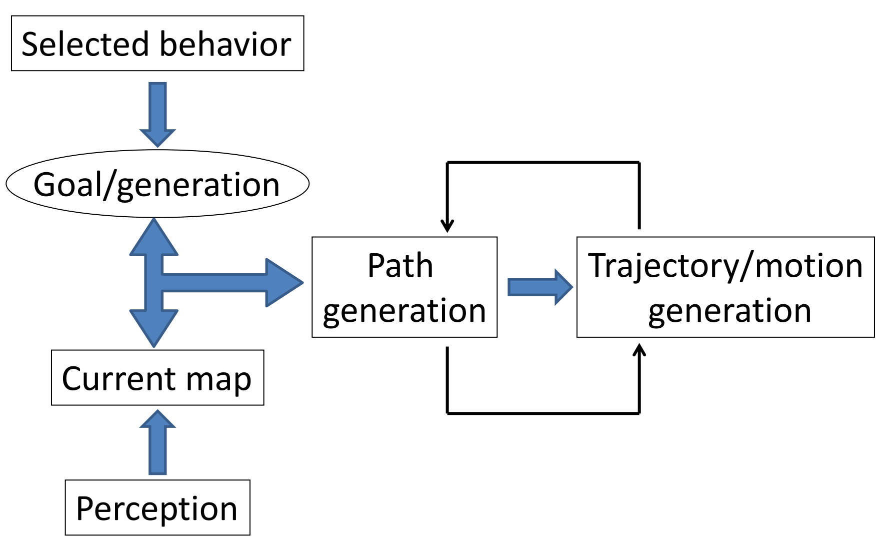
- Detection from multiple sensors
- Fusion of hypotheses
- Tracking filter using context knowledge
  - Road
  - Intersection
  - Zone

Architecture



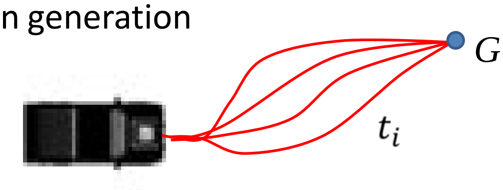
- Behavior selection from mission description in 3 contexts: Road, intersection, zone
- Goal state generation from behaviors

Basic planning loop



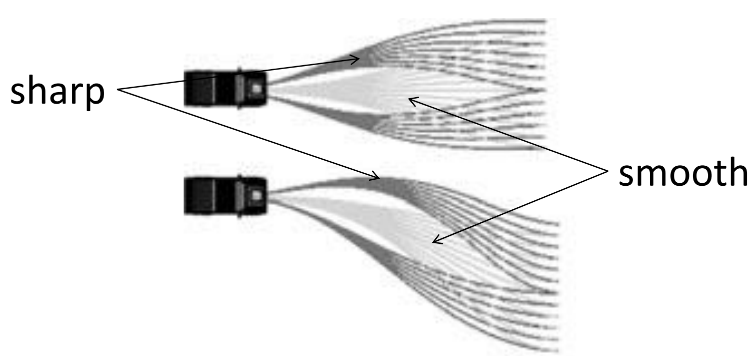
First case: Short-horizon planning (e.g., road tracking)

Trajectory/motion generation

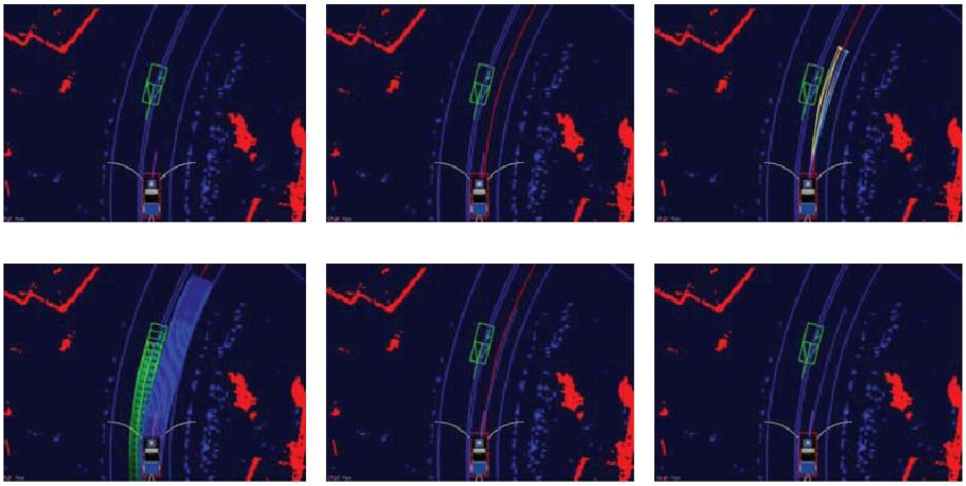


- Moving goal G
- Dynamically generate a family of *feasible* trajectories $\{t_n\}$
- Also "pure pursuit", "receding horizon"

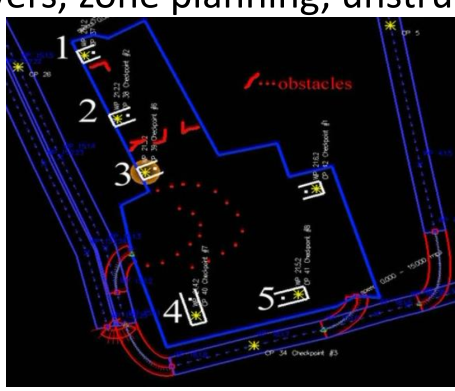
Howard & Kelly. Optimal trajectory generation. IJRR. 2007



- Two sets of trajectories
- Select lowest cost trajectory
- Distance from obstacles/lane boundaries

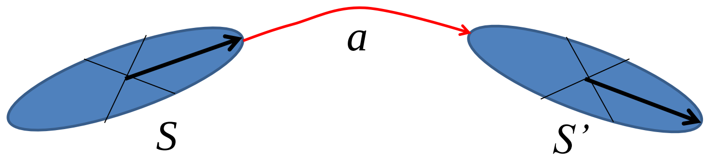


Second case: Long-horizon planning
e.g.: Maneuvers, zone planning, unstructured roads



- Use standard planning technique (A*, D*, etc.)
- Problem: Car has kinematic and dynamic constraints
  - Curvature constraints (turning radius)
  - Speed vs. curvature constraints (if speed high enough)
- Given state s only subset if control input u is allowed

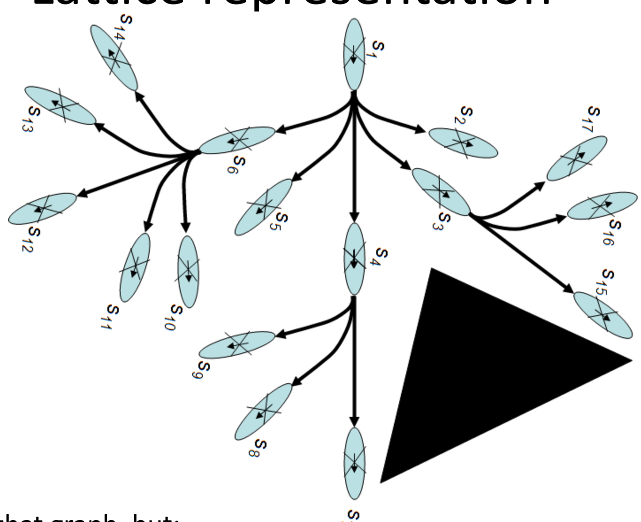
Lattice representation



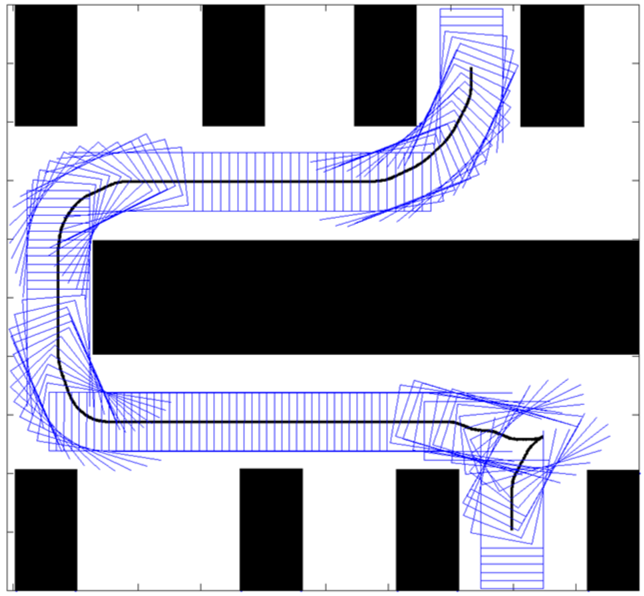
- Graph of states connected by *feasible* actions
- States:
  - $s = (x, y, \theta, v)$ v = translation velocity (forward/backward)
- Actions:
  - For each s: generate the subset of states at distance d that are reachable by a valid trajectory (solved using the previous technique for short-term goals)
  - Reduction: Eliminate redundant actions

Pivtoraiko & Kelly. Generation near optimal spanning control sets….IROS 2005.

Lattice representation



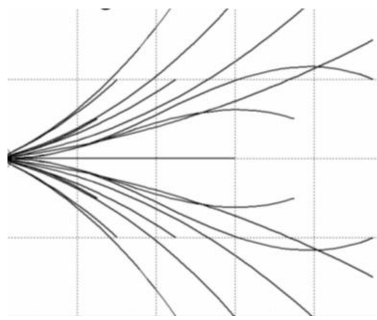
- Planning in that graph, but:
  1. Combinatorics look bad
  2. Bounded amount of time to make decision → Anytime planning
  3. Environment changes because obstacles are discovered → Dynamic planning
  4. What admissible heuristics (in the A* sense)?



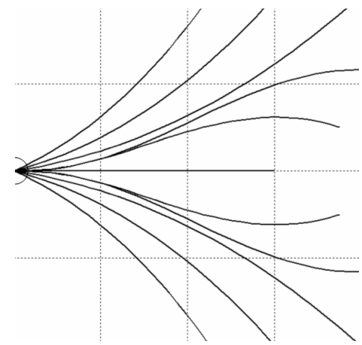
1. Combinatorics

- Observation:
  - Need detailed, high-resolution sampling of the actions at the start and goal but not in between
- Implementation (multiresolution graph):
  - Generate “high-resolution” action graph A_h
  - Subsample action set far from goal and start $A_l \subset A_h$
- Guarantees:
  - Any path in a lower resolution action graph is a valid path in the multiresolution graph)
  - Any path in the multiresolution graph is a valid path in a high-resolution action graph

1. Combinatorics: Multi-resolution



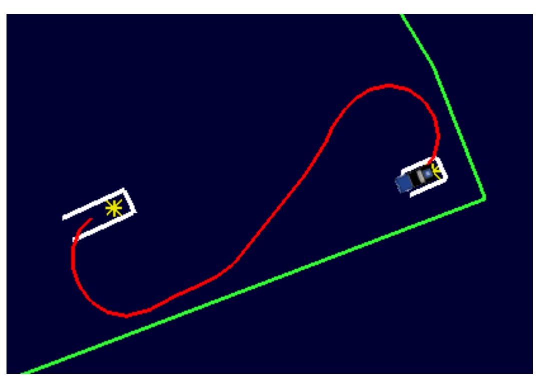
A_h



A_l

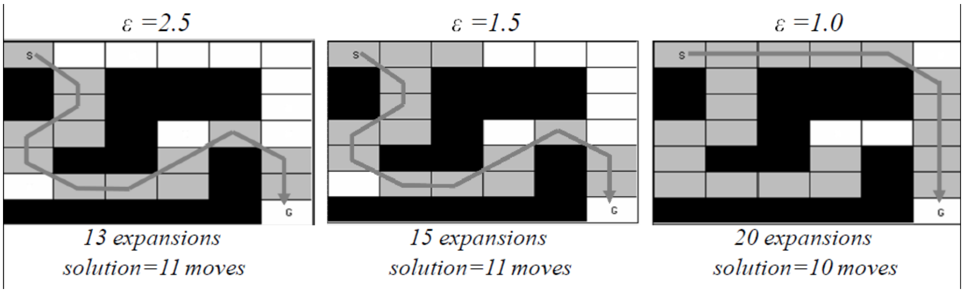
- Observation:
  - Need detailed, high-resolution sampling of the actions at the start and goal but not in between
- Implementation (multiresolution graph):
  - Generate "high-resolution" action graph A_h
  - Subsample action set far from goal and start $A_l \subset A_h$
- Guarantees:
  - Any path in a lower resolution action graph is a valid path in the multiresolution graph)
  - Any path in the multiresolution graph is a valid path in a high-resolution action graph

1. Combinatorics: Multi-resolution



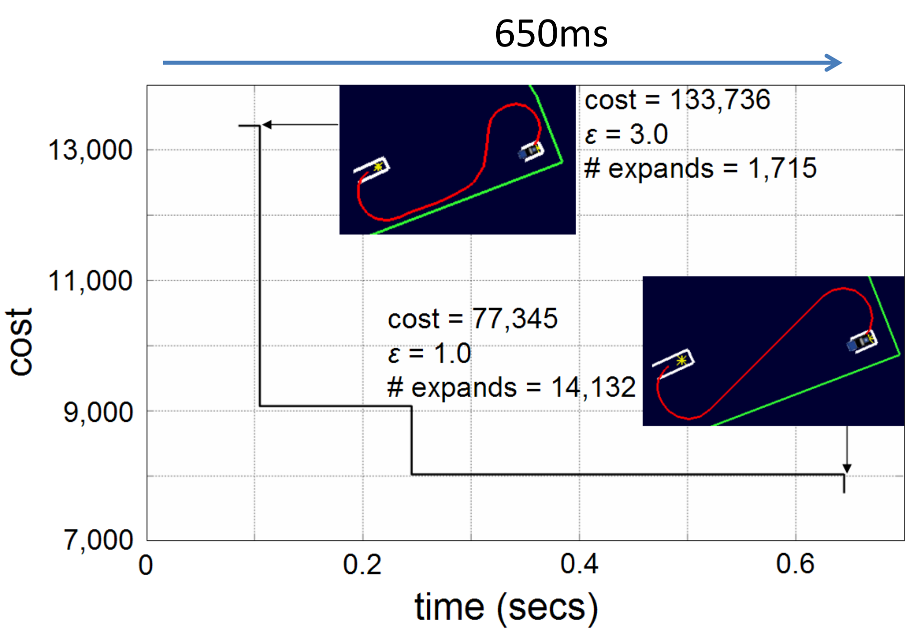
	Expansions	Time
High res	2933	0.19
Low res	1228	0.06

2. Anytime

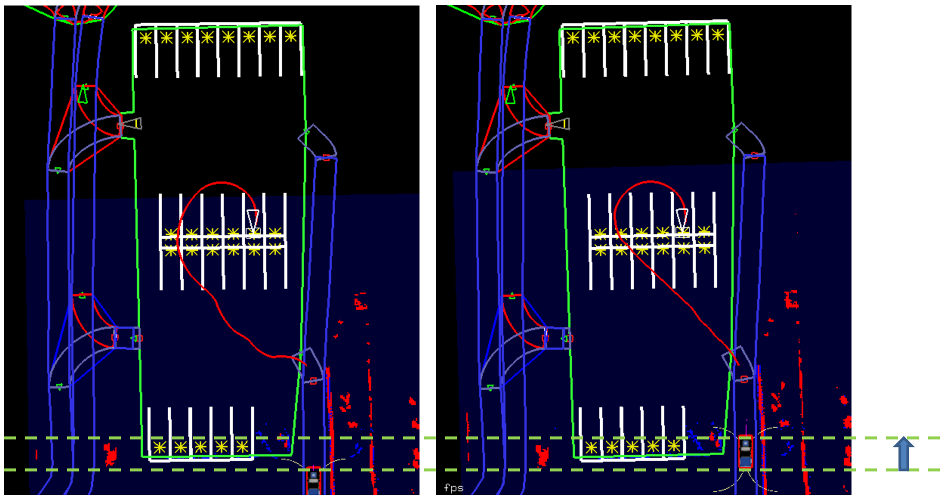


- Car keeps moving continuously and can't stop and wait until plan is complete
- Need answer within some bounded time which may not be enough for full path
- Anytime version of A*:
  - Assume heuristic $h(.)$ is given
  - Plan with $\varepsilon h(.)$ instead of → Fast but sub-optimal solution
  - Decrease ε → Increasingly better solution, converging to optimal

2. Anytime



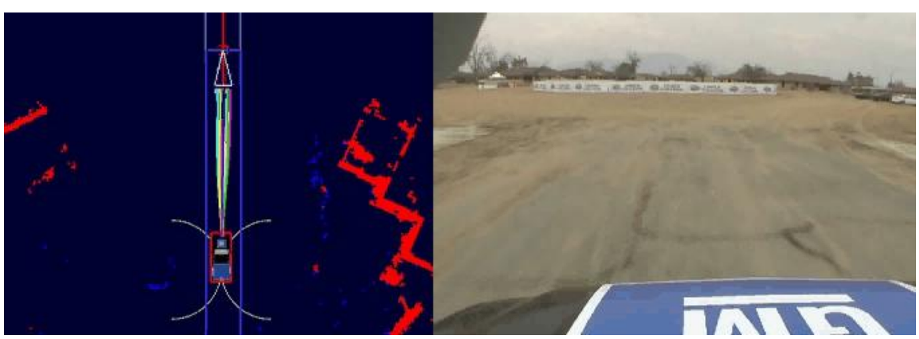
2. Anytime



Pre-planned path $\varepsilon = 3.0$

Refined, optimal path $\varepsilon = 1.0$

3. Dynamic



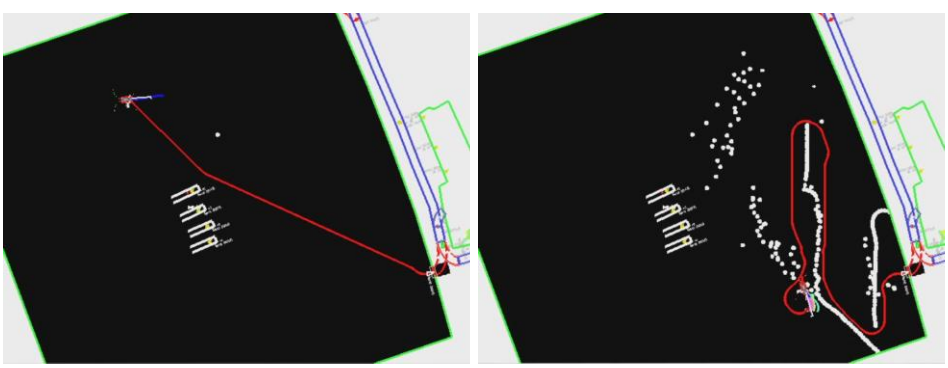
- New obstacles are detected continuously as car moves
- *Dynamic* obstacles change continuously
- Solution: Add dynamic repairing component to the anytime version (e.g., D*)

3. Dynamic

- New obstacles are detected continuously as car moves
- *Dynamic* obstacles change continuously
- Solution: Add dynamic repairing component to the anytime version (e.g., D*)

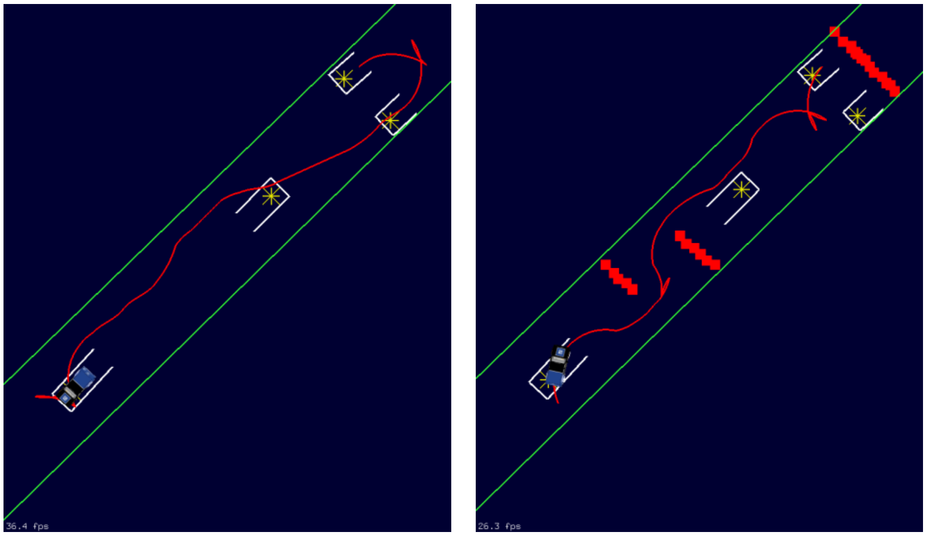
- set ε to large value
- until goal is reached
  - ComputePathReuse() (weighted ε A*)
  - Follow the path until world is updated with new information
  - Update the corresponding edge costs
  - Set s_{start} to the current state of the agent
  - If "significant" changes were observed
    - increase ε or replan from scratch
  - else
    - decrease ε

3. Dynamic: Discovering obstacles

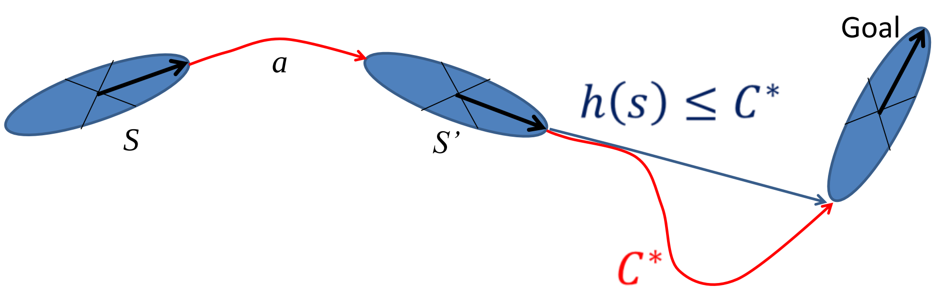


200m x 200m

3. Dynamic: Complex maneuvers

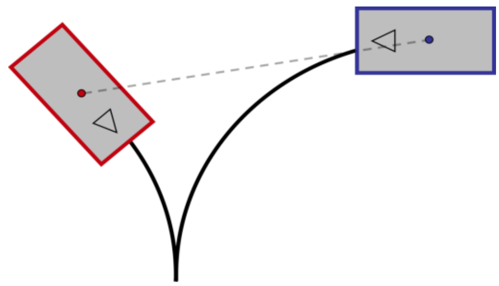


4. Heuristics



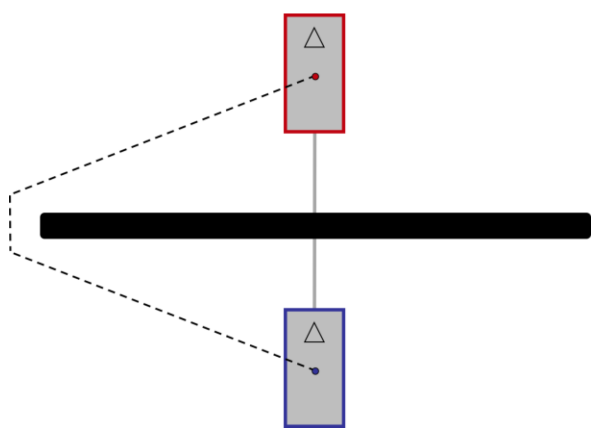
- Everything depends on admissible $h(.)$
- $h(.)$ needs to be admissible and consistent
$$h(s) + c(s, s') \geq h(s')$$
- Two types of factors can be used to evaluate the cost of the path:
  - Mechanism constraints (action graph)
  - Environment constraints (obstacles)

4.a Mechanism Heuristics



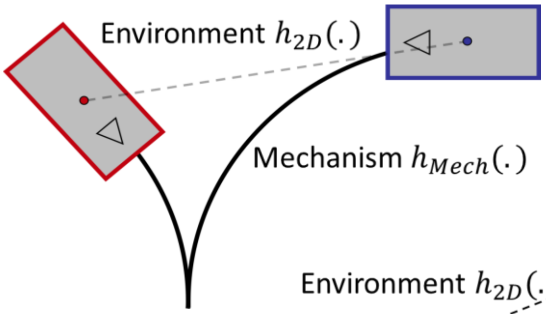
- Compute the path from start to goal using:
  - Full action-state graph
  - With *no* obstacles
- Expensive but:
  - Can be pre-computed *once offline*!
- Fully integrates the physical constraints of the problem
- But can grossly underestimate the path cost

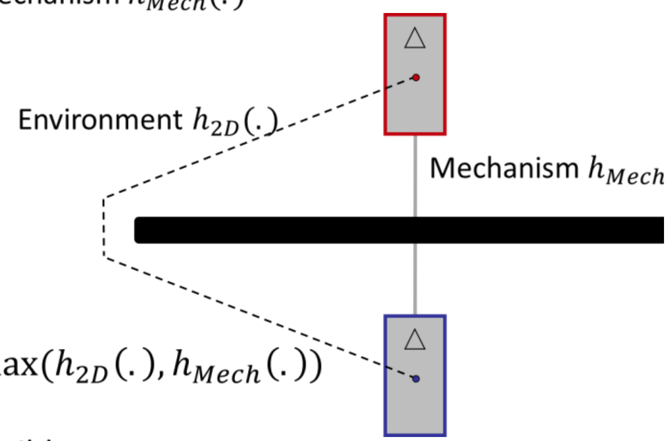
4.b Environment Heuristics



- Ignore the mechanism constraints
- Compute path in 2-D (x,y) grid
- Has be to done online, but *very fast*
- Accounts for obstacles but *may still grossly underestimate* by using mechanically infeasible paths

4. Heuristics



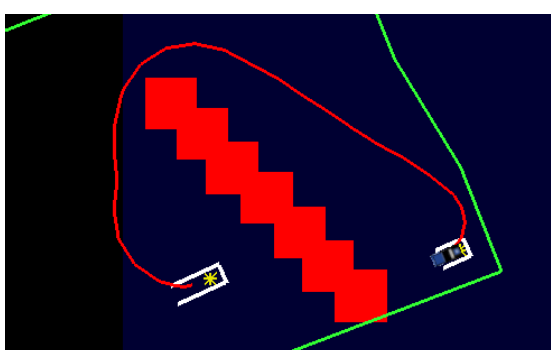


Combination:

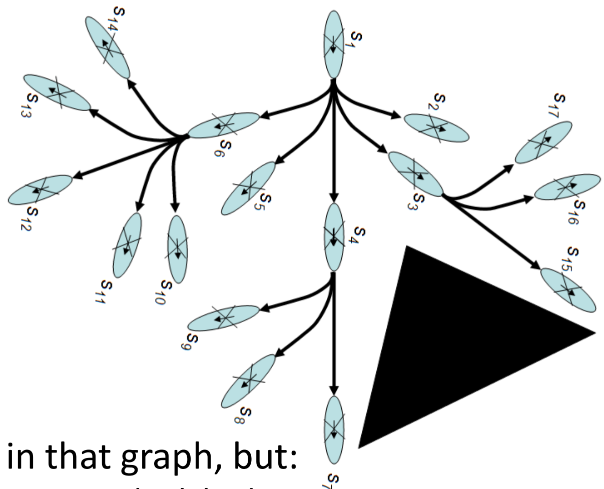
$$h(.) = \max(h_{2D}(.), h_{Mech}(.))$$

Max of admissible heuristics is admissible

4. Heuristics

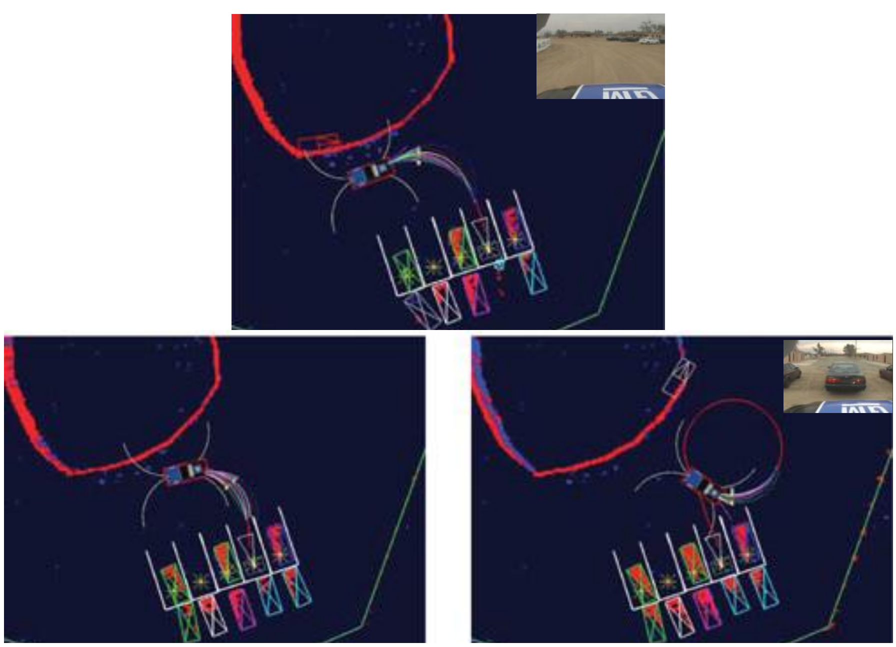


	Expansions	Time
h	2,019	0.06
h_{2D}	26,108	1.30
h_{Mech}	124,794	3.49

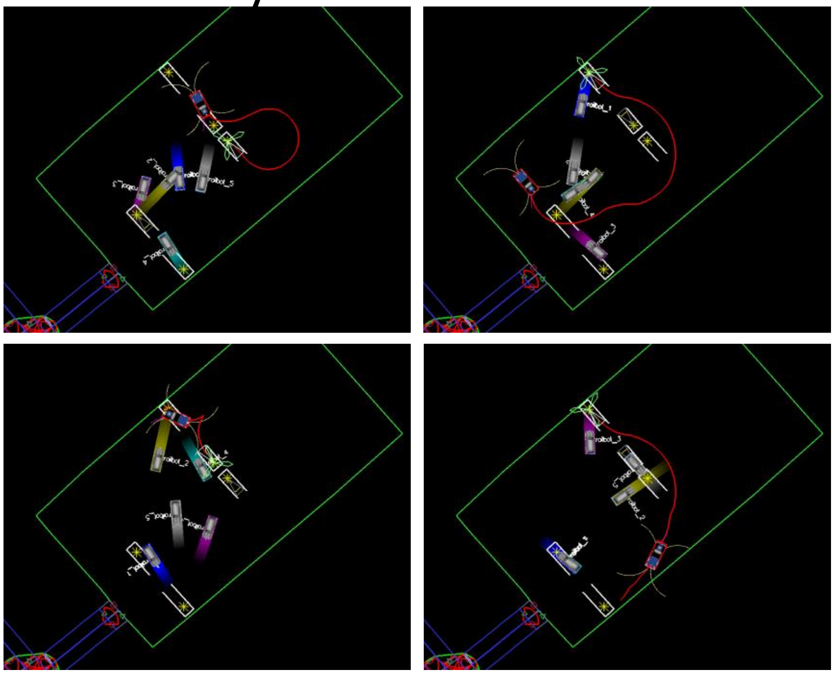


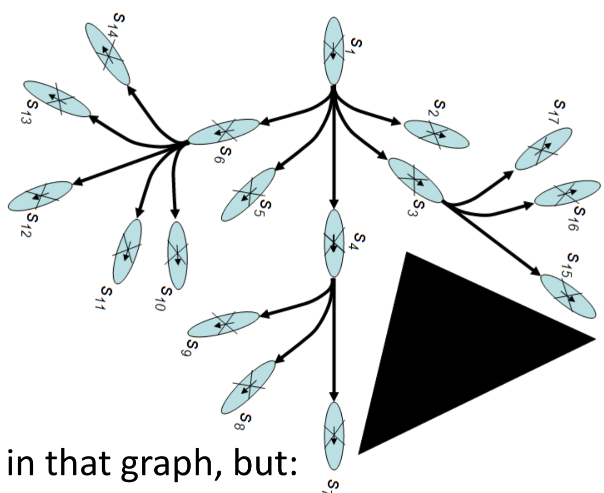
- Planning in that graph, but:
  1. Combinatorics look bad
  2. Bounded amount of time to make decision → Anytime planning
  3. Environment changes because obstacles are discovered → Dynamic planning
  4. What admissible heuristics (in the A* sense)?

Static obstacles



Dynamic obstacles





- Planning in that graph, but:
  1. Combinatorics look bad
  2. Bounded amount of time to make decision → Anytime planning
  3. Environment changes because obstacles are discovered → Dynamic planning
  4. What admissible heuristics (in the A* sense)?