Chapter 7
Control

Part 4

7.4 Intelligent Control

Outline

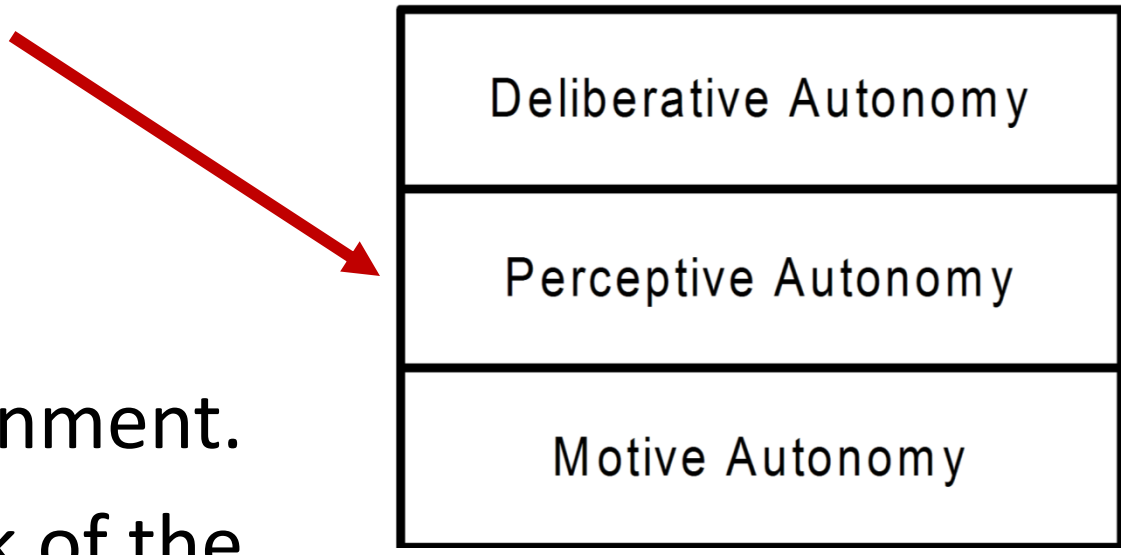
- 7.4 Intelligent Control
  - 7.4.1 Introduction
  - 7.4.2 Evaluation
  - 7.4.3 Representation
  - 7.4.4 Search
  - Summary

Outline

- 7.4 Intelligent Control
  - 7.4.1 Introduction
  - 7.4.2 Evaluation
  - 7.4.3 Representation
  - 7.4.4 Search
  - Summary

Hierarchy

- We are here now …

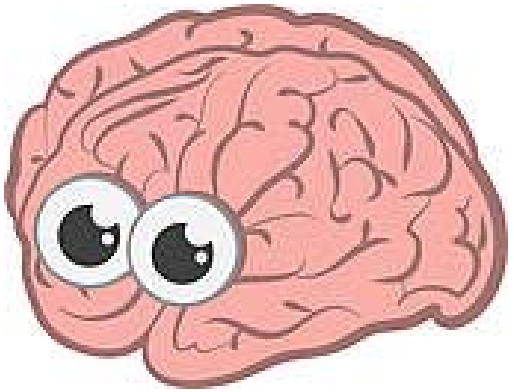


- Responsible for responding to the immediate environment.
- Requires feedback of the state of the environment (e.g. perception).
- May only need relative position estimates.

7.4.1.1 Intelligent Predictive Control

(Perceptive)

- By assumption: Environment is partially unknown and must be measured.
- Don't know beforehand where the obstacles are - or you would have planned around them already.
- "Intelligent" means understanding your surroundings. Hence:
  - IC must be perceptive.
- Perception is discussed later.
  - Here, we will use an environmental model that was produced by perception.



7.4.1.1 Intelligent Predictive Control
(Predictive)

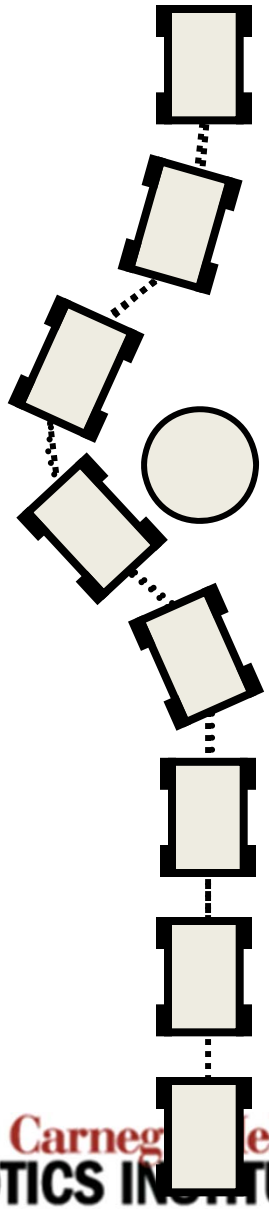
- Latencies and robot dynamics mean it takes time for actions to take effect.
- Robot may also be under-actuated.
- Elements in the scene may be dynamic.
- Hence IC must be predictive.

7.4.1.1 Intelligent Predictive Control
(Reactive)

- However, perception must be done continuously because effective sensor range is limited by:
  - Missing parts (occlusion, limited sensor range)
  - Uncertainty
- Also, prediction of dynamic obstacles is only valid for short periods of time.
- Must:
  - perceive continuously
  - react to what you can see.
  - do it all over again high frequency.

7.4.1.1.1 Generic Intelligent Control Loop

- 1: Consider “all” options for proceeding through space.
- Check each for problems.
- Eliminate those options which are definitely (or probably) problems.
- If any options remain, pick the best from the perspective of mission execution. Goto 1:
- If none remain, do something which reduces your losses
- If you survive that, ask for help, or execute other recovery mechanisms.



7.4.1.1.1 Generic Intelligent Control Loop (Elements of Effective IPC)

- A model of your capacity to move
  - Motion prediction
- A model of the state of the environment
  - Representation
- A capacity to evaluate alternatives for
  - Trajectory evaluation
- A capacity to search through the space of possible motions
  - Optimal control

7.4.1.2 Formulation as Optimal Control

(Objectives and Constraints)

- Motions can be ranked based on cost/utility and satisfaction of hard constraints:
- Simple case:
  - Score each motion (utility)
  - Do not hit obstacles (constraint)
- However, obstacles can also be encoded as cost of traversal and there are motions which do not satisfy feasibility constraints.

7.4.1.2.1 Optimal Control Formulation

(Objectives and Constraints)

- Objectives to Minimize
  - Risk level
  - Path following error
  - Path length to goal
  - Integral speed error.

- Constraints
  - Dynamics (“feasible”) $\dot{\underline{x}} = f(\underline{x}, \underline{u}, t) \quad ; \quad \underline{u} \in U$
  - Don’t hit obstacles (“admissible”) $\underline{x}(t) \notin O$
  - Don’t tip over (“stability”)

7.4.1.2.1 Optimal Control Formulation (Equations)

- Over Time (Trajectory)

$$J[\underline{x}, t_f] = \phi(\underline{x}(t_f)) + \int_{t_0}^{t_f} L(\underline{x}, \dot{\underline{x}}, t)dt$$

$$\dot{\underline{x}} = f(\underline{x}, \underline{u}, t) \quad ; \quad \underline{u} \in U$$

$$\underline{x}(t_0) \in S \qquad \underline{x}(t_f) \in G$$

- Over Space (Path)

$$J[\underline{x}, s_f] = \phi(\underline{x}_f) + \int_{s_0}^{s_f} L(\underline{x}, \underline{u}, s)ds$$

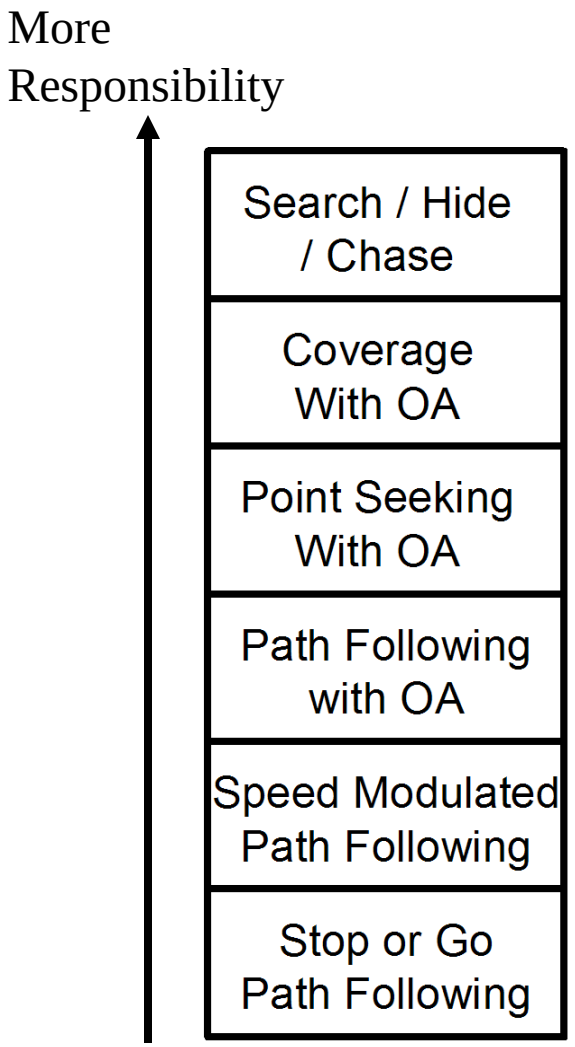
Line Integral

$$\underline{x}(s_0) \in S \qquad \underline{x}(s_f) \in G$$

$$\dot{\underline{x}} = f(\underline{x}, \underline{u}, s) \quad ; \quad \underline{u} \in U$$

7.4.1.2.2 Encoding the Mission in the Objective

- The objective may impart differing levels of responsibility to intelligent control.
- 1) Fixed, detailed path - keep going or stop. AGVs do this in factories.
- 2) Fixed path with speed modulation. Following behavior is a special case of this.
- 3) Follow default path with deviation to avoid obstacles permitted.
- 4) Sparse waypoints to hit with complete authority to plan the paths between them.
- 5) Cover an entire area (e.g. mow the grass).
- 6) Search for something, run from something, or pursue something.



Outline

- 7.4 Intelligent Control
  - 7.4.1 Introduction
  - 7.4.2 Evaluation
  - 7.4.3 Representation
  - 7.4.4 Search
  - Summary

7.4.2 Evaluation

- Methods to compute feasible trajectories were covered earlier in motion prediction (dynamics).
- This section is about how to evaluate trajectories.

7.4.2.1 Cost of a Configuration

- In this view, a cost can be assigned to every configuration.
  - $L[x,u,t] \rightarrow L[x]$
- This is different from cost of a point in the world
  - Vehicle occupies volume
  - Volume depends on orientation
- Can be computed efficiently in some representations and some scenarios.

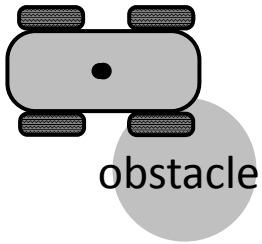
7.4.2.1 Cost of a Configuration (Volume Cost)

- Integrate cost over the volume occupied.

$$L[\underline{x}(s)] = \int_V c(x,y,z)\,dV$$

- Or intersect vehicle and obstacles.

$$L[\underline{x}(s)] = \bigcap_V \{o(x,y,z) \cap v(x,y,z)\}$$



- This can be pre-computed in static, known worlds.
  - Can also be expressed in configuration space as L(x)

An area or a volume property

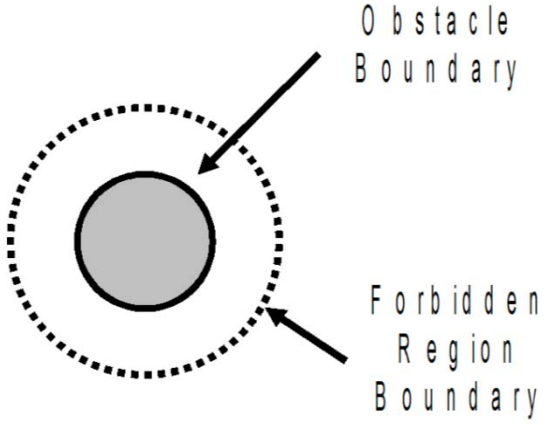
$$J[\underline{x}(t)] = \Phi[\underline{x}(t_0), \underline{x}(t_f)] + \int_{t_0}^{t_f} L[\underline{x}(t)]\,dt$$

7.4.2.1 Cost of a Configuration
(Precision Intersection Calculations)

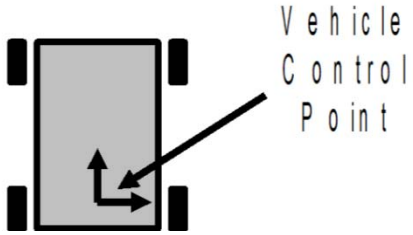
- Obstacles may place different constraints on parts of the vehicle.
  - Point hazards:
    - no part of the vehicle can drive over a 20 foot tall tree.
  - Wheel hazards:
    - wheels cannot drive over a hole - but the undercarriage can.
- Other conditions may depend on orientation.
  - Pose hazards:
    - a slope may be a problem in one orientation but not in another.

7.4.2.1 Cost of a Configuration (Volume of a Vehicle)

- A real vehicle is not a point.
  - A bad point in task space often corresponds to a bad region in state space (or configuration space).
  - Must account for the width and length of the vehicle.



- A real vehicle is not a pancake.
  - Overhanging obstacles occur in factories, warehouses, forests, buildings (tables).
  - Must account for the height of the space underneath them.



7.4.2.2 Cost of a State

(State Dependence)

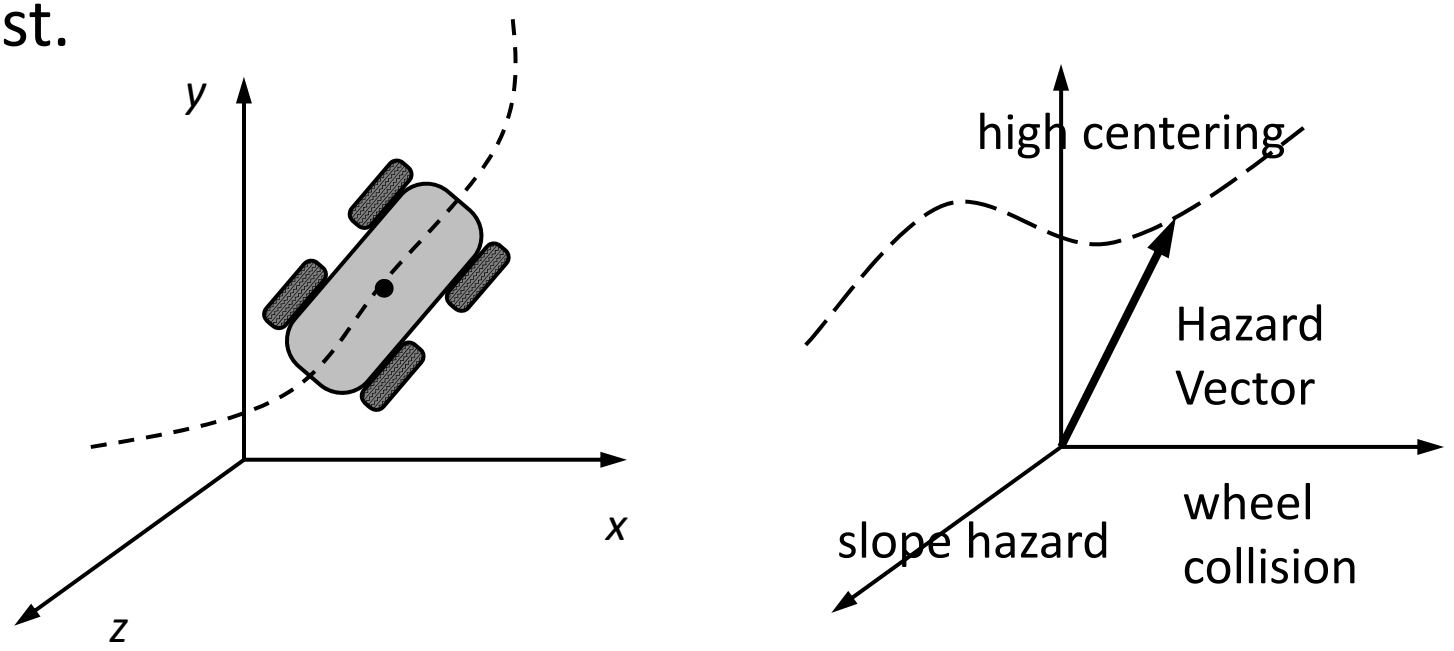
- There are many situations to be avoided
  - They depend on more than pose (e.g. V)
  - They are more properly expressed as a field over state space.
- Examples
  - Rollover relates to lateral acceleration, gravity.
  - Obstacle impact force depends on speed.

7.4.2.2.1 Types of State Dependent Hazards

- Hazards include:
  - loss of control:
    - Yaw instability (skidding); Steep slopes (braking)
  - loss of contact:
    - Rollover, high centering
  - loss of traction:
    - Ice, mud, entrapment hazards
  - collisions: application of damaging forces.
    - Will depend on speed (higher V often worse)
  - risks: uncertain situations to be avoided

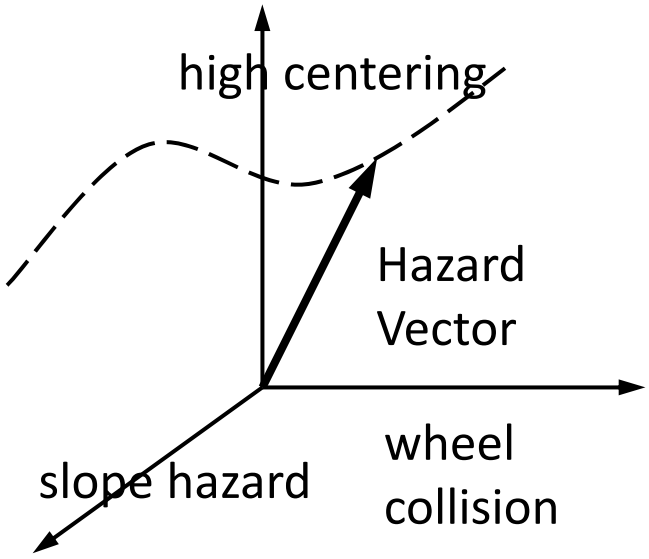
7.4.2.2.2 Hazard Space

- Hazardous states form clusters in state space corresponding to different types of interactions.
- As the vehicle moves in task space, the tip of some kind of hazard vector sweeps through hazard space.
- Comparing options requires the reduction of all hazard attributes for each time on a trajectory to a single scalar cost.



7.4.2.2.3 Consistent Hazard Units

- Must reduce all hazards to some consistent units. This may include any or all of:
  - Severity: 20 degrees is twice as bad as 10?
  - Distance: to go around versus effective distance over.
  - Energy consumed
  - Uncertainty
- Then the length of the vector is meaningful.



7.4.2.3 Cost of a Path

- In optimal control terms, this is our old friend:

$$J = \phi[x(t_f)] + \int_{t_0}^{t_f} L(\underline{x}, \underline{u}, t)dt$$

- Maybe just add up the hazard score along the path?
- Probably makes sense to weight less as distance increases?

7.4.2.4 Models Used in Objectives and Constraints

- Models may be used in the computation of any or all of:
  - Motion Generation
  - Constraints
  - Objective functions

	Attribute Used to generate a motion	**Attribute Used in a Constraint**	**Attribute Used in Objective Function**
Vehicle Model	State (for motion prediction)	volume of vehicle (for collision constraints)	power consumption wheel slip maneuver aggressiveness
Environment Model	Terrain shape or mechanical properties.	volume of obstacles (for collision constraints)	Proximity to obstacles

Outline

- 7.4 Intelligent Control
  - 7.4.1 Introduction
  - 7.4.2 Evaluation
  - 7.4.3 Representation
  - 7.4.4 Search
  - Summary

7.4.3 Representation

(Trajectory Evaluation)

- Methods to compute feasible trajectories were covered earlier in motion prediction (dynamics).
- This section is about how to evaluate trajectories.
- Before we can cost a path, we must cost a point on a path. That means we must model:
  - The path
  - The vehicle
  - The environment

7.4.3.1.1 Motion Constraints

- Dynamic constraints:

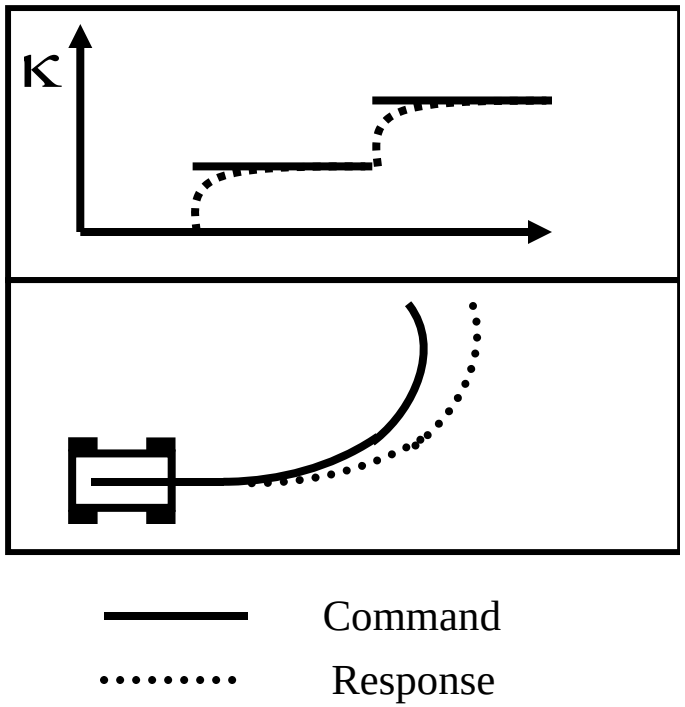
$$\dot{\underline{x}} = f(\underline{x}, \underline{u}, t)$$

- Input/actuation constraints $\underline{u} \in U$

$$|\kappa(s)| < \kappa_{max} \qquad |\dot{\kappa}(s)| < \dot{\kappa}_{max}$$

7.4.3.1.2 Representation of Paths and Trajectories

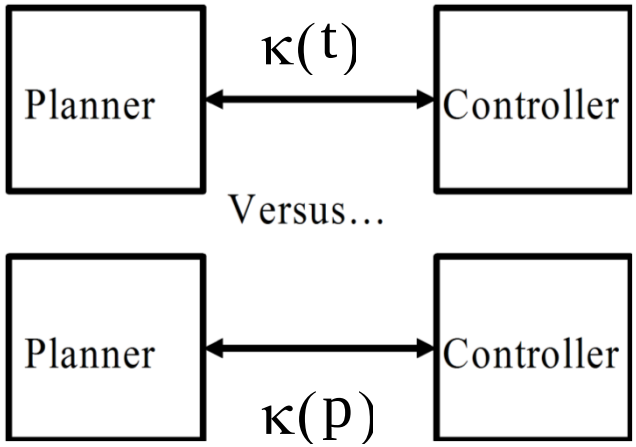
- Two options at least are:
  - Input history: $\underline{u}(t)$
  - State history: $\underline{x}(t)$
- u → x is easy
- x → u is hard (but we know how)
- Sampled versions of these are common:
  - Sequence of curvatures is a sampled $\underline{u}(t)$
  - Ordered sequences of cells or points is a sampled $\underline{x}(t)$



7.4.3.1.3 Compactness and Completeness

of Motion Representations

- Compact representations can streamline communications to low level control.

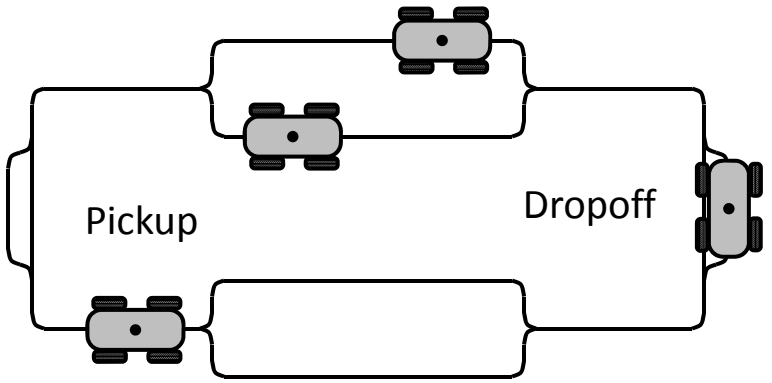


Commanding an entire trajectory (rather than instantaneous curvature) means you only have to do so occasionally.

It also gives lower levels the opportunity to do predictive control.

Motivation: Offline Representations

- Trajectories sometimes must be stored in some off-line representation.
- For AGVs, clothoids, lines, and arcs are a common library of trajectory shapes.
- Complex shapes can always be approximated by a sequence of simpler ones.
- Its a good thing if the representation is both complete and compact



AGV guidepaths can be represented as a sequence of trajectories rather than a large set of points or poses..

7.4.3.2 Representing Configurations

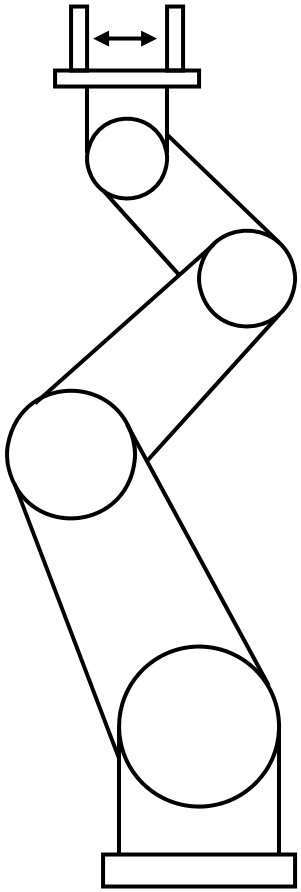
- State space is good for
  - predicting motion.
  - assessing certain hazards.

- Configuration space is good for encoding
  - Articulation
  - Occupied Volume

7.4.3.2 Representing Configurations

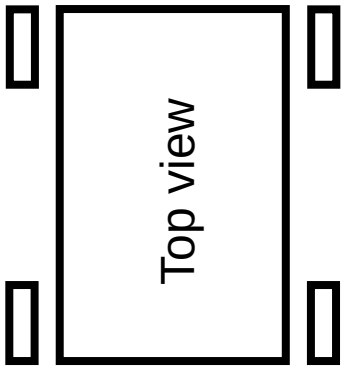
(C Space Definitions)

- A configuration of an object is a specification of the position of every point on the object (with respect to a fixed frame of reference).
- A Configuration Space is the space (i.e. set) of all configurations of the object.
- Informally, this is a set of generalized coordinates which completely determine the position of every point on the object.



7.4.3.2.2 C-Space Dimension

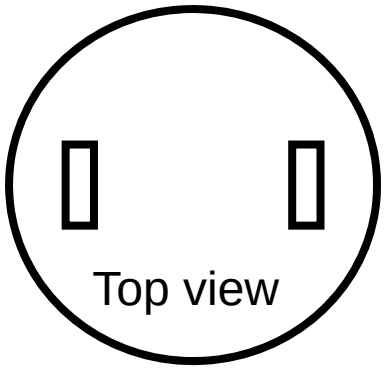
- The number of generalized coordinates required is the dimension of the C Space.
  - Articulations add to the C space dimension.
  - Constraints reduce it.



7.4.3.2.2 C-Space Dimension

(Computing C Space Dimension)

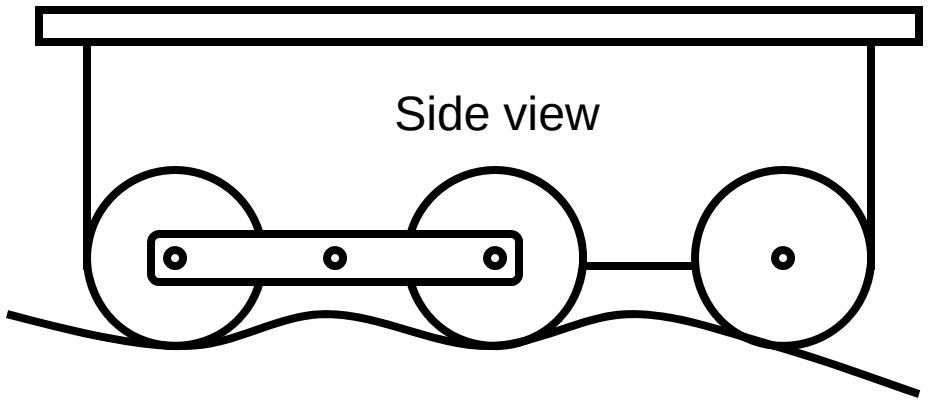
- Start by adding the number of spatial dof of each rigid body comprising the object.
- Then, impose the constraints of articulation
  - kneebone connected to the….
  - → including terrain following



7.4.3.2.2 C-Space Dimension

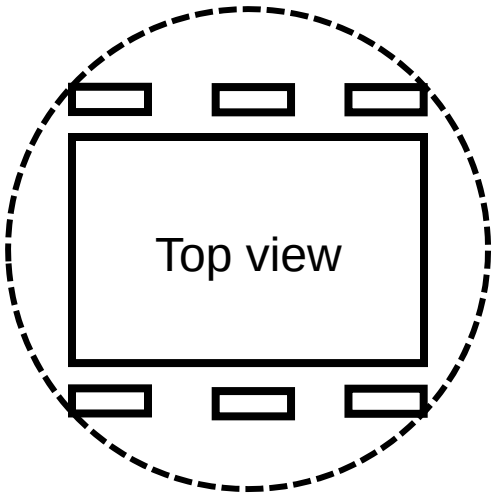
(Symmetry)

- Some dof (e.g. wheels) do not change the occupied volume when they articulate.
- Hence, irrelevant to collision detection.
- Usually, we remove them from the representation.
  - More efficient for Planning
  - Formally however, these dof are still dimensions of C space

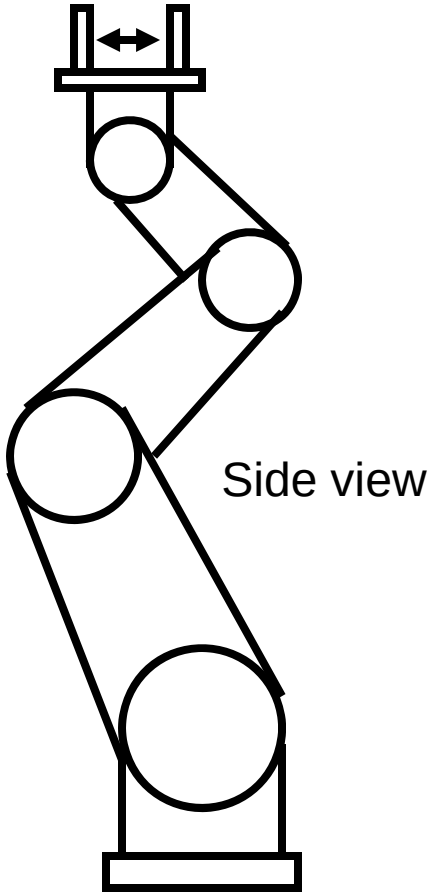


Computation

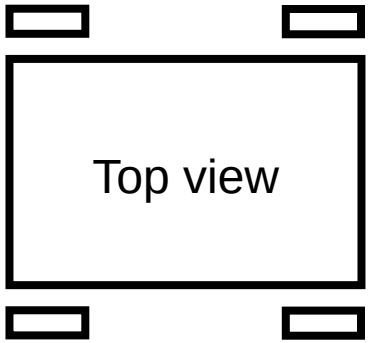
- Computational complexity of search is directly related to:
  - the dimension of C Space
  - the complexity of the obstacles
- At times, it is valuable to approximate a robot shape by a symmetric one (say, by a circle) in order to reduce the dimension of C space.

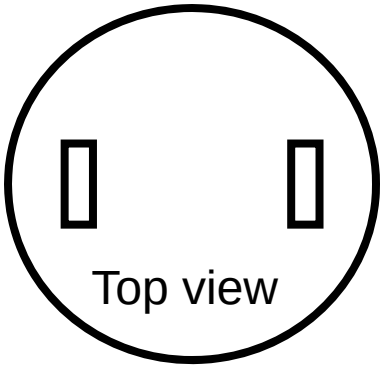


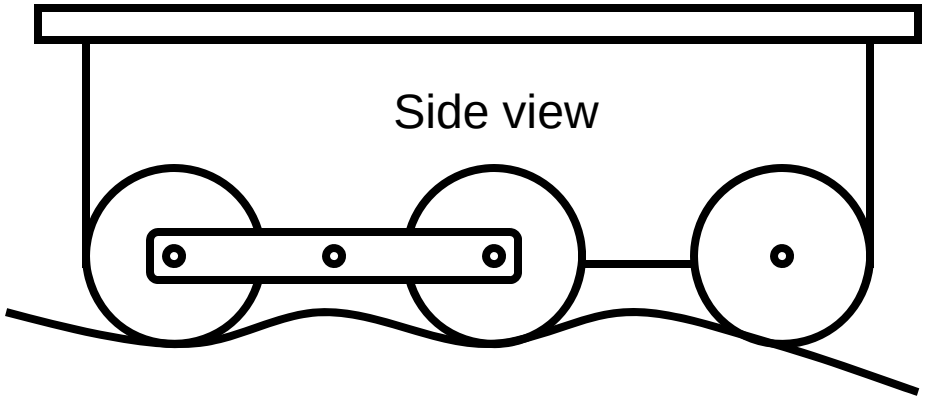
How Many DOF Should We Represent?



Side view







7.4.3.3 Representing the Environment

(Environmental Attributes for Costing)

- This section is about
  - Encoding spatial properties of the environment for trajectory evaluation purposes.
- This section is not about representing elevation for purposes predicting motion.
  - However, both can sometimes be stored in the same data structure.

7.4.3.3 Representing the Environment

(Discriminators for Representation)

- Dynamic range (discrete / continuous / mixture)
- Memory requirements
- Efficiency of intersection calculations
  - Affects collision detection efficiency
- Gradient information available ?
  - Affects search efficiency

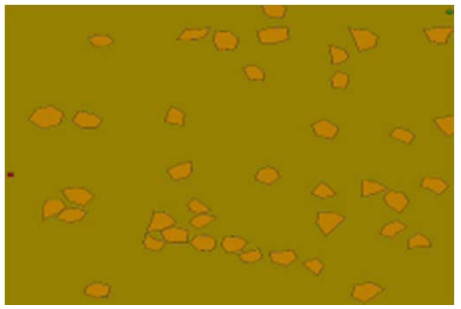
7.4.3.3 Representing the Environment

(Obstacles" and Costs)

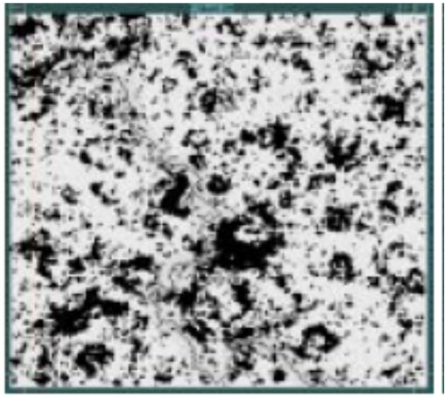
- Literally "obstacles" are impediments to motion.
  - You cannot drive through them
- Robotics thinks of them as places you should not drive
  - Even if you can drive through them
- Assigning a cost or "relative obstacle severity" is also common.

7.4.3.3.1 Set and Field Representations

- Sets → map an object index onto a region of space
  - Typically cost is uniform in the region and binary (meaning obstacle or not)
  - Represent obstacles as objects (which happen to occupy space)
  - Represent position, and perhaps shape as volume or boundary.
  - Can be memory efficient but computationally expensive.
- Fields → map a point in space onto a cost
  - Represent large spatial region as a raster or array.
  - Associate a utility or cost with every point in the workspace.
  - Can be memory inefficient but computationally cheaper than sets.



Set of Binary Obstacles

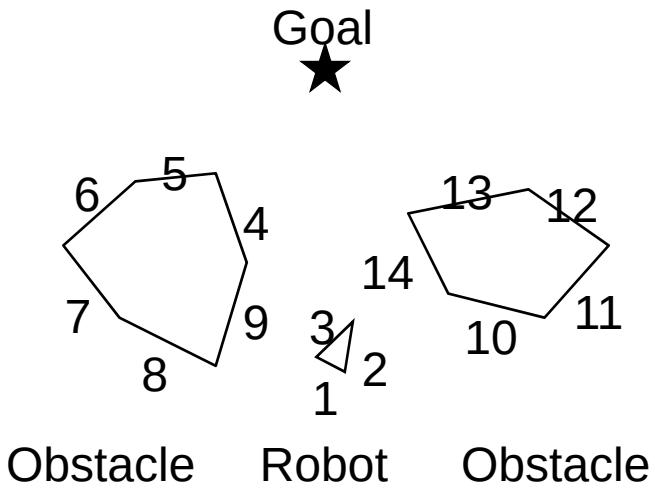


Cost Field

7.4.3.3.2 Shape Representations.

(Boundary Representations [for Sets])

- Collision checking involves checking for boundary intersections.
  - Num_robot_edges * num_obstacle_edges computations without bounding boxes.
- Hence, the planning computational complexity depends on the number of obstacle edges.
- Bounding shapes can be used to quickly eliminate unnecessary intersection checks.

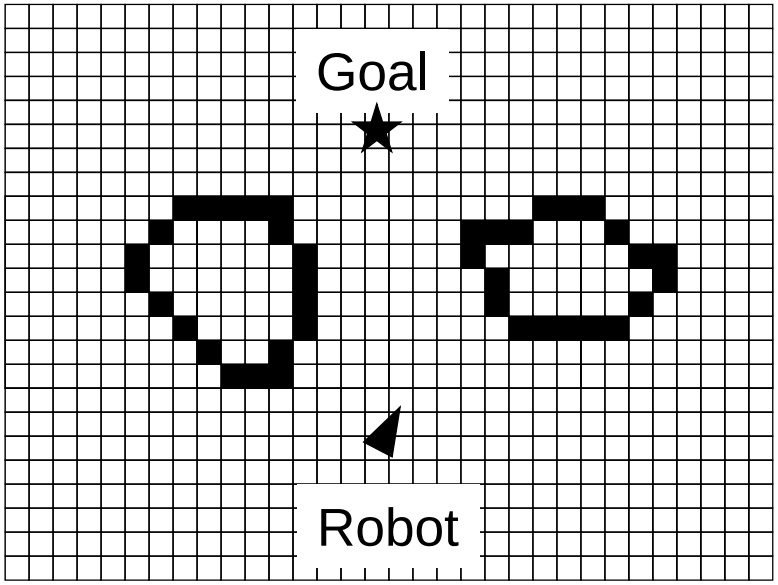


7.4.3.3.3 Obstacles Versus Free Space

- It may be better to represent free space rather than obstacles.

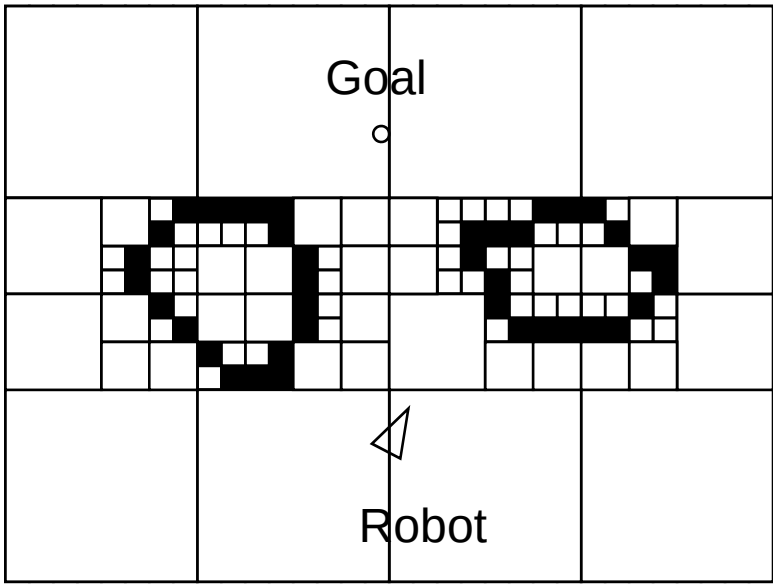
7.4.3.3.4 Sampled Versus Continuum

- Sampled may be best alternative in complex environments
  - i.e. whose continuous representation would be large.
- Intersection calculation is an AND of two rasters.
- Planning computational complexity tends to depend on the resolution of the representation.



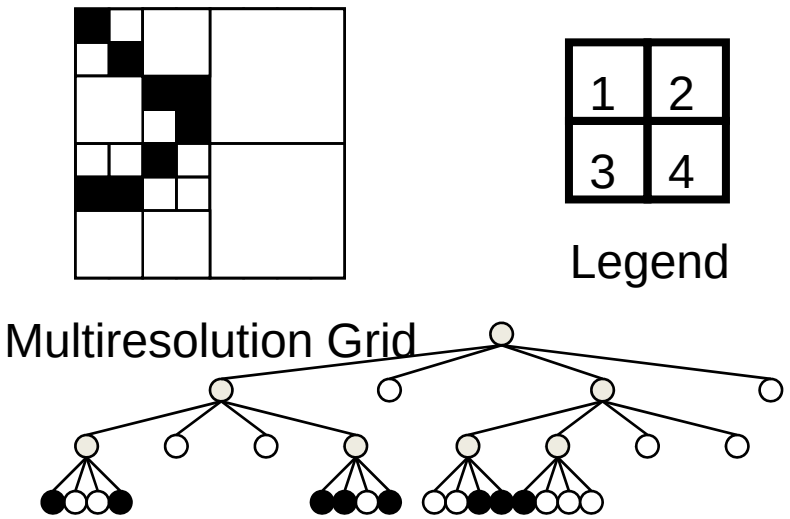
7.4.3.3.5 Hierarchy and Quadtrees

- A popular approach to reducing memory is a kind of hierarchical grid called a quadtree (octree in 3D).



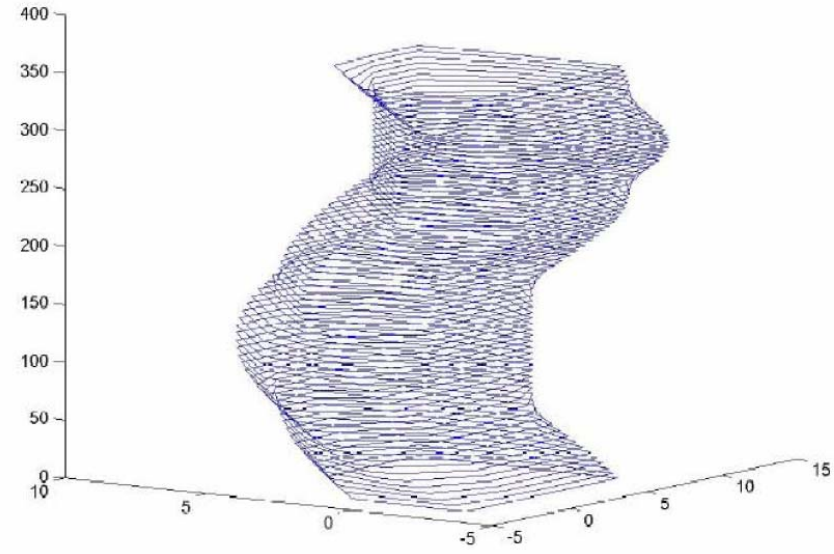
7.4.3.3.5 Hierarchy and Quadtrees

- A tree of nodes that are:
  - filled,
  - unfilled
  - partially filled
- Only the partially filled ones at each level are elaborated.



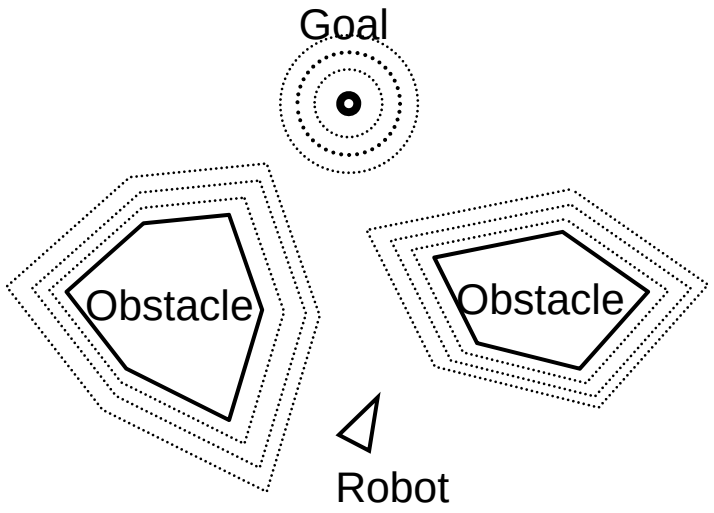
7.4.3.4 Derived Spatial Representations

- Representations derived from the basic geometric or cost information can be useful.
- May make collision checking more efficient.



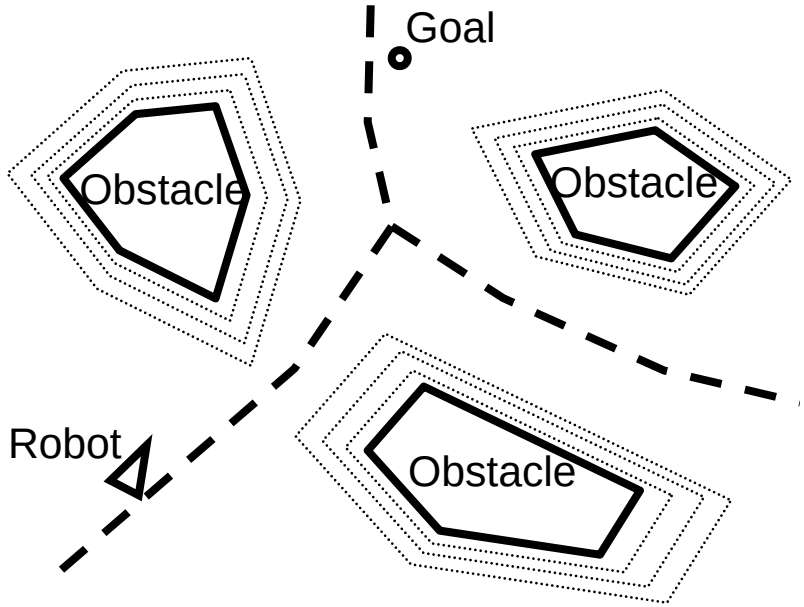
7.4.3.4.1 Potential (& Proximity) Fields

- Proximity (minimum distance to a collision) is a special derived field.
- Individual potentials can simply be added or perhaps combined in more principled ways.
  - For example, a proximity field can be formed as the min distance to any collision.
- Controls, policies, inputs etc. are derived from the field at the present position (e.g follow the gradient).
  - Such representations have a well defined gradient and they can be used in relaxation based search as well as sequential search.



7.4.3.4.2 Voronoi Diagrams

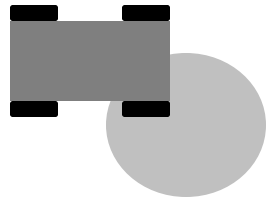
- Subspaces of the original space.
- Can be generated from a field representation.
- Set of all points which are equidistant from at least two obstacle boundaries.
- Local maxima in the proximity field.



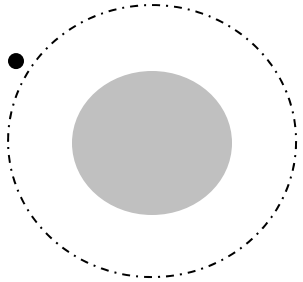
7.4.3.4.3 C-Space Obstacles

(Mapping Volumes to C Space)

- C Space obstacle = set of configurations where a collision in the workspace takes place
  - Compute it with boundaries
  - Compute it with volume intersection
- Precomputes the intersection calculation
  - So its only done once.
- Can also be done as a volume integral for continuous costs
- Not worth it
  - When the environment is dynamic or sensed with noise
  - When only a small region of the workspace will need to be tested



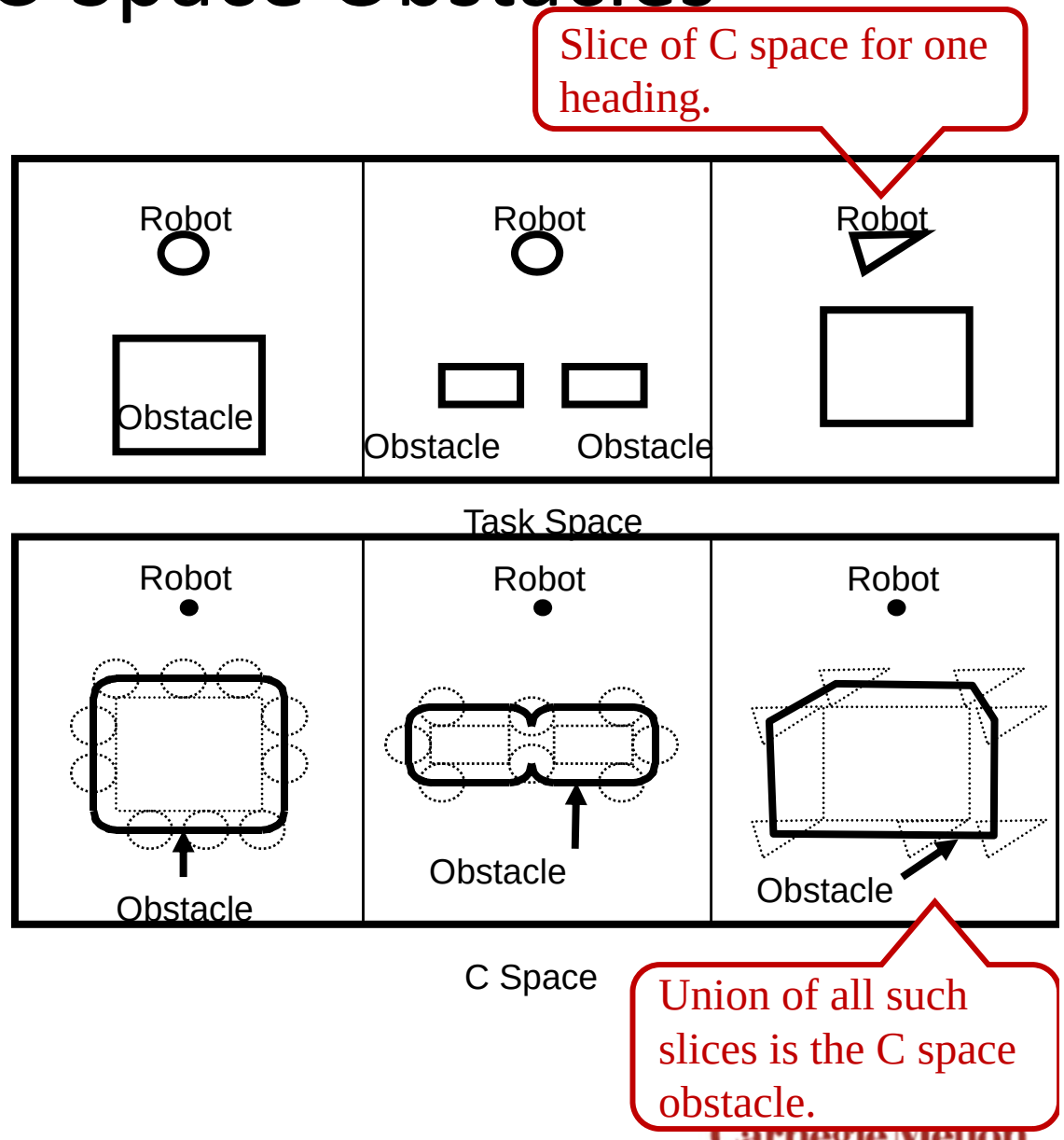
Workspace



Cspace

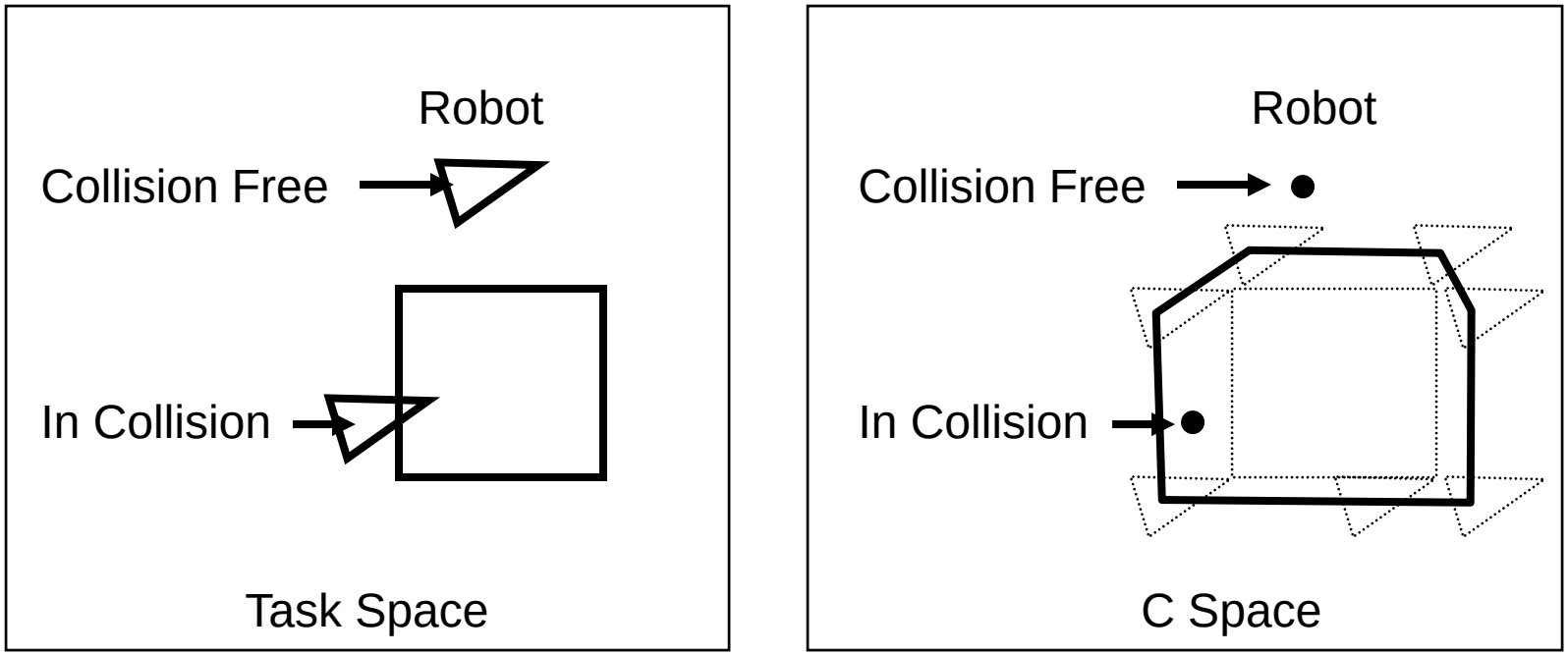
7.4.3.4.3 C-Space Obstacles

- Boundaries of obstacles in the environment can be converted to equivalent obstacles in C Space.
- For every point on the boundary of an obstacle, compute every configuration of the robot which can be in contact with that point.
- Its a property of both the robot shape and the obstacle shape.



7.4.3.4.3 C-Space Obstacles

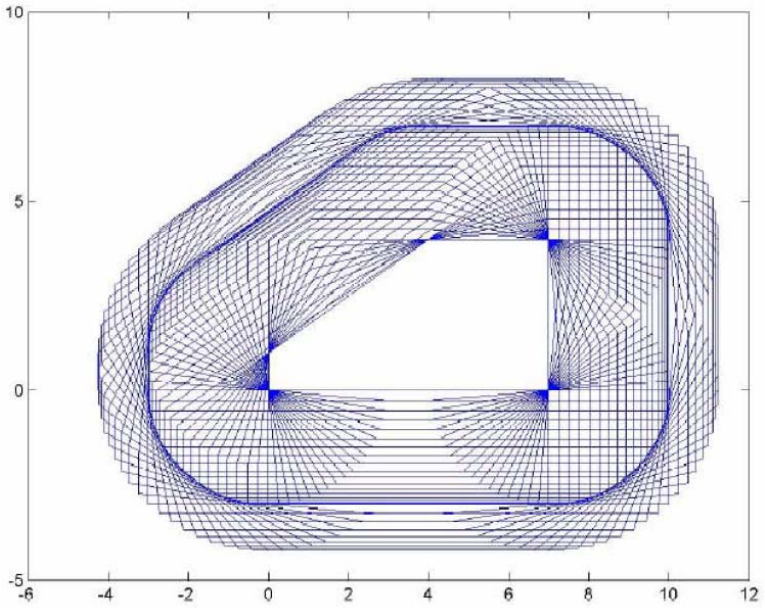
- Now, collisions can be detected cheaply:
  - ask if the point robot is inside a C space obstacle.

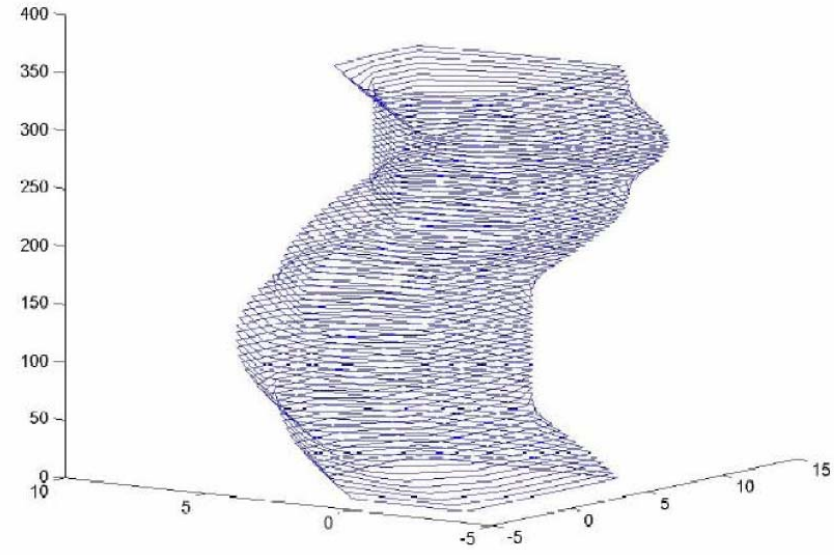


7.4.3.4.3 C-Space Obstacles

(Dimension = 3)

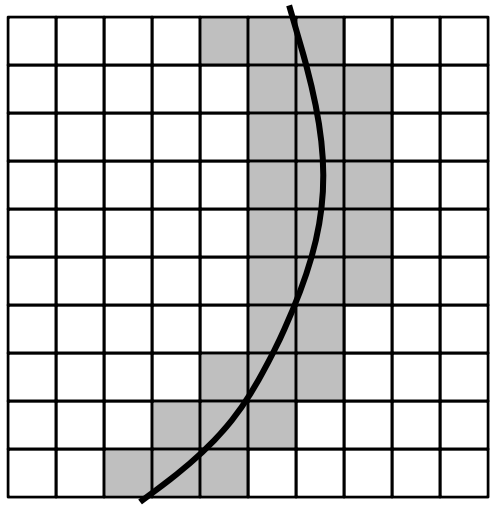
- C Space obstacles can sometimes be hard to compute explicitly.





C Space and Precomputation

- Cost precomputation→ when the environment is static and known
- Cell precomputation→ can be done even if costs are changing.



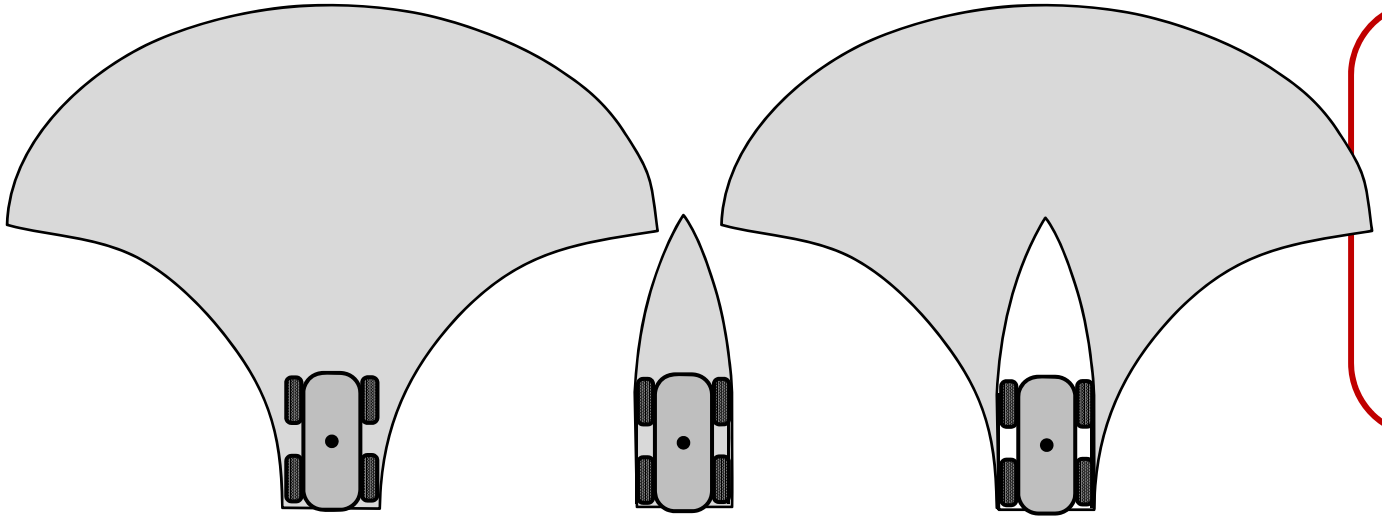
State Space → Workspace Mapping

- Just like
  - obstacles in the workspace can be mapped to regions in C space
  - Regions and hazards in workspace can be mapped to state space

7.4.3.4.4 Partitions of State Space and Work Space (Committed Motion)

- Can segment space in a limited horizon reachable region.
  - Reachable = some point on the vehicle can reach it for some motion.
  - Committed = some point on the vehicle will reach it for every motion.
  - Avoidable = Reachable – committed.

7.4.3.4.4 Partitions of State Space and Work Space (Committed Motion)



reachable committed avoidable

Committed region :

a) grows quickly with speed
b) Bends with initial curvature

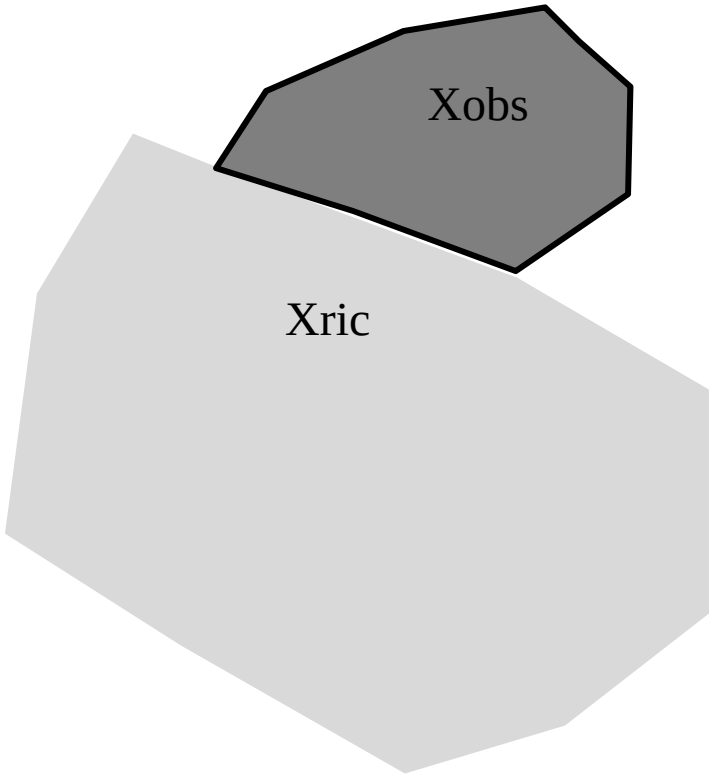
Farther forward Close to centerline

Farther backward Far from centerline

Such a figure can also be drawn for different classes of trajectory (stopping, turning)

7.4.3.4.4 Partitions of State Space and Work Space (Regions of Inevitable Collision)

- State space obstacles.
- RIC = set of all initial states from which entry into an obstacle must eventually occur.
  - Pick a state in Xobs
  - Solve DE backwards for all possible controls to get there.
  - All states occupied for some Xobs and all controls are in Xric



7.4.3.4.5 Incorporating Risk and Uncertainty
(Sources of Uncertainty)

- Interpretation: Assessments of hazards are not accurate.
- Sensing: Localization error implies obstacles may be incorrectly located relative to the vehicle.
- Motion: There is no guarantee that motion prediction/control will do what was planned or predicted.

7.4.3.4.5 Incorporating Risk and Uncertainty
(Techniques for Coping)

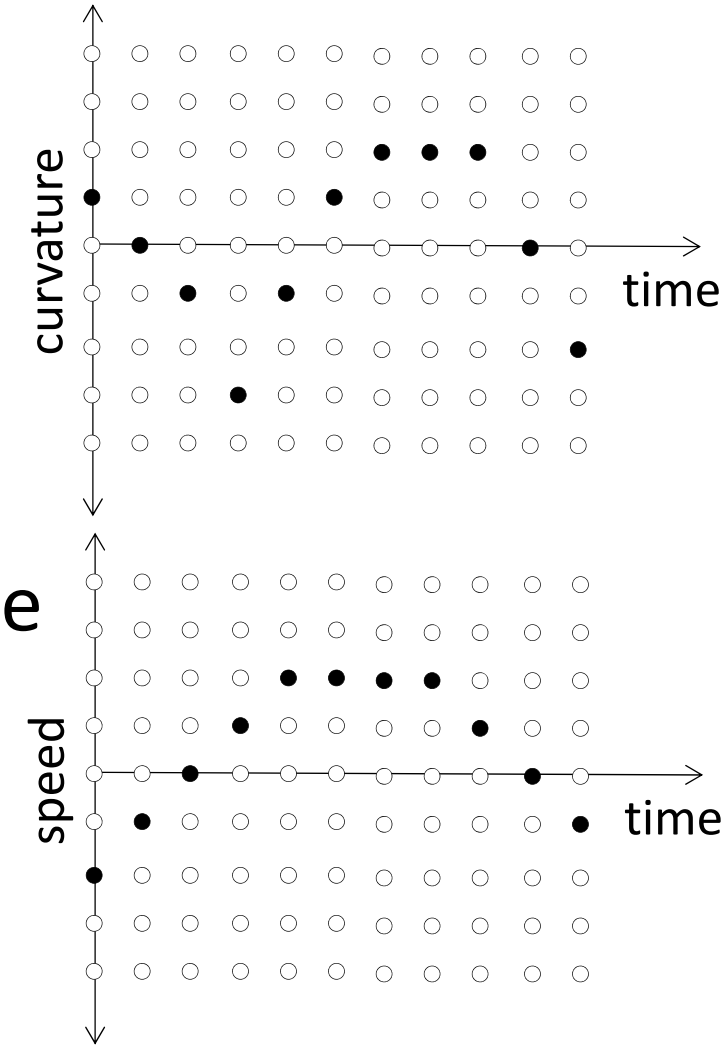
- Margin: Use a deliberately oversized vehicle when assessing collisions.
  - This means you won't be able to squeeze through tight spaces.
- Oversize the Obstacles. Filter the environmental representation to cause high cost to bleed into adjacent areas.
- Explicitly compute motion uncertainty and map uncertainty and do both of the above.

Outline

- 7.4 Intelligent Control
  - 7.4.1 Introduction
  - 7.4.2 Evaluation
  - 7.4.3 Representation
  - 7.4.4 Search
  - Summary

7.4.4.1 Sampling, Discretization, and Relaxation
(Input Discretization and Parameterization)

- In full generality, there is a function space u(t) to search.
- Discretization and parameterization are two options.
- For 10 signal levels and 40 time samples, there are 10^{40} alternatives.
  - Not feasible to search at 10 Hz.



7.4.4.1 Sampling, Discretization, and Relaxation
(Continuum vs Sampling)

- Sampling
  - Avoids local minima
  - Inefficient/impossible in dense obstacles

- Relaxation (Continuum)
  - Finds only local minima
  - Very efficient in dense obstacles
  - Requires gradient information.

7.4.4.2 Constraint Ordering

(Ordering Feasibility and Admissability Constraints)

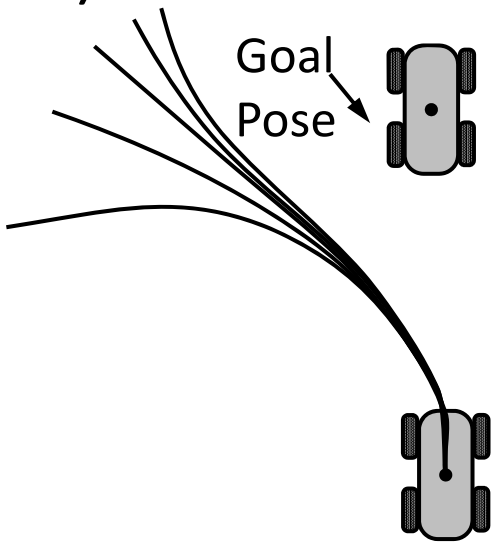
- Option 1:
  - Find good places
  - See if you can go there
- Option 2:
  - Find places you can go
  - See if they are good.

- Imposing the most limiting constraint first is often most efficient.

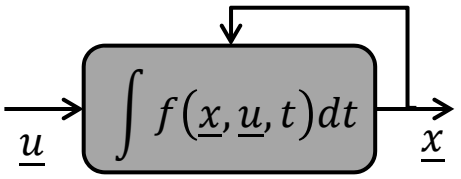
7.4.4.2 Constraint Ordering

(Constraint Ordering Dilemma)

- In which space should we conduct search (= express alternatives)?
- Easy in Input/Control/Action Space:
  - Dynamic feasibility.
  - Actuation limits (e.g turn radius).
- Easy in Work/State/C Space:
  - Obstacle Intersection
  - Following Global Guidance
  - Enforcing workspace constraints.
  - Ensuring good separation.
- What's easy in one is hard in the other.



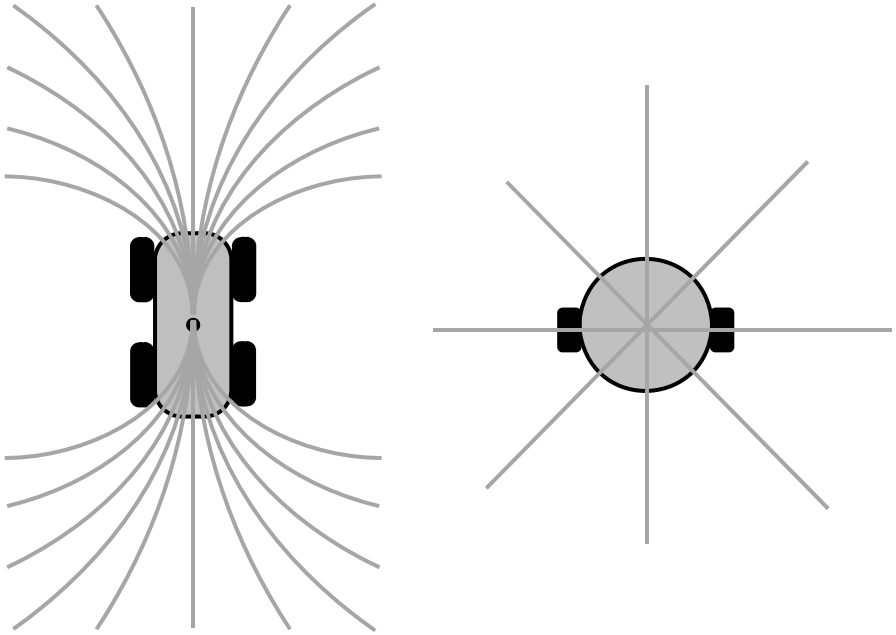
$Forward: \dot{\underline{x}} = f(\underline{x}, \underline{u}, t)$



$Inverse: \underline{u} = ?(\underline{x})$

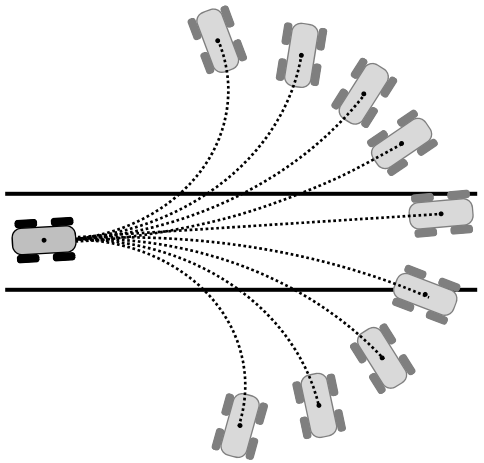
7.4.4.2.1 Search Coordinates

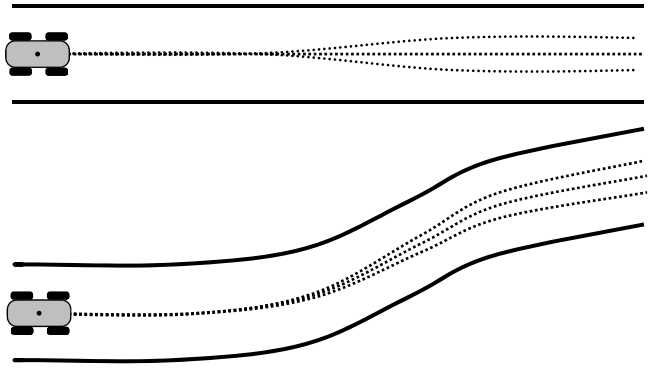
- There is value in artificially limiting mobility.
- Almost any vehicle can be driven by considering only arcs.
- A differentially steered vehicle can be driven by considering compositions of point turns and line segments

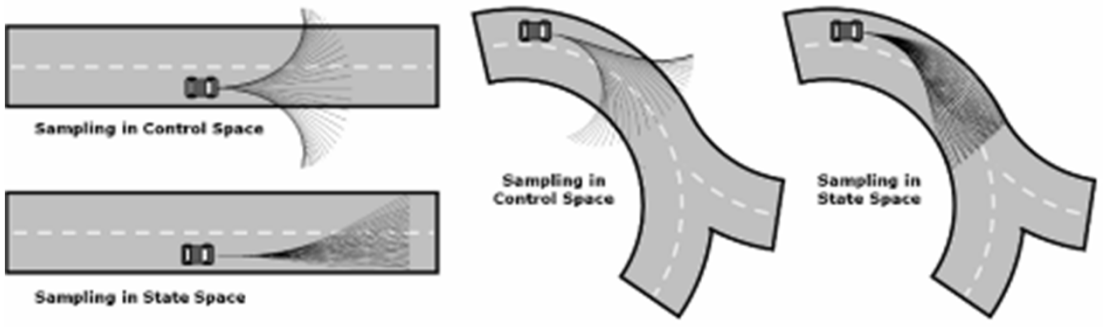


7.4.4.2.2 Environmental Constraints and Guidance

- Sometimes there is value in limiting maneuverability artificially to respect and exploit environmental structure.
  - Admissability first.
- This focuses the search and eliminates wasted computation.

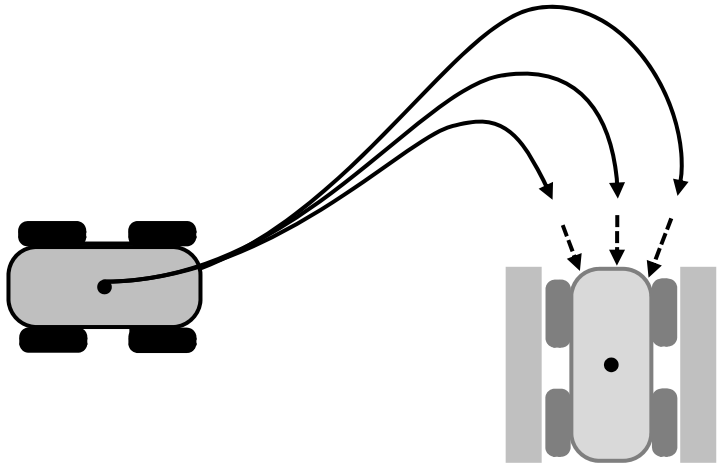






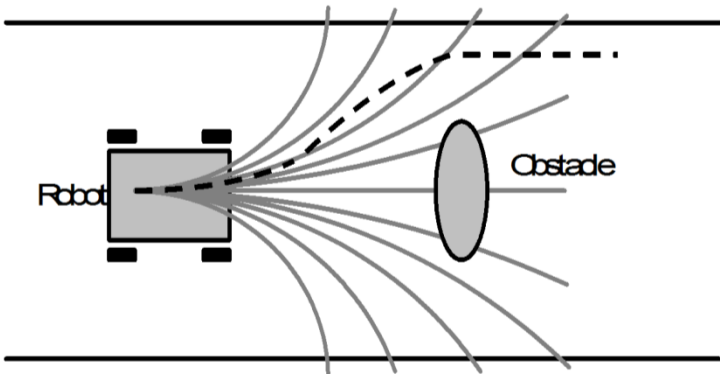
7.4.4.2.2 Environmental Constraints and Guidance (Global Guidance)

- Need to hit the terminal state fairly precisely to make the tight maneuver.



7.4.4.2.2 Environmental Constraints and Guidance (Workspace Constraints with Obstacles)

- Here workspace constraints and admissibility leave only a small region in trajectory space.
  - feasible
  - admissable



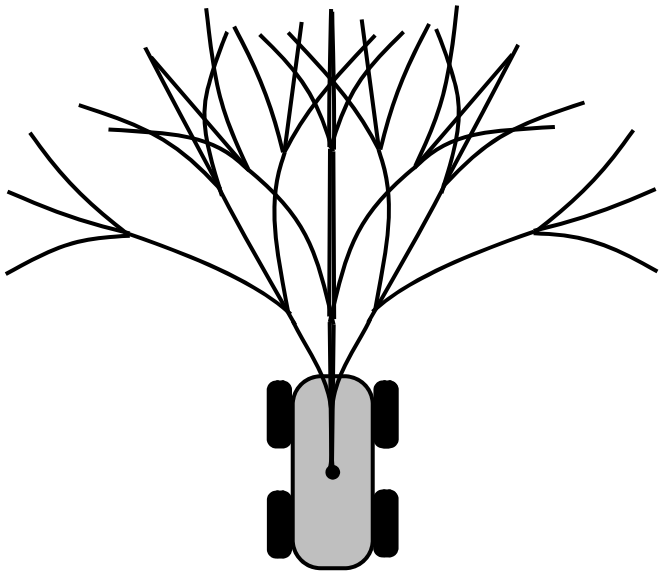
The problem of avoiding the obstacle and staying on the road is solveable - but solution is not in the space of arcs. Only a compound turn (left, then right) will work.

7.4.4.3 Efficient Search

- Above section tries to conduct the search in a space that satisfies the constraints intrinsically.
- This section looks at how to accelerate the search itself.

7.4.4.3.1 Mitigating Effects

- Distinct inputs do not necessarily generate very distinct outputs.
- The environmental representation is not of infinite resolution.
  - So a continuum search is not necessary.
- Often there are many solutions and any one is good enough.
- Sometimes can search in priority order (e.g. straight first) in sparse obstacles.



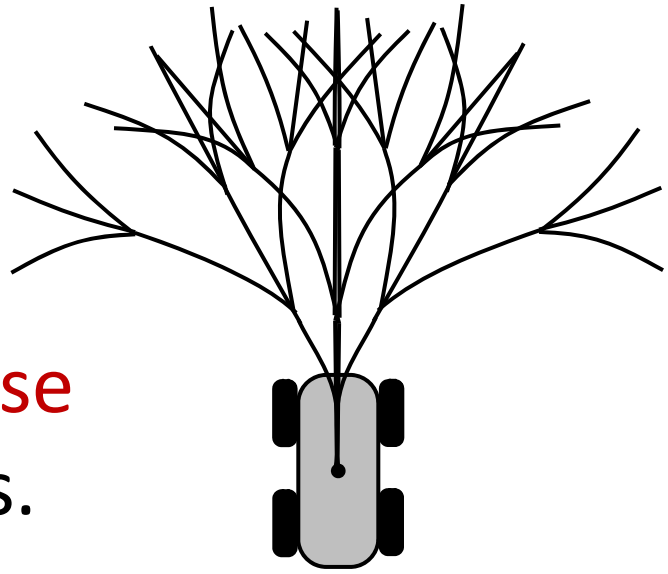
7.4.4.3.2 Reusing Computations

- When the environment is cluttered, search efficiency matters more.
- Can exploit tree structure to reuse the component path evaluations.
- Total length opposite is:

$$(3 + 3^2 + 3^3)\frac{s}{3} = (1 + 3 + 3^2)s = \boxed{13s}$$

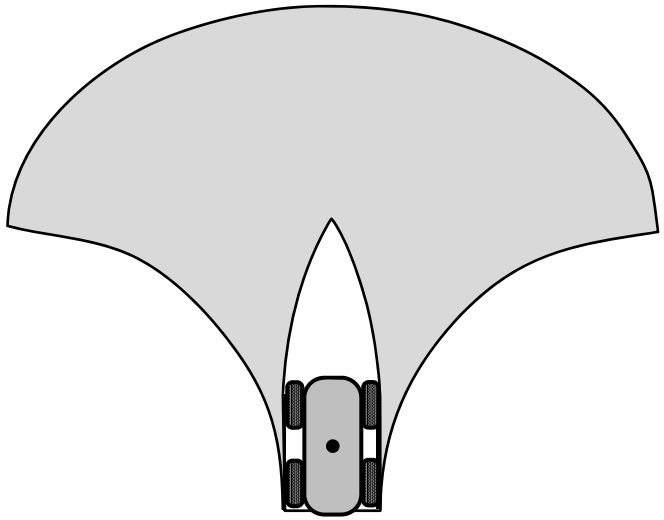
- Whereas 27 paths require:

$$3^3 s = \boxed{27s}$$



7.4.4.3.3 Exploiting Committed Motion

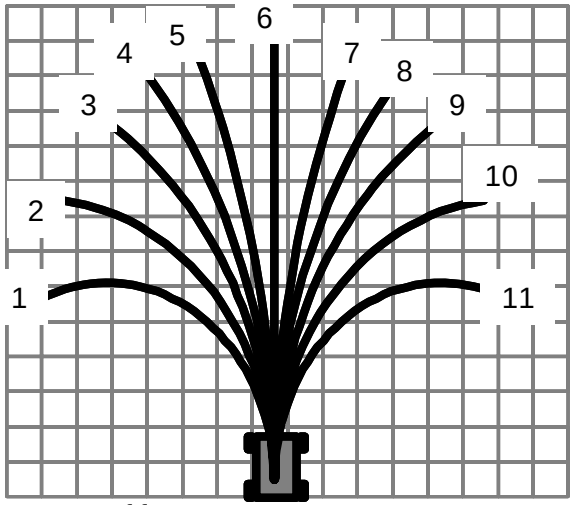
- Makes no sense to search for obstacles in committed region.
- Makes a big difference at high speeds.

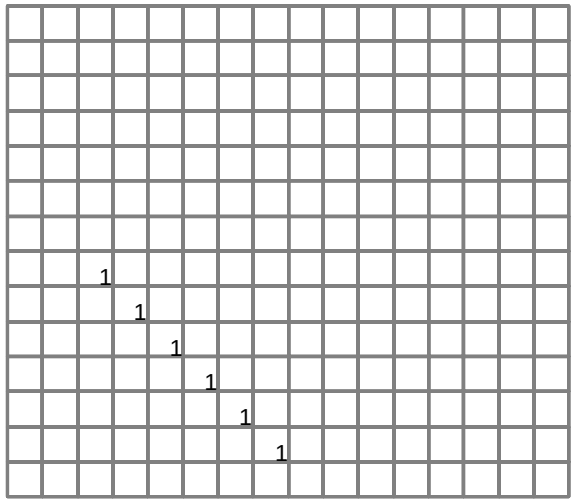


avoidable

Trajectory Caching

- Invert the input-to-state mapping in a lookup table:
  - Assumes fixed terrain shape
  - May be more efficient to visit each map cell rather than each possible input.





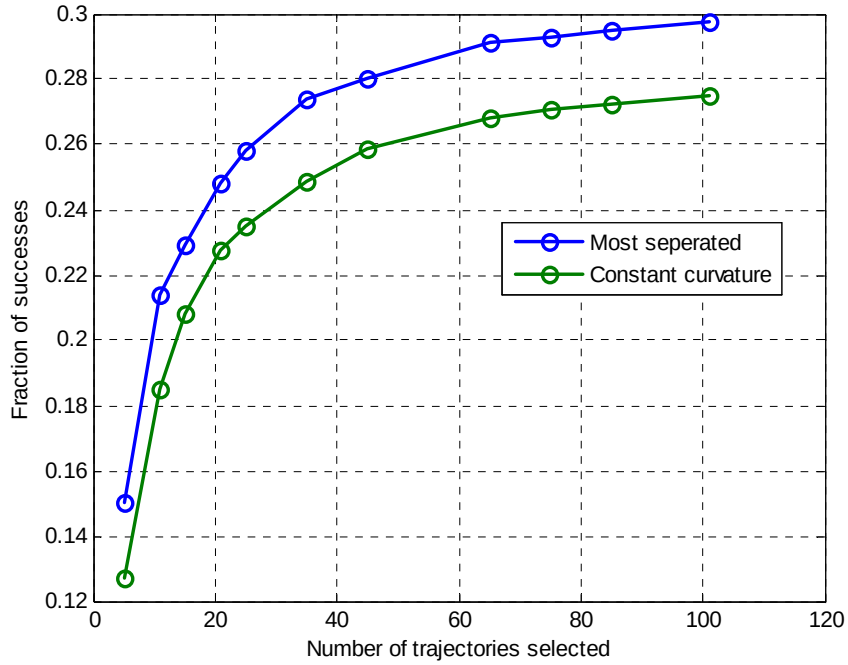
- This gives "Input space obstacles" (c.f. C space).

7.4.4.4 Search Space Design

- Tradeoffs and desirable characteristics of the search space itself.

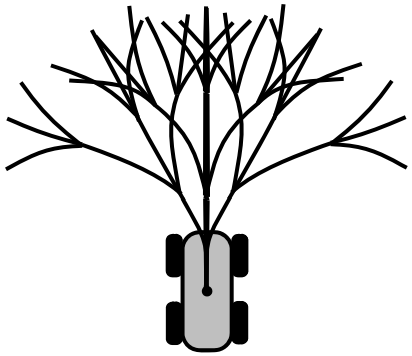
7.4.4.4.1 Mutual Separation

- Not all search spaces are created equal.
  - More “separated” is statistically better.
- What’s more…
  - Relaxation of a finite set of alternatives can improve matters dramtaically.

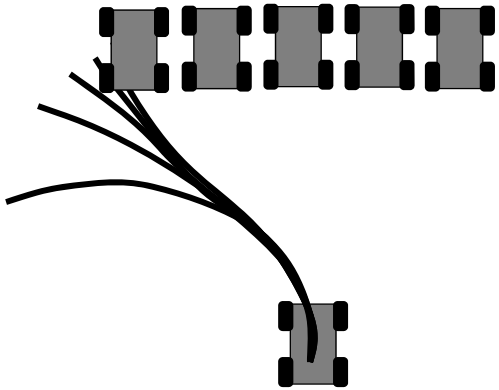


Feasibility Versus Separation

- Would like to:
  - Span the space of feasible motions (arcs do not!)
  - While sampling as uniformly as possible.
- Problem:
  - First is easy to do in input space.
  - Second is easy to do in state space.



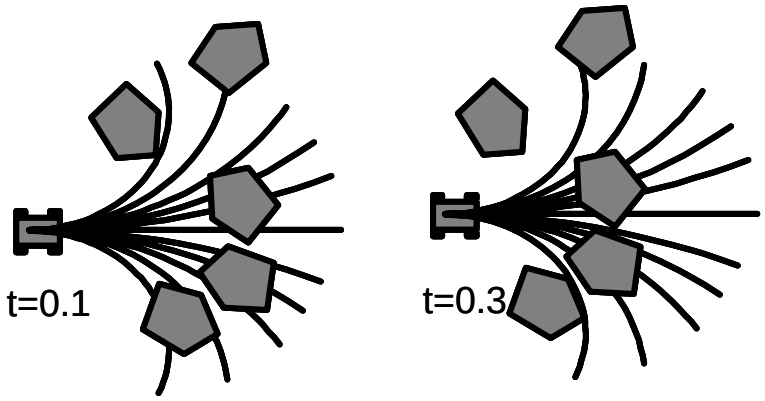
Feasible, Not Well Separated

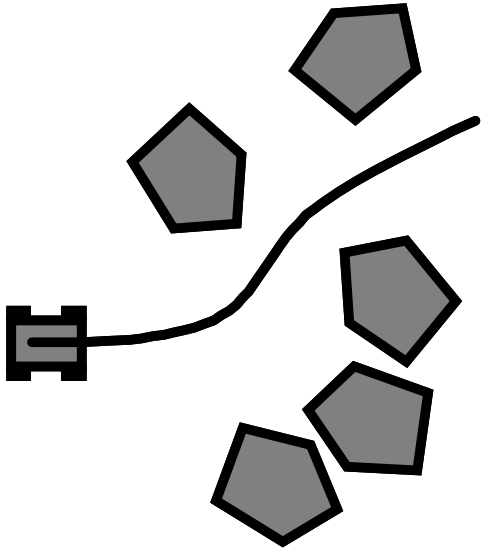


Well Separated,. Not Feasible

7.4.4.4.2 Completeness

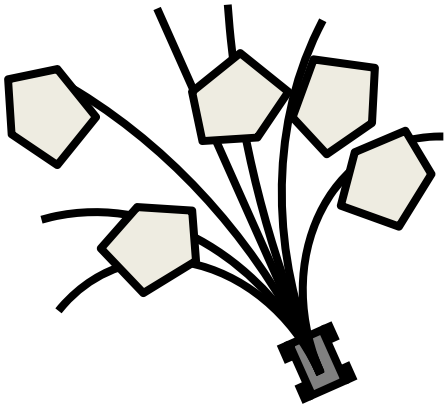
- The natural assumption that:
  - any set of reasonable trajectories ...
  - ... searched often enough ...
  - ... can generate any path is ...
- WRONG
- Solutions must be safe as far as the stopping distance even if only a small amount will be executed.
  - Because you may not have the option to change your mind.



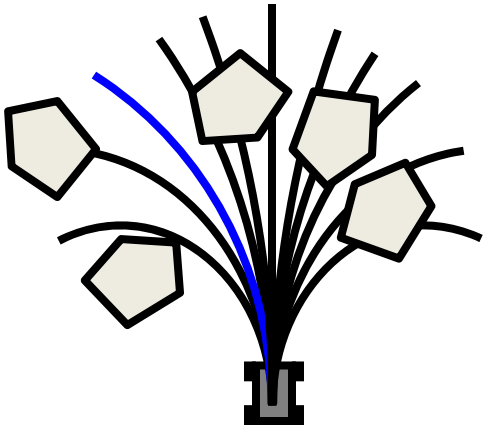


7.4.4.4.3 Robustness to Control Uncertainty via Persistence

- Plan instability can cause a rare solution to be lost. Two solutions:
- Persistence:
  - Make sure the next search includes the last solution in case it's the only one.
- Relaxation
  - Deform the search space to regenerate the old solution.



t=0.3

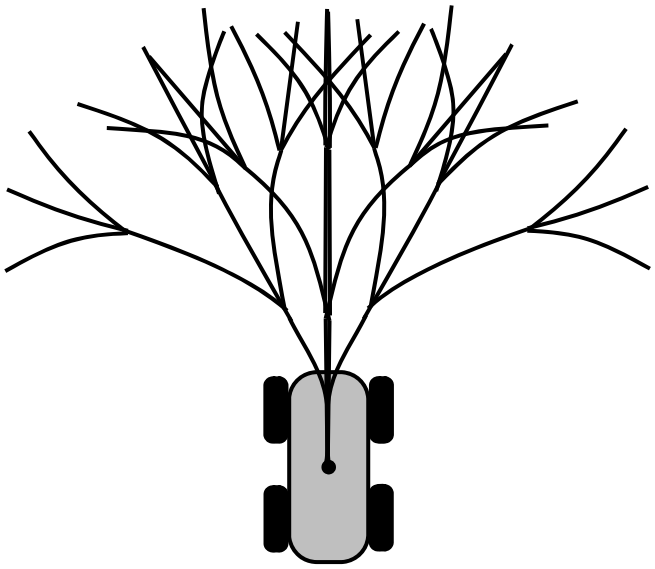


t=0.1

7.4.4.4.3 Robustness to Control Uncertainty

(Search Space Persistence)

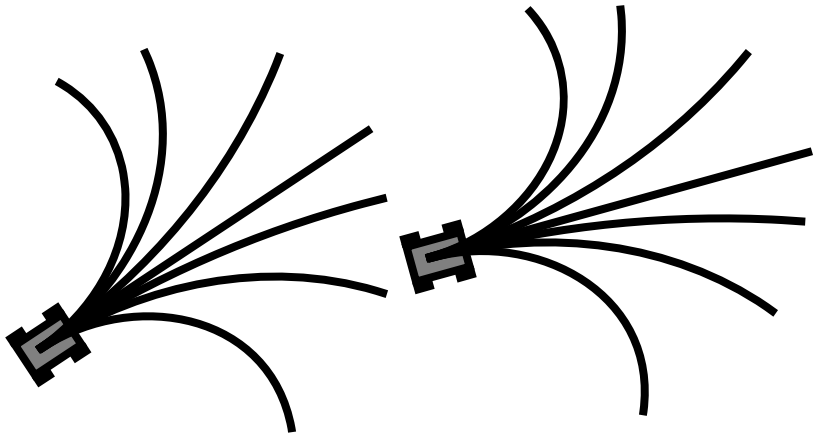
- Simple technique is to start next iteration from intended pose rather than actual pose.
- Special case:
  - Execute one segment at right and then replan from the fork point.



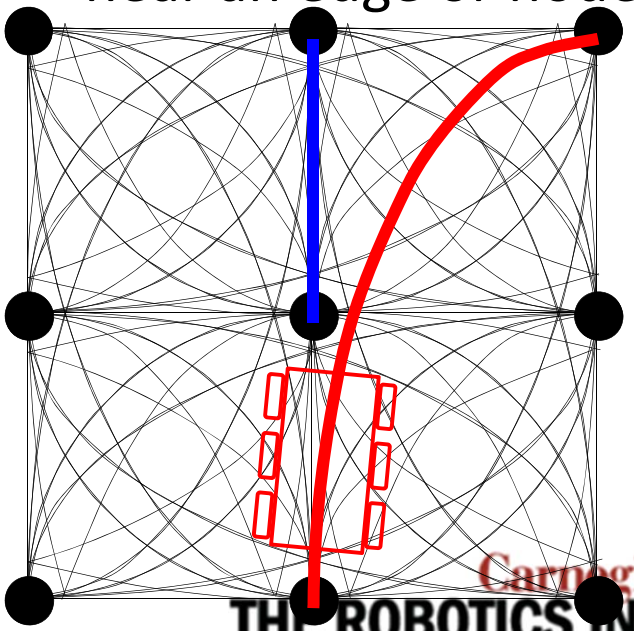
7.4.4.4.3 Robustness to Control Uncertainty

(Search Space Persistence)

- Search spaces fixed to robot (RHMPC)…
  - are not stable
  - hard to reuse computation
  - BUT robot is always on an edge and node.



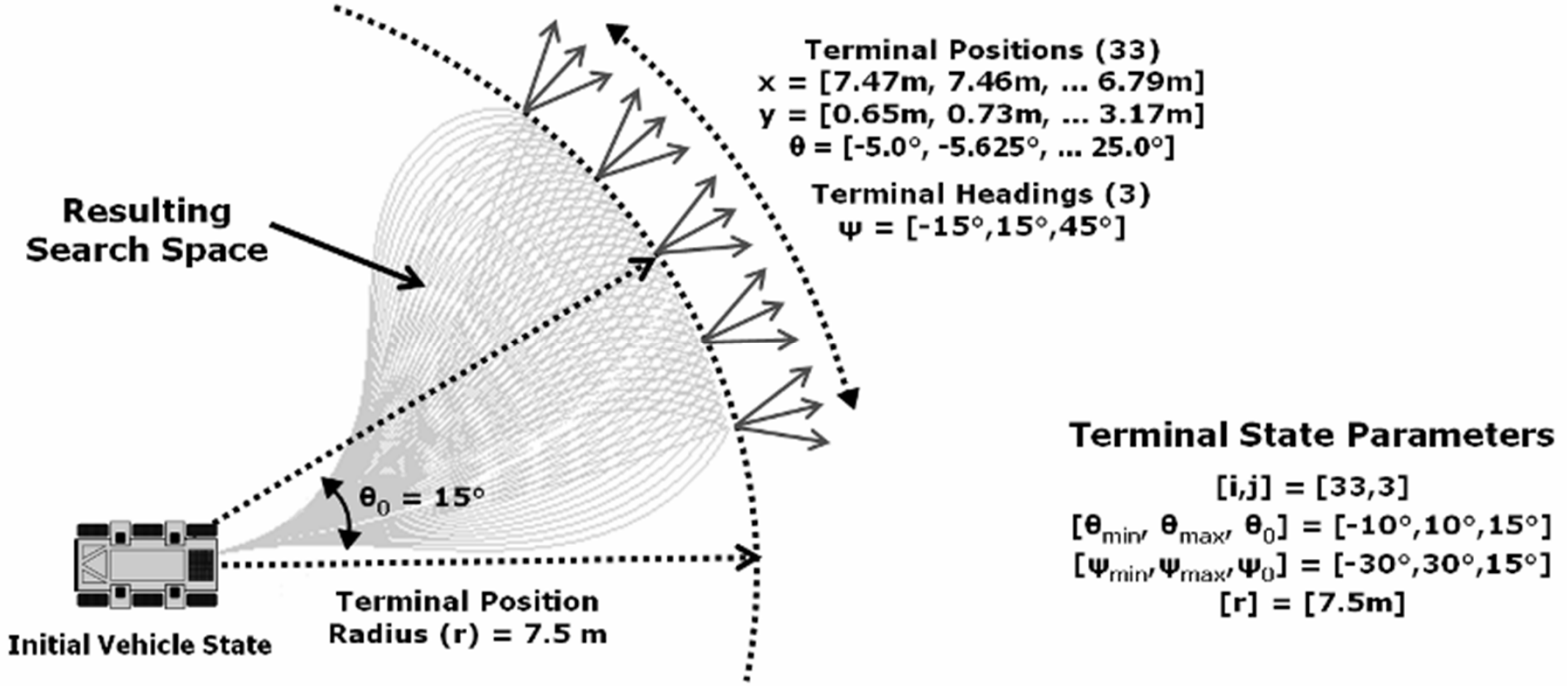
- Search spaces fixed to ground….
  - are stable
  - easy to reuse computation
  - BUT robot may not be near an edge or node.



Ultimate OA System

- **Feasible:** Generates feasible motions only.
- **Admissible:** Exploits global guidance and satisfies global constraints.
- **Efficient:** Samples feasible set uniformly in the workspace.

Ultimate System ?



- Shape of feasible set in workspace is computed off-line and stored in lookup tables.
- Impose workspace constraints on that.
- Sample regularly in state space with trajectory generation.

Outline

- 7.4 Intelligent Control
  - 7.4.1 Introduction
  - 7.4.2 Evaluation
  - 7.4.3 Representation
  - 7.4.4 Search
  - Summary

Summary

- Avoiding obstacles is a kind of planning problem.
  - Motion prediction.
  - Trajectory Evaluation
  - Search
- Its a real-time problem.
  - If the choice is between smart and fast, semi-dumb robots rule here.
- Dynamics matter in many ways.
- Cleverness of several kinds are possible.

Summary

- Alternative courses of action are evaluated based on models of environmental interaction.
- A constrained optimization formulation applies.
  - Obstacles and dynamics are constraints
  - Feasible paths evaluated for utility.
- A large number of options exist for the representations used in planning models.
  - Each has its own issues and advantages from the perspective of computational complexity.