16-350
Planning Techniques for Robotics

Planning Representations:
Lattice-based Graphs, Explicit vs. Implicit Graphs

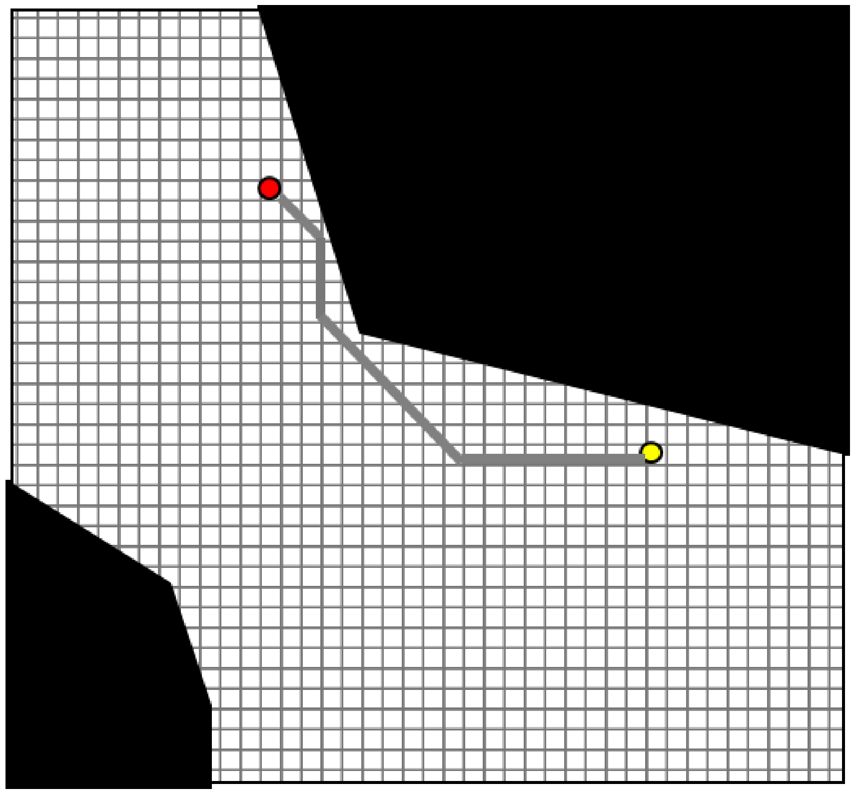
Maxim Likhachev

Robotics Institute

Carnegie Mellon University

Beyond Planning for Omnidirectional Robots

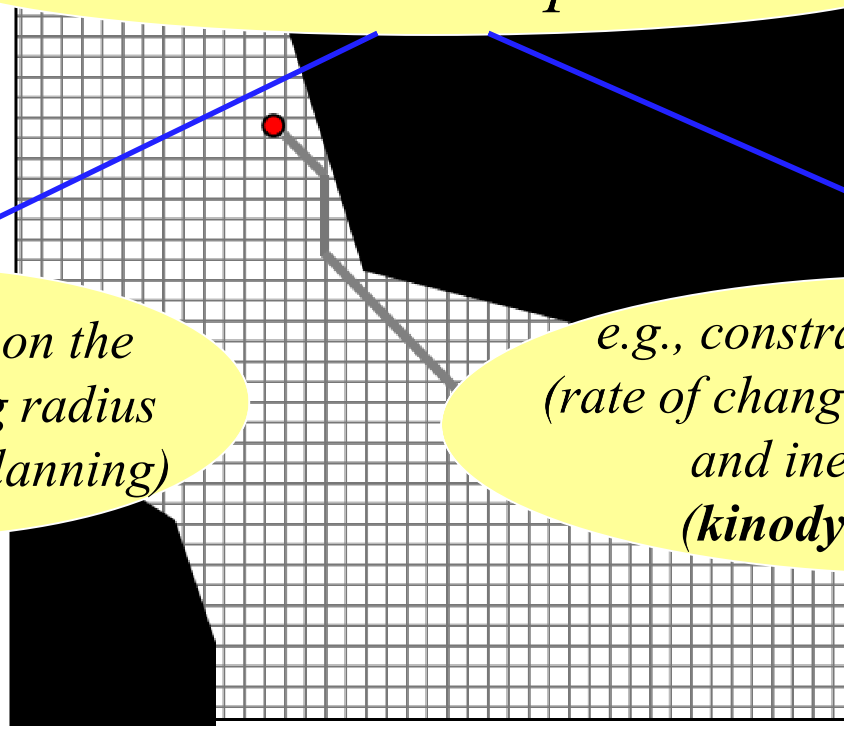
What's wrong with using
Grid-based Graphs when planning
for non-omnidirectional robots?

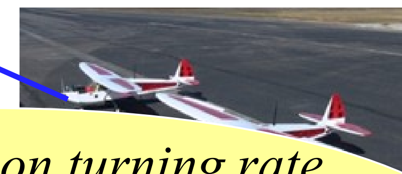


Beyond Planning for Omnidirectional Robots

What's wrong with using
Grid-based Graphs when planning
for non-omnidirectional robots?

"Can't turn in place"





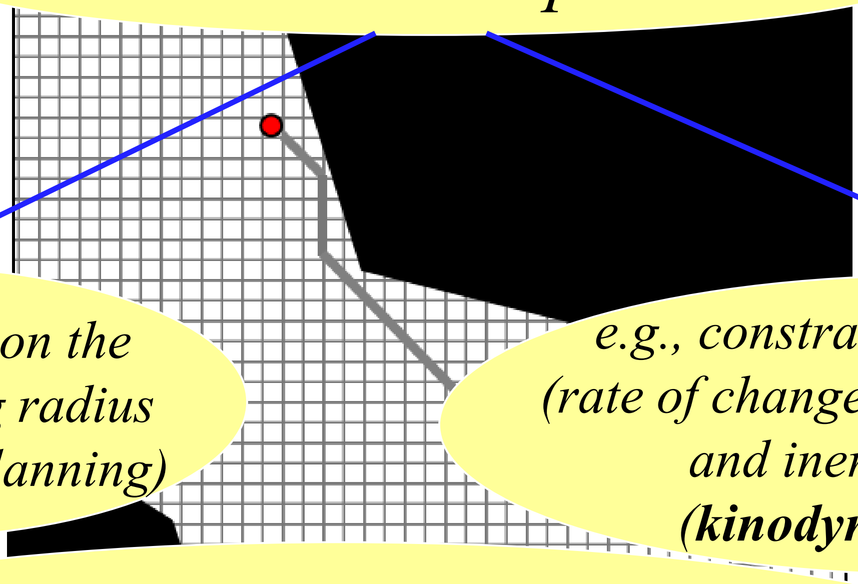
*e.g., constraints on the minimum turning radius (still **kinematic** planning)*

*e.g., constraints on turning rate (rate of change in wheel orientation) and inertial constraints (**kinodynamic** planning)*

Beyond Planning for Omnidirectional Robots

What's wrong with using
Grid-based Graphs when planning
for non-omnidirectional robots?

"Can't turn in place"



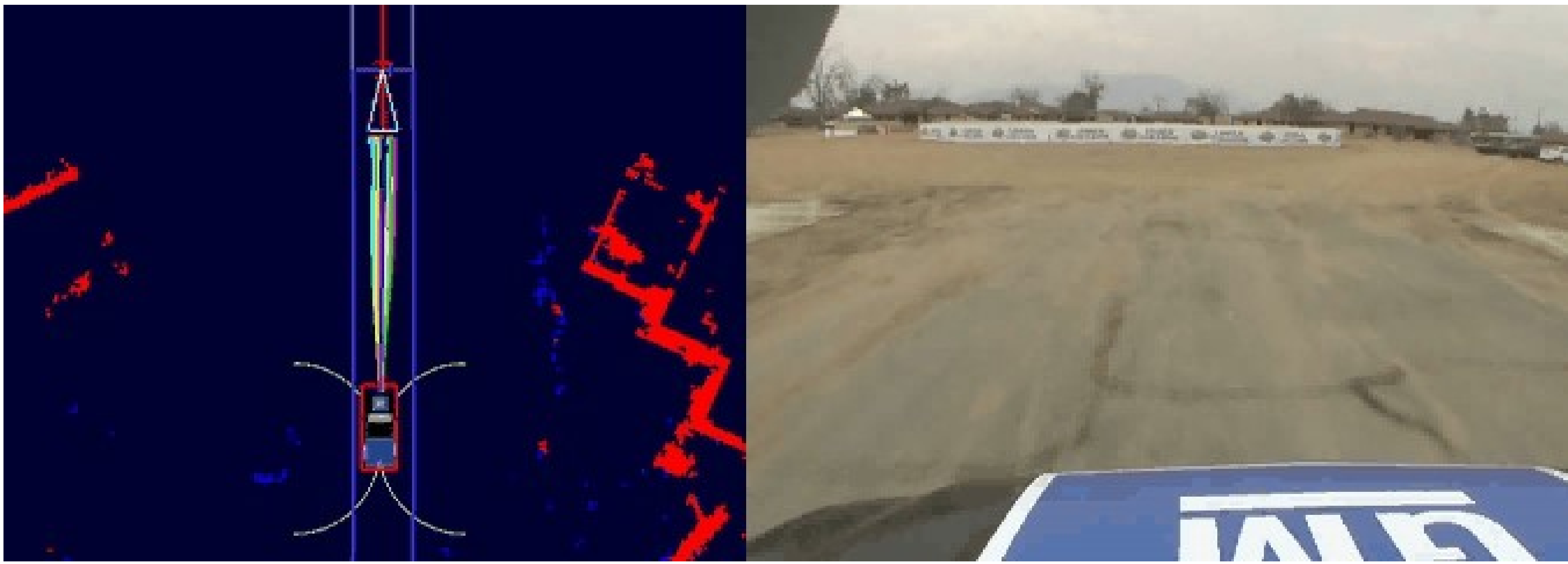
*e.g., constraints on the minimum turning radius (still **kinematic** planning)*

*e.g., constraints on turning rate (rate of change in wheel orientation) and inertial constraints (**kinodynamic** planning)*

***Kinodynamic planning**:*
Planning representation includes $\{X, \dot{X}\}$,
where X-configuration and $\dot{X}$*-derivative of X (dynamics of X)*

Beyond Planning for Omnidirectional Robots

(x,y,Θ,v) planning
with Anytime D (Anytime Incremental A*) on Lattice Graphs*



Beyond Planning for Omnidirectional Robots

(x,y,Θ) planning with ARA-based algorithm on Lattice Graphs*

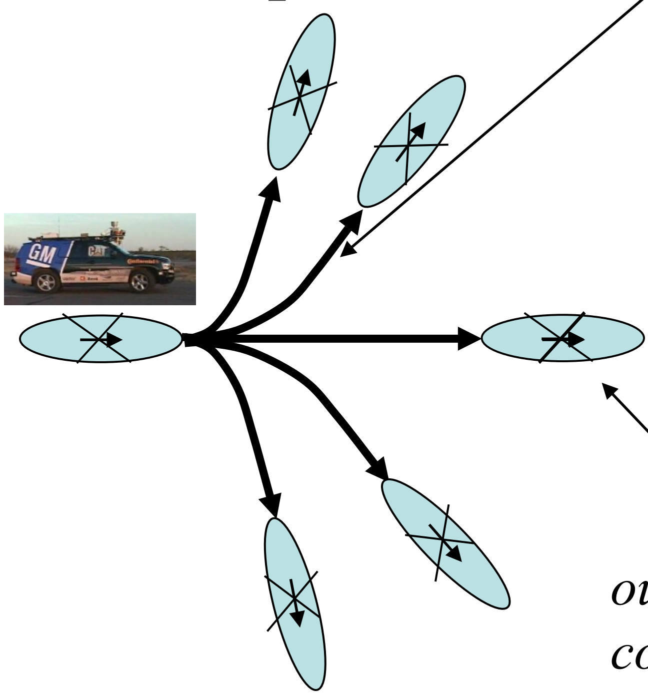
Joint work with V. Kumar (Upenn), I. Kaminer (NPS) and V. Dobrokhodov (NPS)
[thakur et al., '13]

Lattice Graphs

- Graph *{V, E}* where
  - *V*: centers of the grid-cells
  - *E*: motion primitives that connect centers of cells via short-term **feasible** motions

each transition is feasible
(typically, constructed beforehand)

motion primitives

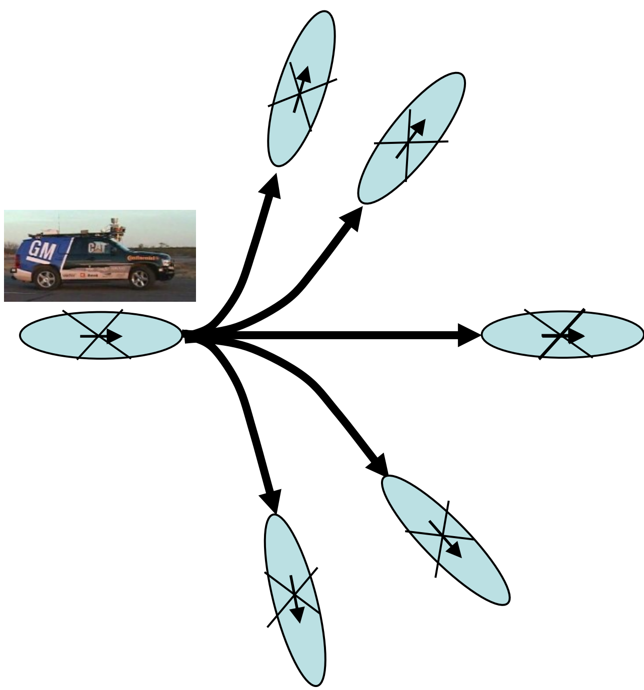


outcome state is the center of the
corresponding cell in a grid

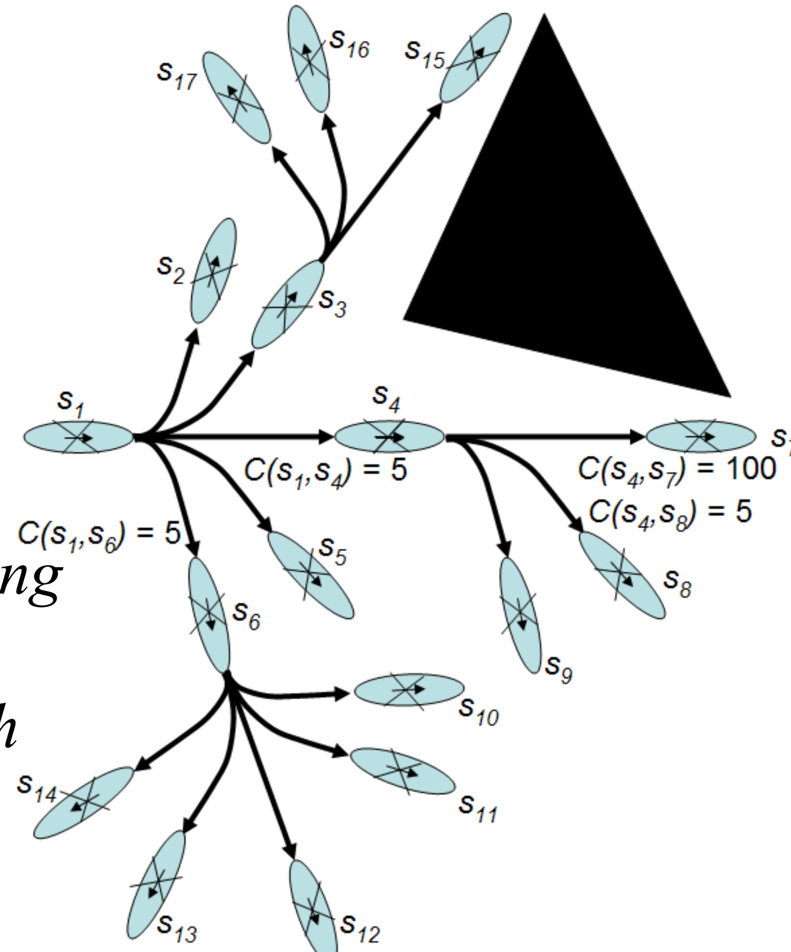
Lattice Graphs

- Graph *{V, E}* where
  - *V*: centers of the grid-cells
  - *E*: motion primitives that connect centers of cells via short-term **feasible** motions

motion primitives



replicate it during planning to generate lattice graph

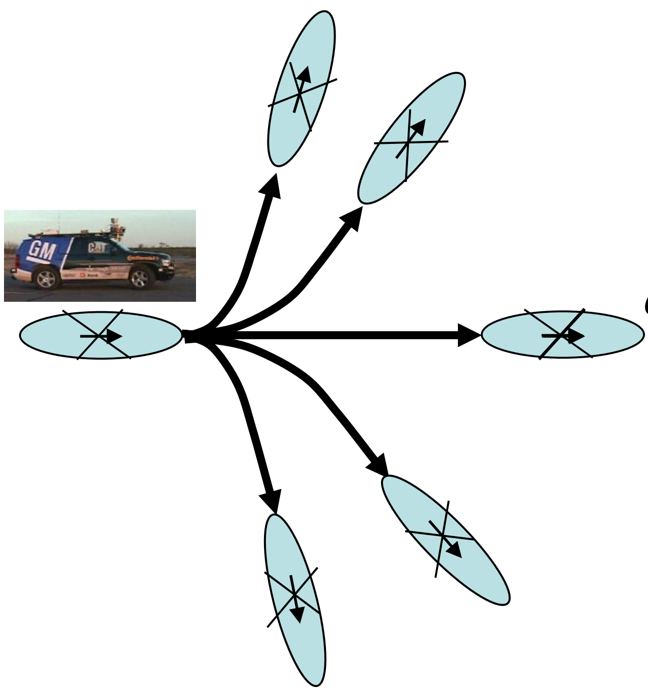


Lattice Graphs

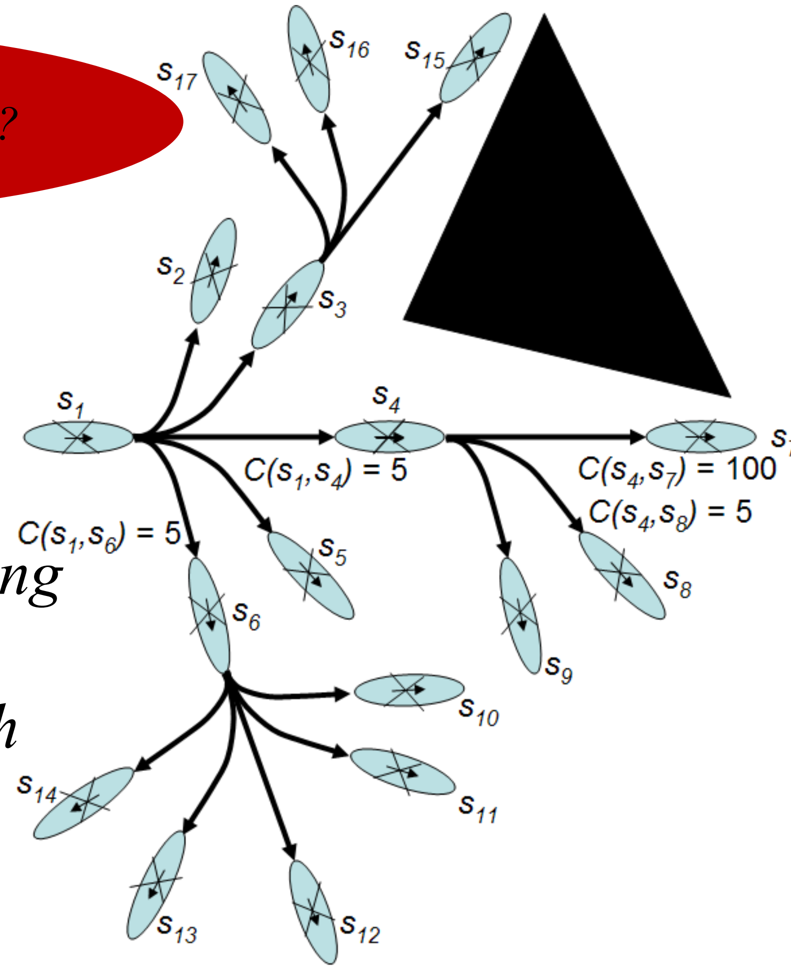
- Graph *{V, E}* where
  - *V*: centers of the grid-cells
  - *E*: motion primitives that connect centers of cells via short-term **feasible** motions

How do edgecosts get assigned?

motion primitives



replicate it during planning to generate lattice graph



Lattice Graphs

- **Board example** for *(x,y,Θ)* planning for a unicycle model (minimum turning radius)

Planning as Graph Search Problem

1. Construct a graph representing the planning problem

2. Search the graph for a (hopefully, close-to-optimal) path

The two steps above are often interleaved

Planning as Graph Search Problem

1. Construct a graph representing the planning problem

2. Search the graph for a (hopefully, close-to-optimal) path

The two steps above are often interleaved

Interleaving Search and Graph Construction

Graph Search using an **Explicit Graph** (allocated prior to the search itself):

1. *Create the graph G = {V, E} in-memory*

2. *Search the graph*

Using Explicit Graphs
is typical for low-D (i.e., 2D) problems in Robotics
(with the exception of PRMs, covered in a later lecture)

Interleaving Search and Graph Construction

Graph Search using an **Implicit Graph** (allocated as needed by the search):

1. *Instantiate Start state*

2. *Start searching with the Start state using functions*

   a) *Succs = GetSuccessors (State s, Action)*
   b) *ComputeEdgeCost (State s, Action a, State s')*

   and allocating memory for the generated states

Using Implicit Graphs
is critical for most (>2D) problems
in Robotics

Interleaving Search and Graph Construction

- **Board example** for deciding whether to use an Explicit graph or Implicit graph

- Planning for (x, y, Θ, v) for
  - 20 by 20 m environment discretized into 25 cm cells with 8 heading Θ values and 2 velocity v values for a point robot

Is it feasible to use Explicit Graph
(memory and pre-computation time reqs)?

Interleaving Search and Graph Construction

- **Board example** for deciding whether to use an Explicit graph or Implicit graph

- Planning for *(x,y,Θ,v)* for
  - 200 by 200 m environment discretized into 25 cm cells with 16 heading *Θ* values and 2 velocity *v* values for a real vehicle

Is it feasible to use Explicit Graph (memory and pre-computation time reqs)?

Summary

- Lattice graphs are grid-based graphs with transitions based on short feasible motion primitives (not necessarily to the neighboring cells)

- Implicit graphs is a typical way to do planning for non-trivial planning in Robotics
  - reduces memory requirements
  - reduces pre-computation time