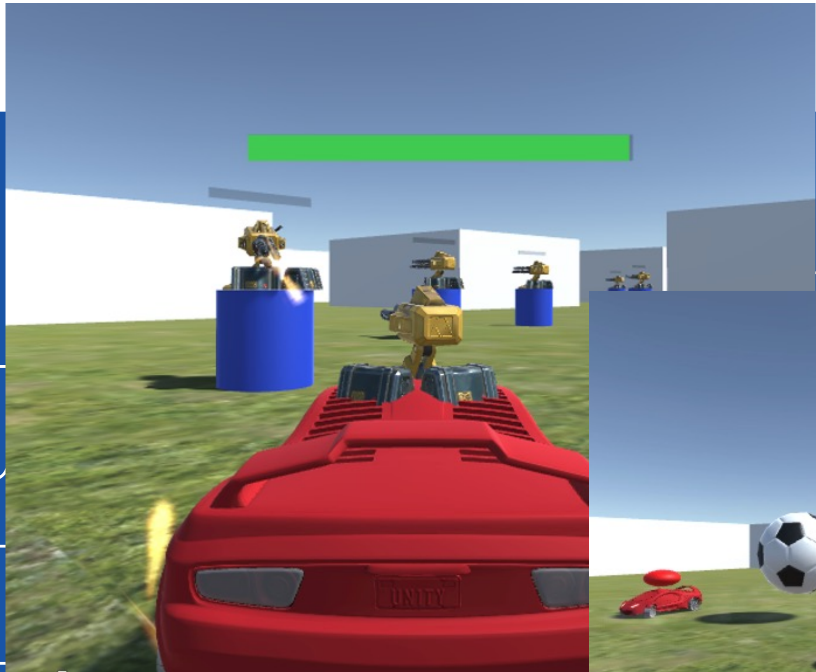
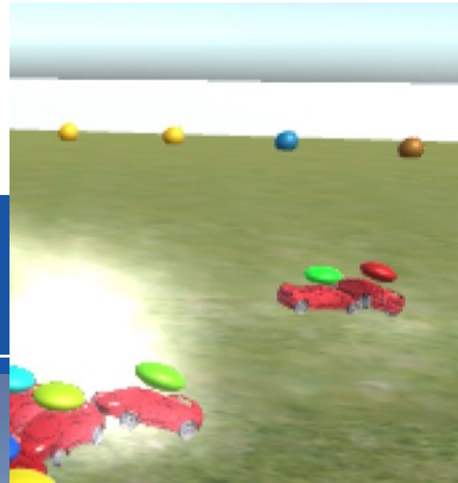
Artificial Intelligence and Multi Agent Systems DD2438

Lecture 2





Pairs for Assignment 1

- By now all pairs should be up and running
- If you or your teammate is dropping the course, please let me know ASAP! …

Outline of lecture

- Administrative Notes
  - You should be working in the groups now!
  - *Initial Progress* is expected on Tuesday next week
  - Group managers are listed in Canvas (on group page)
  - Todo:
    - Prepare slides
    - Upload/share slides
    - Fill in progress report
    - Peer review slides of others (2 questions)
    - Present slides at meeting

- Motion Planning for the Motion Models ...

Todo for Next Week

View Course Stream
View Course Calendar
View Course Notifications

To Do

You have to attend the lecture in DD2438 on Tuesday (tomorrow)
DD2438 VT23 Artificial Intelligence and Multi Agent Systems (agent23)
Jan 16 at 3:48pm

Meeting preparations Week 4 (slides, progress report, peer review)
DD2438 VT23 Artificial Intelligence and Multi Agent Systems (agent23)
Jan 23 at 8pm

Meeting preparations Week 5 (slides, progress report, peer review)
DD2438 VT23 Artificial Intelligence and Multi Agent Systems (agent23)
Jan 30 at 8pm

A1 Meeting preparations

These instruction are a more detailed version of the general ones.

On the day before a meeting with an assignment **you need to do the following:**

- Update progress tables, and best results so far
- Create slides (using Google slides, see example) describing your current status
    - Add a link to your slides here
        - This will be used for peer review and presentation, so make sure everyone with the link can read and comment your w
    - Upload a pdf-version of your slides in the corresponding canvas "assignment"
        - This is used for book-keeping, to check that you met the deadline etc.
- Peer feedback: After the deadline pick 2 other groups from the list (choose one freely, and one that is right below you in the list) read their slides and provide at least 2 constructive comments or questions to each group (this is done individually, in Google slides)

When **preparing the slides and your presentation** consider the following:

- See example slides here.
- Focus on what is **interesting**, do not tell you friends what they already know.
- What did you do, why?
- What are you planning to do, why?
- Do you have any **assumptions, observations** regarding algorithm performance?
- Make the presentation **short**, only a few slides (but plots and videos are interesting)

Template of Slides

https://docs.google.com/presentation/d/1AEhrvdoOA-Nr_5EoMbjozHjY8whdiub3kXRUeMwCLQE/edit#slide=id.g766abca800_0_7

Remember: Ask Questions in Canvas

Per Andersson
Yesterday

Hi,

will we know whether there will be diagonal walls or even round corners on the evaluation mazes, or will the mazes be comprised of blocks as the examples are?

Per

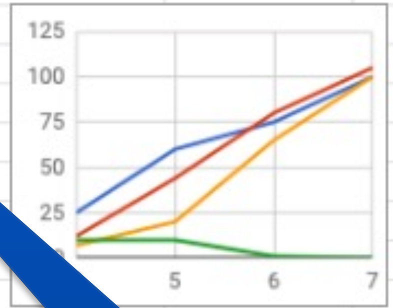
Progress Report (link in Canvas)

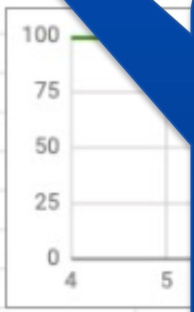
Fill in data the day before each meeting

Note: Effort is in % out of the 200h (for 2 persons working 2.5 weeks full time)

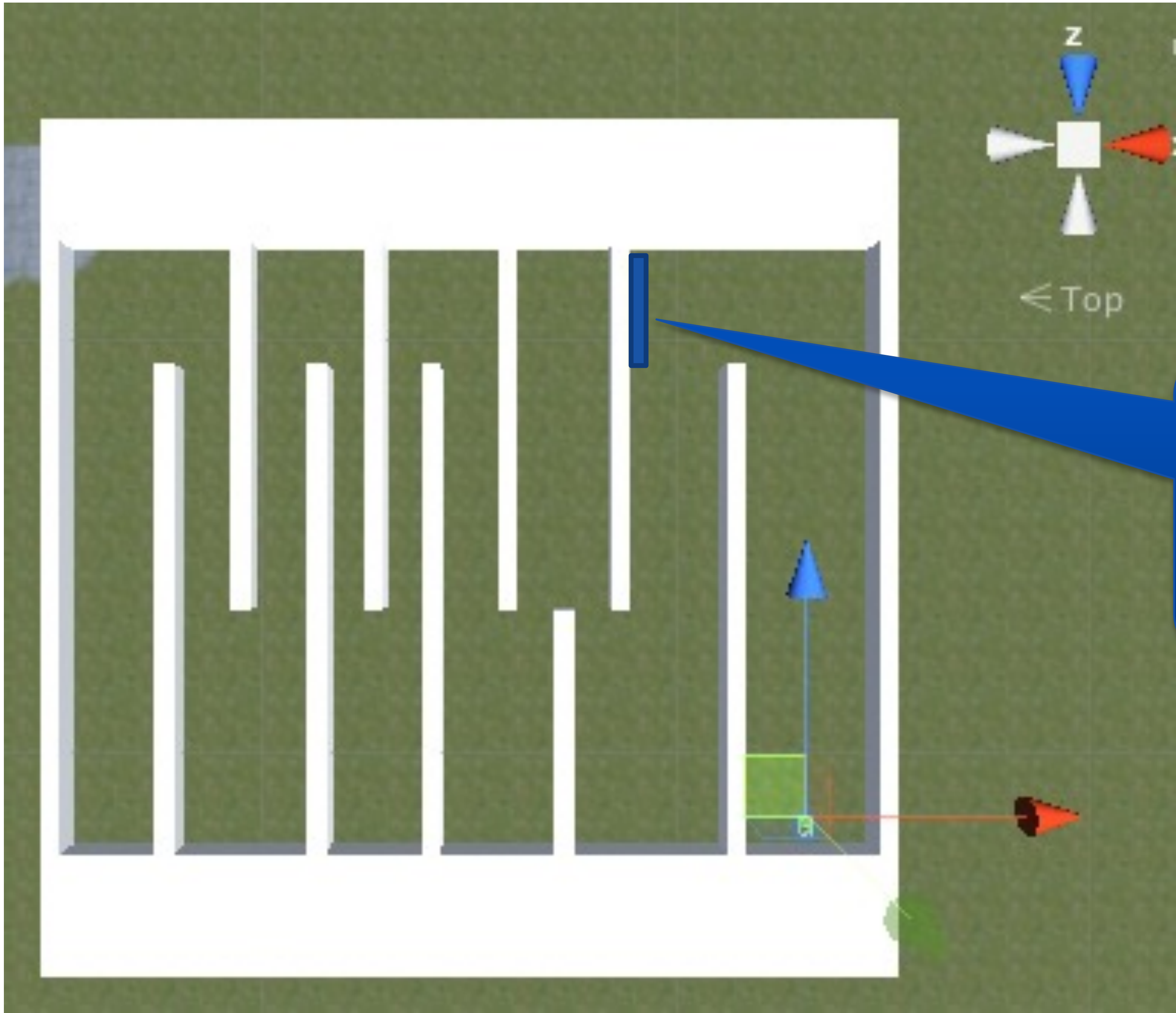
Let the number of each week be given by the effort spent between the meeti … that week and the previous meeting.

		Week number		Final Meetin	eport Handin
Group 0		4	5	6	7
(example)	Planned Effort (accumulated):	25	60	75	100
	Actual Effort (accumulated):	12	44	80	105
	Actual Progress:	7	20	65	100
	Risk of Failure:	10	10	1	0
	Comments on Risk:	(only needed if >5%)			
	Comments on Progress:	(only needed if large gaps between first 3 curves)			
	Project management:	(what approach are you following and how is it working)			
		Week number		Final Meeting	Report Handin
Group 1		4	5	6	7
(Name 1)	Planned Effort (accumulated):	99	99	99	99
(Name 2)	Actual Effort (accumulated):	99	99	99	99
	Actual Progress:	99	99	99	99
	Risk of Failure:	99	99	99	1
	Comments on Risk:				
	Comments on Progress:				
	Project management:				



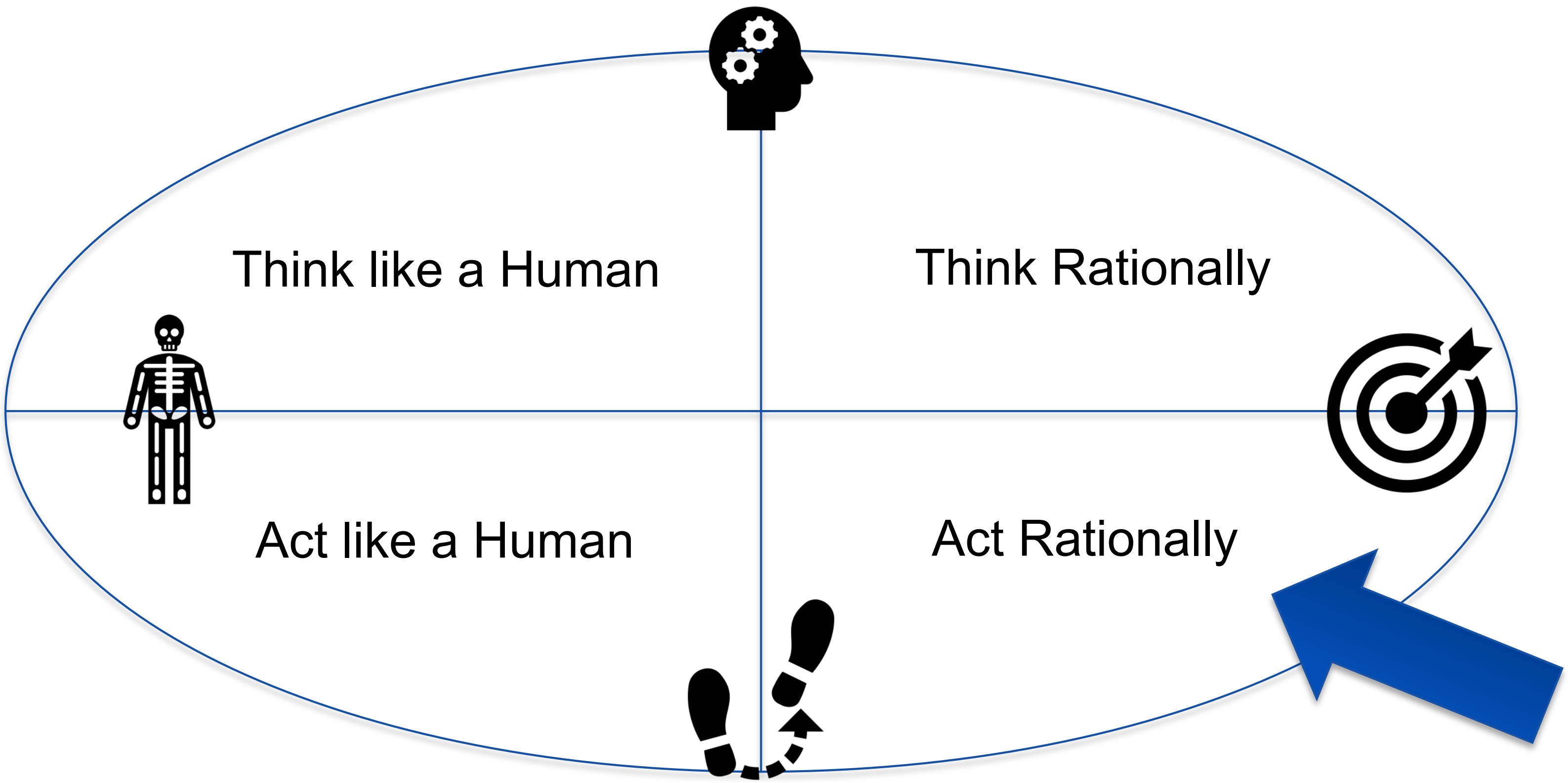


Add data for “week between this and previous meeting”



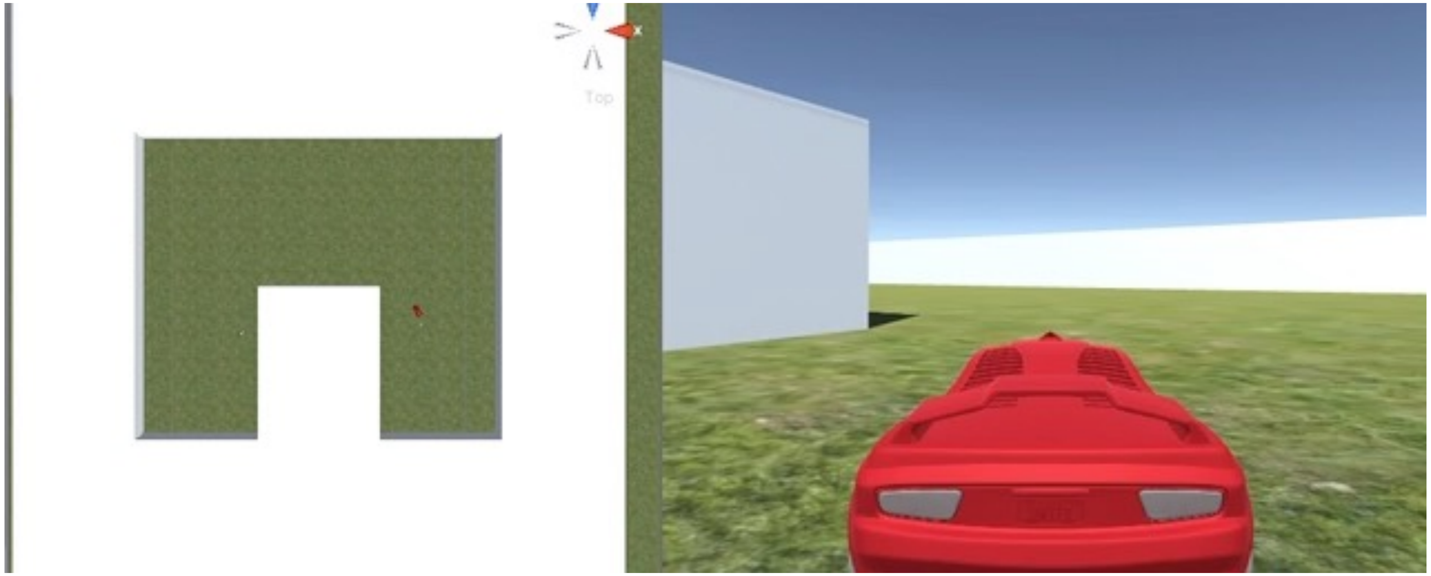
Note that grid sizes can be smaller than the car

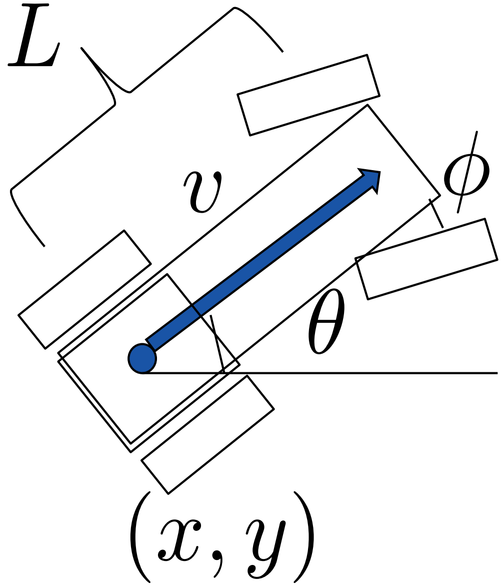
Artificial Intelligence

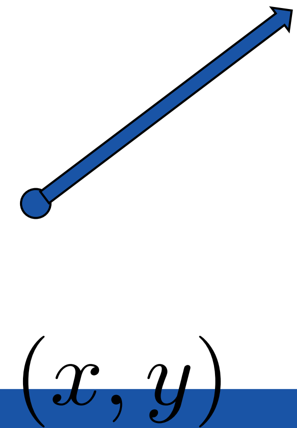


“Get the car to the goal as fast as possible”

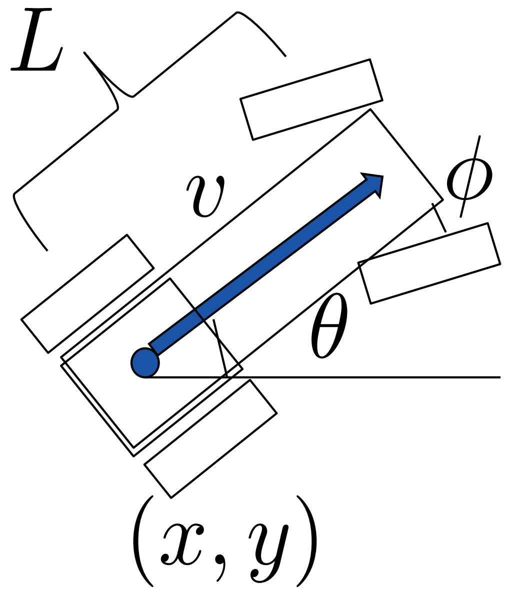
The two vehicle models: Car and Drone







Summary of vehicle models



Complexity

- Discrete state space
- Kinematic point model
- Dynamic point model
- Differential drive
- Car model
- More detailed models

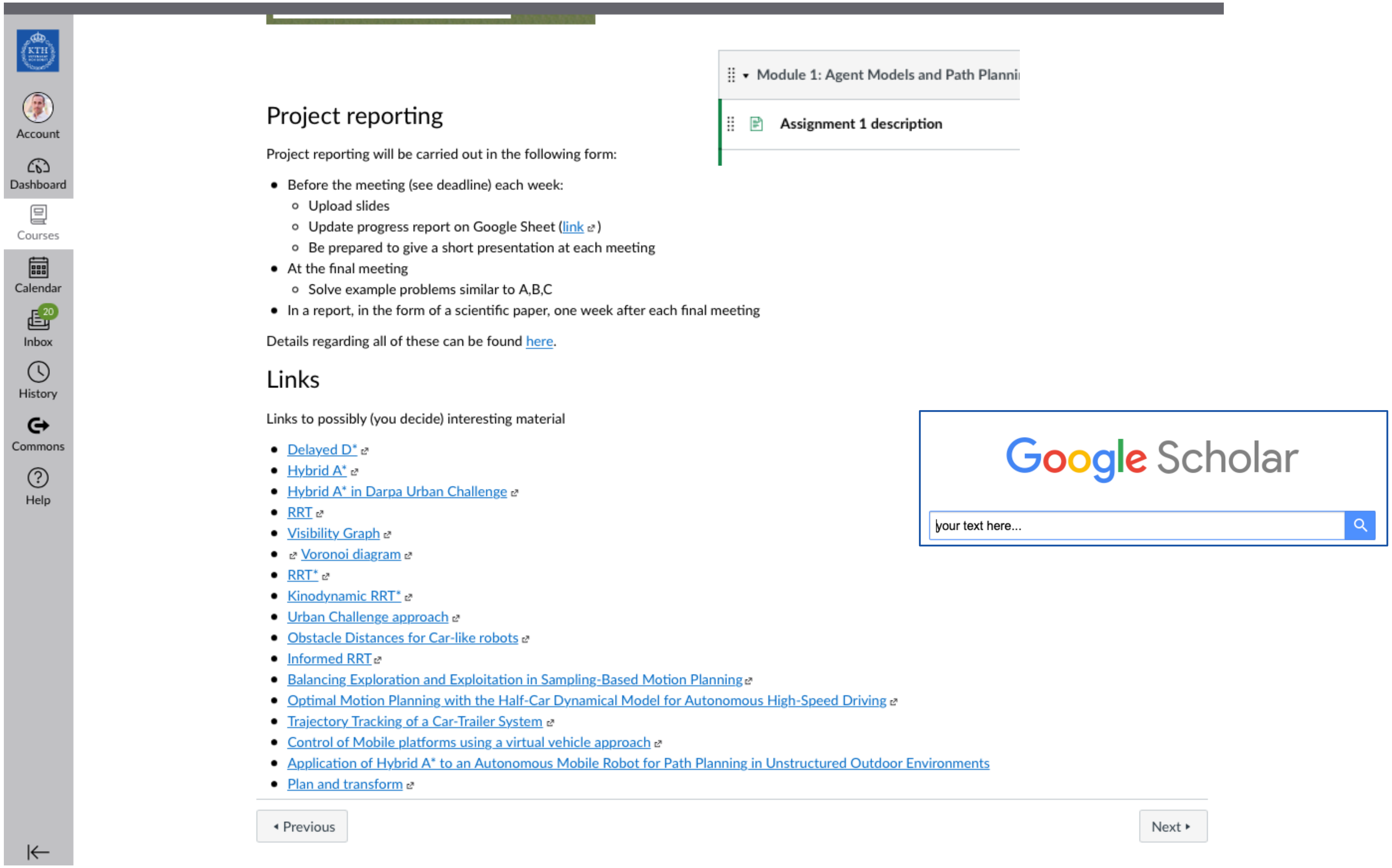
$$\begin{aligned}\dot{x} &= v\cos\theta,\\ \dot{y} &= v\sin\theta,\\ \dot{\theta} &= \frac{v}{L}\tan\phi.\end{aligned}$$

Short overview of Path Planning Algorithms

Initial Note:

- These slides are meant to give you **a rough idea** of how the different algorithms work
- You are **not** expected to be able to solve the assignment by just reading the slides
- You are expected to
  - Search for information (even beyond listed references)
  - Combine and modify ideas
  - Do problem solving
  - Discuss at meetings

References can be found here…



Module 1: Agent Models and Path Planni

Assignment 1 description

Account
Dashboard
Courses
Calendar
20
Inbox
History
Commons
Help

Project reporting

Project reporting will be carried out in the following form:

- Before the meeting (see deadline) each week:
  - Upload slides
  - Update progress report on Google Sheet (link)
  - Be prepared to give a short presentation at each meeting
- At the final meeting
  - Solve example problems similar to A,B,C
- In a report, in the form of a scientific paper, one week after each final meeting

Details regarding all of these can be found here.

Links

Links to possibly (you decide) interesting material

- Delayed D*
- Hybrid A*
- Hybrid A* in Darpa Urban Challenge
- RRT
- Visibility Graph
- Voronoi diagram
- RRT*
- Kinodynamic RRT*
- Urban Challenge approach
- Obstacle Distances for Car-like robots
- Informed RRT
- Balancing Exploration and Exploitation in Sampling-Based Motion Planning
- Optimal Motion Planning with the Half-Car Dynamical Model for Autonomous High-Speed Driving
- Trajectory Tracking of a Car-Trailer System
- Control of Mobile platforms using a virtual vehicle approach
- Application of Hybrid A* to an Autonomous Mobile Robot for Path Planning in Unstructured Outdoor Environments
- Plan and transform

Previous
Next

Google Scholar

your text here...

… but searching on your own is encouraged …

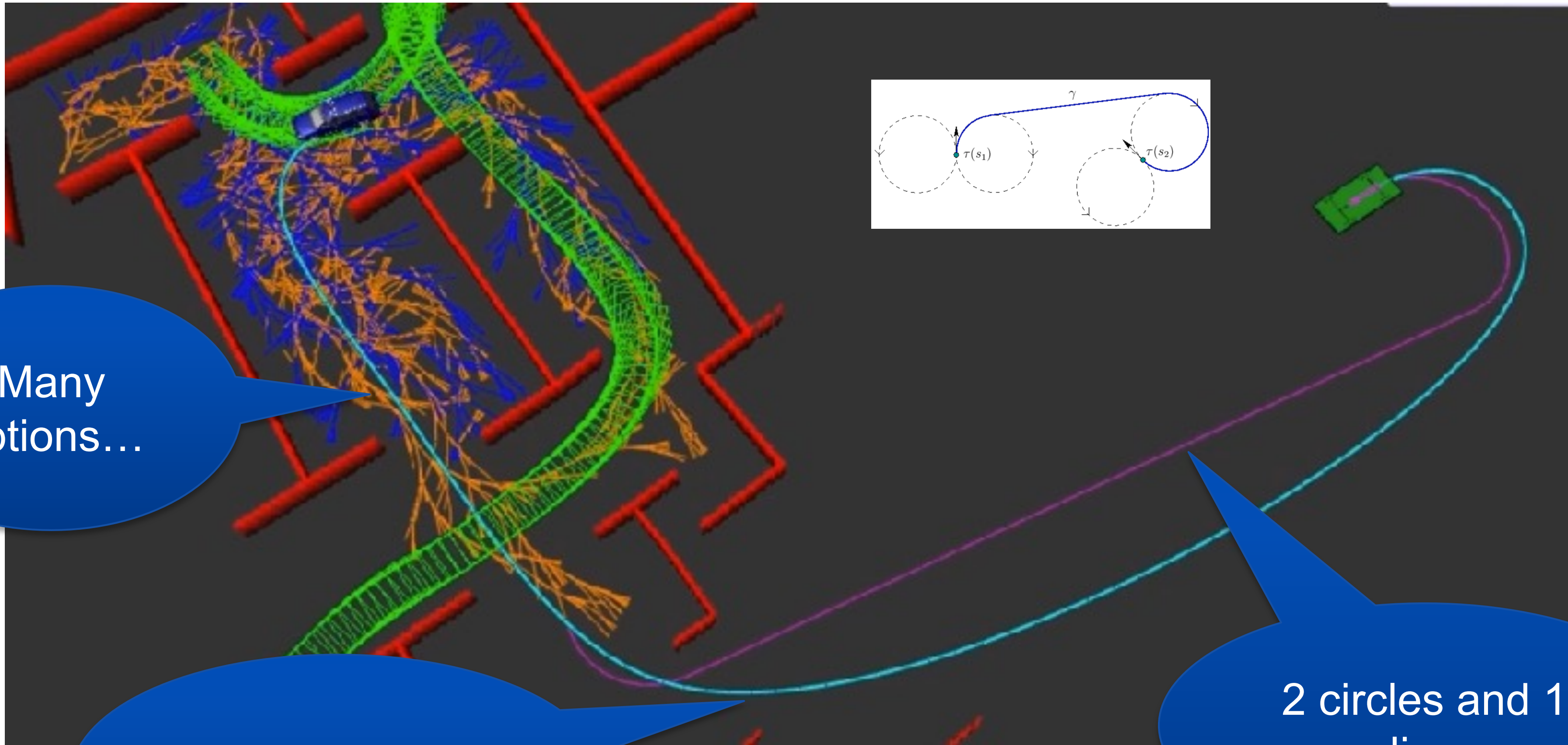
In general: Path planning is hard

- A complete algorithm finds a path if one exists and reports no path exists otherwise.
- Several variants of the path planning problem have been proven to be NP-hard.
- A complete algorithm may take exponential time.

Planning for Autonomous Driving

How is this done?

Some observations



The foundation for many planning algorithms

Shortest Path in a Graph

A Graph G=(V,E)

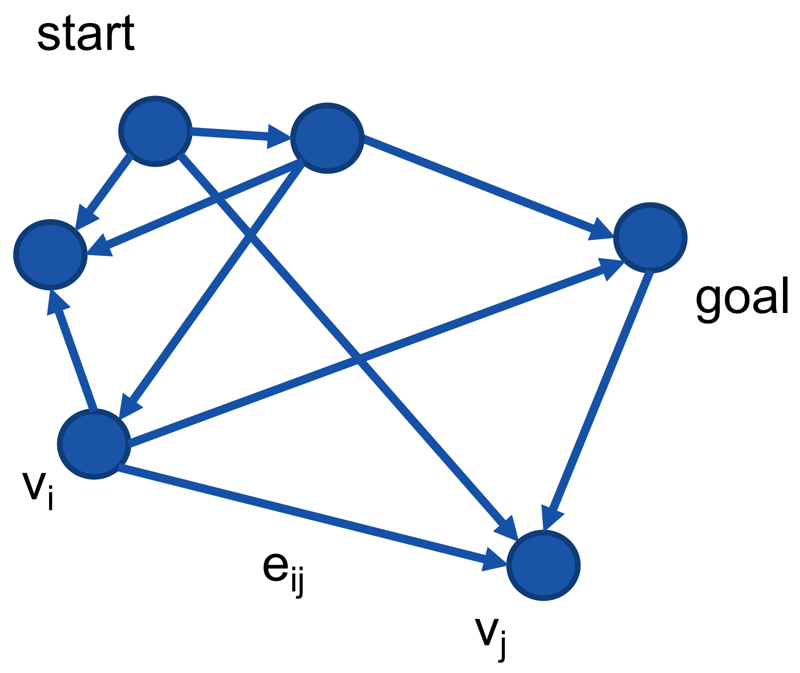
Vertices (v_i)

Edges $e_{ij}=(v_i, v_j)$

Costs c_{ij}

Solved by

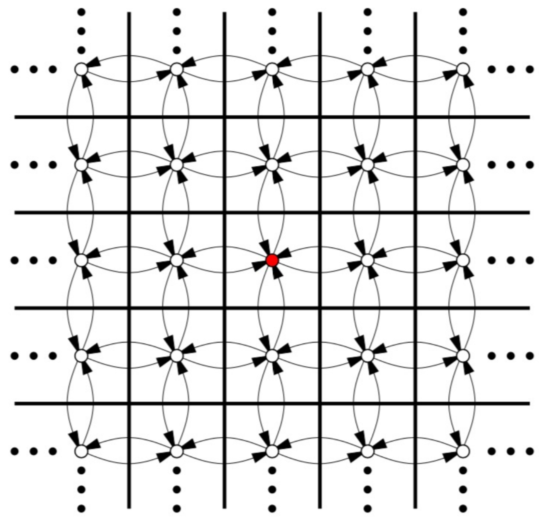
- Dijkstas algorithm
- A*

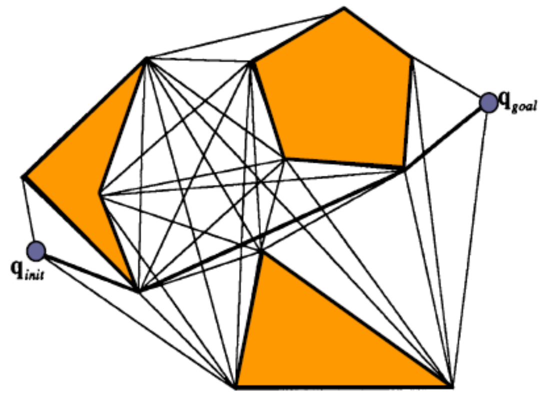


How do we use A*?

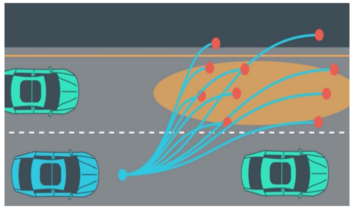
- How to choose the Graph?
- How to set Costs?

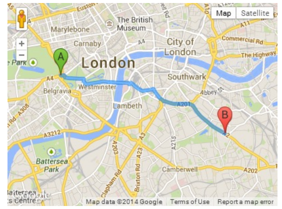
- Take robot into account
  - Motion model
  - Size/shape





Common Approaches





Continuous Problem

Configuration/State Space Problem

Size/shape

Motion model

Discretized Problem

Sample & Search

A*

Graph search

RRT etc
Motion model

Motion model

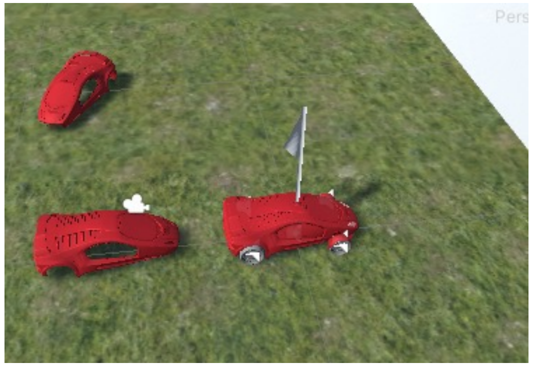
Post process

Motion model

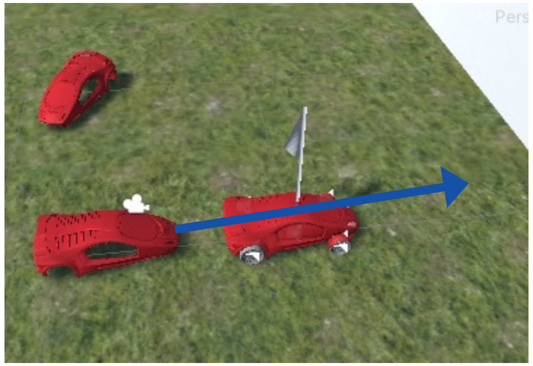
Drive

Configuration space and State space

- **Configuration**:
  - Specification of **every position** of the agent
  - Enough to check collisions
  - Ex: (x, y, theta) of a car

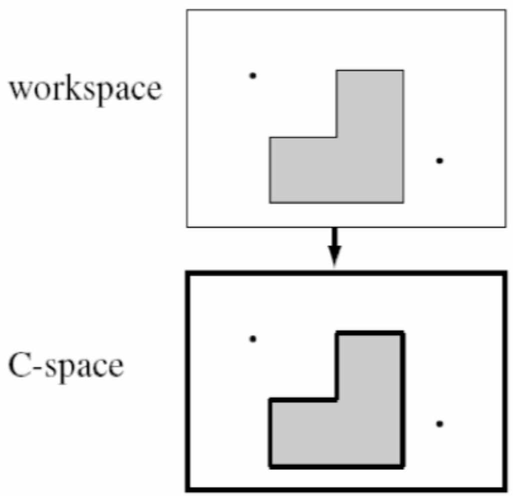


- **State**:
  - Specification of **every position and velocity** of the agent
  - Enough to predict the future
  - Ex: (x,y, theta, velocity) of a car
  - Configuration space is subset of State space



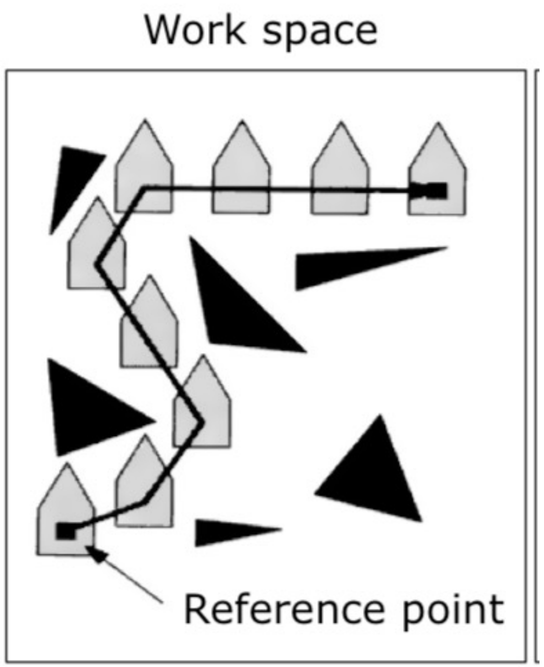
Configuration Space Obstacles

- Part of C-space that induces a collision somewhere

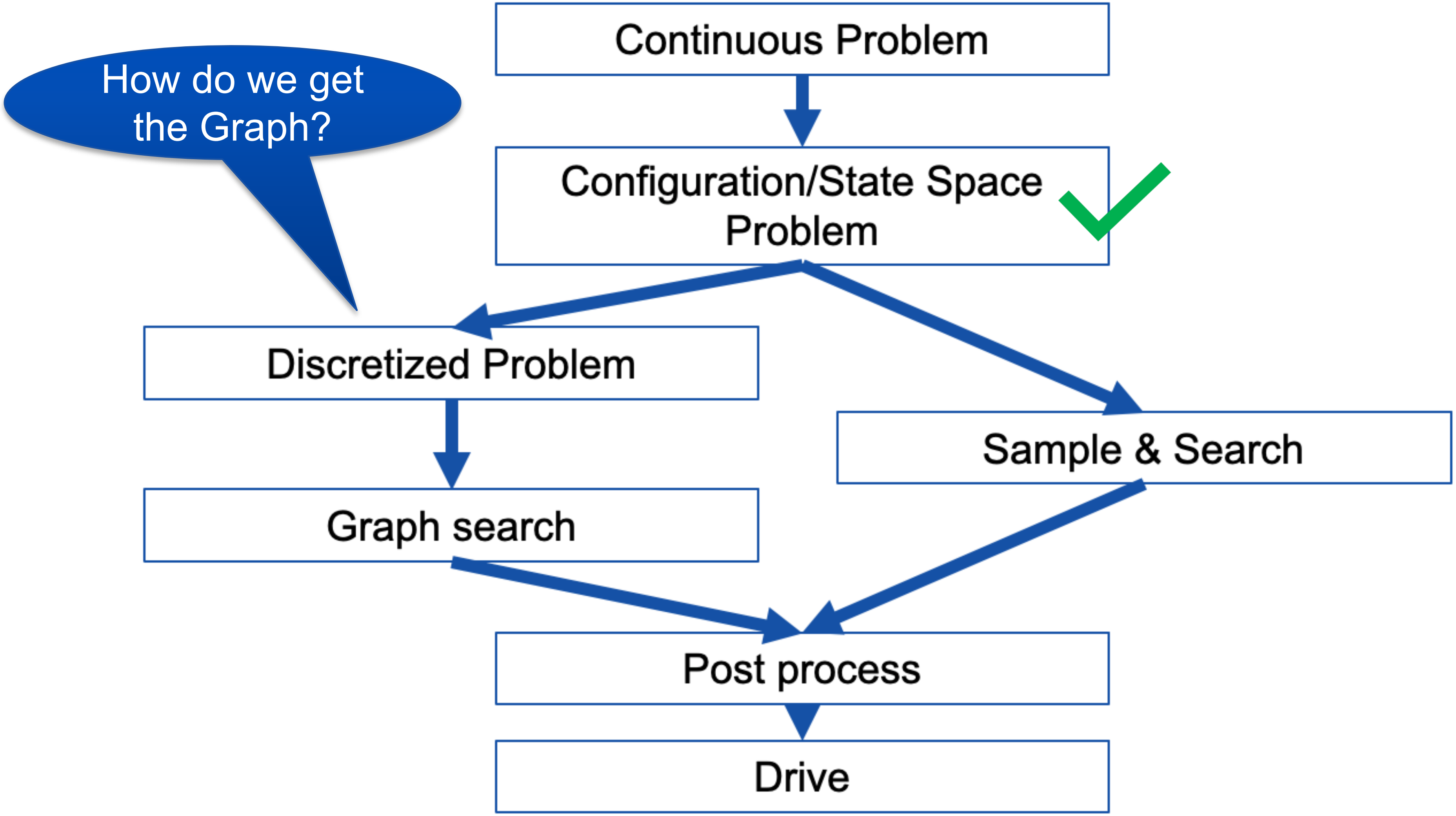


Configuration Space Obstacles

- Part of C-space that induces a collision somewhere

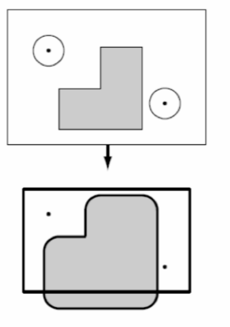


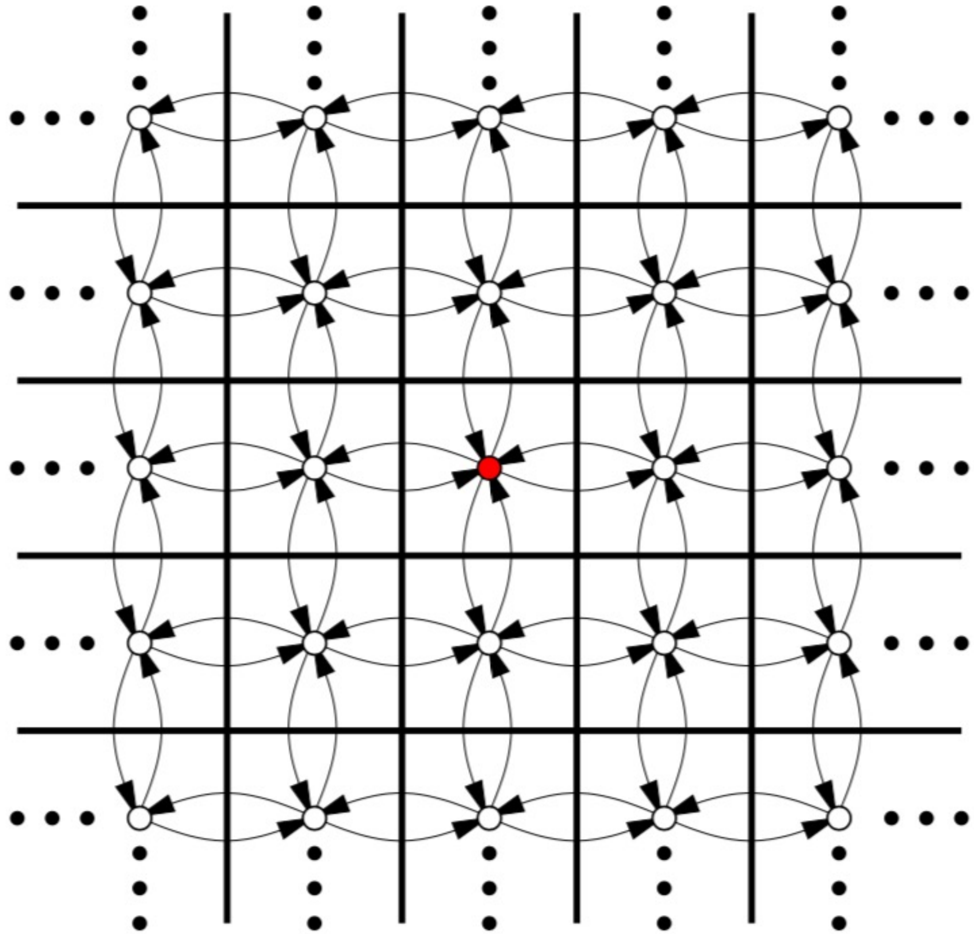
Common Path Planning Approach



How to make a Graph from C-space?

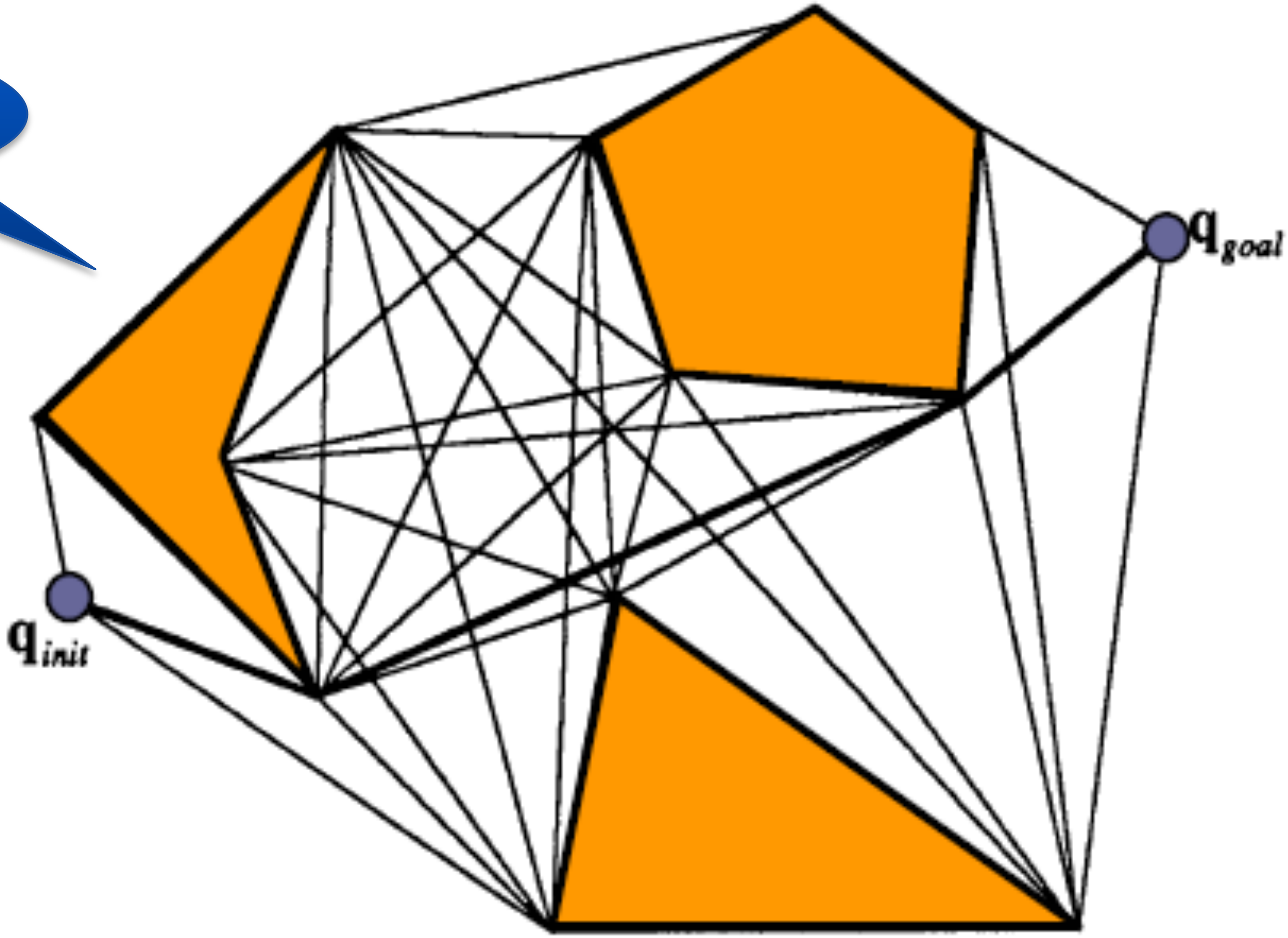
A grid

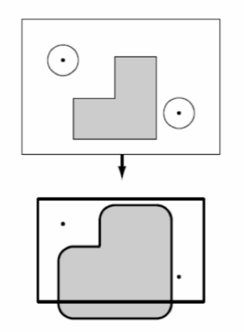




How to make a Graph from C-space?

A Visibility Graph

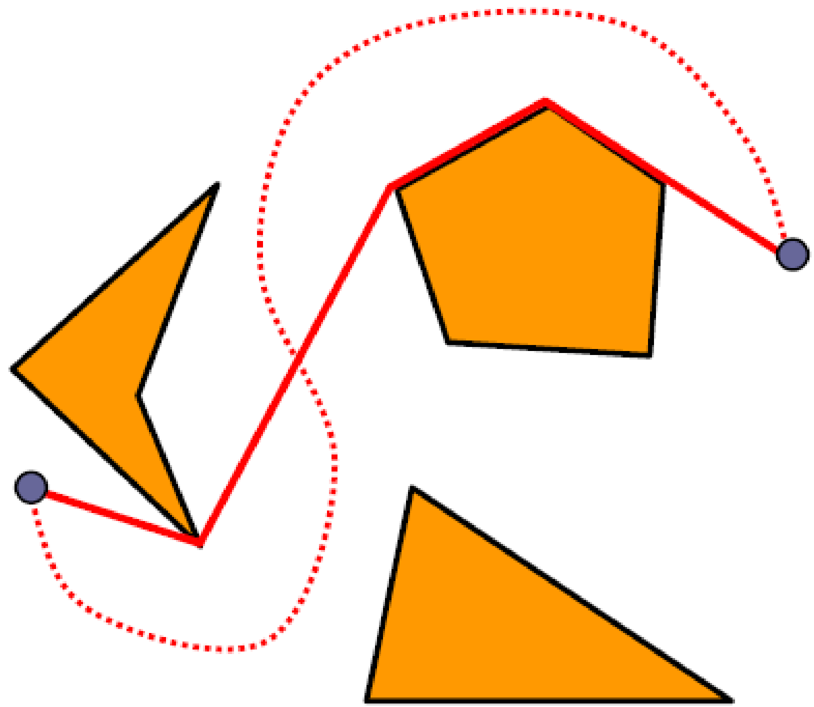




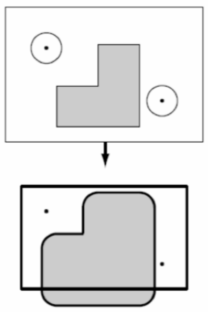
How to make a Graph from C-space?

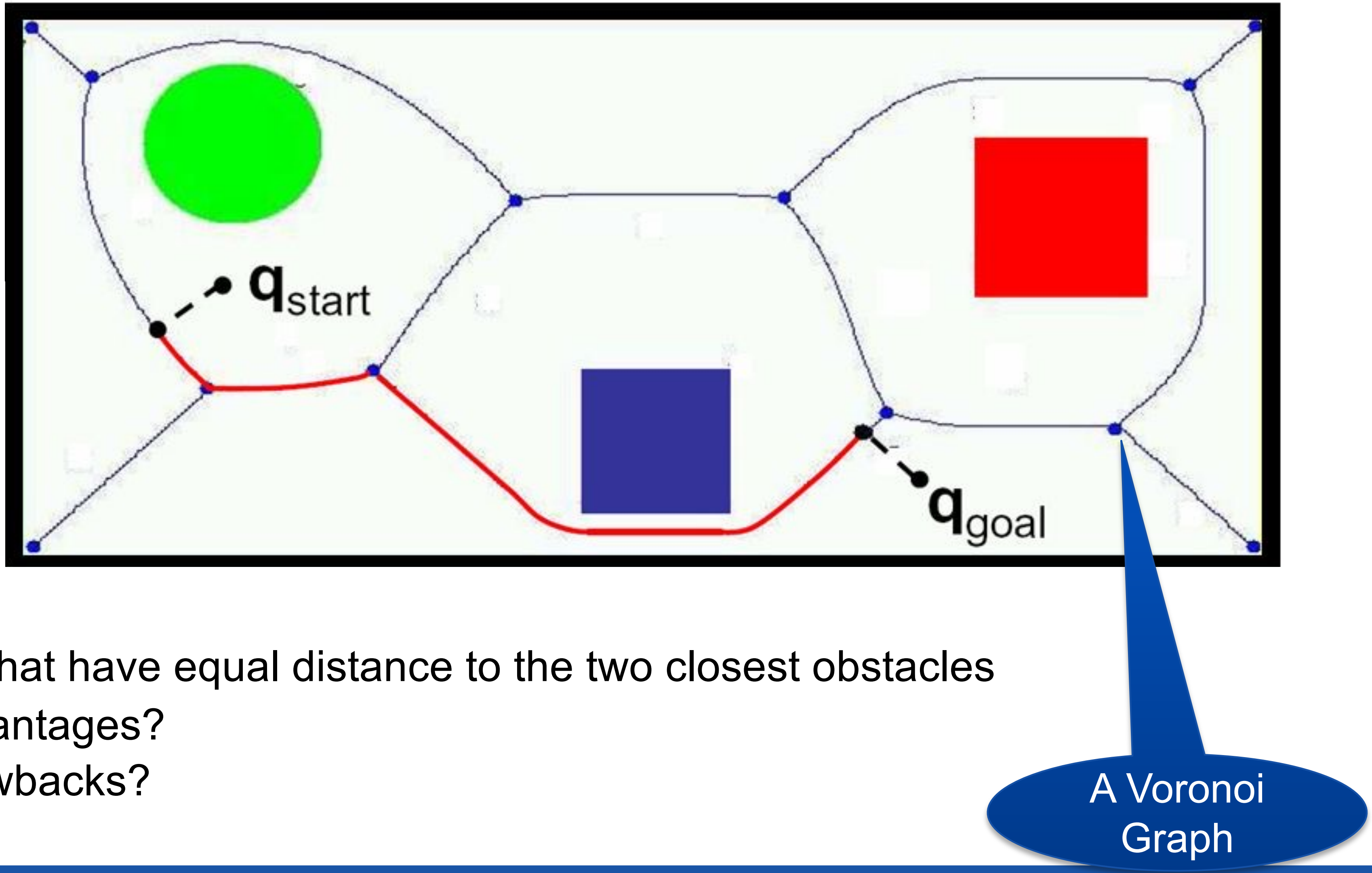
Observation:

- Shortening any path gives a visibility graph path
- Advantages?
- Drawbacks?



How to make a Graph from C-space?





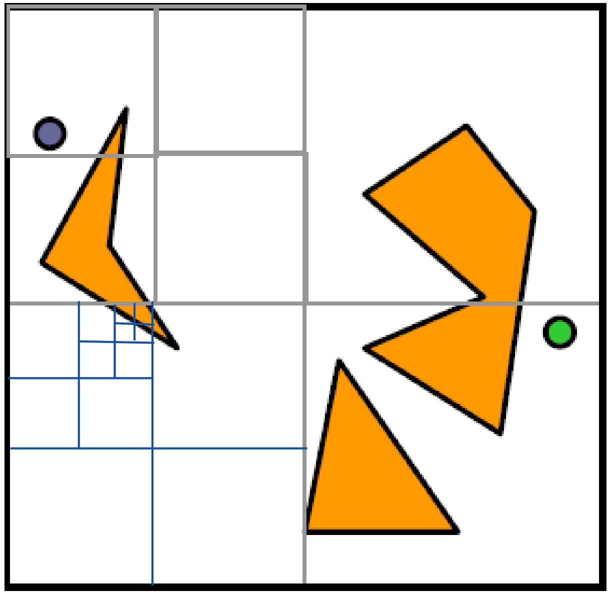
Points that have equal distance to the two closest obstacles

- Advantages?
- Drawbacks?

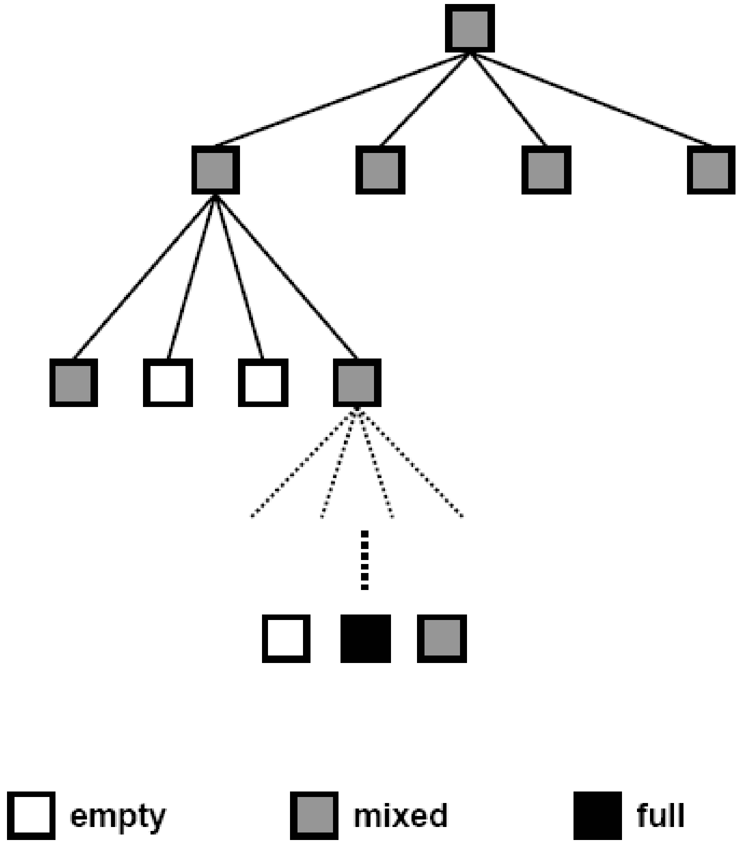
How small should we make the grids?

7	6	5	6	7	8	9	10	11		19	20	21	22
6	5	4	5	6	7	8	9	10		18	19	20	21
5	4	3	4	5	6	7	8	9		17	18	19	20
4	3	2	3	4	5	6	7	8		16	17	18	19
3	2	1	2	3	4	5	6	7		15	16	17	18
2	1	0	1	2	3	4	5	6		14	15	16	17
3	2	1	2	3	4	5	6	7		13	14	15	16
4	3	2	3	4	5	6	7	8		12	13	14	15
5	4	3	4	5	6	7	8	9	10	11	12	13	14
6	5	4	5	6	7	8	9	10	11	12	13	14	15

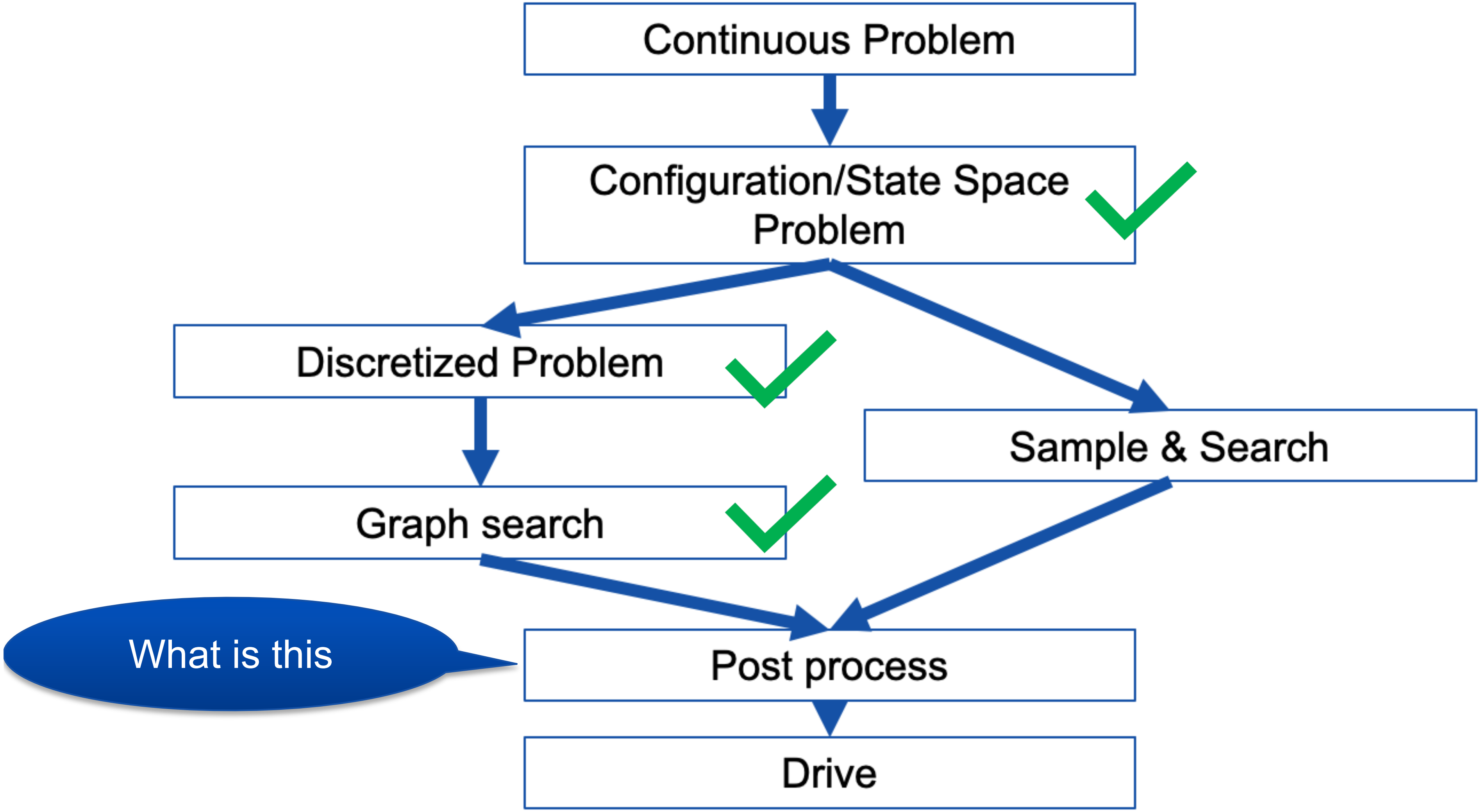
High resolution in narrow areas
Low resolution in open areas…



Quadtree
Decomposition

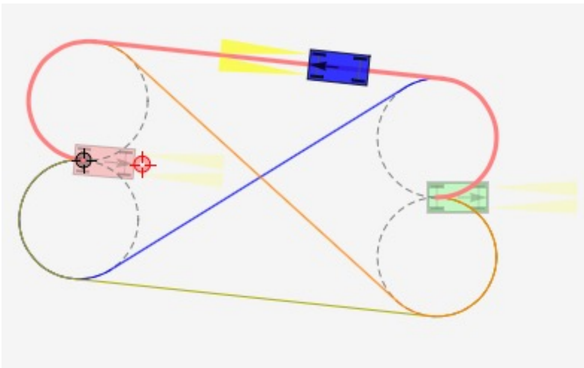


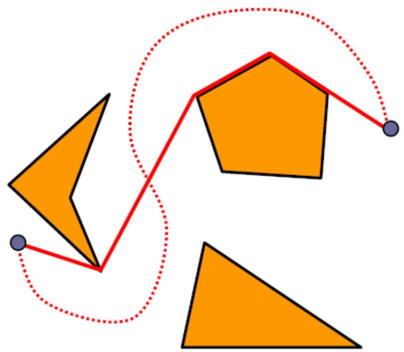
Common Path Planning Approach



What about undrivable trajectories?

Can a car drive any path?

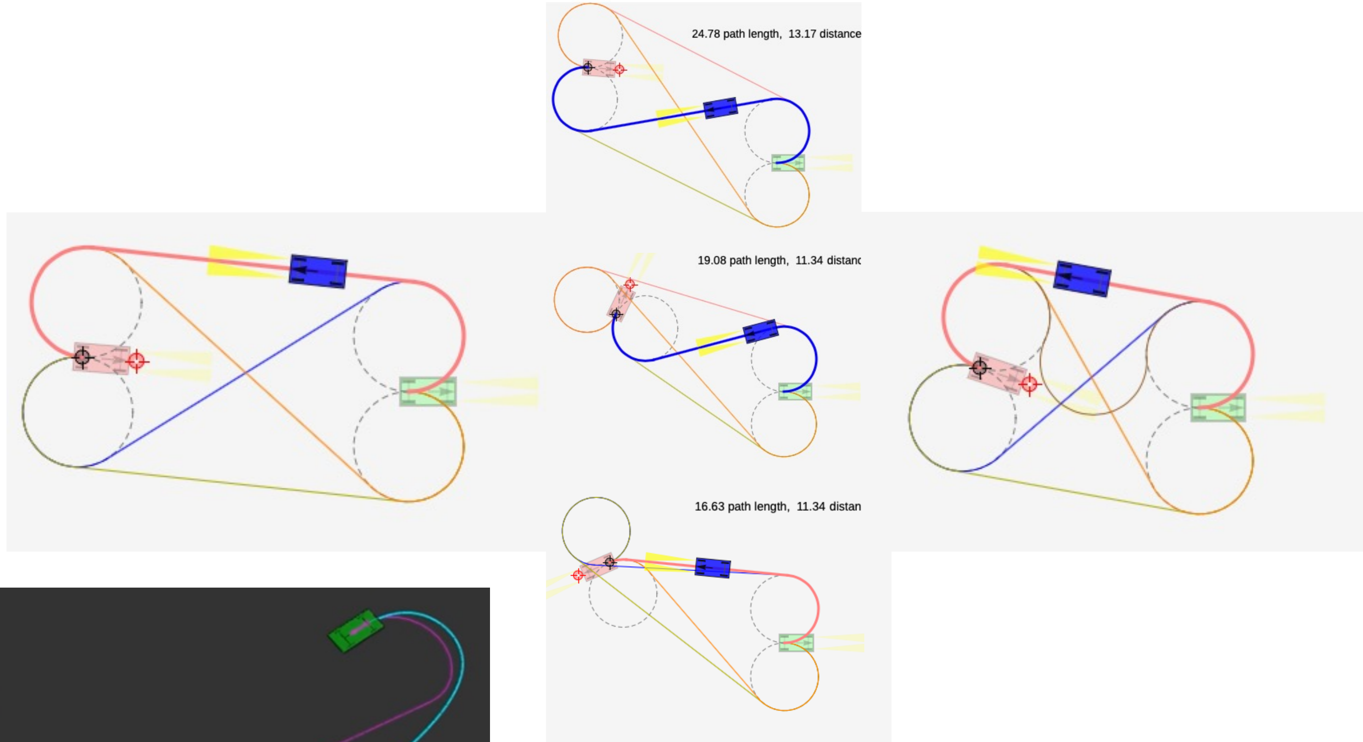


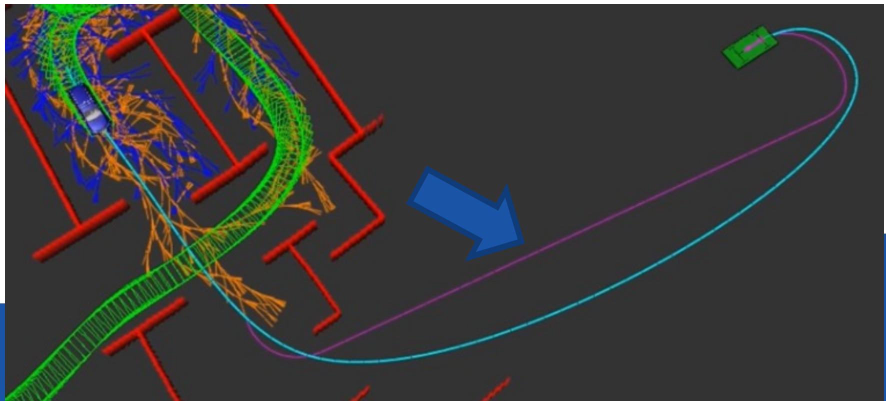


Kinematic

Dubins car

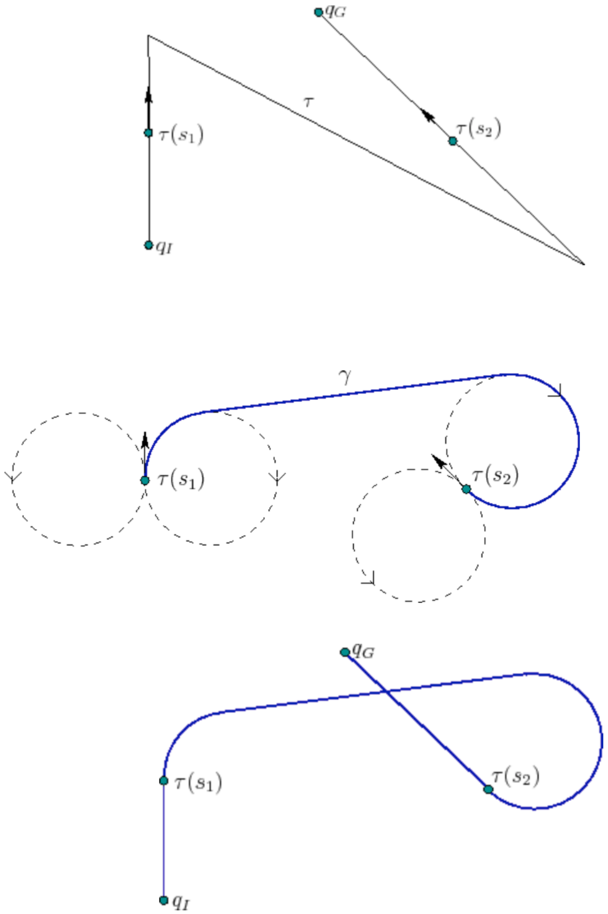
The optimal path for a car can be created using at most 3 circles and 1 straight line





Can we fix an undrivable path?

Plan and Transform



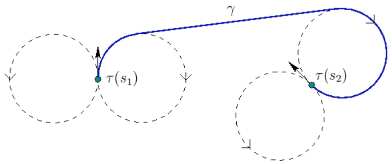
Algorithm

1. Plan a short non-traversable path
2. Pick two points on path
3. Connect with traversable sub-path
4. Iterate from 2, until whole path is traversable

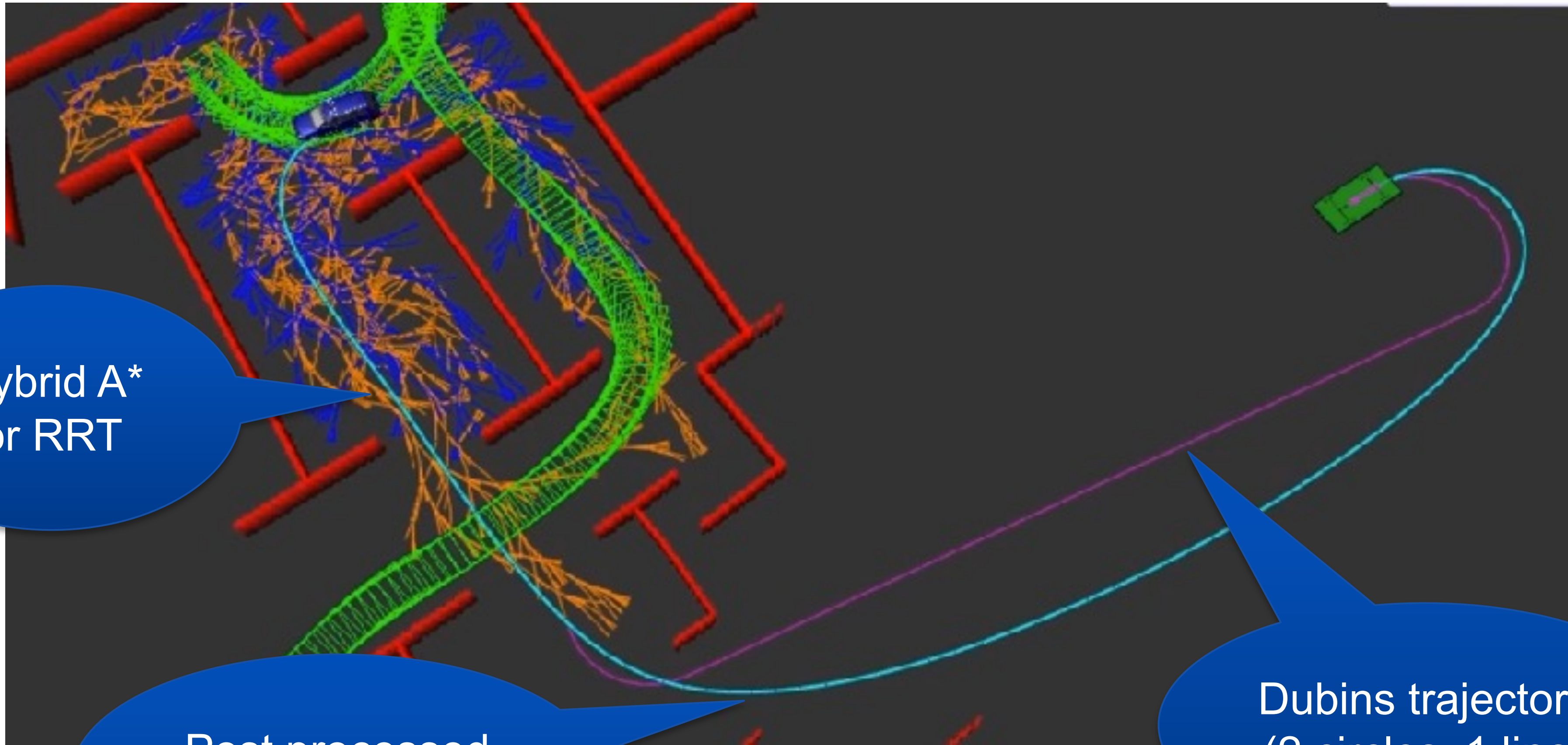
Not always possible
Hard to know when to stop
Can yield very good solutions for visibility graph

not	always
possible	...

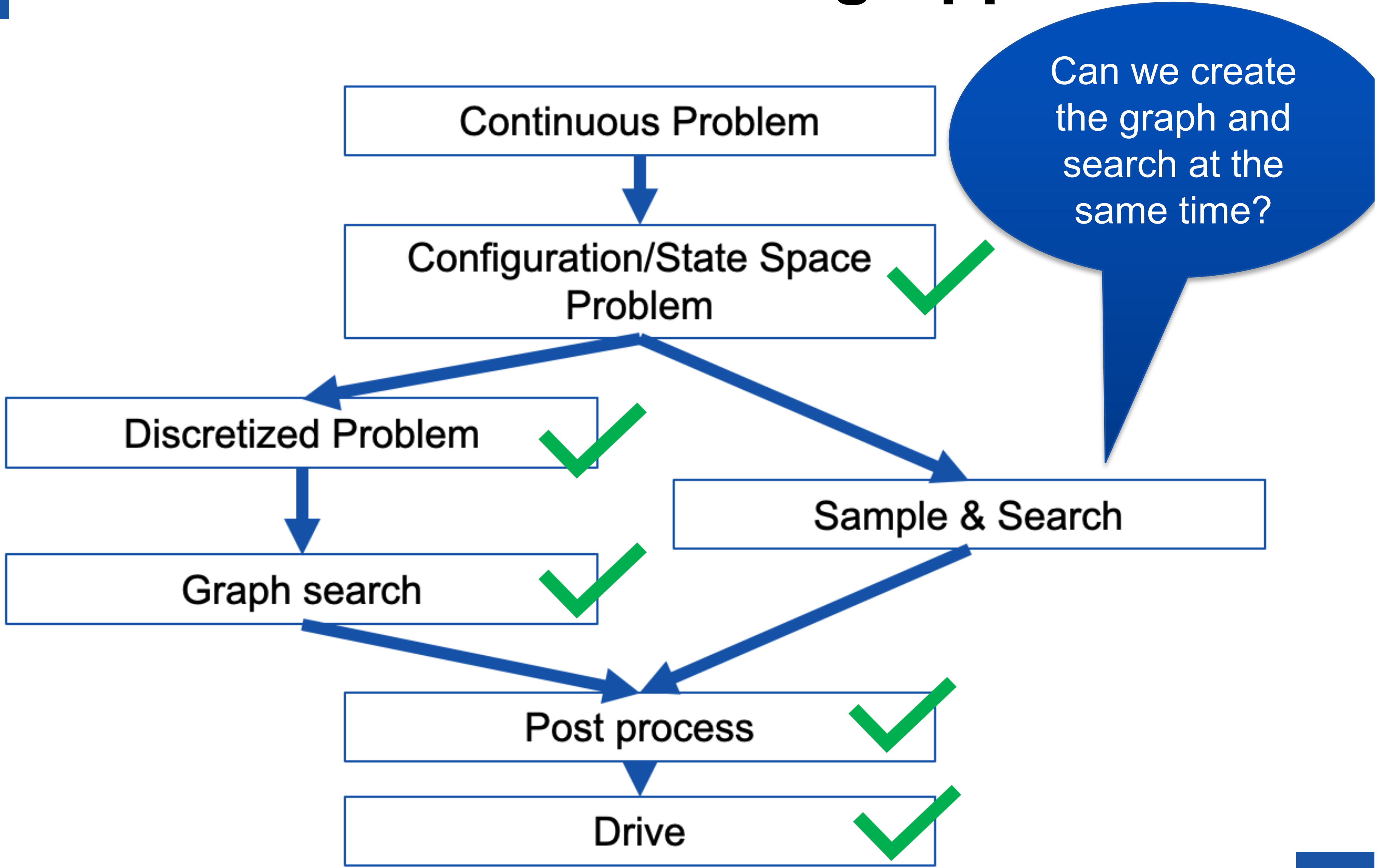
Post processing can also improve performance of drivable trajectories



Smoothen trajectory to enable higher speed using
- Optimization problem
- Heuristic method

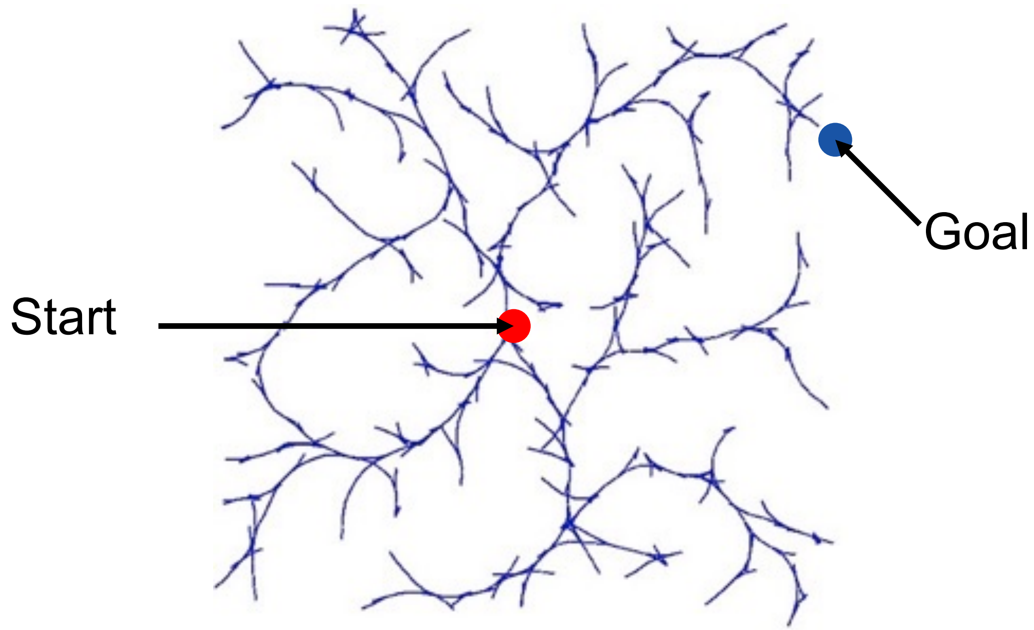


Common Path Planning Approach

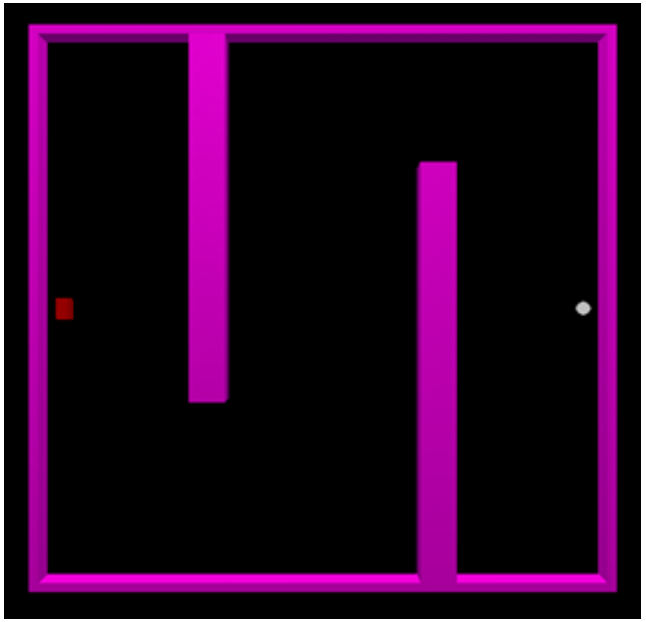


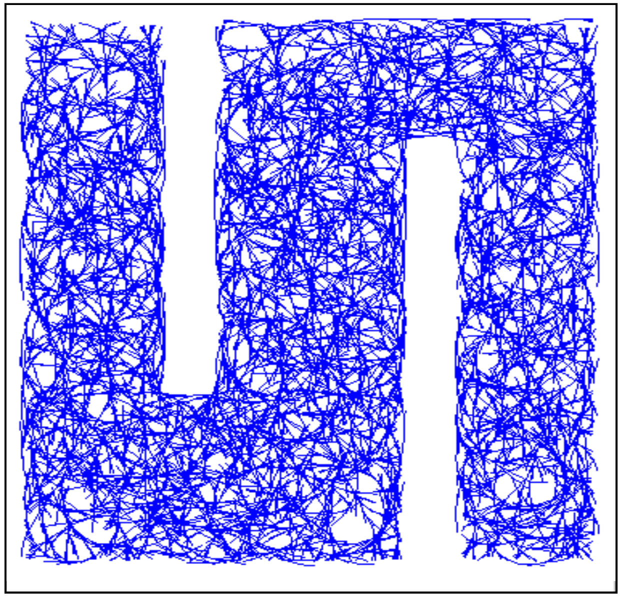
Sample & Search: RRT

RRT: Rapidly Exploring Random Trees



Example: Simple RRT Planner

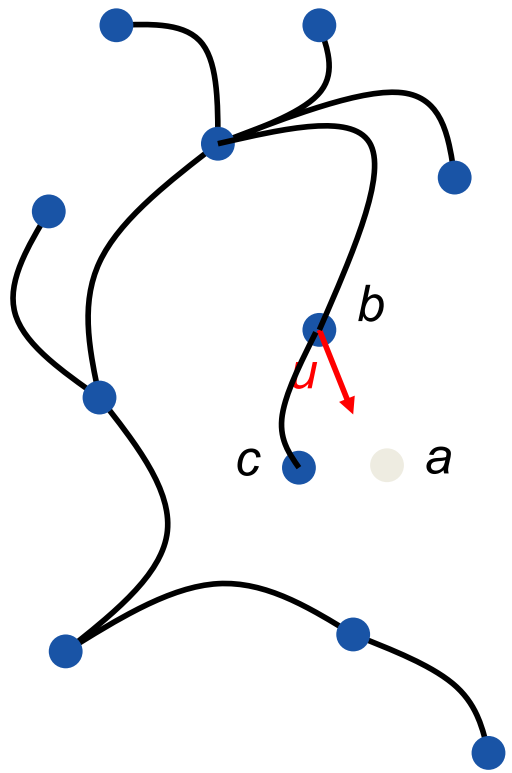




Building an RRT

To extend an RRT:

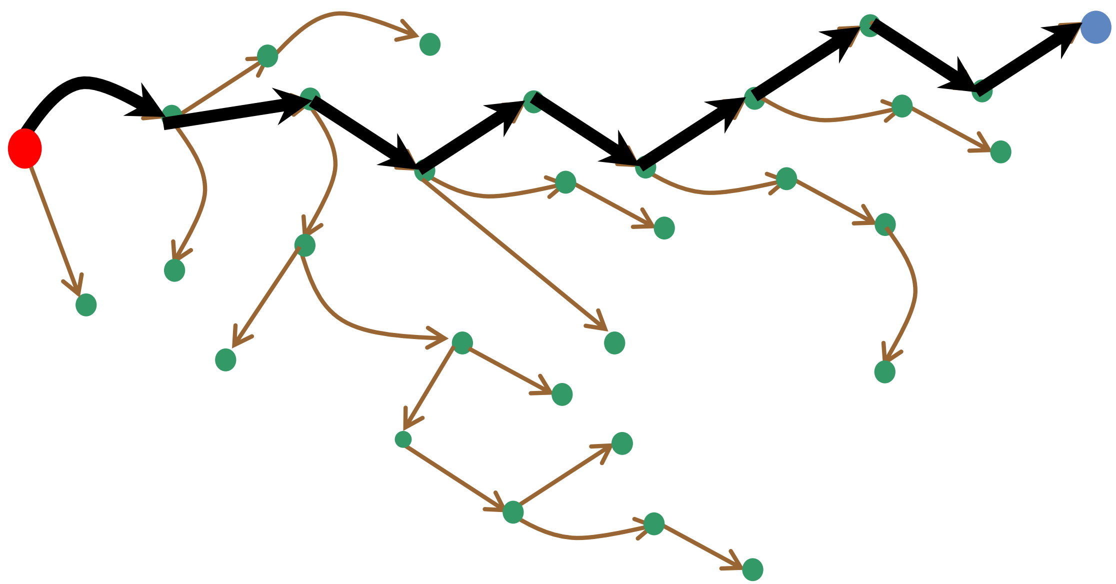
- Pick a **random** point *a* in *X*
- Find *b*, the node of the tree closest to *a*
- Find control inputs *u* to steer the robot from *b* to *a*
- Apply control inputs *u* for time δ, so robot reaches *c*
- If no collisions occur in getting from *a* to *c*, add *c* to RRT and record *u* with new edge



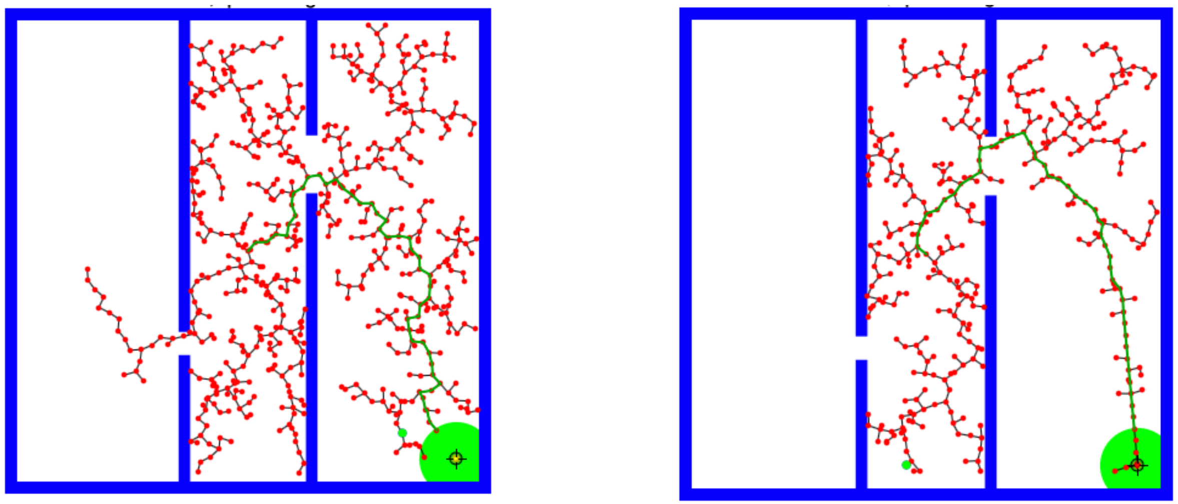
Executing the Path

Once the RRT reaches s_{goal}

- Backtrack along tree to identify edges that lead from s_{start} to s_{goal}
- Drive robot using control inputs stored along edges in the tree



Example: Simple RRT Planner



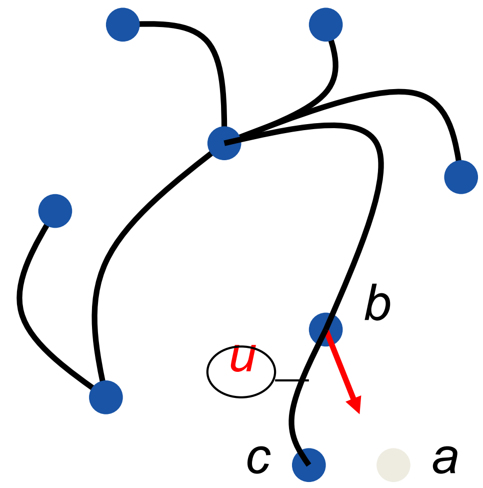
Problem: ordinary RRT explores *X* uniformly

- slow convergence

Solution: bias distribution towards the goal

- Pick the goal point with X% probability

Things to think about…



To extend an RRT

- Pick a random point a in X
- Find b, the node of the tree closest to a
- Find control inputs u to steer the robot from b to a
- Apply control inputs u for time δ, so robot reaches c
- If no collisions occur in getting from a to c, add c to RRT and record u with new edge

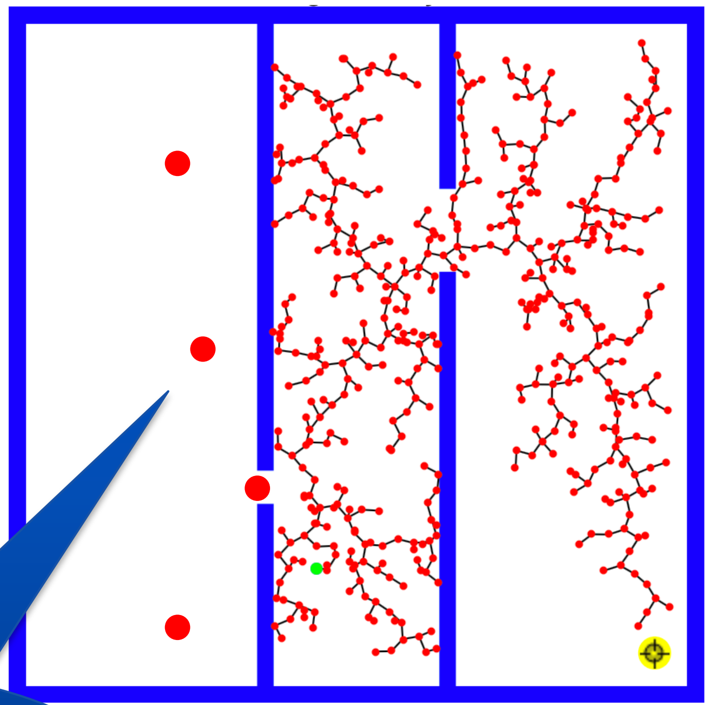
- (x,y)?
- (x,y,theta)?
- (x,y,theta,v)?

Closest in what sense?

Things to think about…

To extend an RRT

- Pick a random point *a* in *X*
- Find *b*, the node of the tree closest to *a*
- Find control inputs *u* to steer the robot from *b* to *a*
- Apply control inputs *u* for time δ, so robot reaches *c*
- If no collisions occur in getting from *a* to *c*, add *c* to RRT and record *u* with new edge



Why are there no Nodes here

Consider sampling bias

- In narrow gaps
- Along optimal grid path
- …

Additional improvement: Bidirectional Planners

Build two RRTs, from start and goal state

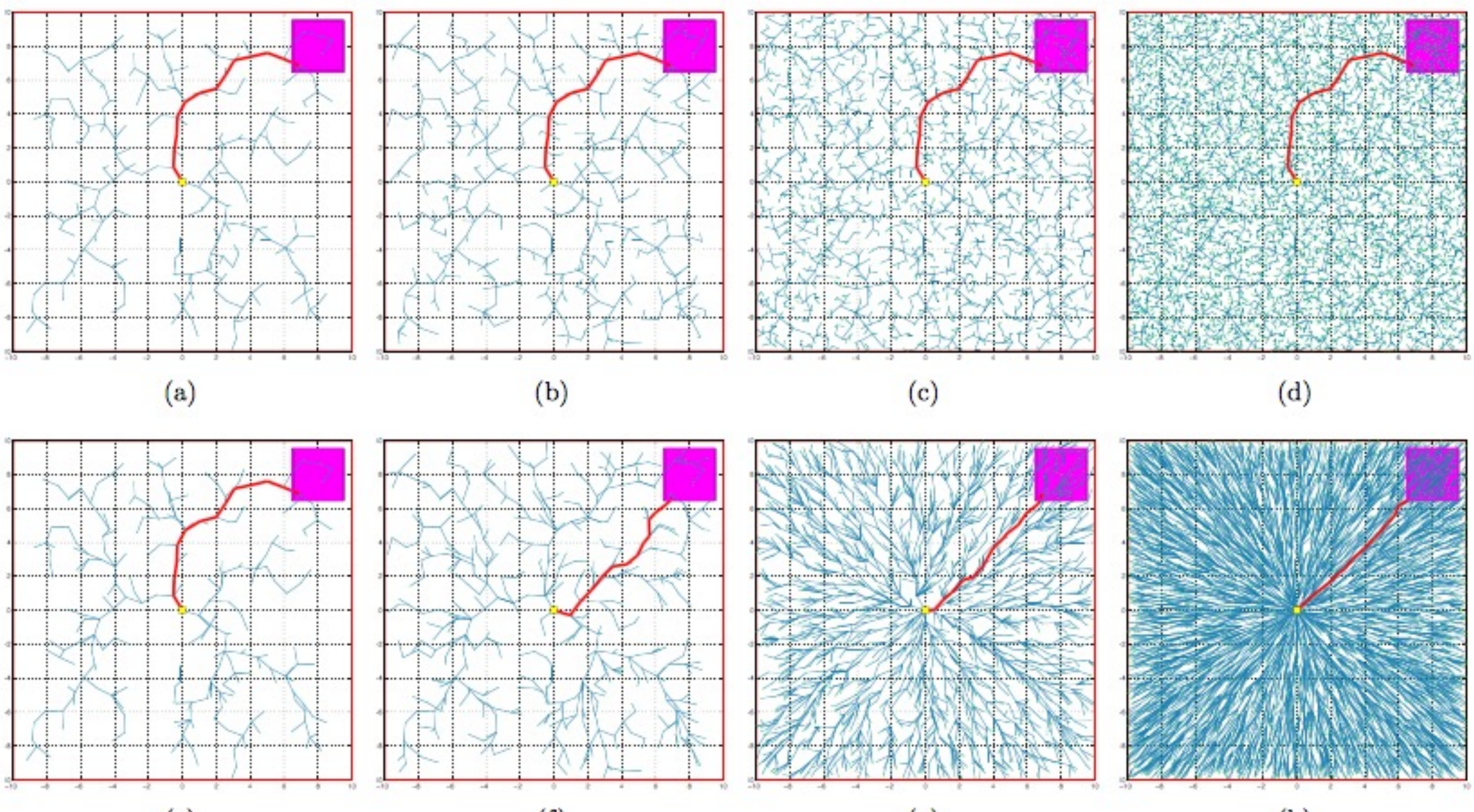
Complication: need to connect two RRTs

- **bias** the distribution, so that the trees meet

Some notes on RRT

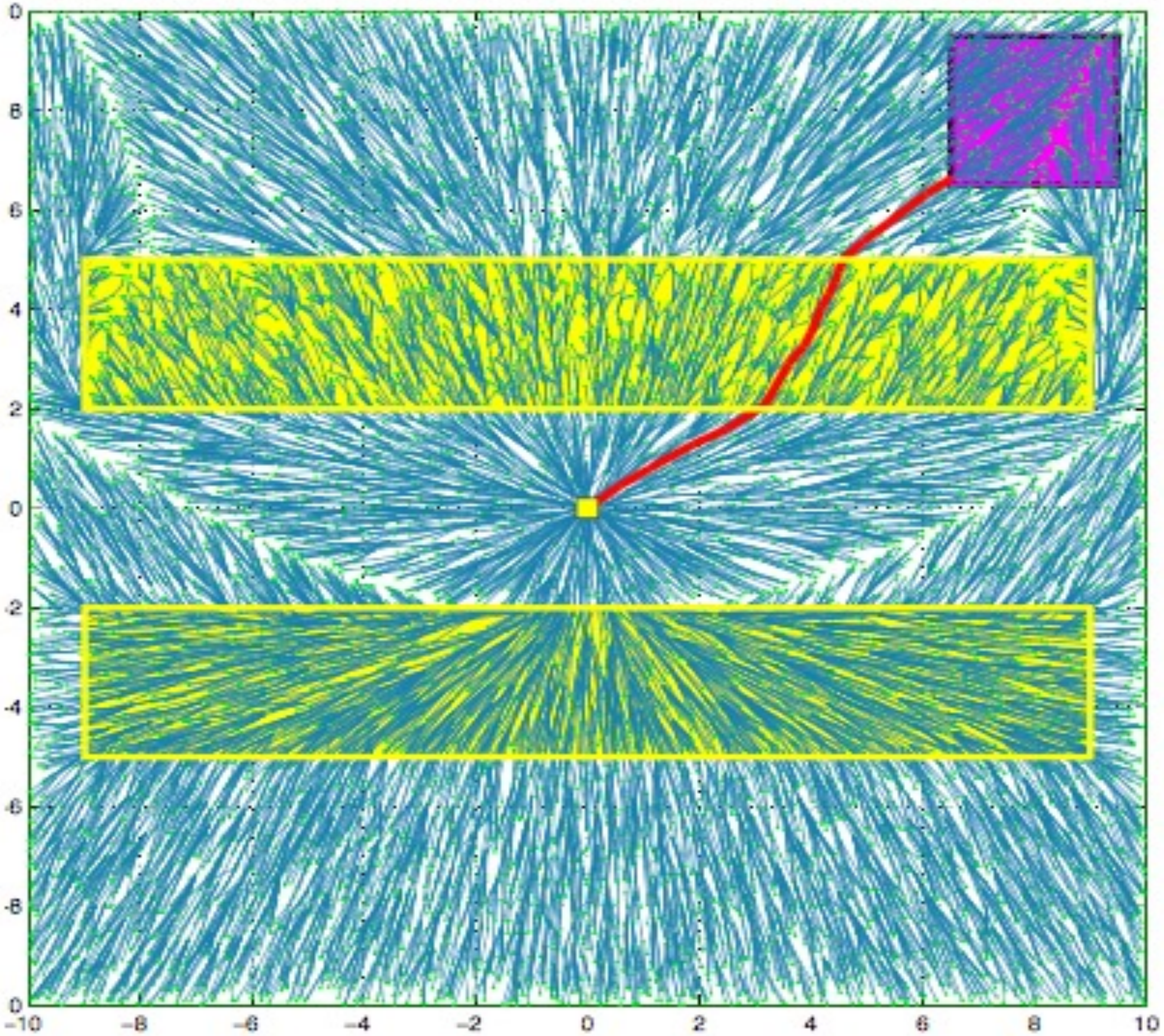
- RRT finds **one** solution with probability →1
  - Quality is not perfect…
- Brake through in 2011 (Karaman and Frazzoli)
  - RRT*
- RRT* finds **optimal** solution with probability → 1

RRT vs RRT* (Karaman and Frazzoli)


(a) (b) (c) (d)

(e) (f) (g) (h)

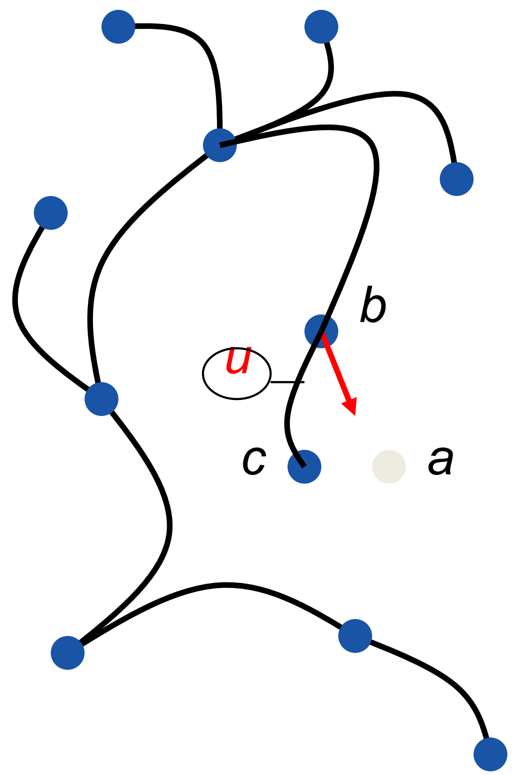
RRT* (High cost and Low cost regions)



How does the RRT* work?

Same start as RRT...

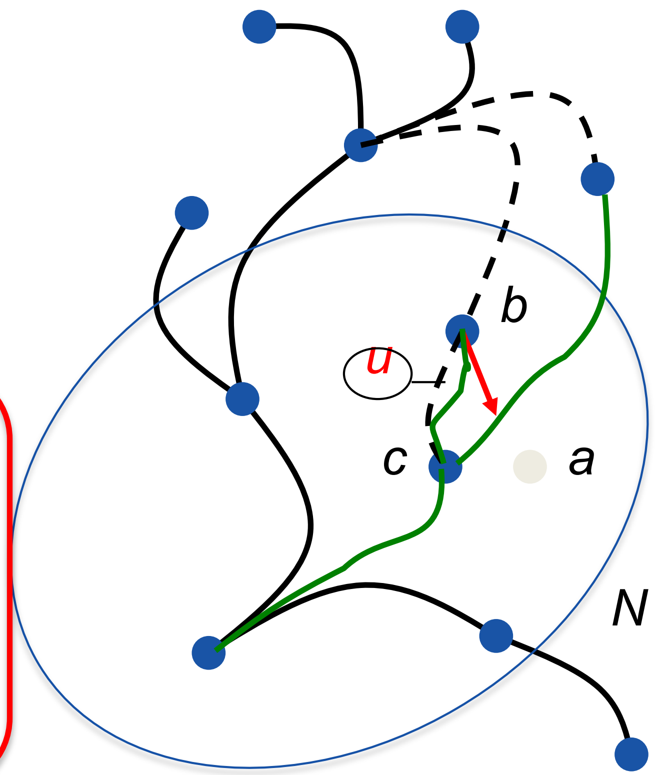
- Pick a **random** point ***a*** in *X*
- Find ***b***, the node of the tree closest to ***a***
- Find control inputs ***u*** to steer the robot from *b* to *a*
- Apply control inputs *u* for time δ, so robot reaches *c*
- If no collisions occur in getting from *a* to *c*, ~~add *c* to RRT and record *u* with new edge~~



How does the RRT* work?

Same start as RRT...

- Pick a **random** point ***a*** in X
- Find ***b***, the node of the tree closest to ***a***
- Find control inputs ***u*** to steer the robot from b to a
- Apply control inputs u for time δ, so robot reaches c
- If no collisions occur in getting from a to c
  - Find set of Neighbors N of c
  - Choose Best parent!
  - Try to adopt Neighbors (if good)



RRT* (2011, original)

a

b

c

Neighbors N

Find best parent

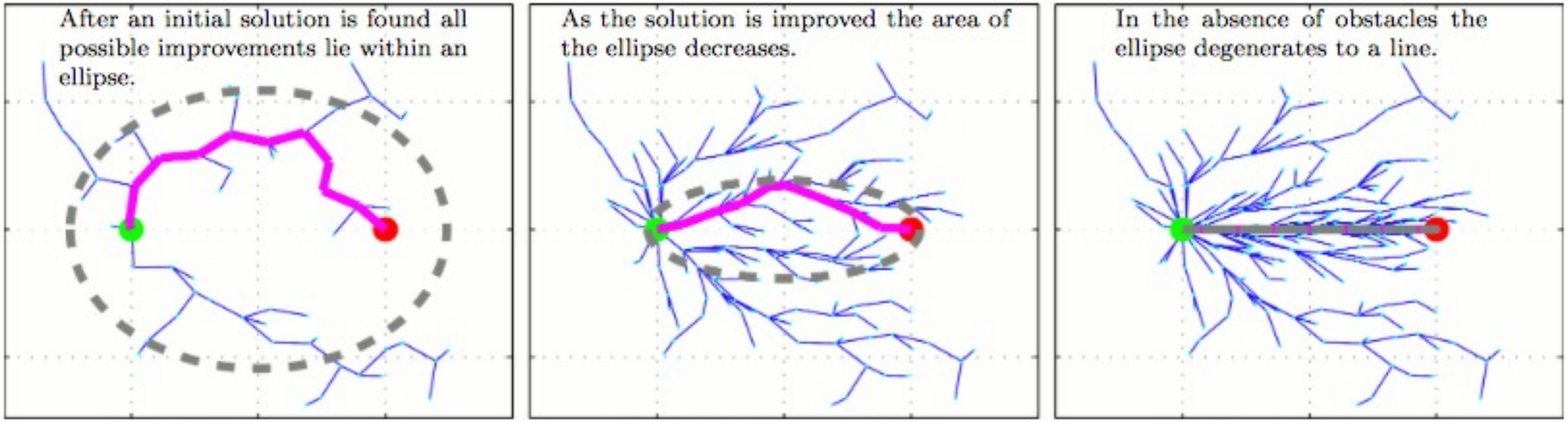
Adopt new children (if improvment)

Algorithm 6: RRT*

```
V ← {x_init}; E ← ∅;
for i = 1, ..., n do
    x_rand ← SampleFree_i;
    x_nearest ← Nearest(G = (V, E), x_rand);
    x_new ← Steer(x_nearest, x_rand) ;
    if ObtacleFree(x_nearest, x_new) then
        X_near ← Near(G = (V, E), x_new, min{γ_RRT*(log(card (V))/ card (V))^{1/d}, η}) ;
        V ← V ∪ {x_new};
        x_min ← x_nearest; c_min ← Cost(x_nearest) + c(Line(x_nearest, x_new));
        foreach x_near ∈ X_near do                // Connect along a minimum-cost path
            if CollisionFree(x_near, x_new) ∧ Cost(x_near) + c(Line(x_near, x_new)) < c_min then
                x_min ← x_near; c_min ← Cost(x_near) + c(Line(x_near, x_new))
        E ← E ∪ {(x_min, x_new)};
        foreach x_near ∈ X_near do                // Rewire the tree
            if CollisionFree(x_new, x_near) ∧ Cost(x_new) + c(Line(x_new, x_near)) < Cost(x_near)
            then x_parent ← Parent(x_near);
            E ← (E \ {(x_parent, x_near)}) ∪ {(x_new, x_near)}
return G = (V, E);
```

Informed RRT*
(2013, Gamell et al., searching within ellipsoids)

Key observation!



59 iterations, $c_{best} = 148.24$
(a)

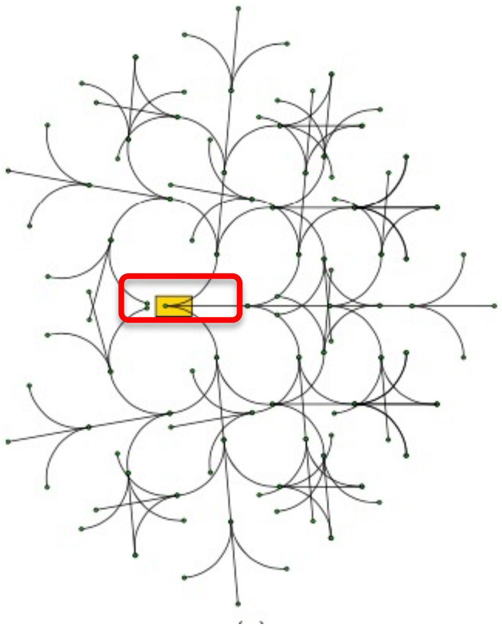
175 iterations, $c_{best} = 107.12$
(b)

1142 iterations, $c_{best} = 100$
(c)

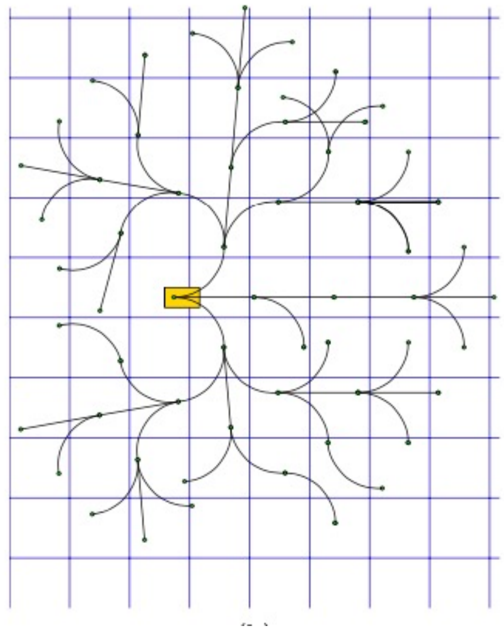
Other ways of biasing sampling are possible: I.e. Close to Visibility Graph solution…

What about A* and grids?

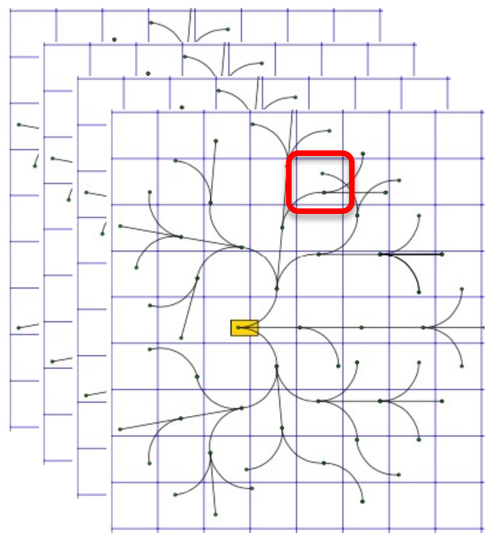
Hybrid A*



If we just build a search tree we get copies of same state

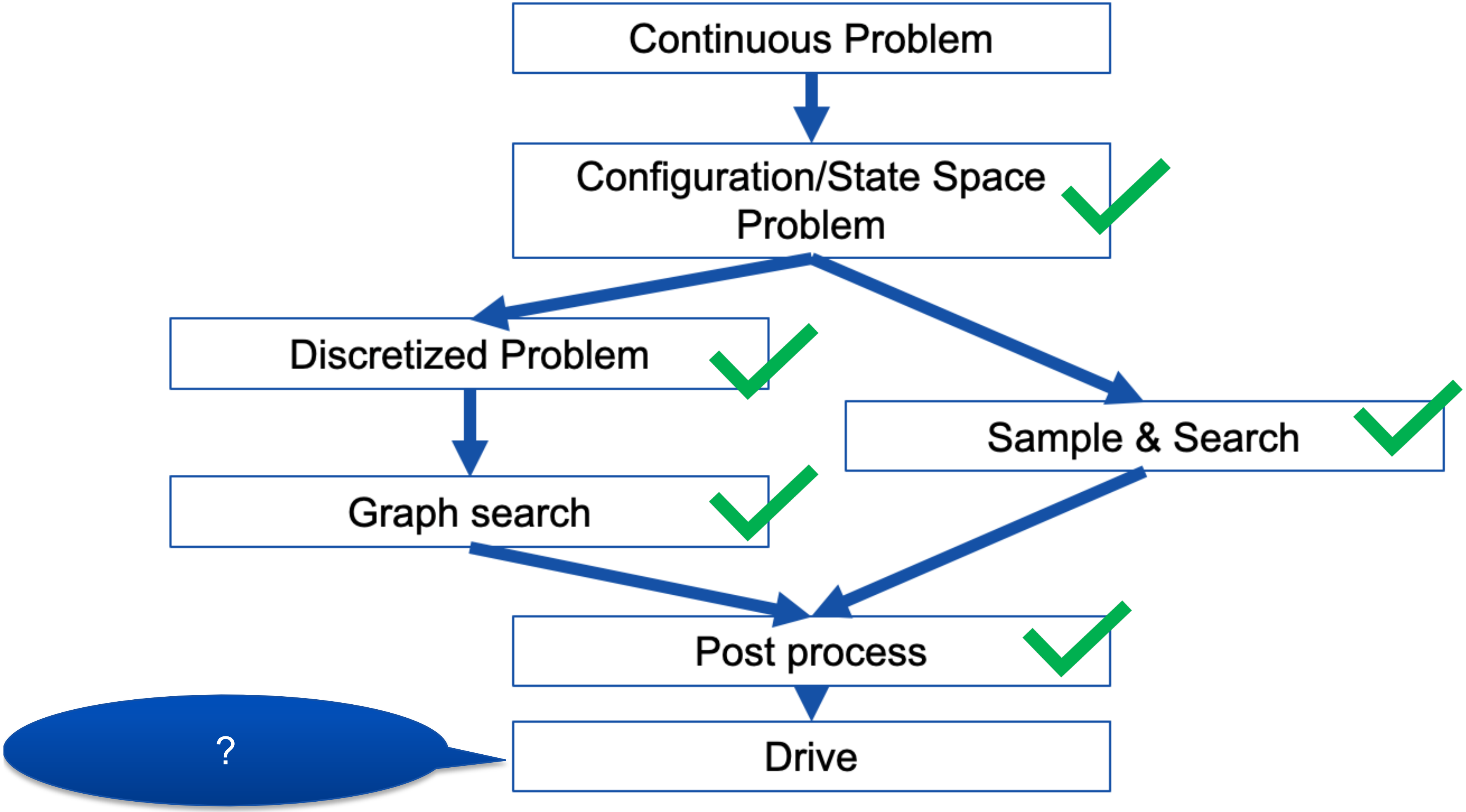


Allowing just one state in each grid



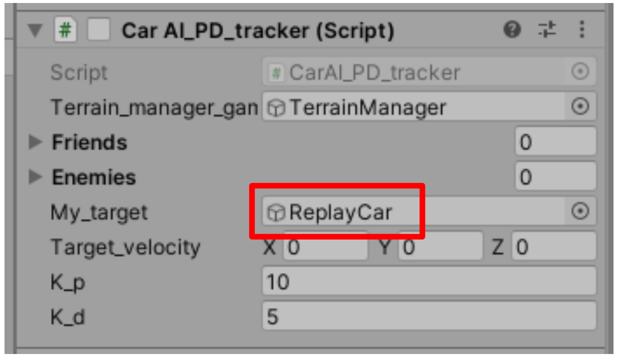
Allowing 4 states in each grid: theta=(0,pi/2,pi,3pi/2)

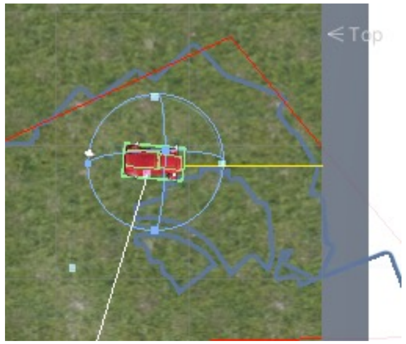
Common Path Planning Approach



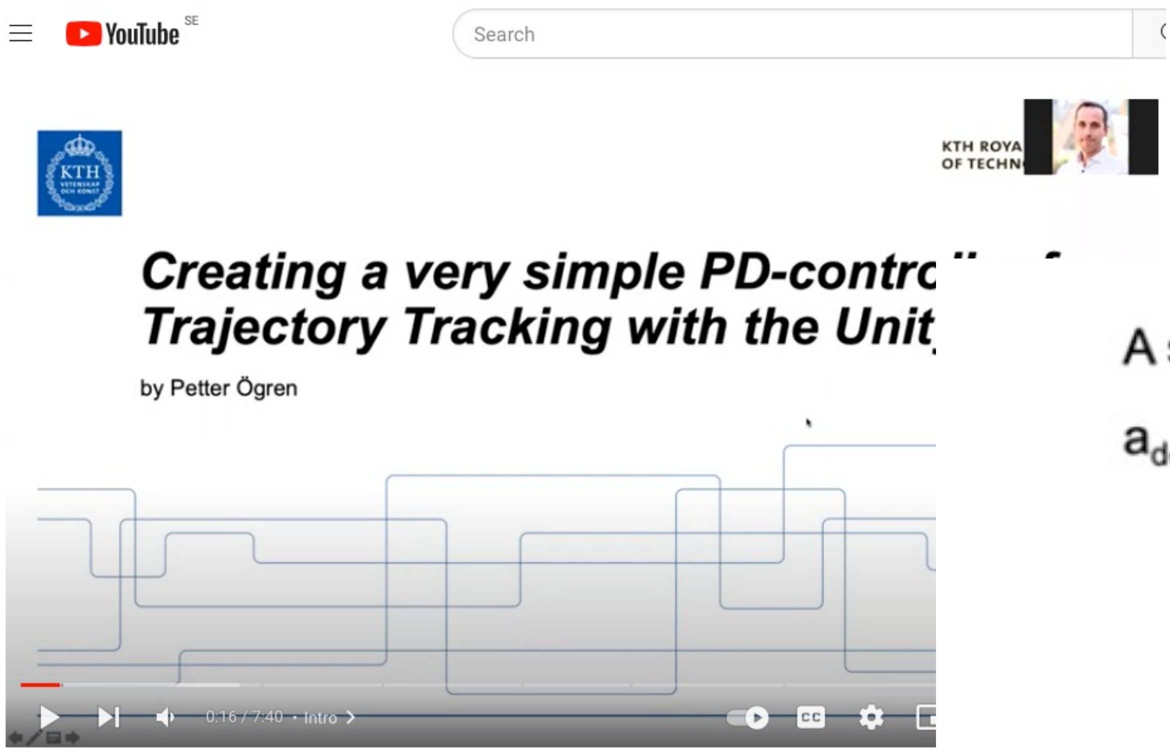
How to drive along your planned path?

- Draw inspiration from
  - CarAI_PD_tracker.cs
  - Google “Pure pursuit controller”
- How fast should you go?
- What happens if you collide?
- What is best?
  - Advanced planning and simple driving
  - Simple planning and advanced driving

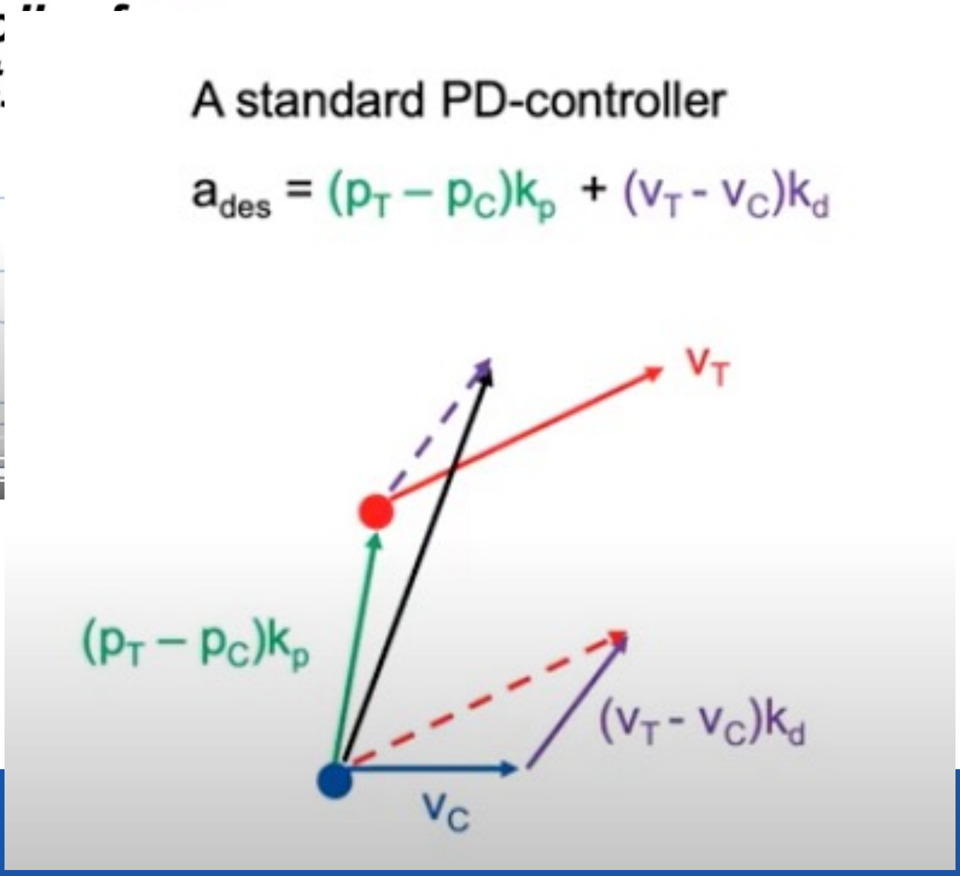




Driving along a path (youtube video)

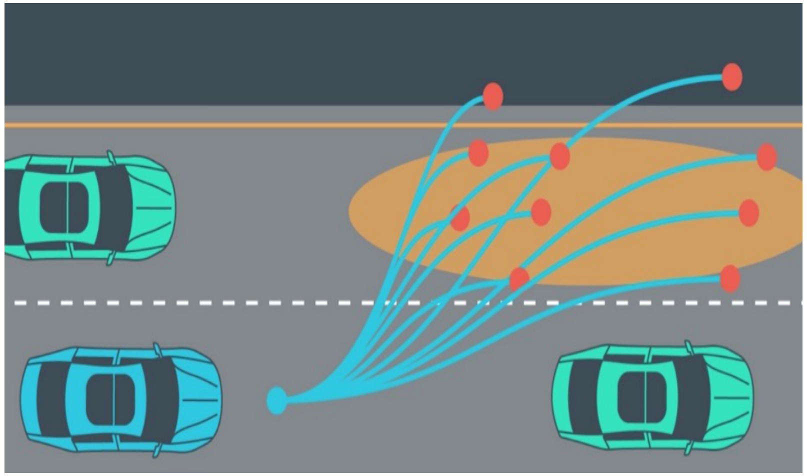


Creating a simple PD-controller for trajectory tracking with a Unity car

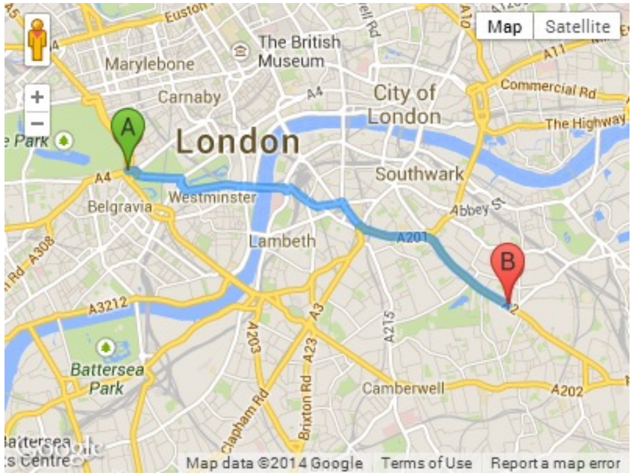


Multi Horizon Path Planning

5 sec horizon
Full vehicle model

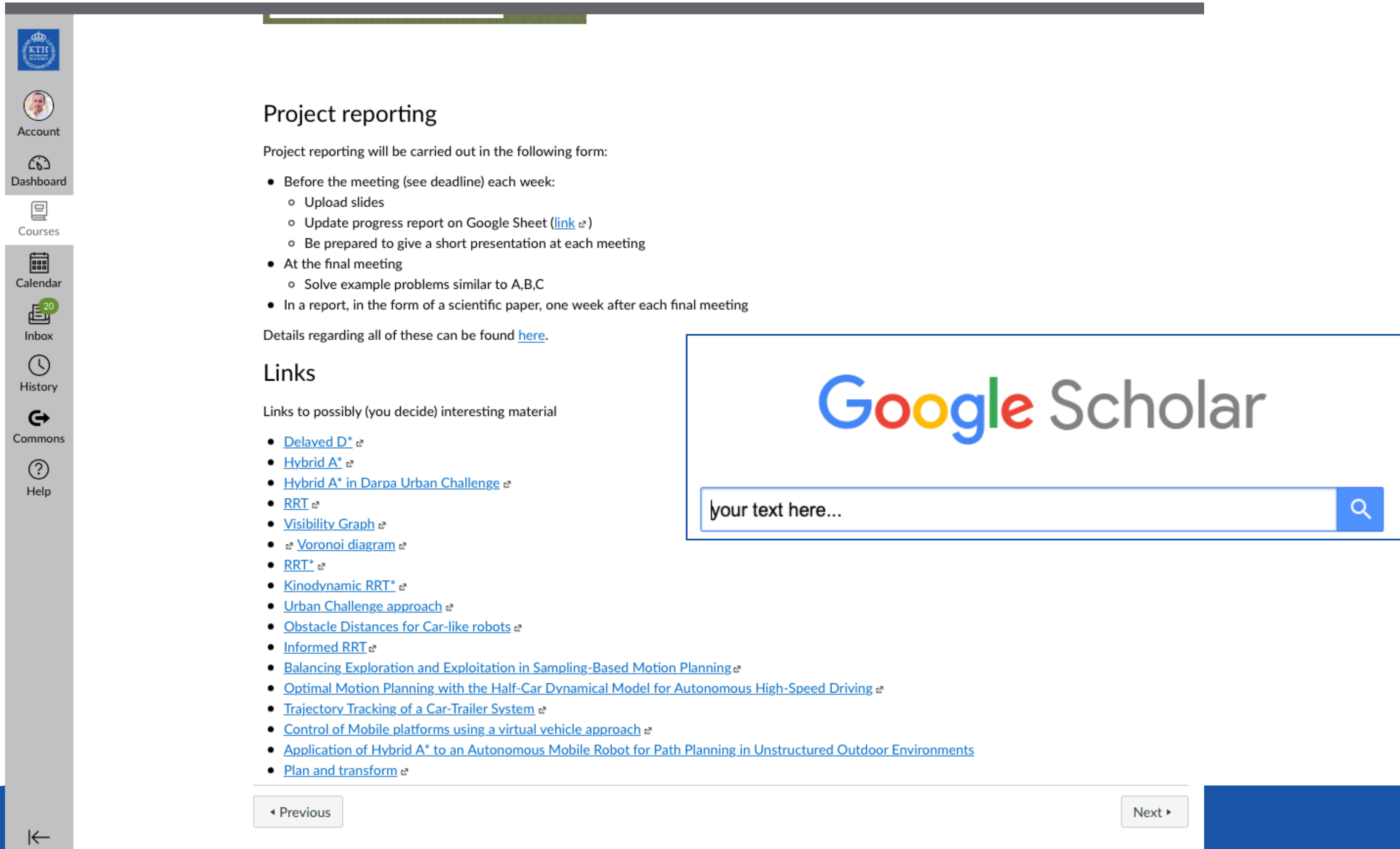


5 hour horizon
Position is enough
(not heading/speed)



Remember:

You need at least **5 references described** in your report



The End