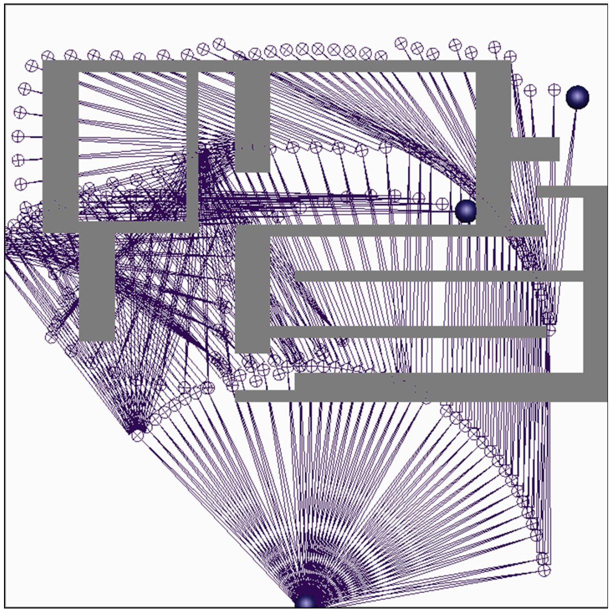
15-780: Graduate AI
Lecture 14. Spatial Search; Uncertainty

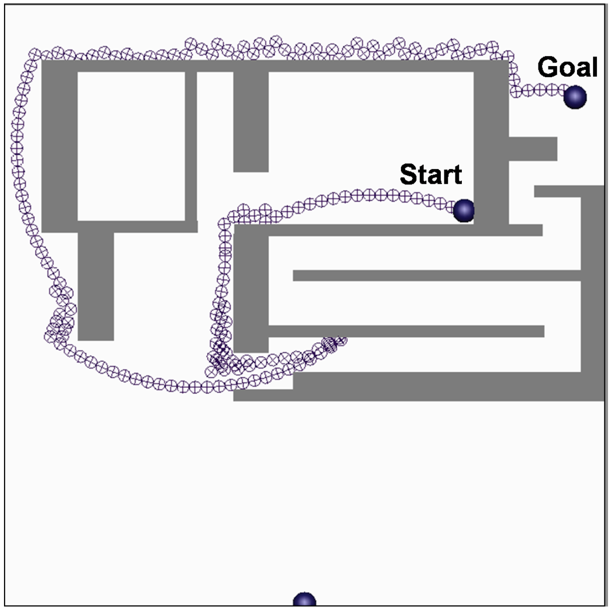
Geoff Gordon (this lecture)
Tuomas Sandholm
TAs Sam Ganzfried, Byron Boots

Spatial Planning

Plans in Space…



Optimal Solution



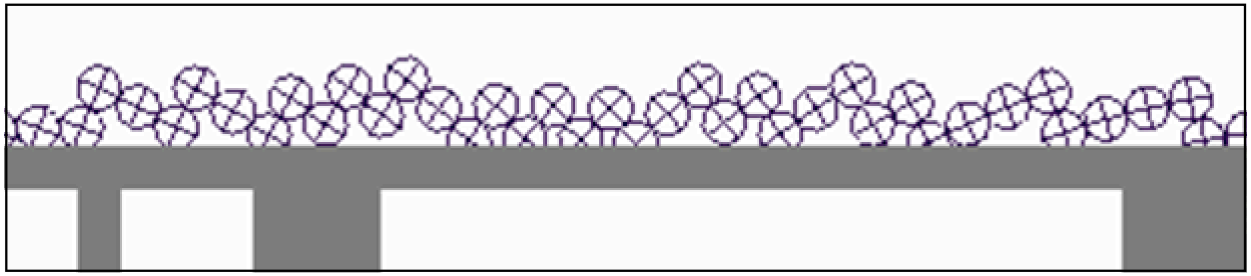
End-effector Trajectory

- *A* can be used for many things*
- *Here, A* for spatial planning (in contrast to, e.g., jobshop scheduling)*

What’s wrong w/ A* guarantees?

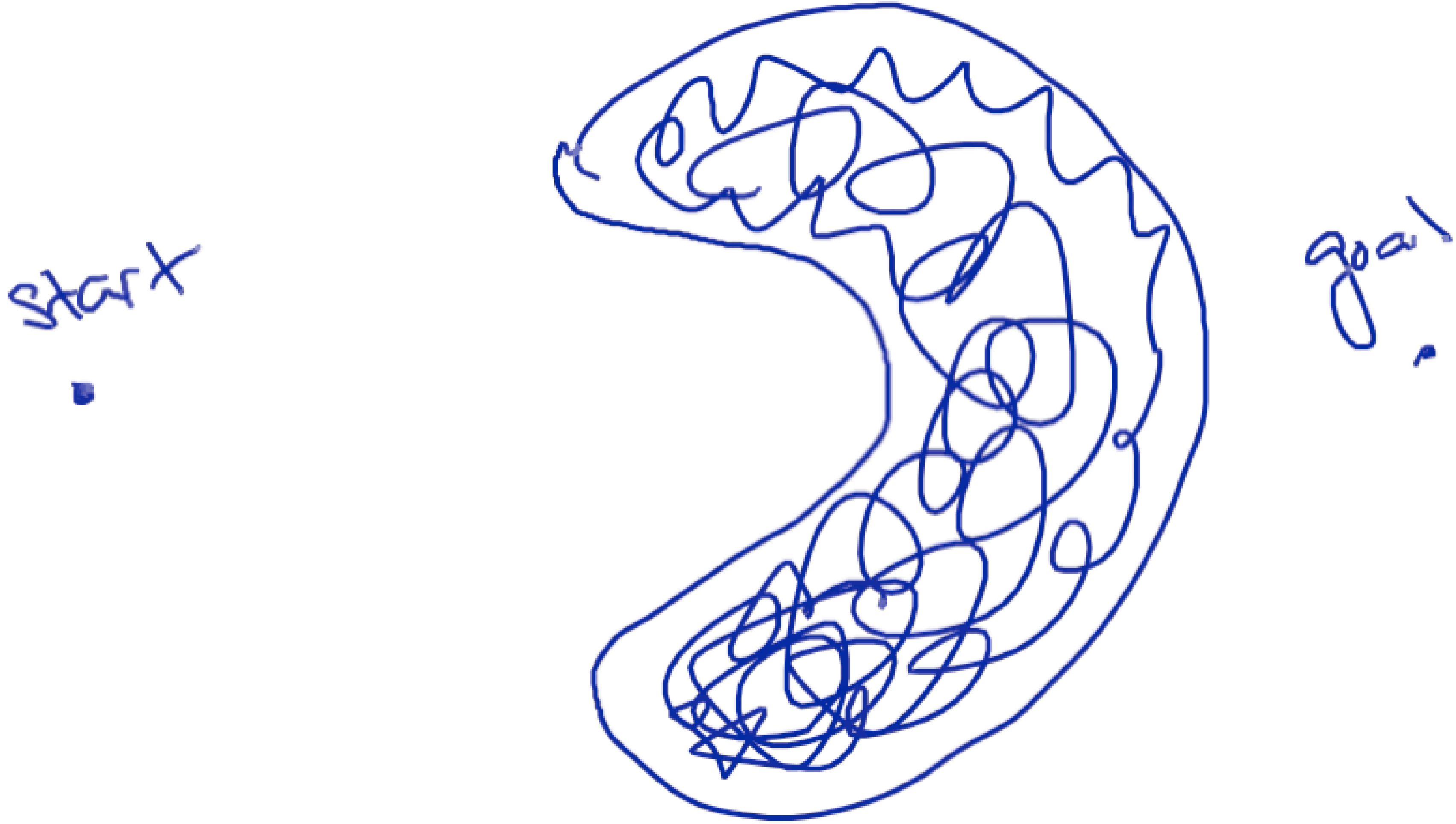
- *(**optimality**) A* finds a solution of cost g**
- *(**efficiency**) A* expands no nodes that have f(node) > g**

What's wrong with A*?

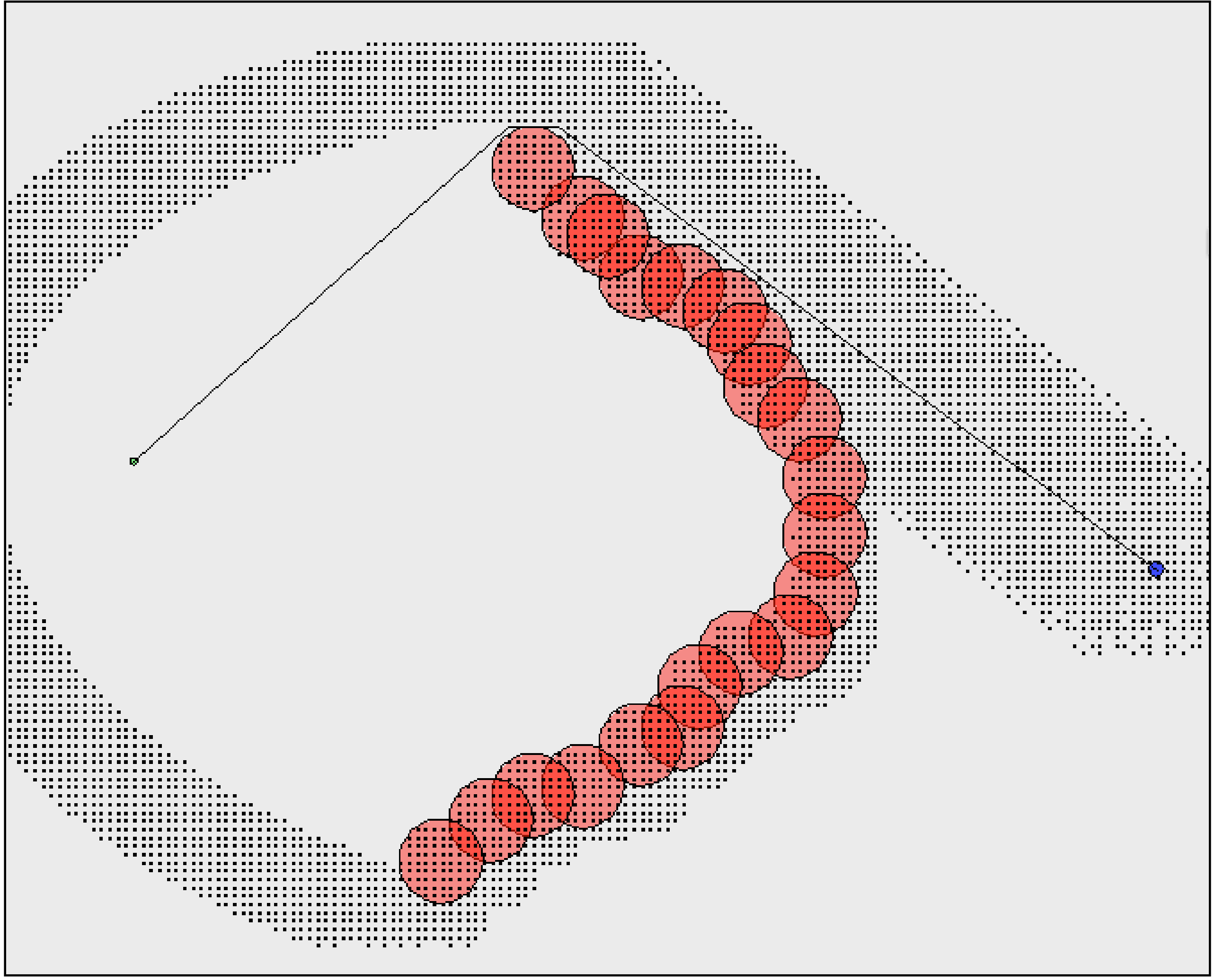


- *Discretized space into tiny little chunks*
    - *a few degrees rotation of a joint*
    - **Lots** *of states ⇒ lots of states w/ $f \leq g^*$*
- *Discretized actions too*
    - *one joint at a time, discrete angles*
- *Results in jagged paths*

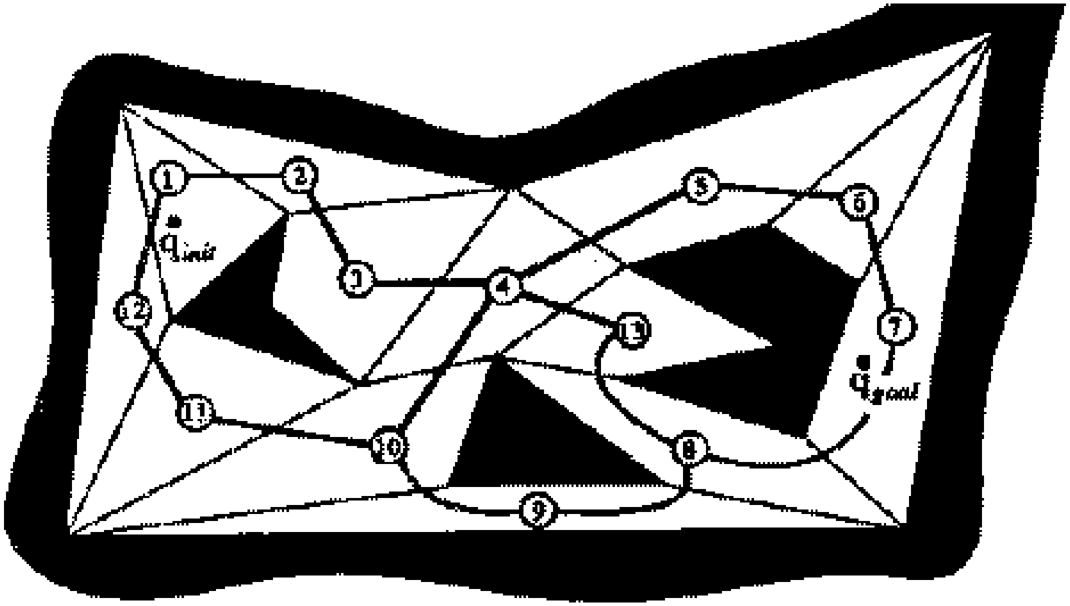
What's wrong with A*?



Snapshot of A*



Wouldn’t it be nice…

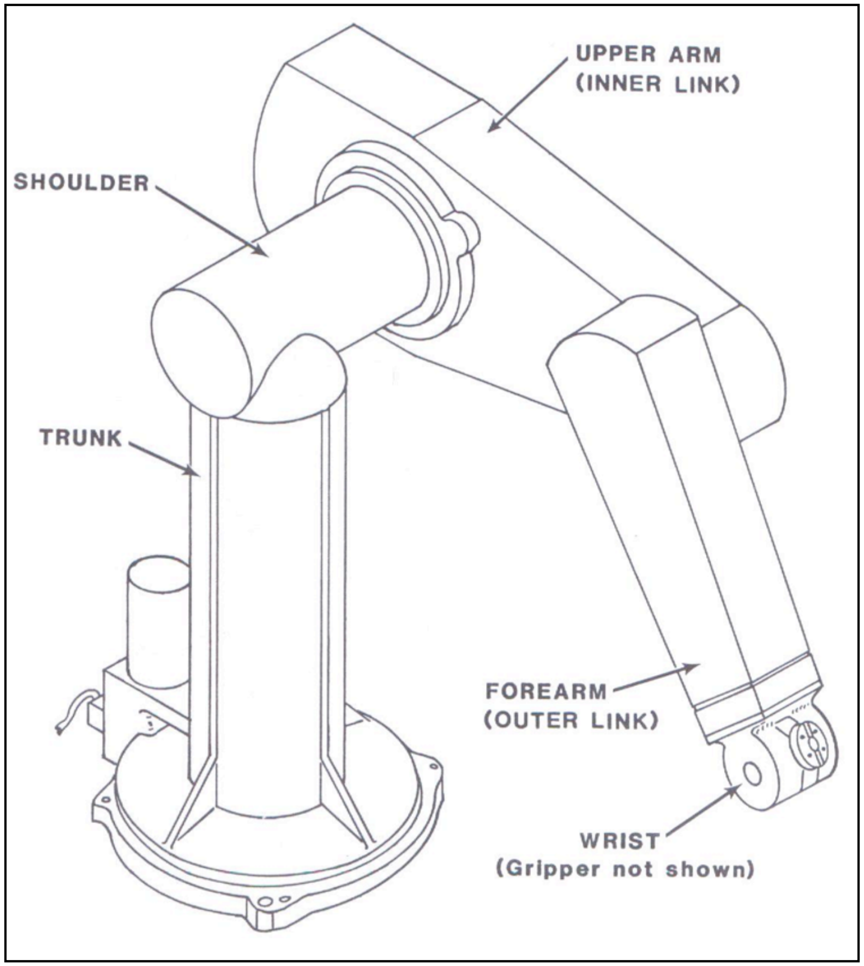


- *… if we could break things up based more on the real geometry of the world?*
- Robot Motion Planning *by Jean-Claude Latombe*

Physical system

- *A moderate number of real-valued coordinates*
- *Deterministic, continuous dynamics*
- *Continuous goal set (or a few pieces)*
- *Cost = time, work, torque, …*

Typical physical system



A kinematic chain

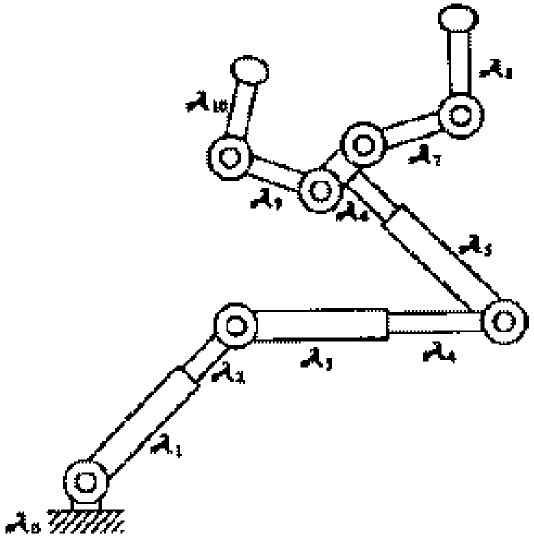


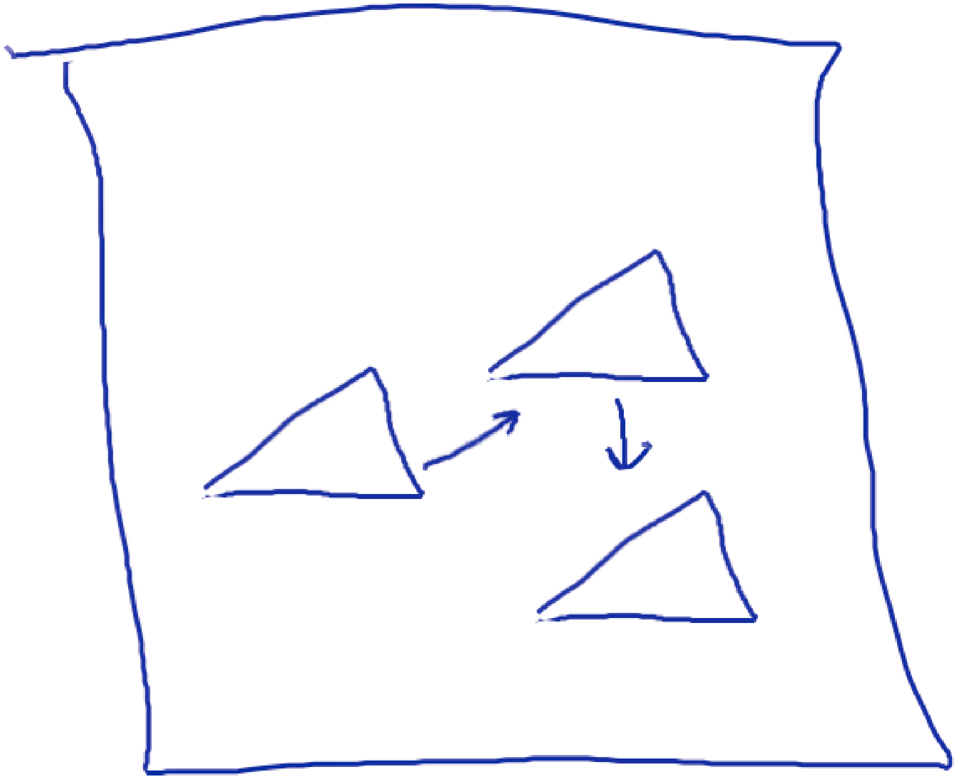
Fig. 11. Structure of the 10-DOF manipulator.

- *Rigid links connected by joints*
  - *revolute or prismatic*
- *Configuration*

  $$\mathbf{q} = (q_1, q_2, \ldots)$$

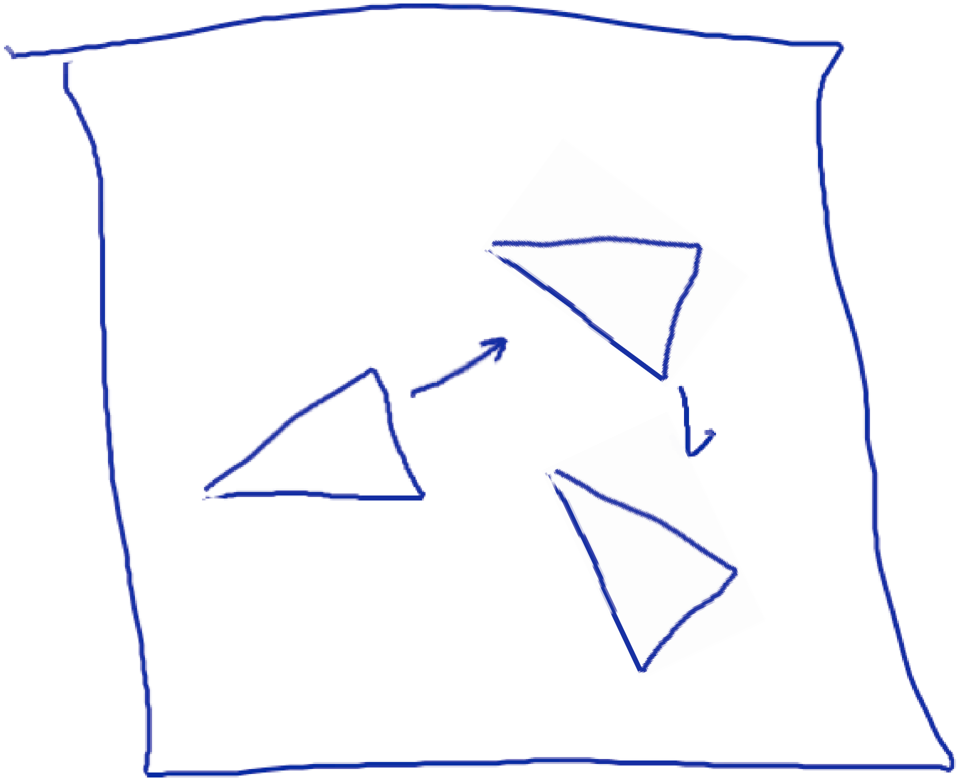
  q_i = *angle or length of joint i*
- *Dimension of* **q** = *"degrees of freedom"*

Mobile robots



- *Translating in space = 2 dof*

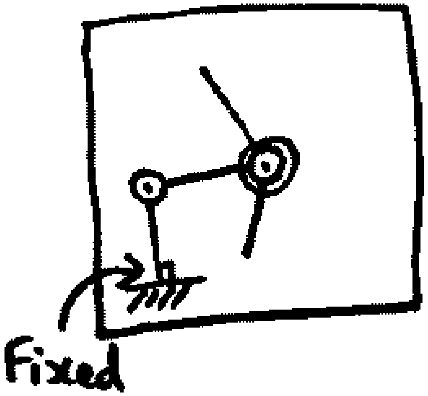
More mobility



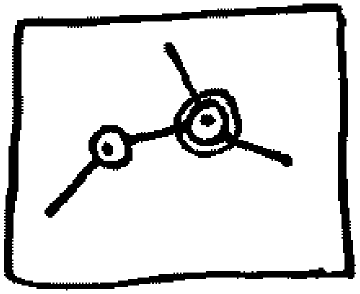
- *Translation + rotation = 3 dof*

Q: How many dofs?

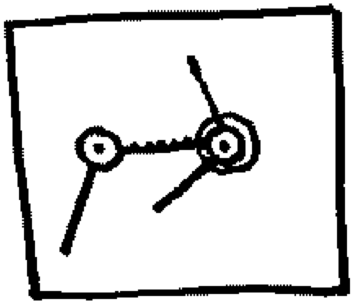
- *3d translation & rotation*



How many dofs?



Free flying
How many dofs?

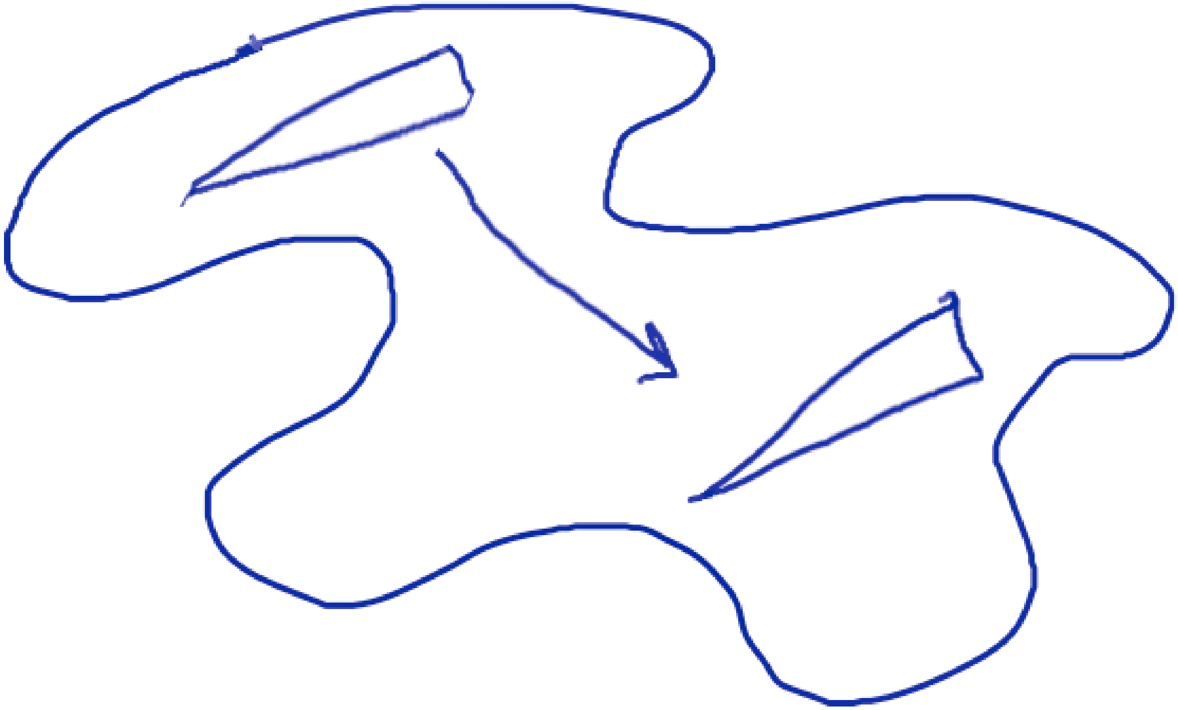


Midline ____ must always be horizontal.
How many DOFs?

The configuration $\underset{\sim}{q}$ has one real valued entry per DOF.

credit: Andrew Moore

Robot kinematic motion planning



- *Now let's add obstacles*

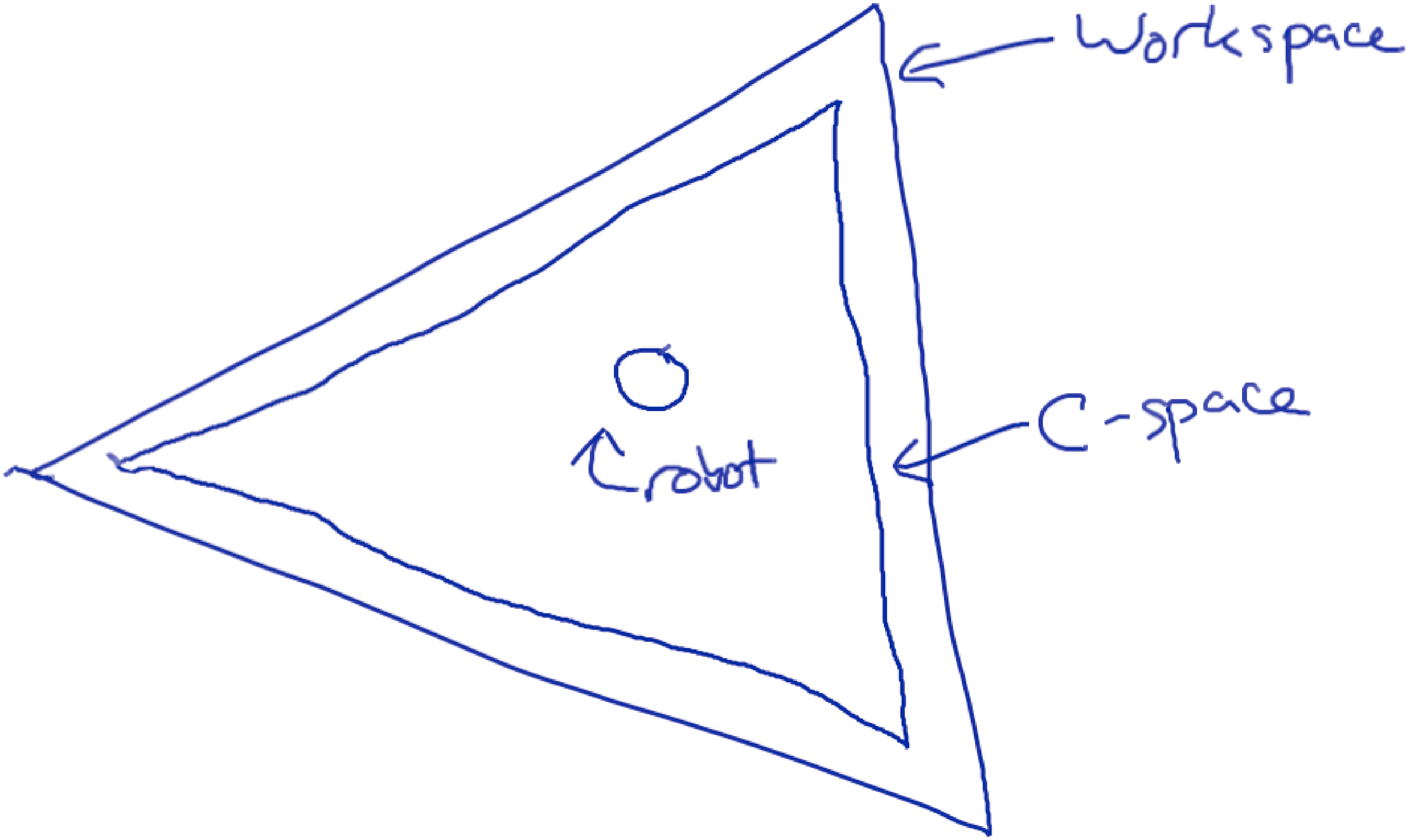
Configuration space

- *For any configuration* **q**, *can test whether it intersects obstacles*
- *Set of legal configs is "configuration space" C (a subset of a dof-dimensional vector space)*
- *Path is a continuous function from [0,1] into C with* $q(0) = \mathbf{q}_s$ *and* $q(1) = \mathbf{q}_g$

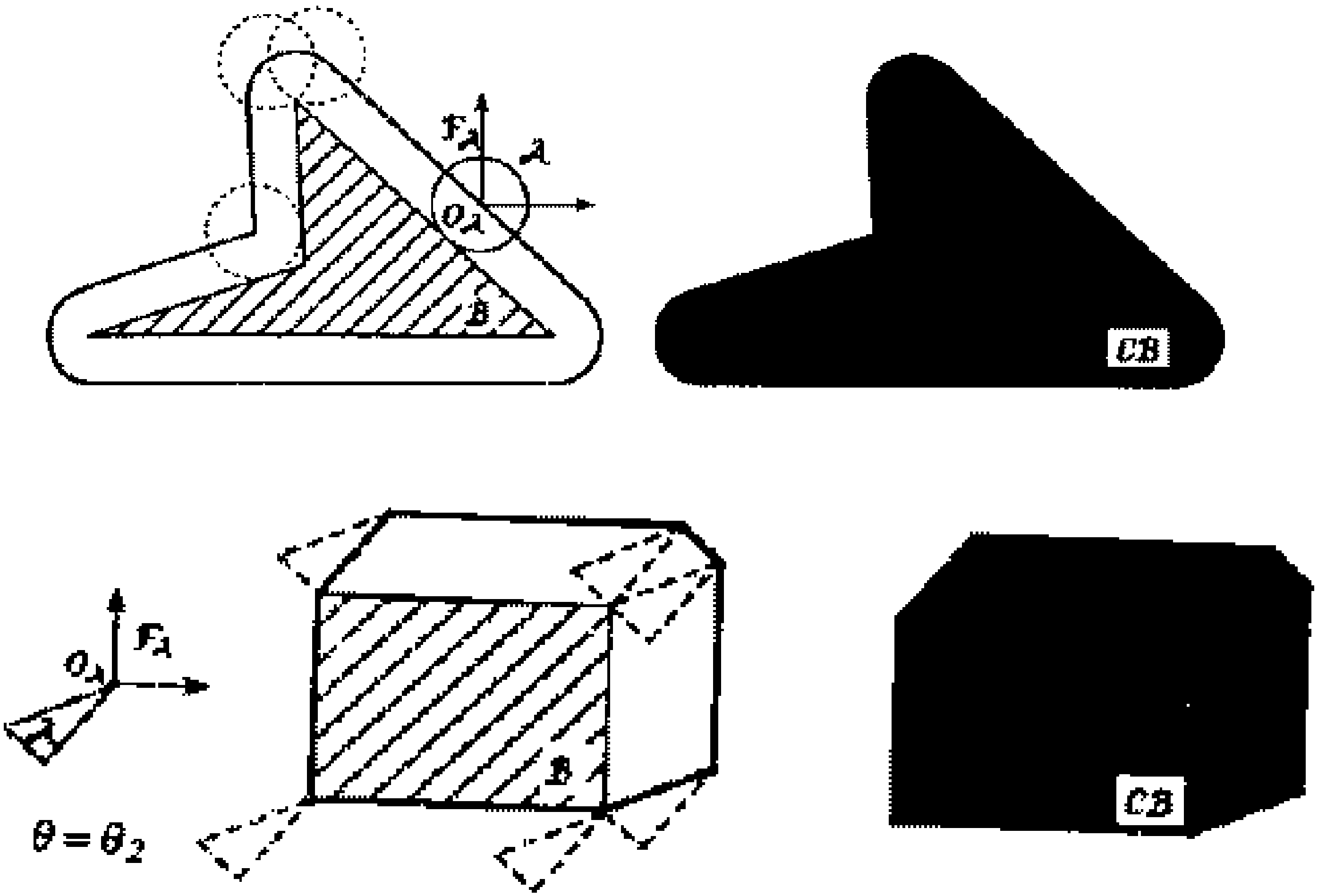
Note: dynamic planning

- *Includes inertia as well as configuration*
- $\mathbf{q}, \mathring{\mathbf{q}}$
- *Harder, since twice as many dofs*
- *More later…*

C-space example



More C-space examples



Another C-space example

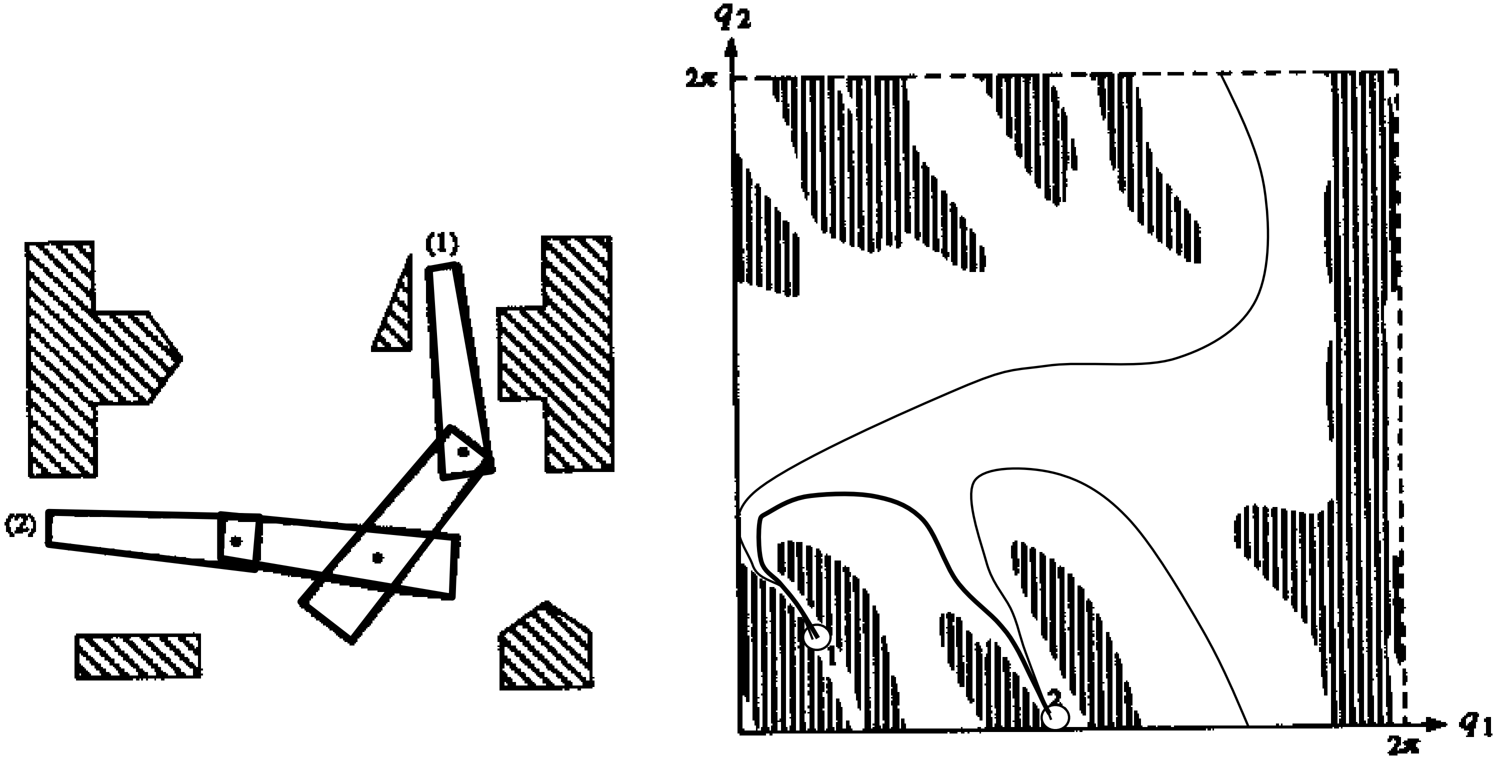
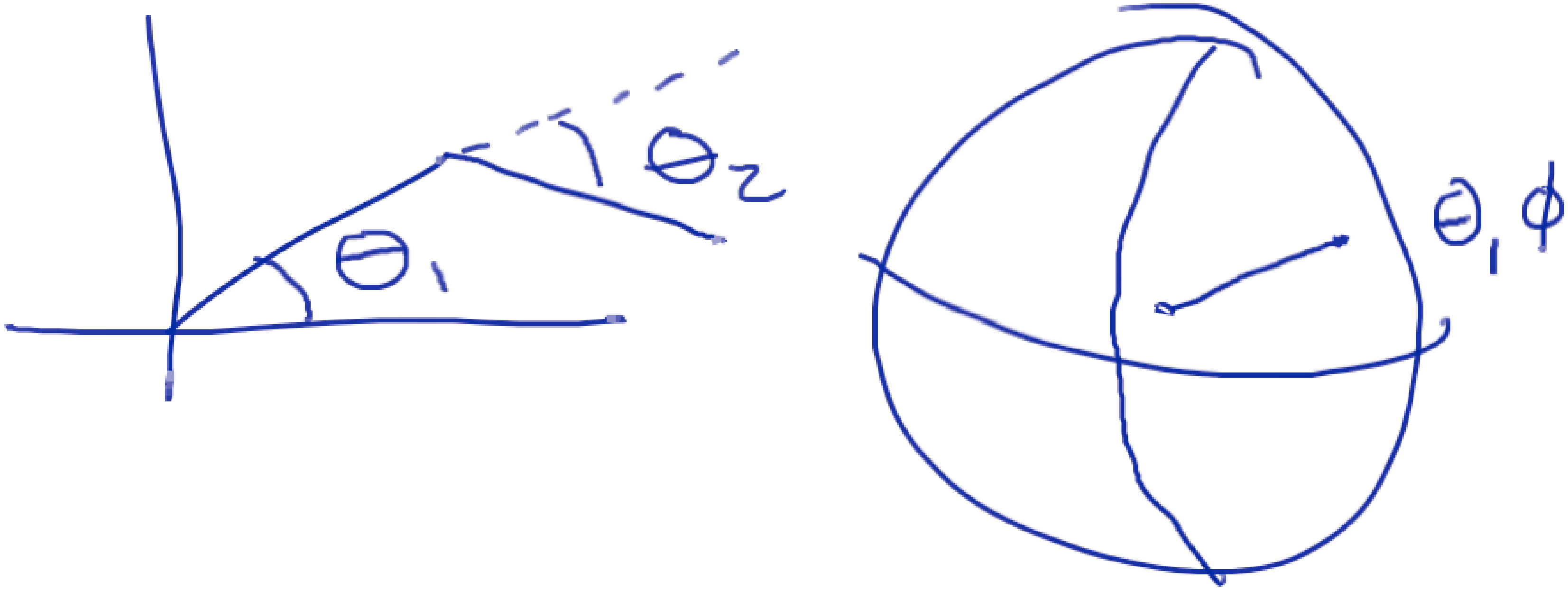


image: J Kuffner

Topology of C-space

- *Topology of C-space can be something other than the familiar Euclidean world*
- *E.g. set of angles = unit circle = SO(2)*
  - *not [0, 2π) !*
- *Ball & socket joint (3d angle) ⊆ unit sphere = SO(3)*

Topology example

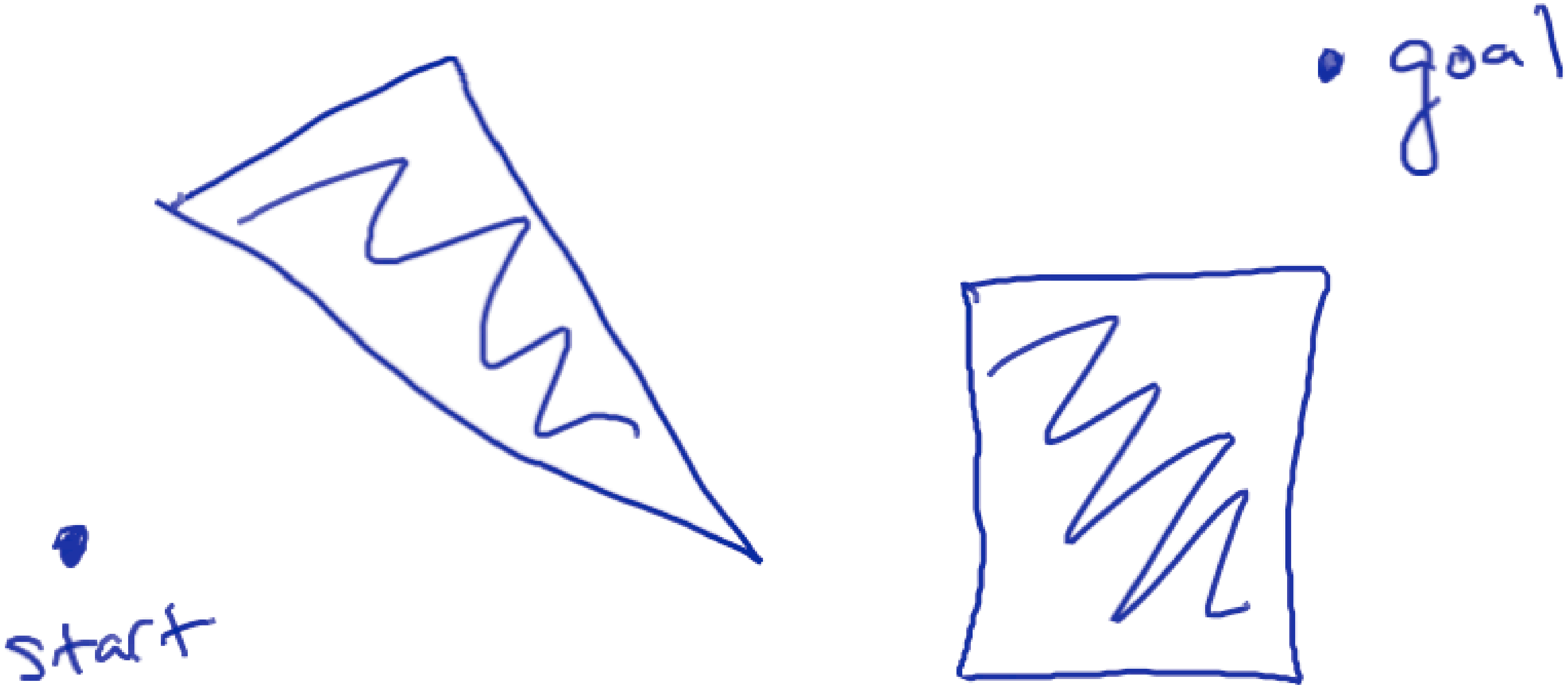


- *Compare L to R: 2 planar angles v. one solid angle — both 2 dof (and neither the same as Euclidean 2-space)*

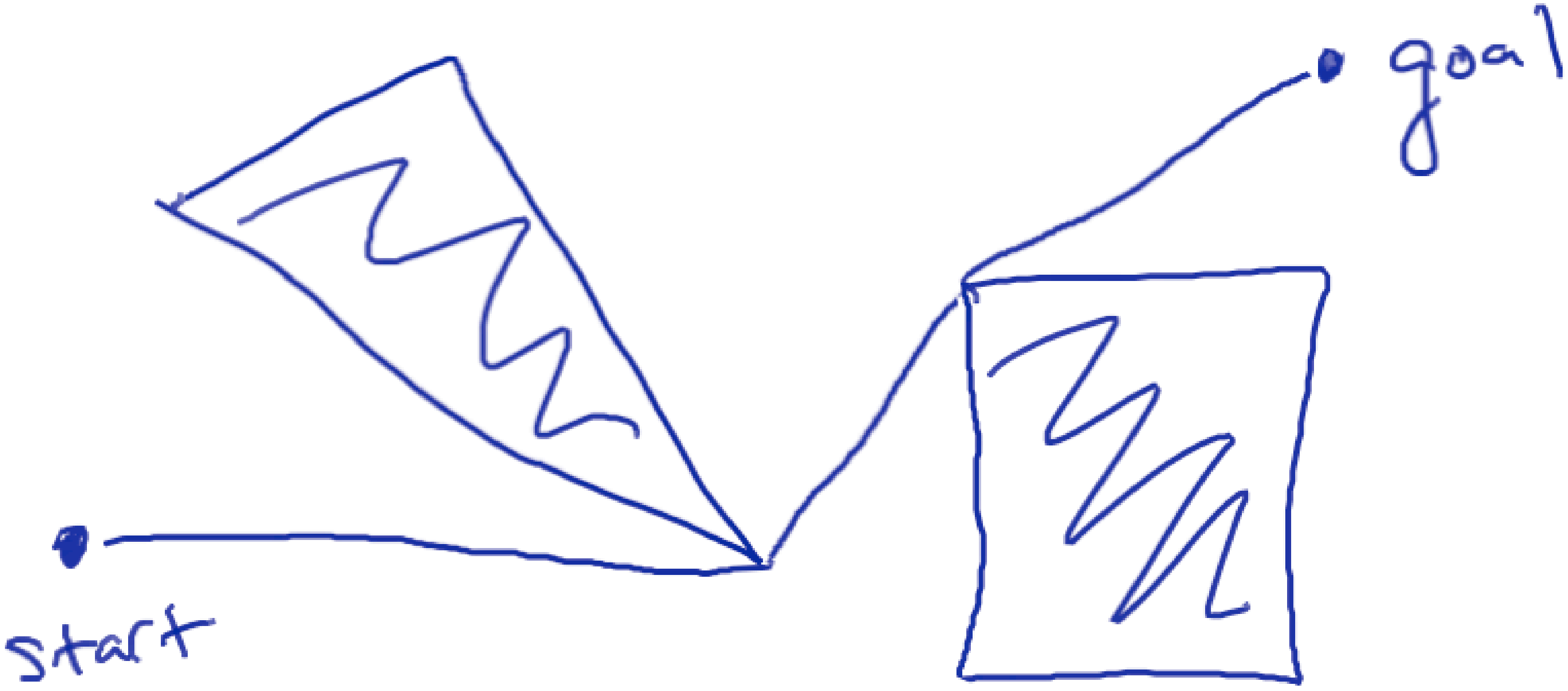
Back to planning

- *Complaint with A* was that it didn't break up space intelligently*
- *How might we do better?*
- *Lots of roboticists have given lots of answers!*

Shortest path in C-space



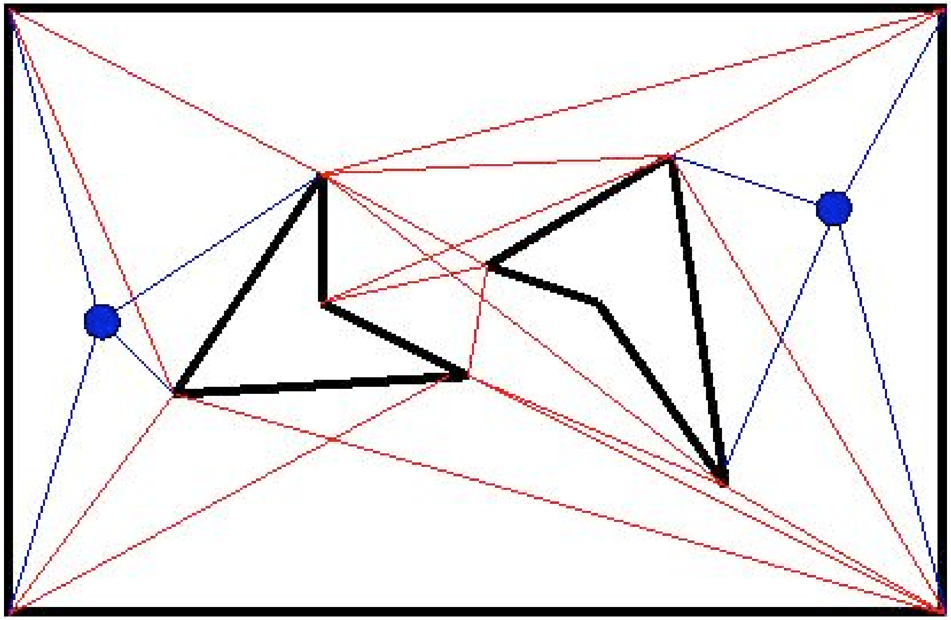
Shortest path in C-space



Shortest path

- *Suppose a planar polygonal C-space*
- *Shortest path in C-space is a sequence of line segments*
- *Each segment's ends are either start or goal or one of the vertices in C-space*
- *In 3-d or higher, might lie on edge, face, hyperface, …*

Visibility graph



http://www.cse.psu.edu/~rsharma/robotics/notes/notes2.html

Naive algorithm

```
For i = 1 … points
    For j = 1 … points
        included = t
        For k = 1 … edges
            if segment ij intersects edge k
                included = f
```

Complexity

- *Naive algorithm is* $O(n^3)$ *in planar C-space*
- *For algorithms that run faster,* $O(n^2)$ *and* $O(k + n \log n)$*, see [Latombe, pg 157]*
  - k *= number of edges that wind up in visibility graph*
- *Once we have graph, search it!*

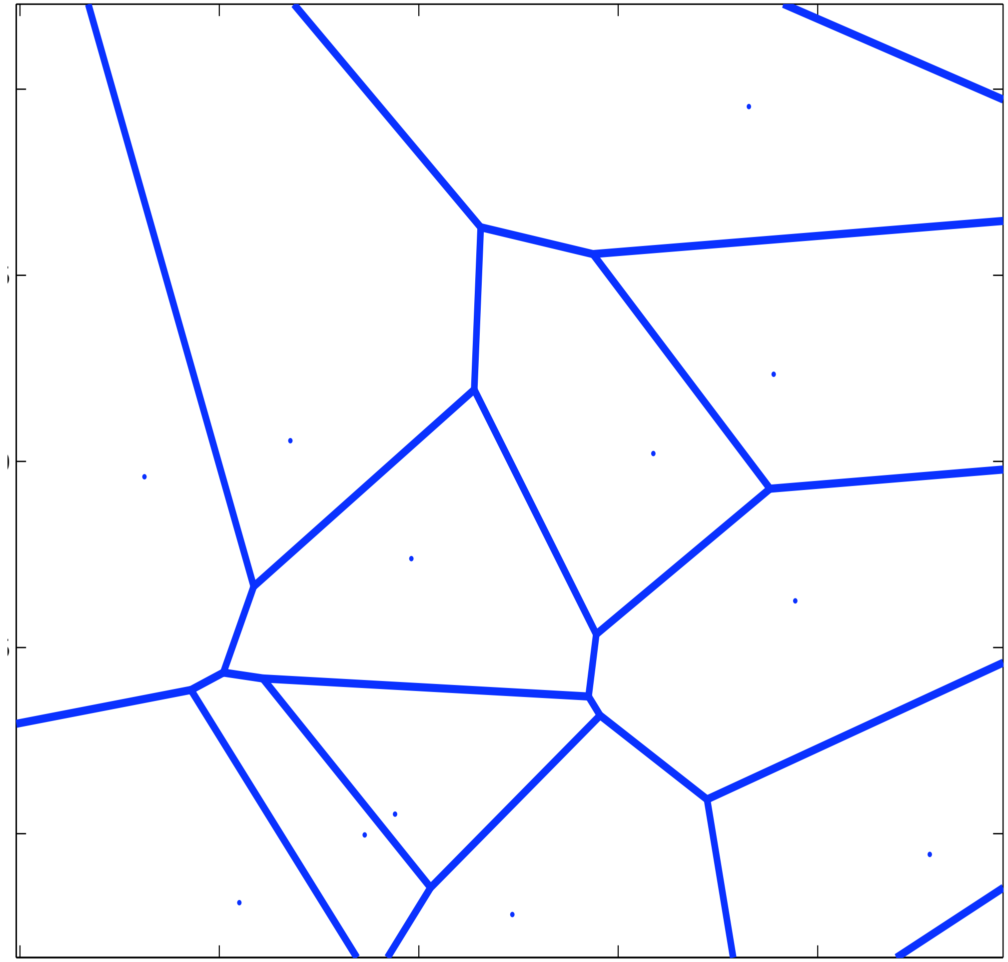
Discussion of visibility graph

- *Good: finds shortest path*
- *Bad: complex C-space yields long runtime, even if problem is easy*
    - *get my 23-dof manipulator to move 1mm when nearest obstacle is 1m*
- *Bad: no margin for error*

Getting bigger margins

- *Could just pad obstacles*
  - *but how much is enough? might make infeasible…*
- *What if we try to stay as far away from obstacles as possible?*

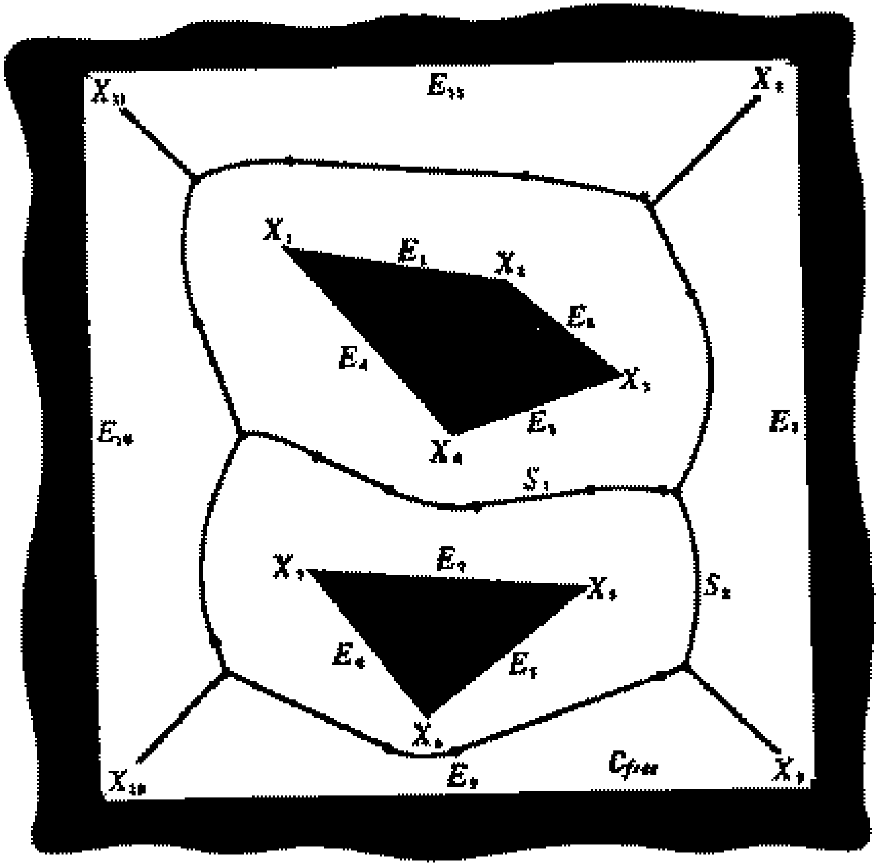
Voronoi graph



Voronoi graph

- *Given a set of point obstacles*
- *Find all places that are equidistant from two or more of them*
- *Result: network of line segments*
- *Called* Voronoi graph
- *Each line stays as far away as possible from two obstacles while still going between them*

Voronoi from polygonal C-space



Voronoi from polygonal C-space

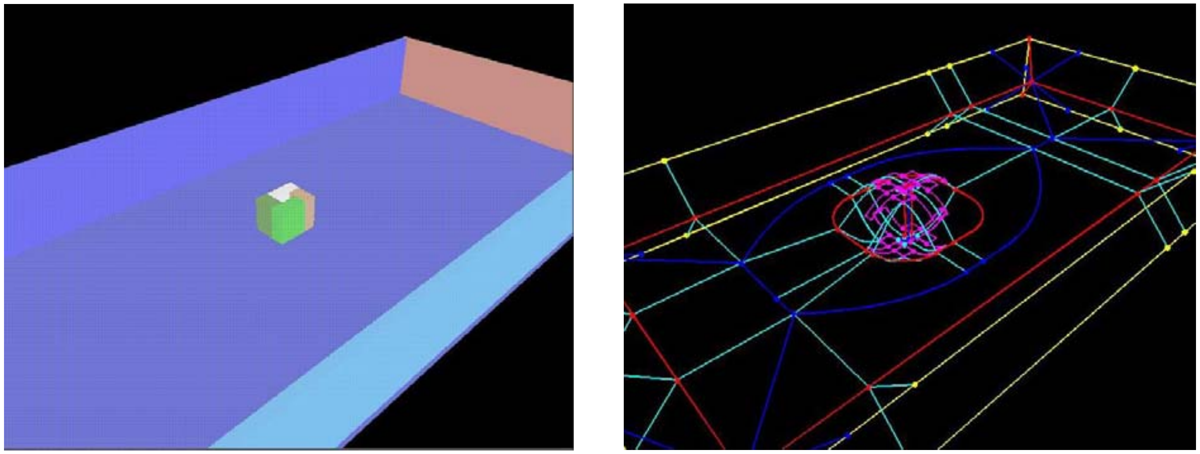
- *Set of points which are equidistant from 2 or more closest points on border of C-space*
- *Polygonal C-space in 2d yields lines & parabolas intersecting at points*
  - *lines from 2 points*
  - *parabolas from line & point*

Voronoi method for planning

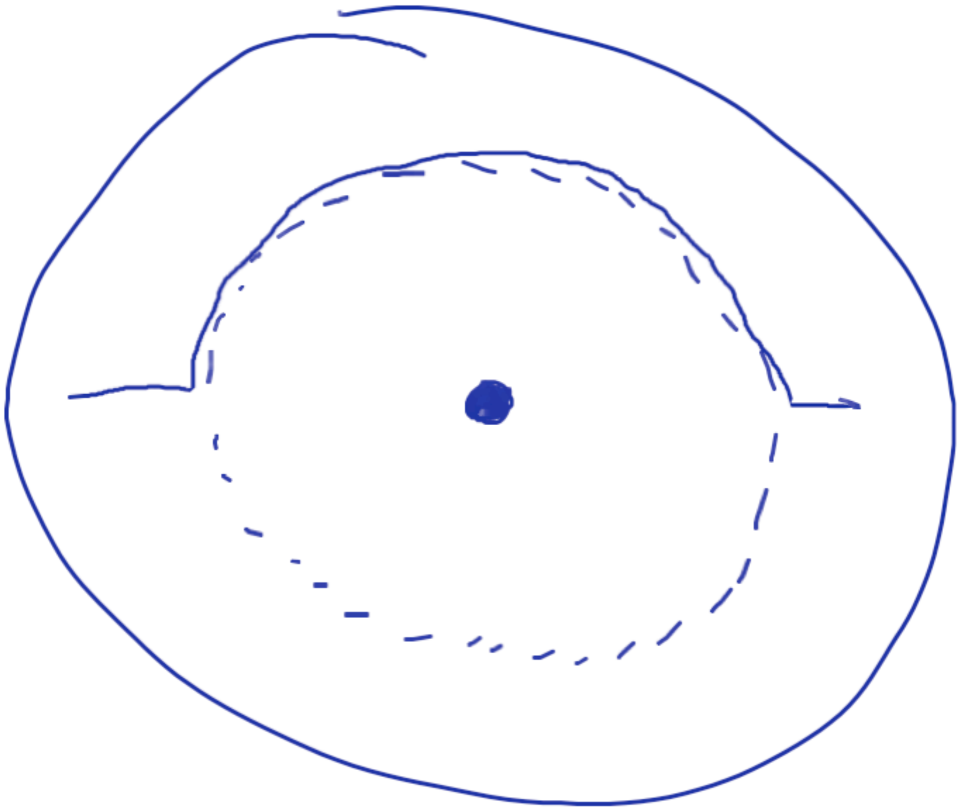
- *Compute Voronoi diagram of C-space*
- *Go straight from start to nearest point on diagram*
- *Plan within diagram to get near goal (e.g., with A*)*
- *Go straight to goal*

Discussion of Voronoi

- *Good: stays far away from obstacles*
- *Bad: assumes polygons*
- *Bad: gets kind of hard in higher dimensions (but see Howie Choset's web page and book)*

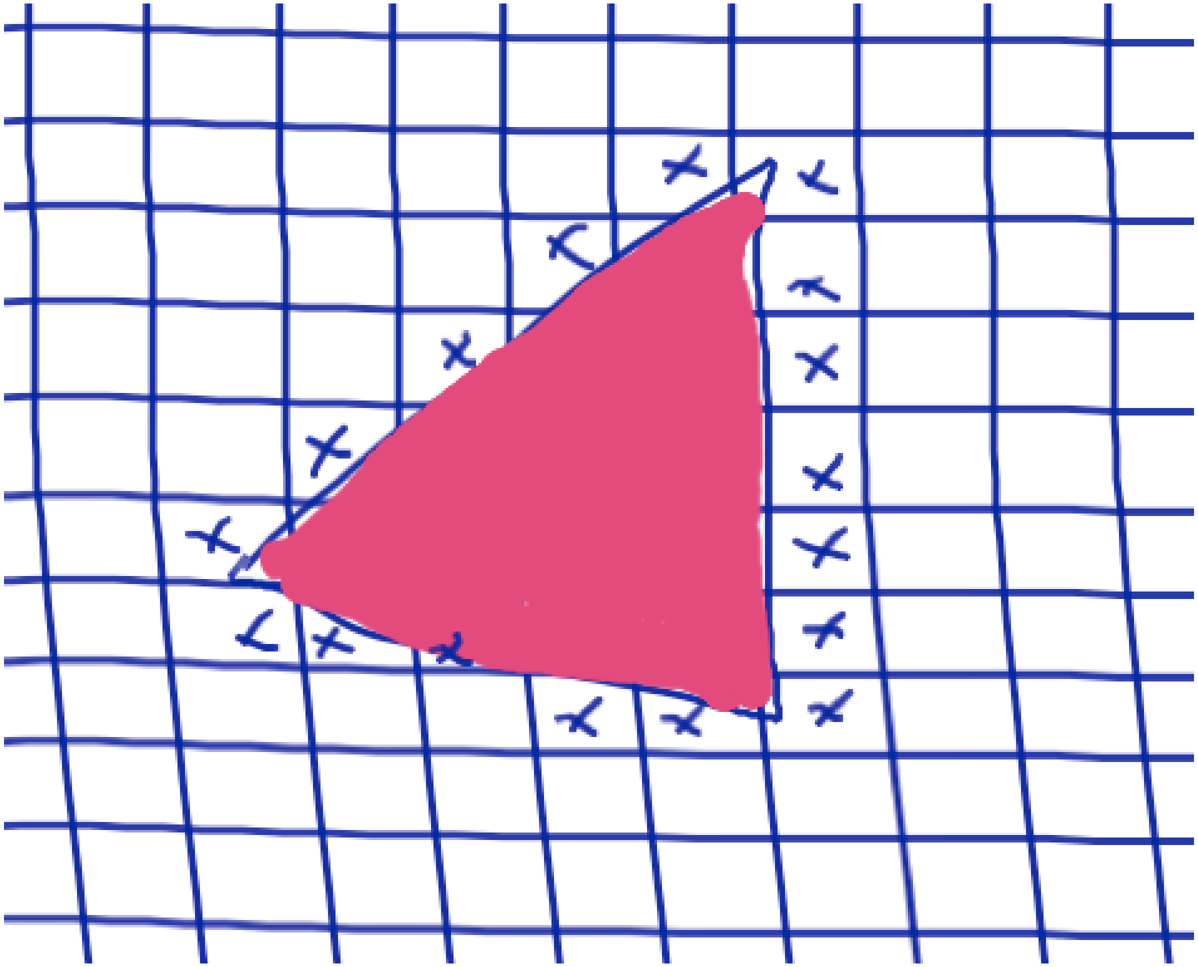


Voronoi discussion



- *Bad: kind of gun-shy about obstacles*

(Approximate) cell decompositions



Planning algorithm

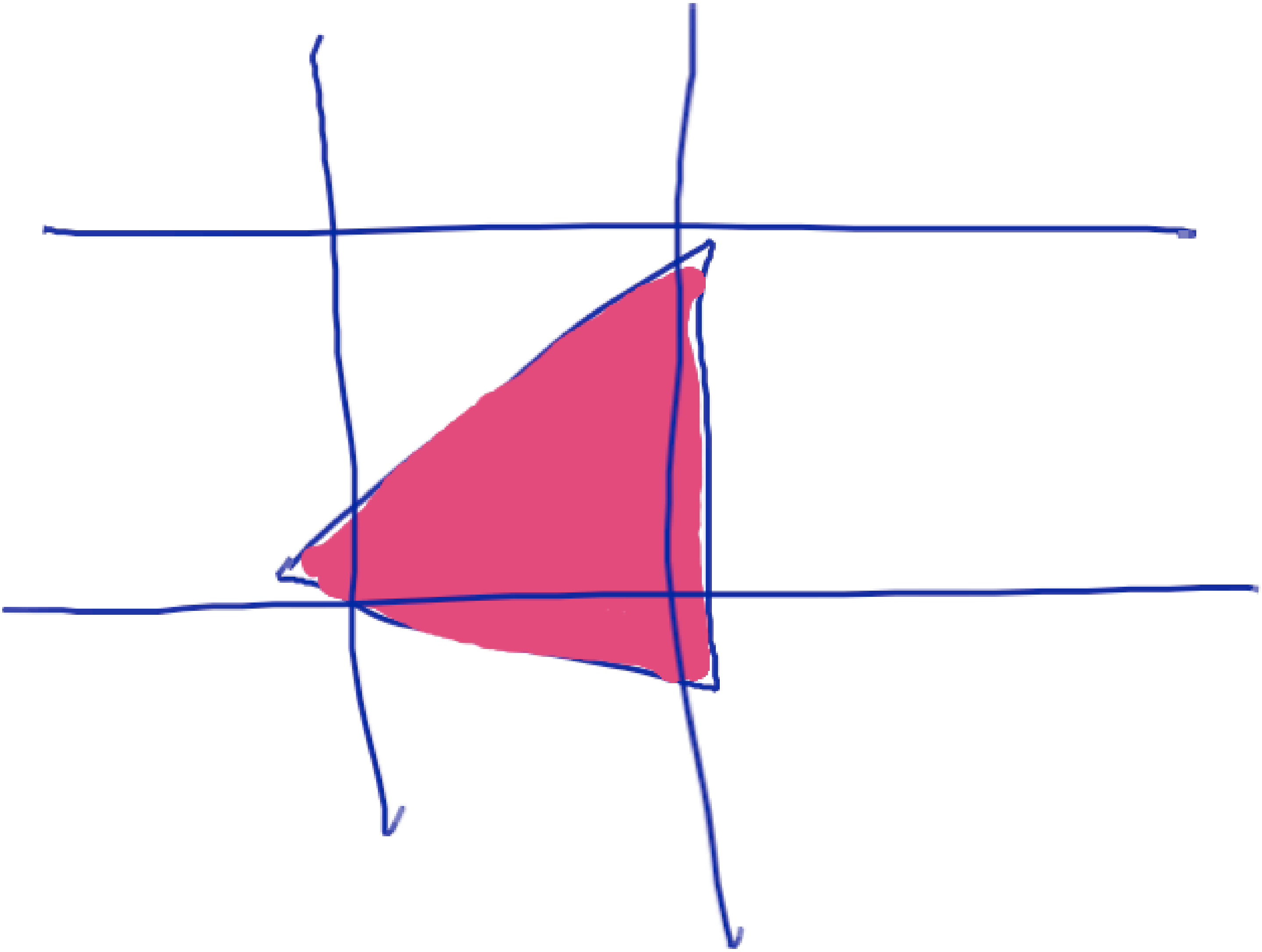
- *Lay down a grid in C-space*
- *Delete cells that intersect obstacles*
- *Connect neighbors*
- *A**
- *If no path, double resolution and try again*
    - *never know when we’re done*

Approximate cell decomposition

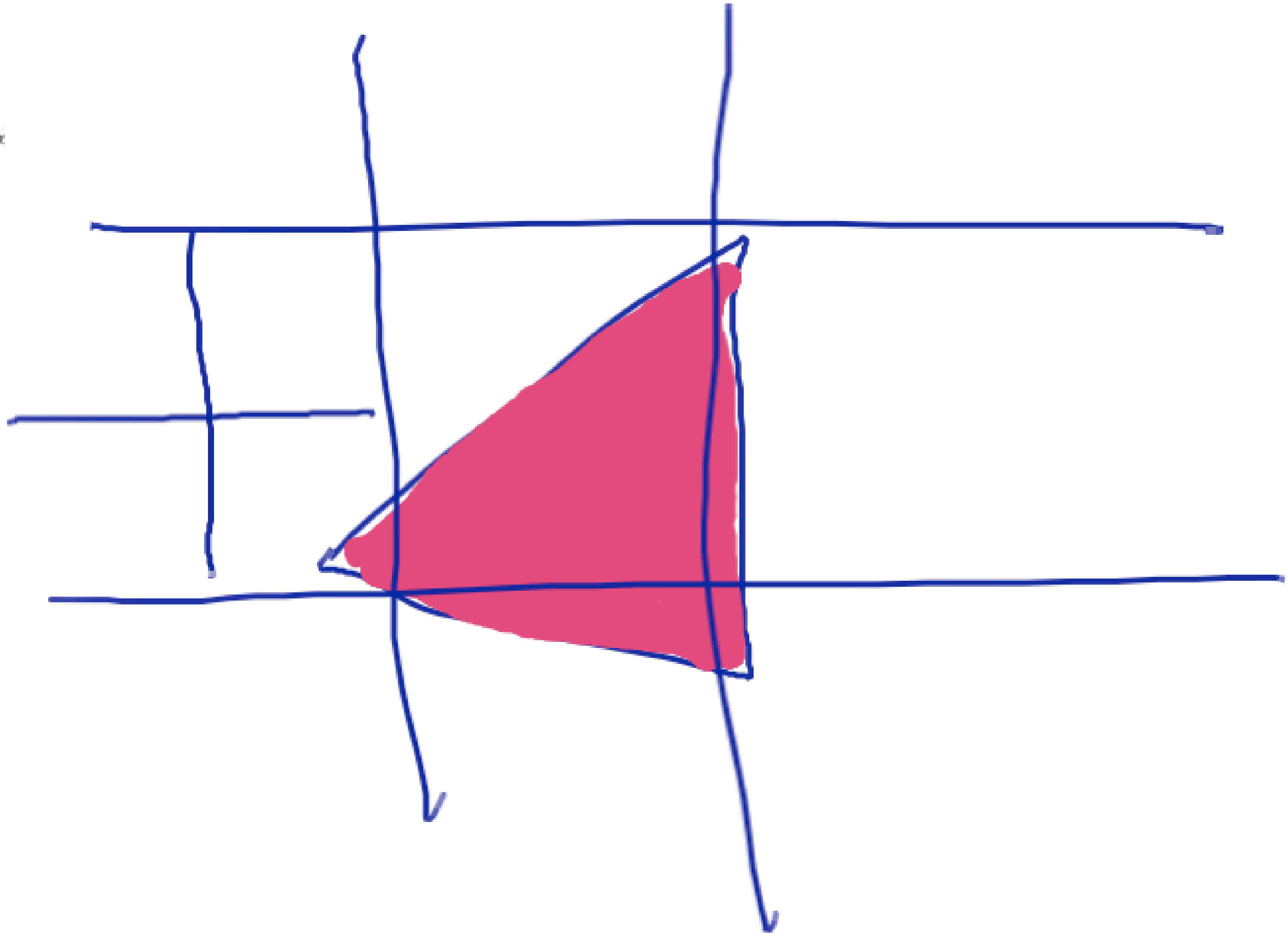
- *This decomposition is what we were using for A* in examples from above*
- *Works pretty well except:*
  - *need high resolution near obstacles*
  - *want low res away from obstacles*

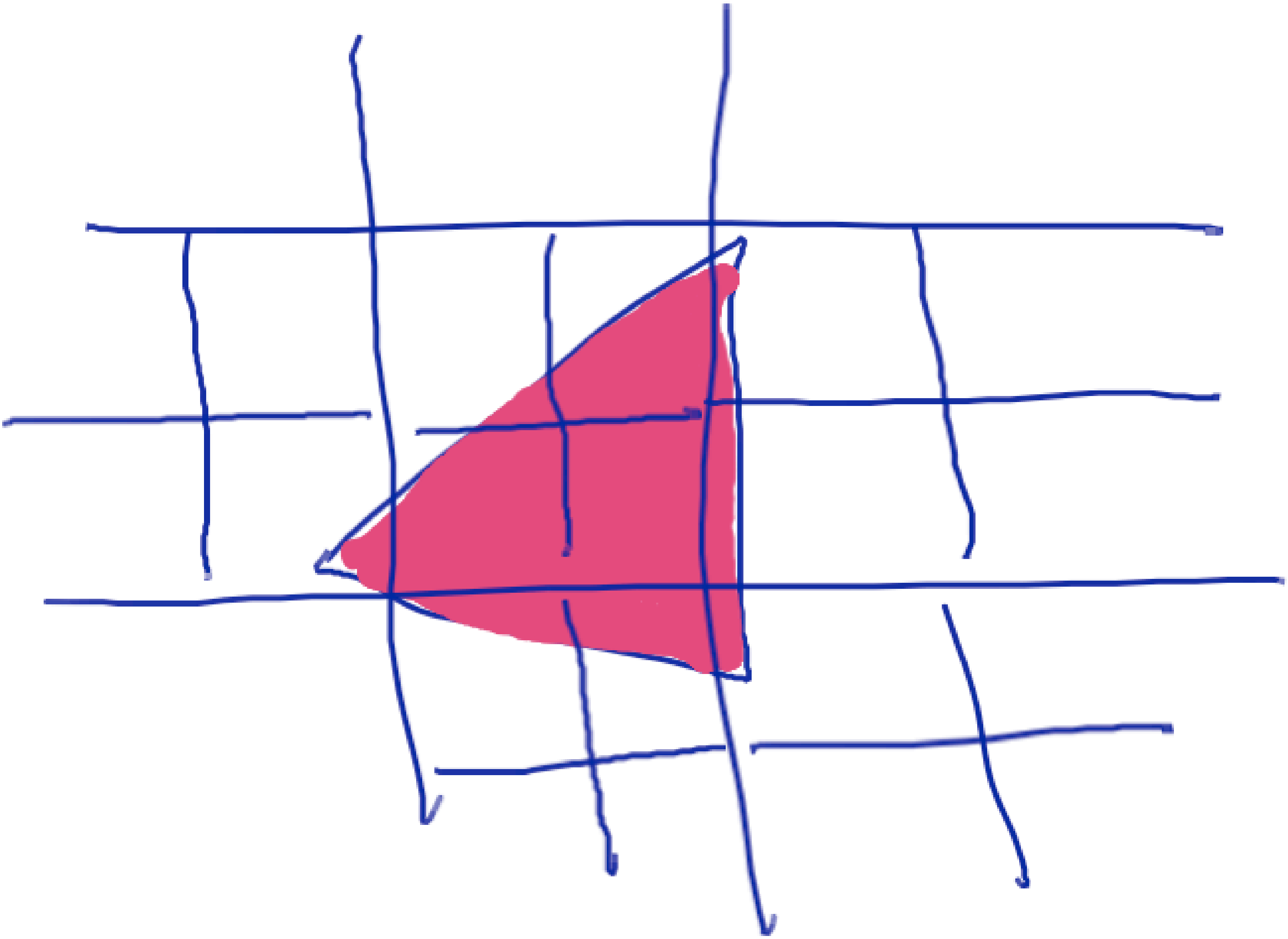
Fix: variable resolution

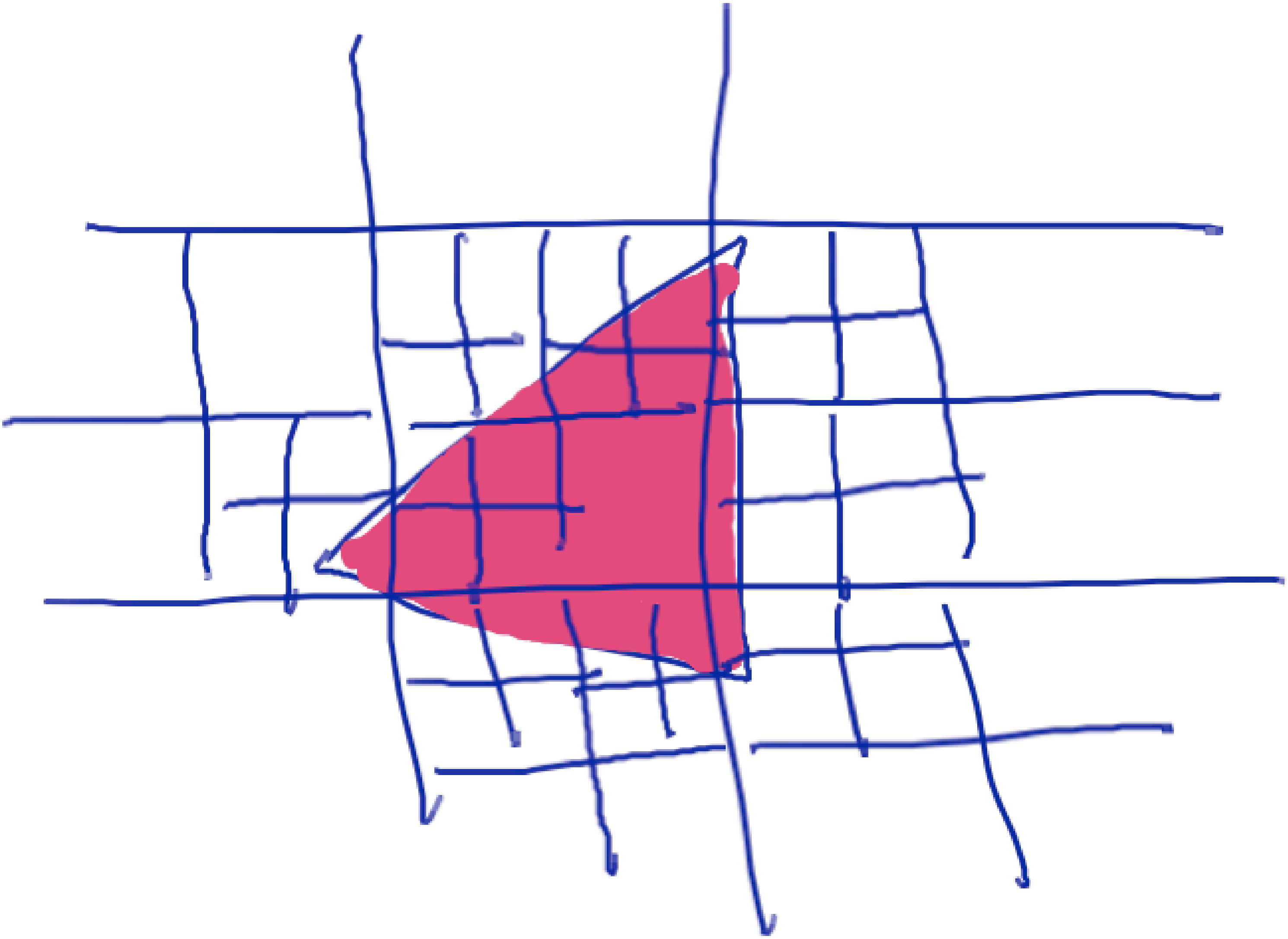
- *Lay down a coarse grid*
- *Split cells that intersect obstacle borders*
  - *empty cells good*
  - *full cells also don't need splitting*
- *Stop at fine resolution*
- *Data structure: quadtree*



[illegible] [illegible]







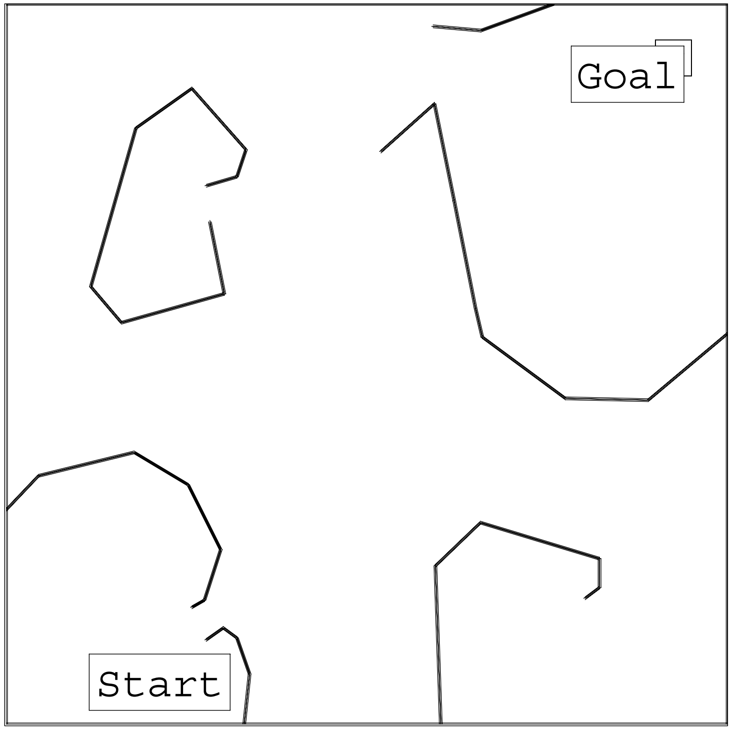
Discussion

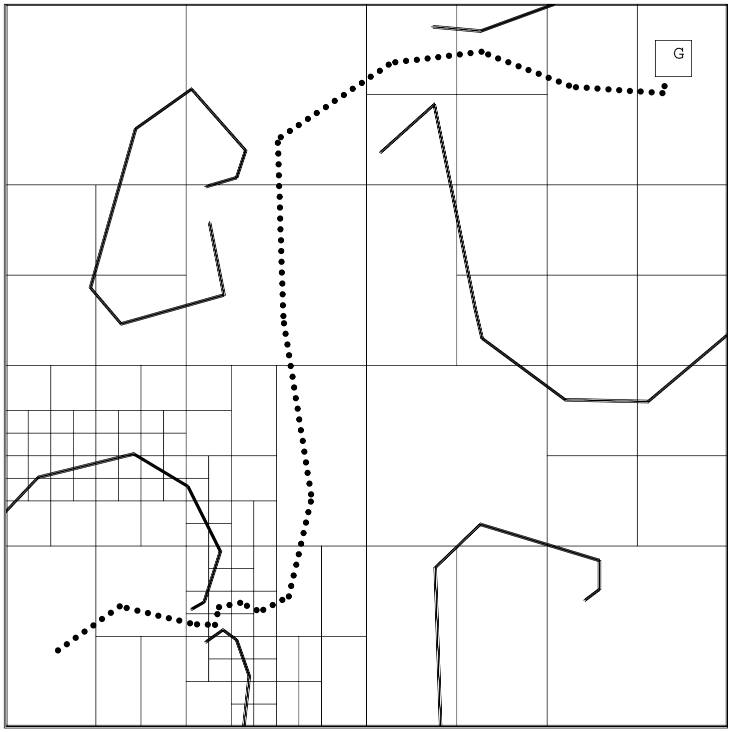
- *Works pretty well, except:*
    - *Still don't know when to stop*
    - *Won't find shortest path*
    - *Still doesn't really scale to high-d*

Better yet

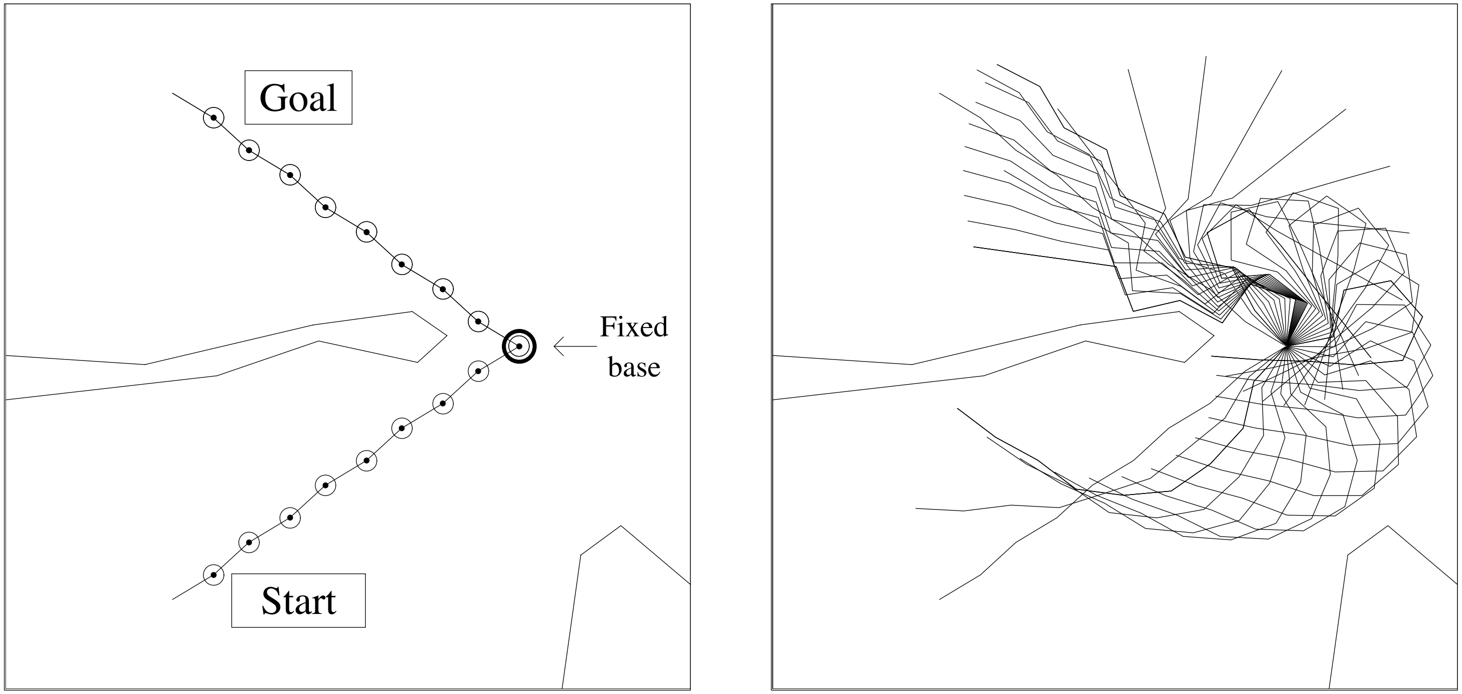
- *Adaptive decomposition*
- *Split only cells that actually make a difference*
  - *are on path from start*
  - *make a difference to our policy*

An adaptive splitter: parti-game





9dof planar arm



85 partitions total

Parti-game paper

- *Andrew Moore and Chris Atkeson*. The Parti-game Algorithm for Variable Resolution Reinforcement Learning in Multidimensional State-spaces
- *http://www.autonlab.org/autonweb/14699.html*

Randomness in search

Rapidly-exploring Random Trees

- *Put landmarks into C-space*
- *Break up C-space into Voronoi regions around landmarks*
- *Put landmarks densely only if high resolution is needed to find a path*
- *Will not guarantee optimal path (*)*

RRT assumptions

- *RANDOM_CONFIG*
    - *samples from C-space*
- *EXTEND(***q***,* **q***')*
    - *local controller, heads toward* **q***' from* **q**
    - *stops before hitting obstacle*
- *FIND_NEAREST(***q***, Q)*
    - *searches current tree Q for point near* **q**

Path Planning with RRTs

RRT = Rapidly-Exploring Random Tree

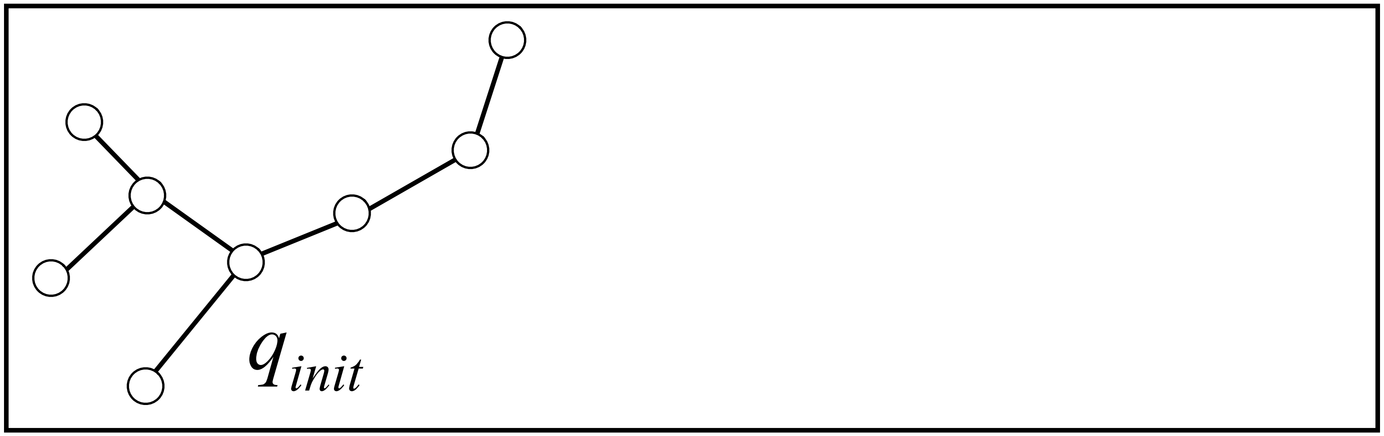
```
BUILT_RRT($q_{init}$) {
    T = $q_{init}$
    for k = 1 to K {
        $q_{rand}$ = RANDOM_CONFIG()
        EXTEND(T, $q_{rand}$);
    }
}
```

```
EXTEND(T, q) {
    $q_{near}$ = FIND_NEAREST(q, T)
    $q_{new}$ = EXTEND($q_{near}$, q)
    T = T + ($q_{near}$, $q_{new}$)
}
```

[Kuffner & LaValle , ICRA'00]

Path Planning with RRTs

RRT = Rapidly-Exploring Random Tree



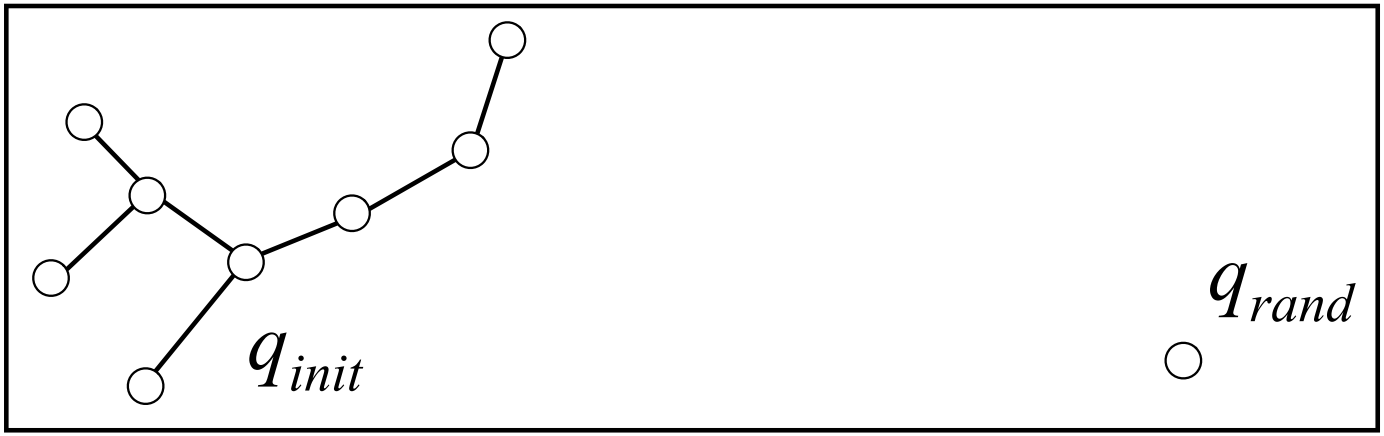
```
BUILT_RRT($q_{init}$) {
    T = $q_{init}$
    for k = 1 to K {
        $q_{rand}$ = RANDOM_CONFIG()
        EXTEND(T, $q_{rand}$);
    }
}
```

```
EXTEND(T, q) {
    $q_{near}$ = FIND_NEAREST(q, T)
    $q_{new}$ = EXTEND($q_{near}$, q)
    T = T + ($q_{near}$, $q_{new}$)
}
```

[Kuffner & LaValle , ICRA'00]

Path Planning with RRTs

RRT = Rapidly-Exploring Random Tree



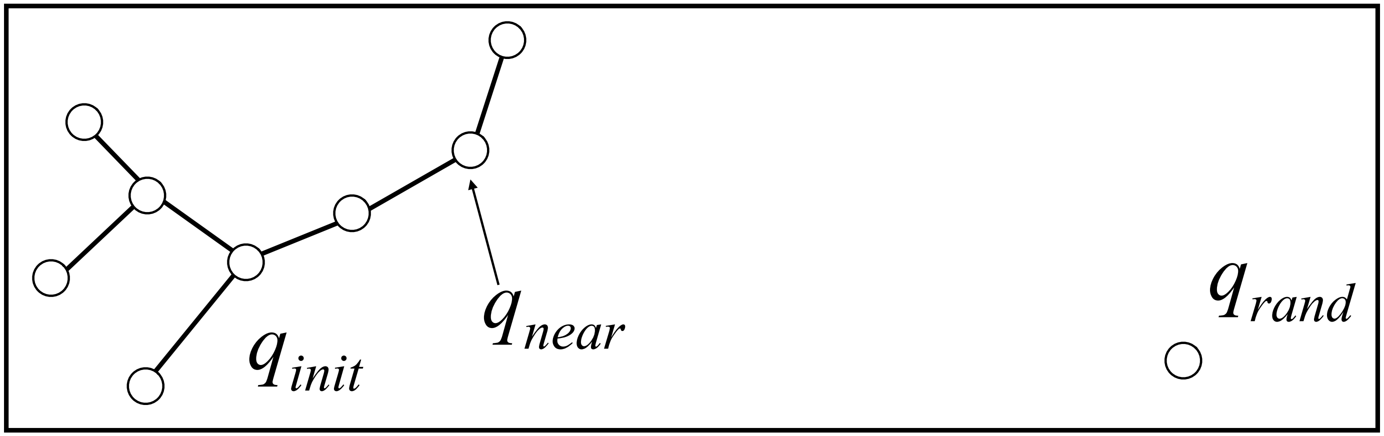
```
BUILT_RRT($q_{init}$) {
    T = $q_{init}$
    for k = 1 to K {
        $q_{rand}$ = RANDOM_CONFIG()
        EXTEND(T, $q_{rand}$);
    }
}
```

```
EXTEND(T, q) {
    $q_{near}$ = FIND_NEAREST(q, T)
    $q_{new}$ = EXTEND($q_{near}$, q)
    T = T + ($q_{near}$, $q_{new}$)
}
```

[Kuffner & LaValle , ICRA'00]

Path Planning with RRTs

RRT = Rapidly-Exploring Random Tree



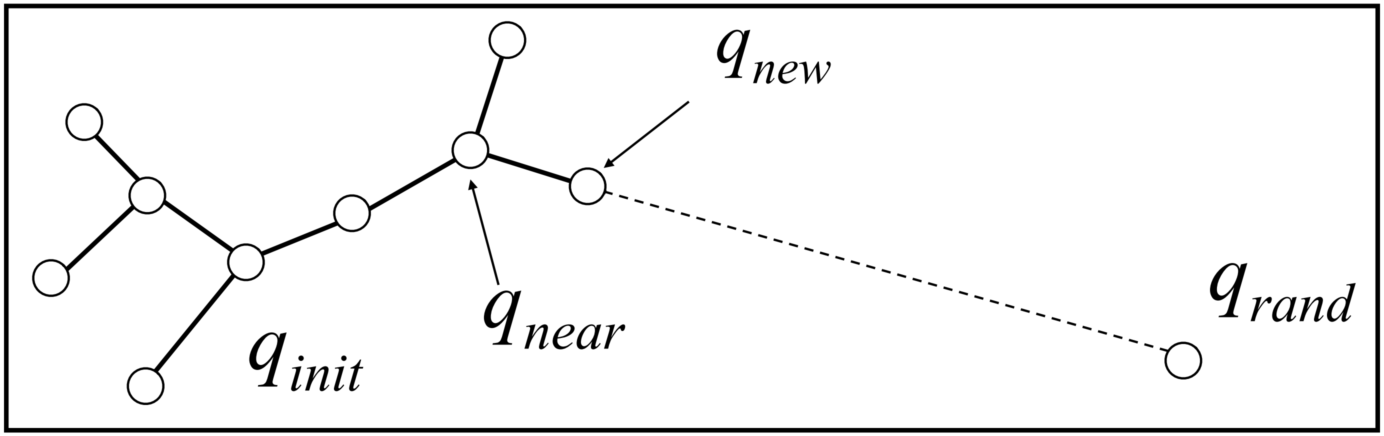
```
BUILT_RRT($q_{init}$) {
    T = $q_{init}$
    for k = 1 to K {
        $q_{rand}$ = RANDOM_CONFIG()
        EXTEND(T, $q_{rand}$);
    }
}
```

```
EXTEND(T, q) {
    $q_{near}$ = FIND_NEAREST(q, T)
    $q_{new}$ = EXTEND($q_{near}$, q)
    T = T + ($q_{near}$, $q_{new}$)
}
```

[Kuffner & LaValle , ICRA'00]

Path Planning with RRTs

RRT = Rapidly-Exploring Random Tree

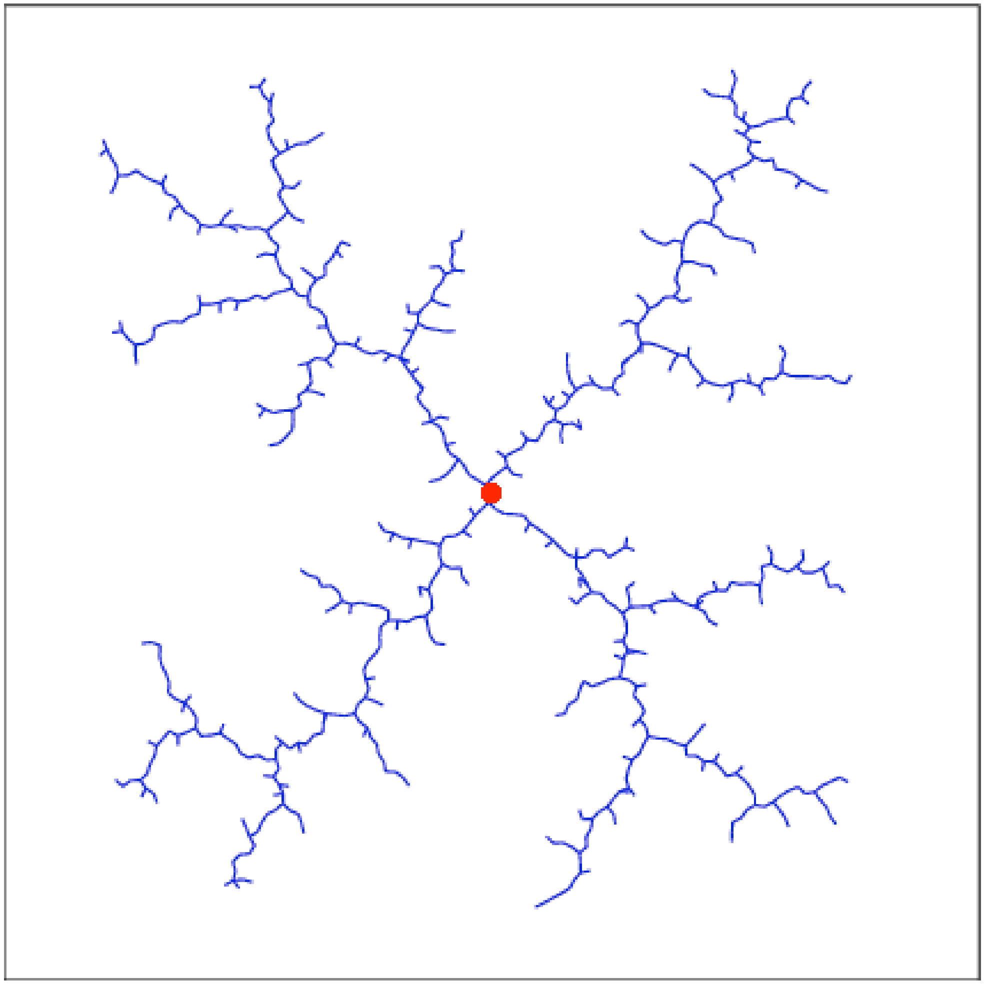


```
BUILT_RRT($q_{init}$) {
    T = $q_{init}$
    for k = 1 to K {
        $q_{rand}$ = RANDOM_CONFIG()
        EXTEND(T, $q_{rand}$);
    }
}
```

```
EXTEND(T, q) {
    $q_{near}$ = FIND_NEAREST(q, T)
    $q_{new}$ = EXTEND($q_{near}$, q)
    T = T + ($q_{near}$, $q_{new}$)
}
```

[Kuffner & LaValle , ICRA’00]

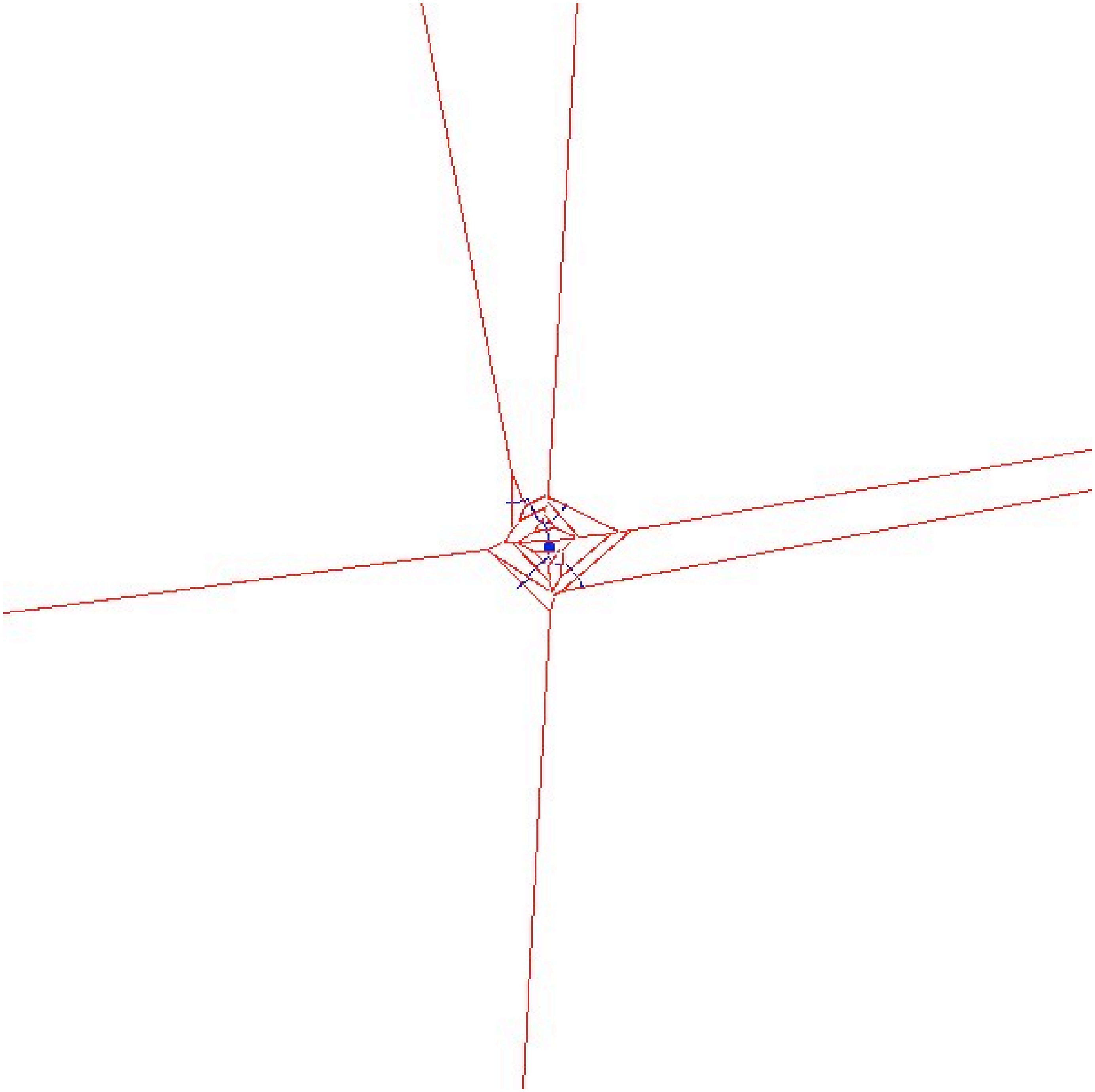
RRT example

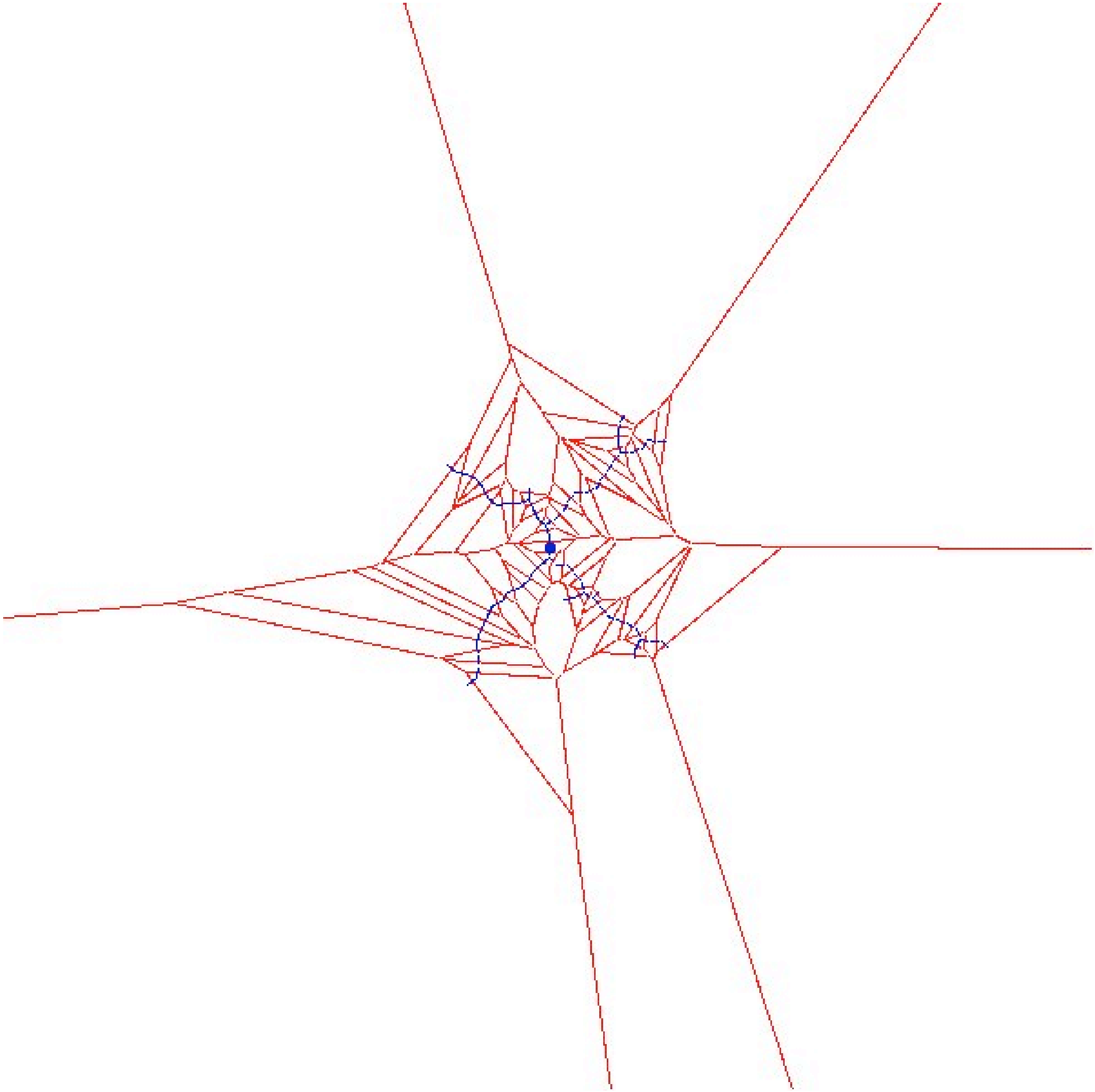


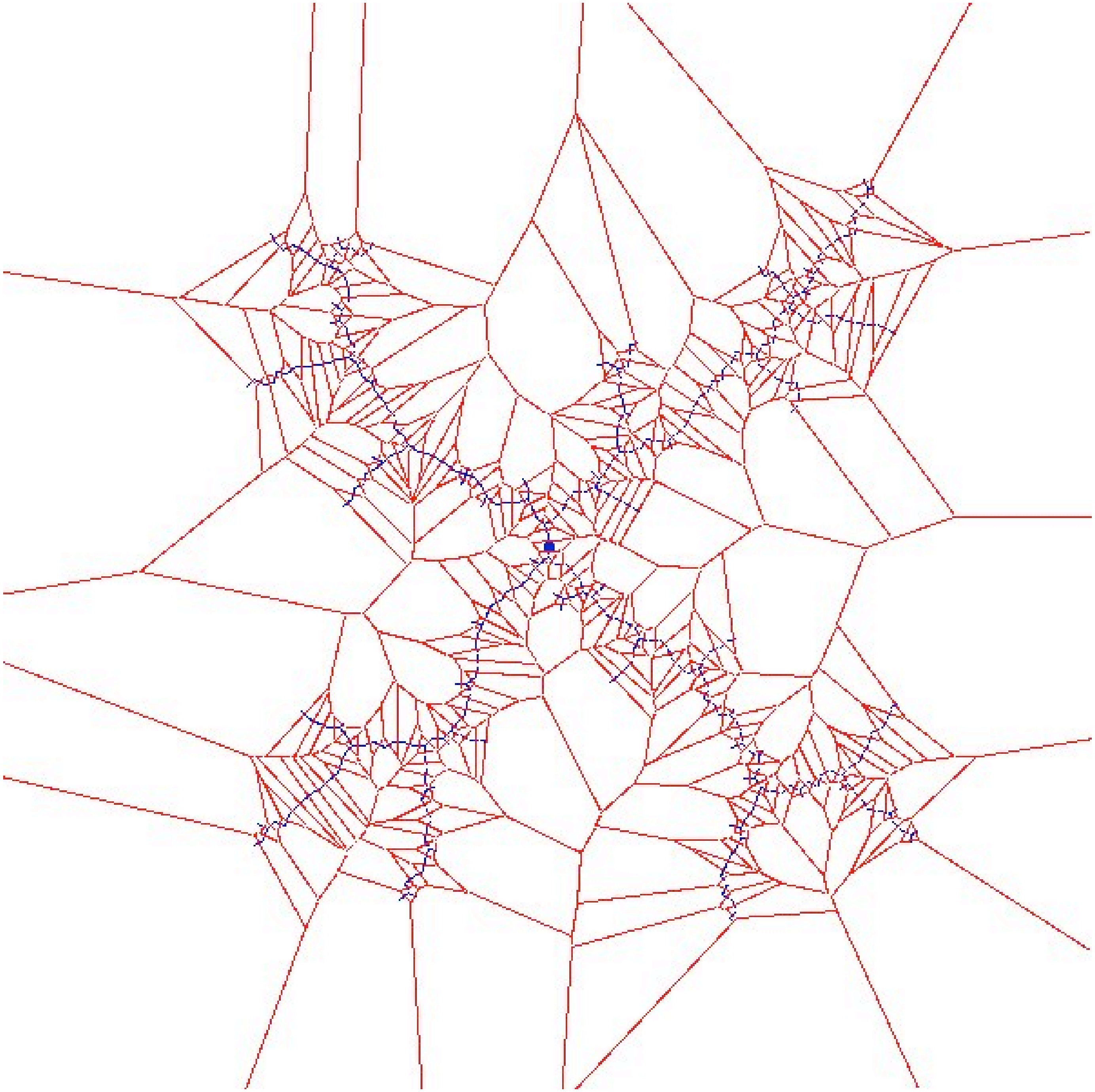
Planar holonomic robot

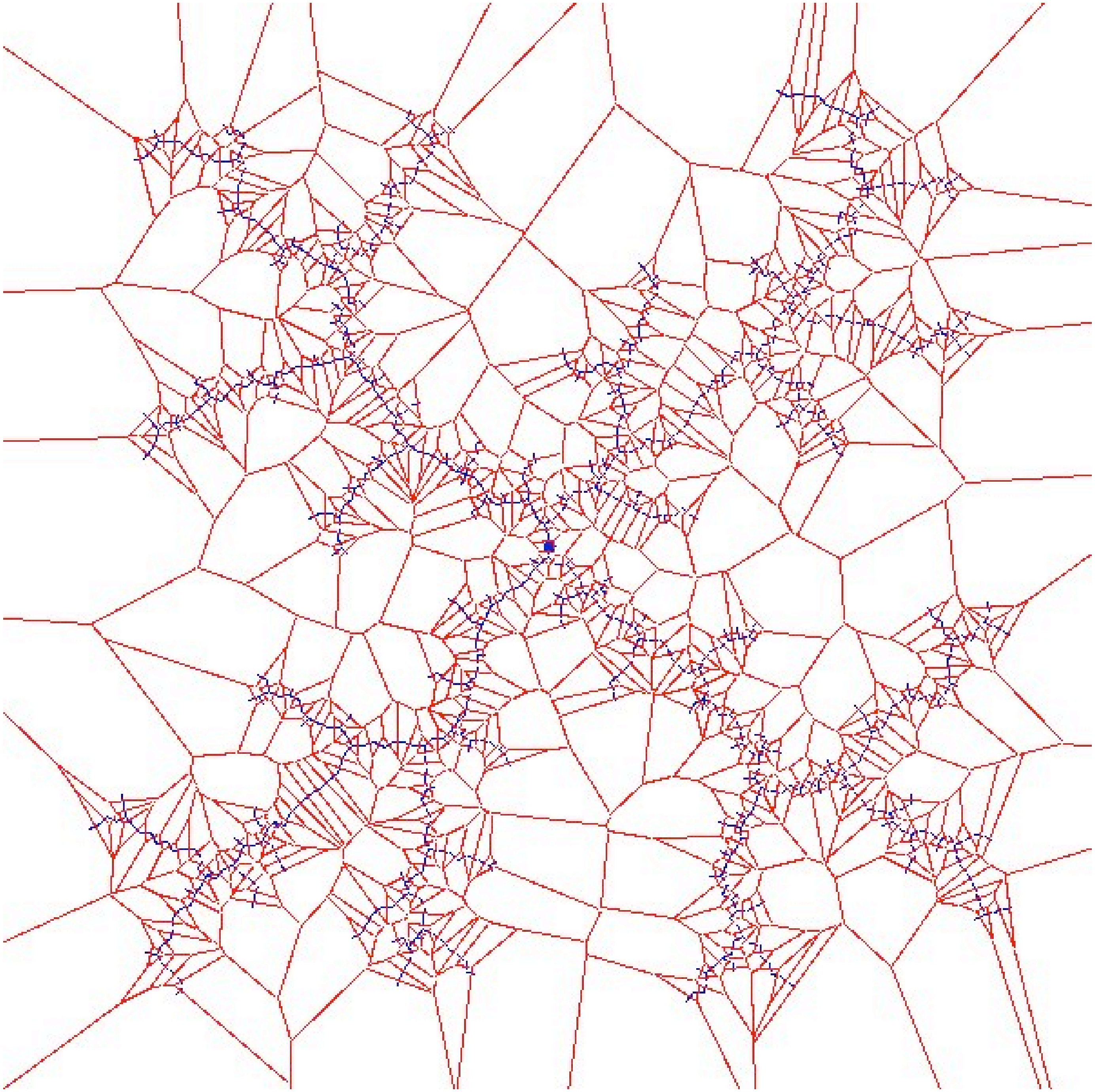
RRTs explore coarse to fine

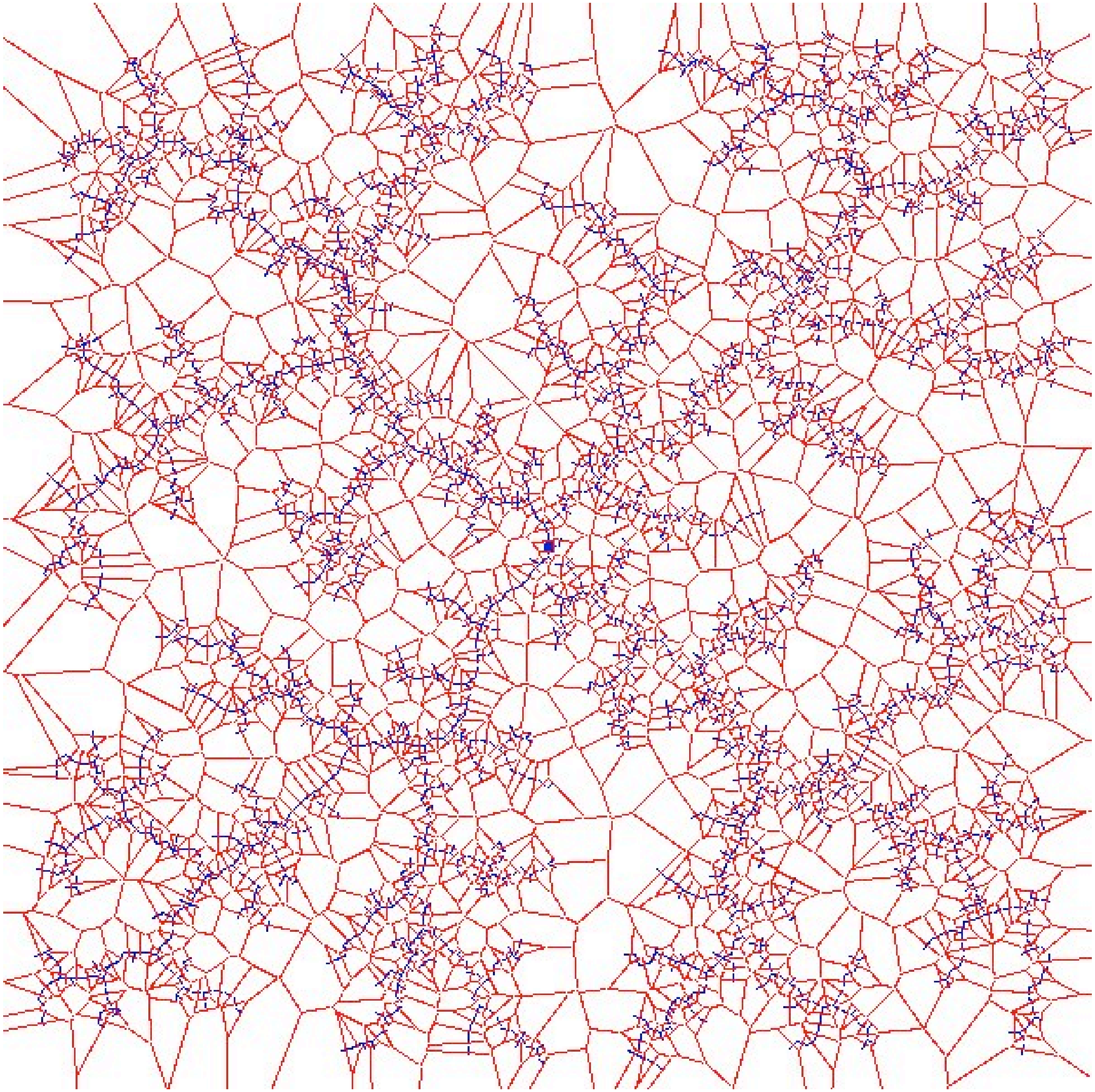
- *Tend to break up large Voronoi regions*
    - *higher probability of* q_{rand} *being in them*
- *Limiting distribution of vertices given by RANDOM_CONFIG*
    - *as RRT grows, probability that* q_{rand} *is reachable with local controller (and so immediately becomes a new vertex) approaches 1*



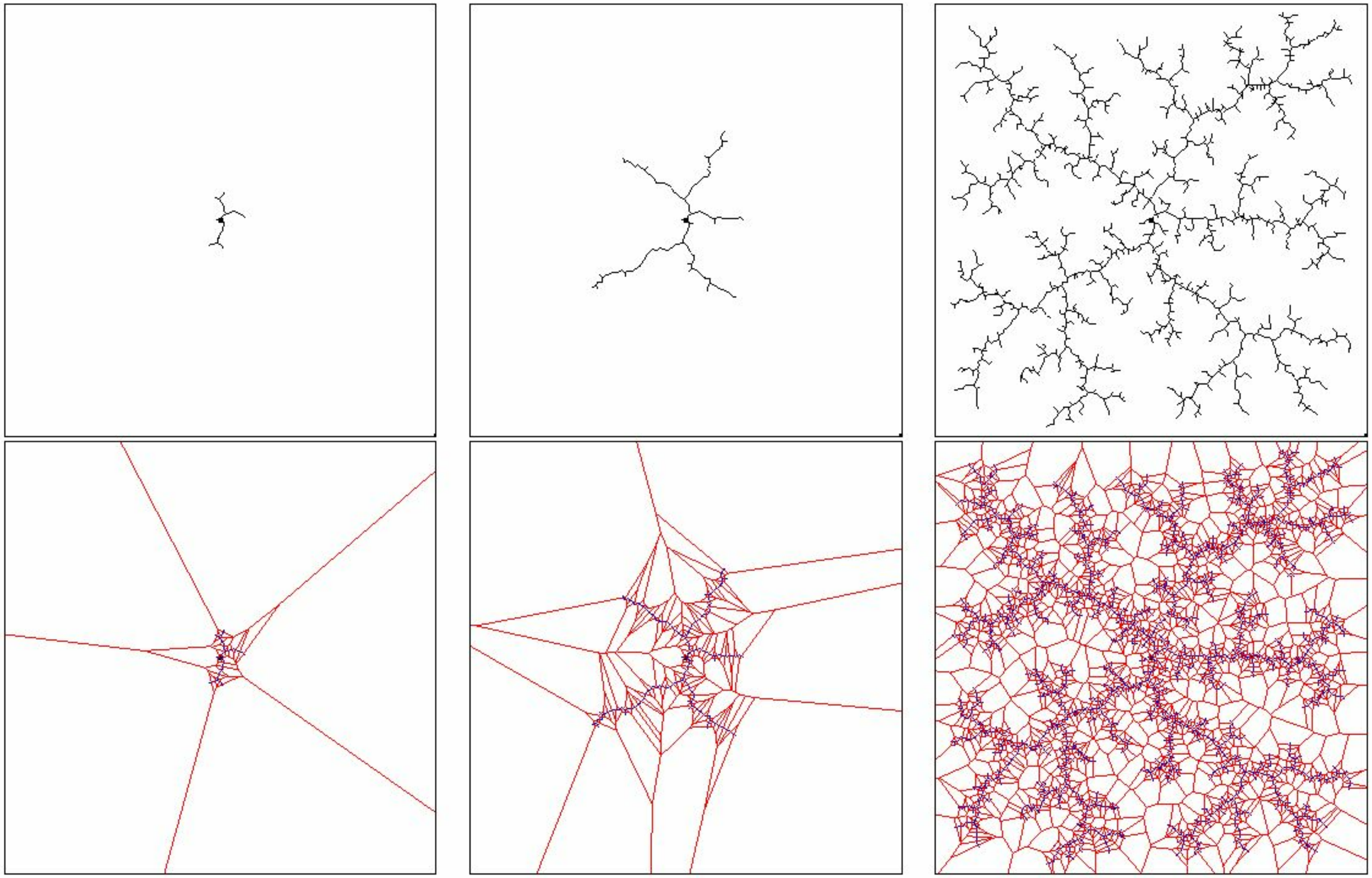




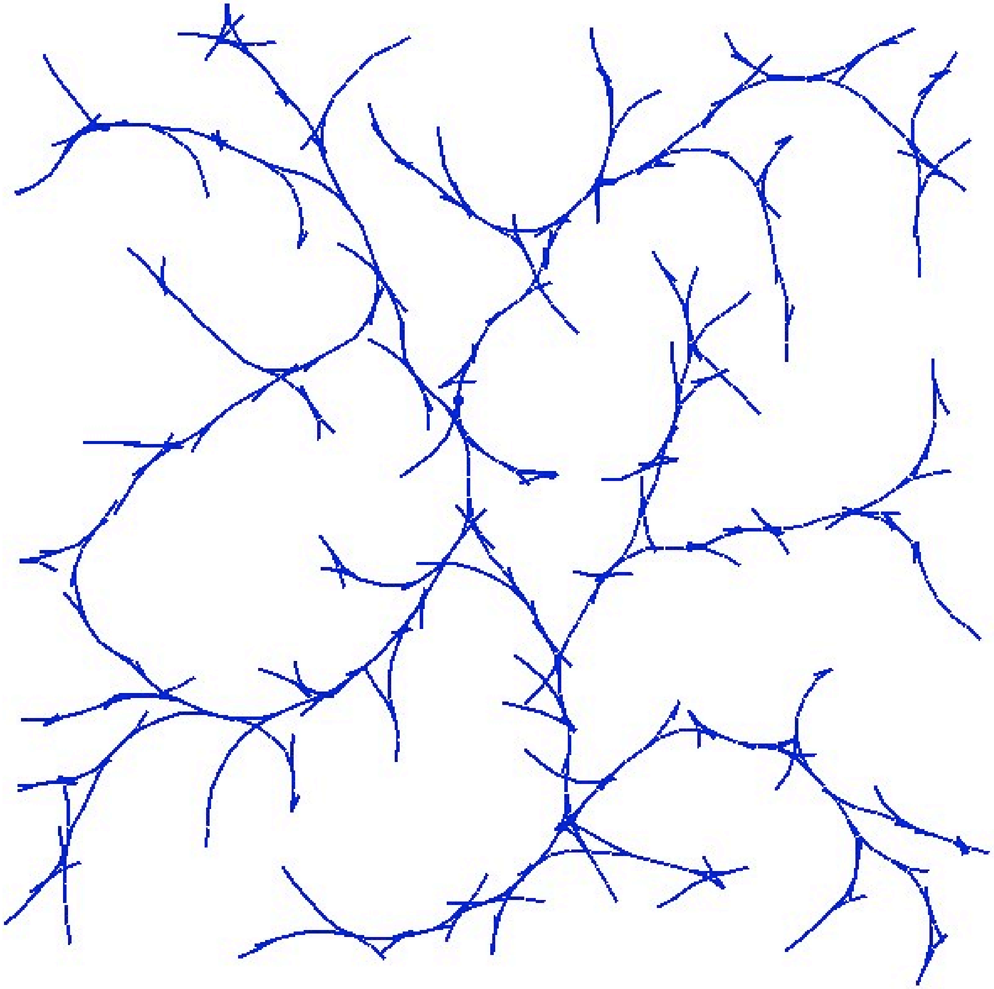




RRT example

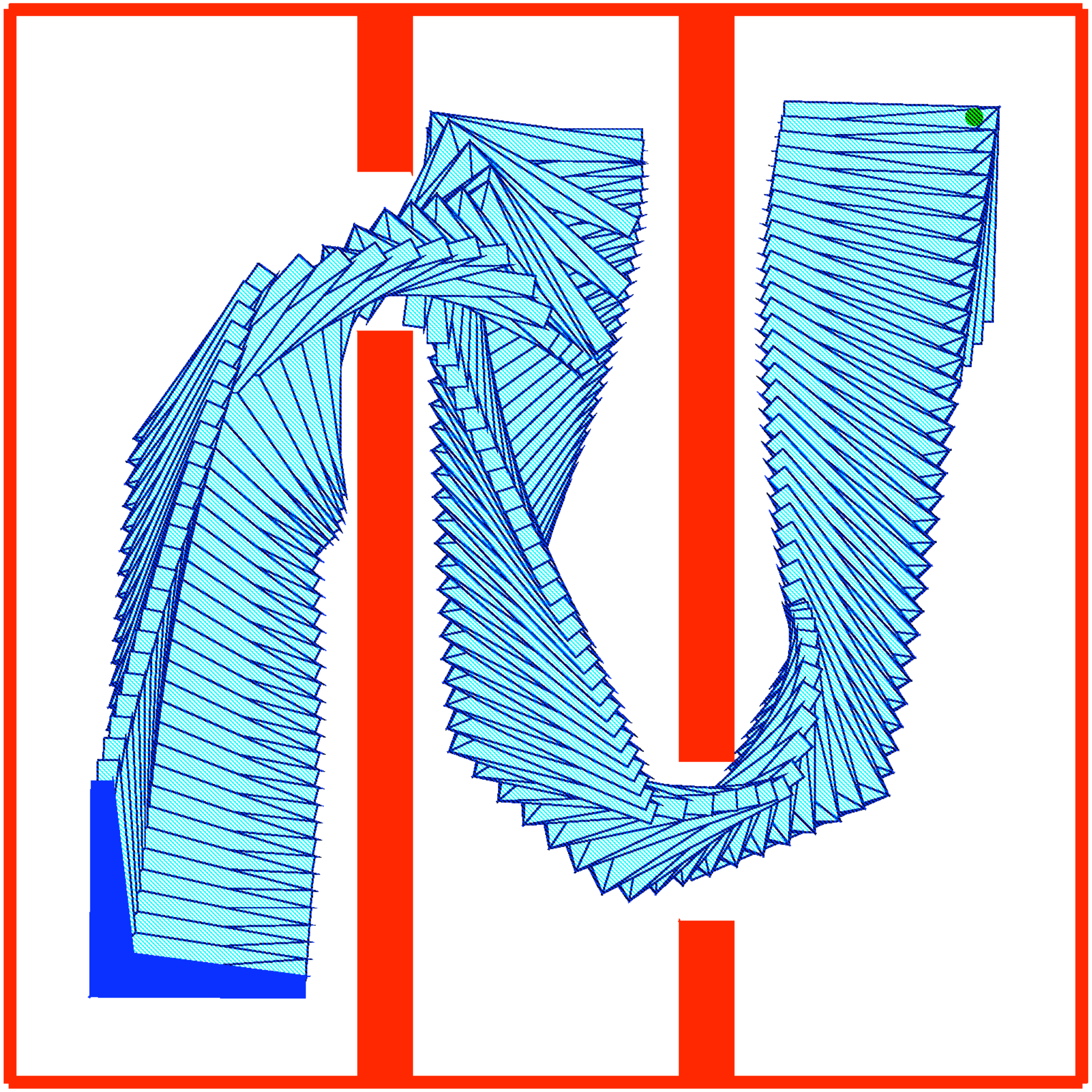


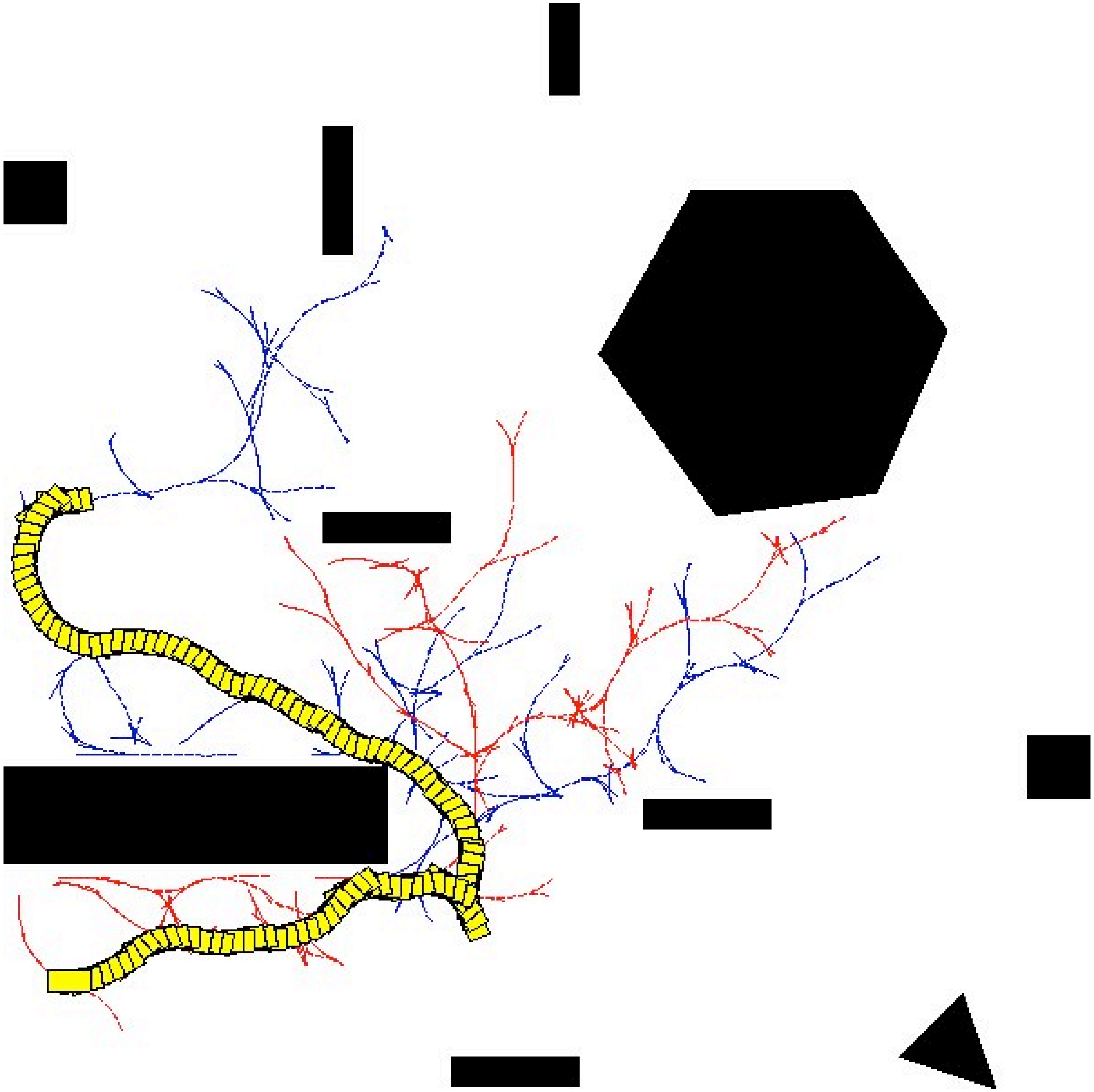
RRT for a car (3 dof)

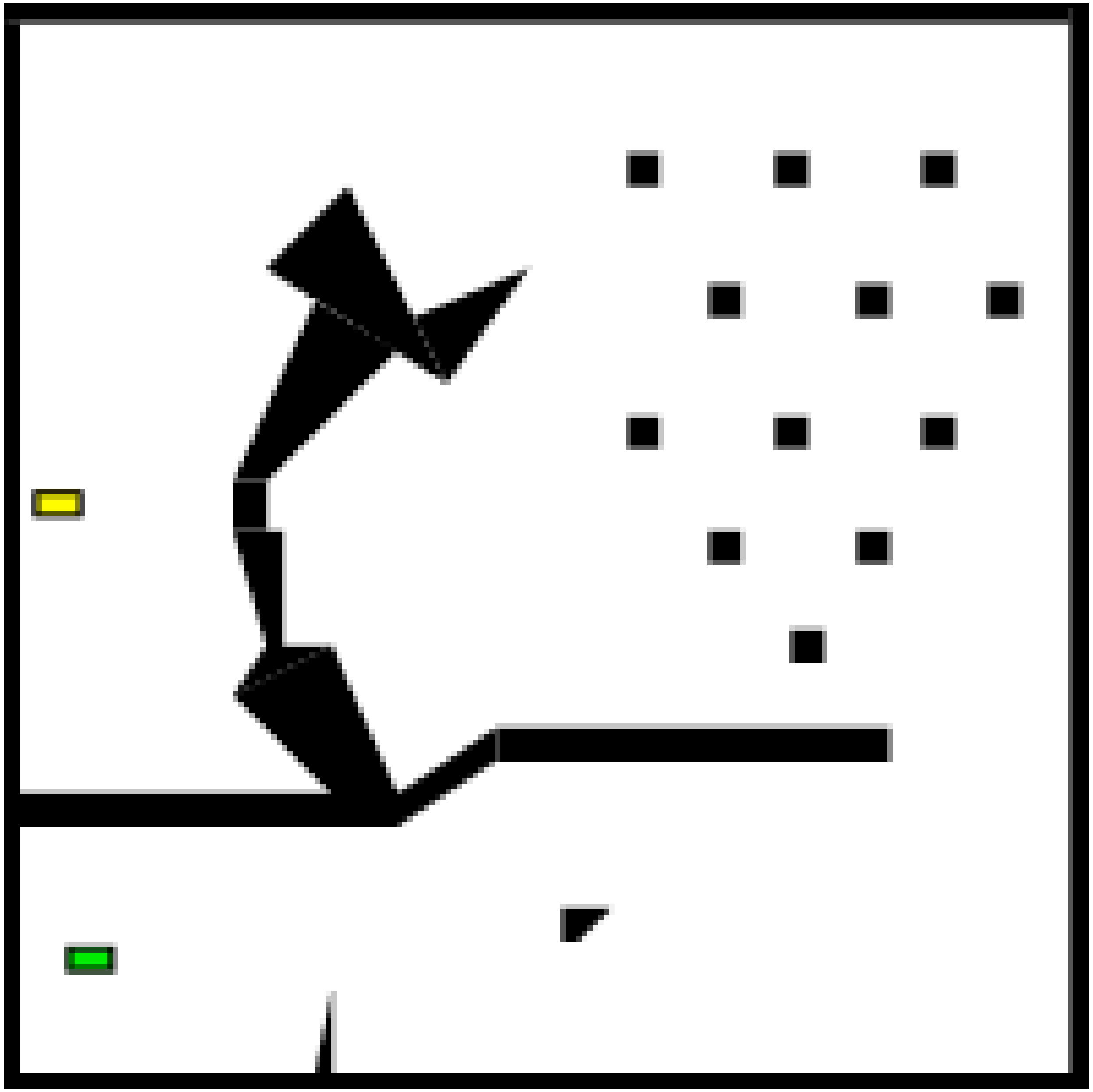


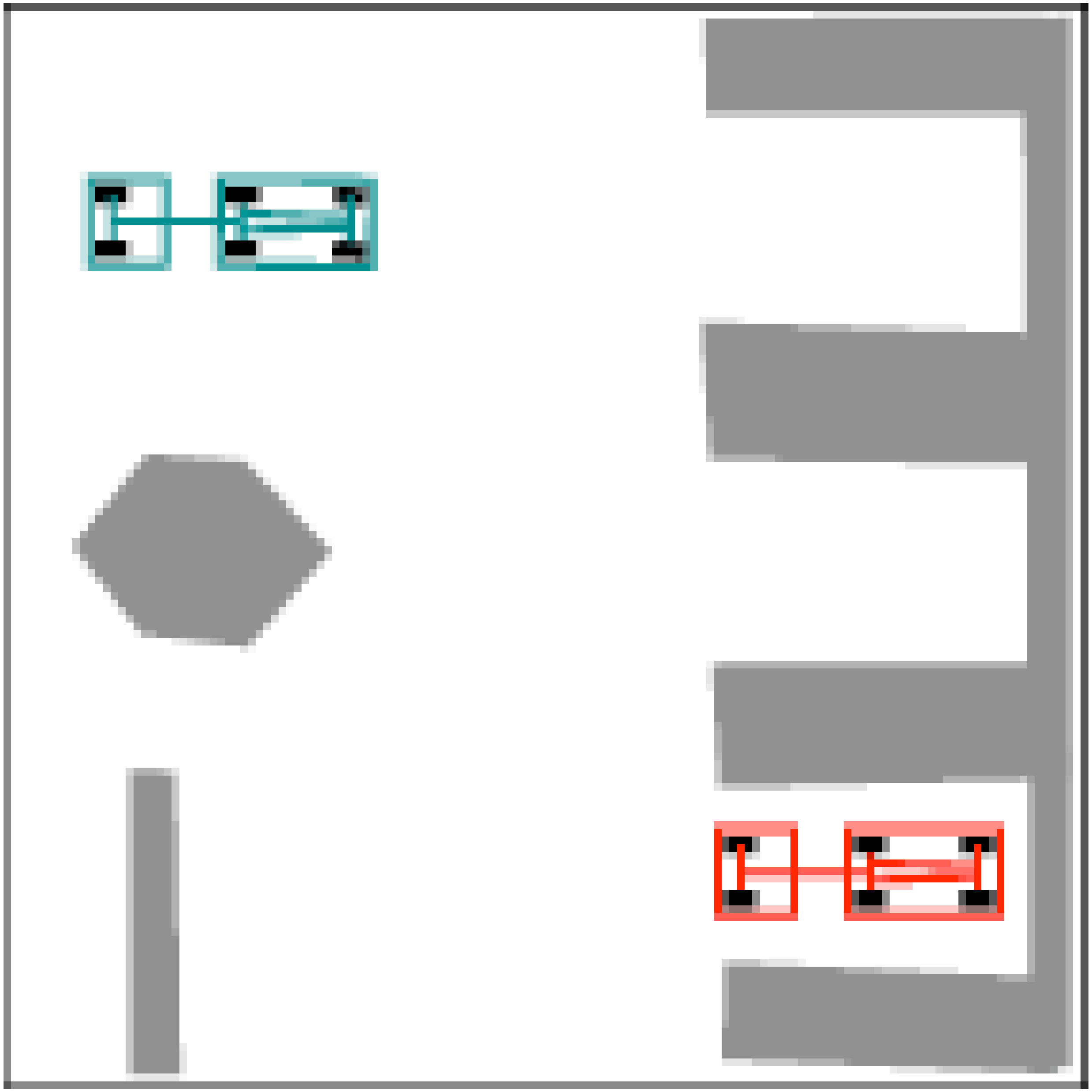
Planning with RRTs

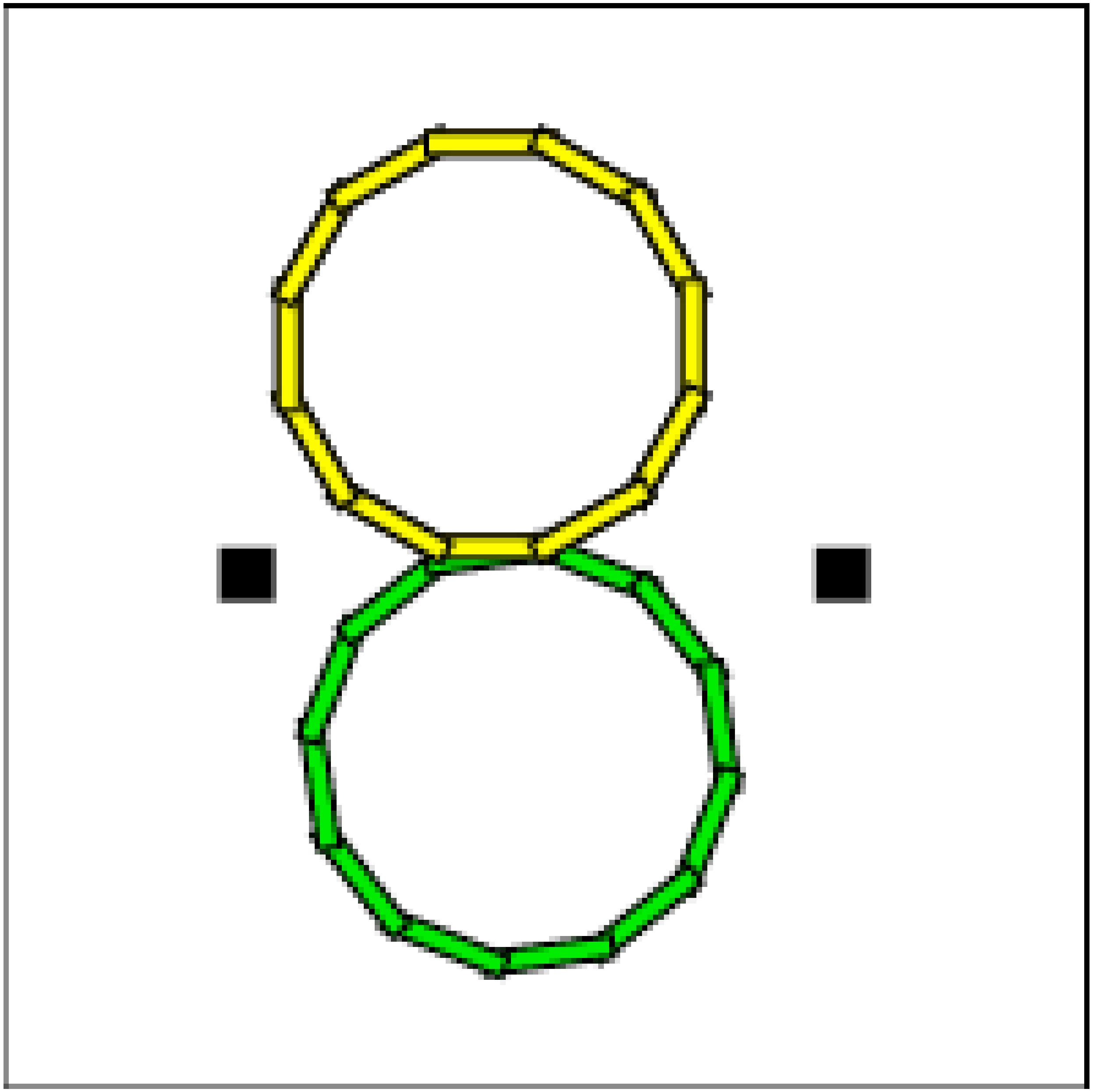
- *Build RRT from start until we add a node that can reach goal using local controller*
- *(Unique) path: root → last node → goal*
- *Optional: cross-link tree by testing local controller, search within tree using A**
- *Optional: grow forward and backward*











What you should know

- *C-space*
- *Ways of splitting up C-space*
  - *Visibility graph*
  - *Voronoi*
  - *Cell decomposition*
  - *Variable resolution or adaptive cells (quadtree, parti-game)*
- *RRTs*

AI +
Uncertainty

Uncertainty is ubiquitous

- *Random outcomes (coin flip, who's behind the door, …)*
- *Incomplete observations (occlusion, sensor noise)*
- *Behavior of other agents (intentions, observations, goals, beliefs)*

Propositional v. lifted

- *We've talked about both propositional and lifted* ***logical*** *representations*
  - *SAT v. FOL*
- *Can add uncertainty to both*
- *We'll do just propositional; uncertainty in lifted representations is a topic of current research*

Review: probability

Random variables

- *Uncertain analog of propositions in SAT or variables in MILP*
- *Tomorrow’s weather, change in AAPL stock price, grade on HW4*
- ***Observations** convert random variables into fixed realizations*

Notation

- *X=x means r.v. X is realized as x*
- *P(X=x) means probability of X=x*
  - *if clear from context, may omit "X="*
  - *instead of P(Weather=rain), just P(rain)*
- *P(X) means a function: x → P(X=x)*
- *P(X=x, Y=y) means probability that both realizations happen simultaneously*

Discrete distributions

		AAPL price		
		up	*same*	*down*
Weather	*sun*	*.09*	*.15*	*.06*
	rain	*.21*	*.35*	*.14*

		15-780		
		A	*A-*	*B+*
HW4	*A*	*.21*	*.17*	*.07*
	A-	*.17*	*.15*	*.06*
	B+	*.07*	*.06*	*.04*

Joint distribution

- ***Atomic event**: a joint realization of all random variables of interest*
- ***Joint distribution**: assigns a probability to each atomic event*

AAPL

Weather

	up	*same*	*down*
sun	*.09*	*.15*	*.06*
rain	*.21*	*.35*	*.14*

$e = sun, same$

$P(e) = 0.15$

Events

		AAPL		
		up	*same*	*down*
Weather	*sun*	*.09*	*.15*	*.06*
	rain	*.21*	*.35*	*.14*

- *An* ***event*** *is a set of atomic events*
  - $E = \{ (sun, same), (rain, same) \}$
- $P(E) = \sum_{e \in E} P(e) = .15 + .35 = .5$
- *This one is written "AAPL = same"*

AND, OR, NOT

- *For events E, F:*
  - *E AND F means E ∩ F*
  - *E OR F means E ∪ F*
  - *NOT E means U – E*
- *U is set of* ***all*** *possible joint realizations*

Marginal: eliminating unneeded r.v.s

AAPL price

		up	same	down
Weather	sun	.09	.15	.06
	rain	.21	.35	.14

Weather	sun	0.3
	rain	0.7

$$.09 + .15 + .06 = .3$$

$$.21 + .35 + .14 = .7$$

Law of Total Probability

- *For any X, Y*
  - $P(X) = P(X, Y{=}y_1) + P(X, Y{=}y_2) + \ldots$

Conditional: incorporating observations

15-780

HW4

	A	*A-*	*B+*
A	*.21*	*.17*	*.07*
A-	*.17*	*.15*	*.06*
B+	*.07*	*.06*	*.04*

15-780

A	*.41*
A-	*.35*
B+	*.24*

$$P(15\text{-}780=A \mid HW4=B+) = .04 / (.07+.06+.04)$$

Conditioning

- *In general, divide a row or column by sum*
  - $P(X \mid Y=y) = P(X, Y=y) / P(Y=y)$
- $P(Y=y)$ *is a marginal probability, gotten by row or column sum*
- $P(X \mid Y=y)$ *is a row or column of table*
- *Thought experiment: what happens if we condition on an event of zero probability?*

Notation

- $P(X \mid Y)$ *is a function:* $x, y \rightarrow P(X=x \mid Y=y)$
- *Expressions are evaluated separately for each realization:*
  - $P(X \mid Y)\, P(Y)$ *means the function* $x, y \rightarrow P(X=x \mid Y=y)\, P(Y=y)$

Independence

- *X and Y are* ***independent*** *if, for all possible values of y,* $P(X) = P(X \mid Y{=}y)$
  - *equivalently, for all possible values of x,* $P(Y) = P(Y \mid X{=}x)$
  - *equivalently,* $P(X, Y) = P(X)\, P(Y)$
- *Knowing X or Y gives us no information about the other*

Bayes Rule

- *For any X, Y, C*
  - $P(X \mid Y, C)\, P(Y \mid C) = P(Y \mid X, C)\, P(X \mid C)$
- *Simple version (without context)*
  - $P(X \mid Y)\, P(Y) = P(Y \mid X)\, P(X)$
- *Proof: both sides are just* $P(X, Y)$
  - $P(X \mid Y) = P(X, Y) / P(Y)$ *(by def'n of conditioning)*

Continuous distributions

- *What if X is real-valued?*
    - *Atomic event:* $(X \le x)$, $(X \ge x)$
        - *any* other subset via AND, OR, NOT*
    - $X=x$ *has zero probability**
- $P(X=x, Y=y)$ *means*
    - $\lim P(x \le X \le x+\varepsilon, Y = y) / \varepsilon$
    - *as* $\varepsilon \to 0+$

Factor Graphs

Factor graphs

- *Strict generalization of SAT to include uncertainty*
- *Factor graph = (variables, constraints)*
  - *variables: set of discrete r.v.s*
  - *constraints: set of (possibly)* ***soft*** *or* ***probabilistic*** *constraints called* ***factors*** *or* ***potentials***

Factors

- *Hard constraint:* $X + Y \geq 3$
- *Soft constraint:* $X + Y \geq 3$ *is more probable than* $X + Y < 3$, *all else equal*
- *Domain: set of relevant variables*
  - $\{X, Y\}$ *in this case*

Factors

Hard

X

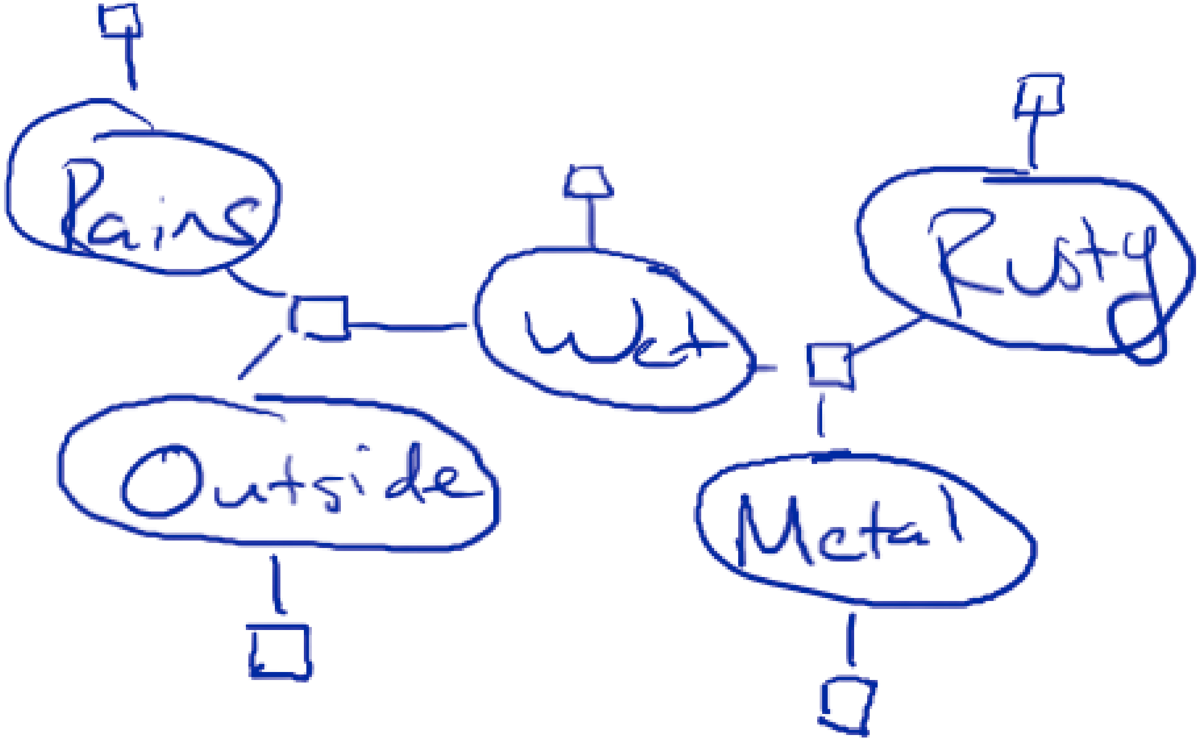
		0	*1*	*2*
	0	*0*	*0*	*0*
Y	*1*	*0*	*0*	*1*
	2	*0*	*1*	*1*

Soft

X

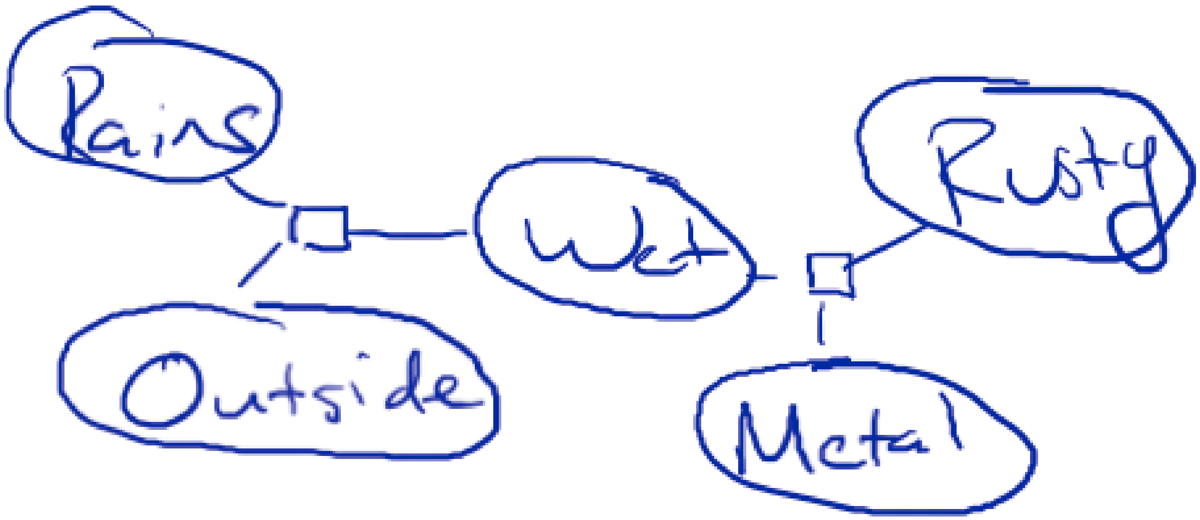
		0	*1*	*2*
	0	*1*	*1*	*1*
Y	*1*	*1*	*1*	*3*
	2	*1*	*3*	*3*

Factor graphs



- *Variables and factors connected according to domains*

Factor graphs



- *Omit single-argument factors from graph to reduce clutter*

Factor graphs: probability model

$$P(Ra, W, O, M, Ru) =$$

$$\phi(Ra, W, O, M, Ru)/Z$$

$$\phi(Ra, W, O, M, Ru) =$$

$$\phi_1(Ra, W, O)\,\phi_2(O, M, Ru)$$

$$\phi_3(Ra)\,\phi_4(W)\,\phi_5(O)$$

$$\phi_6(M)\,\phi_7(Ru)$$

Normalizing constant

$$Z = \sum_{Ra} \sum_{W} \sum_{O} \sum_{M} \sum_{Ru} \phi(Ra, W, O, M, Ru)$$

- *Also called* ***partition function***

Factors

Ra	O	W	Φ_1
T	T	T	3
T	T	F	1
T	F	T	3
T	F	F	3
F	T	T	3
F	T	F	3
F	F	T	3
F	F	F	3

W	M	Ru	Φ_2
T	T	T	3
T	T	F	1
T	F	T	3
T	F	F	3
F	T	T	3
F	T	F	3
F	F	T	3
F	F	F	3

Unary factors

Ra	Φ_3
T	*1*
F	*2*

W	Φ_4
T	*1*
F	*1*

O	Φ_5
T	*5*
F	*2*

M	Φ_6
T	*10*
F	*1*

Ru	Φ_7
T	*1*
F	*3*

Inference Qs

- *Is* $Z > 0$*?*
- *What is* $P(E)$*?*
- *What is* $P(E_1 \mid E_2)$*?*
- *Sample a random configuration according to* $P(.)$ *or* $P(. \mid E)$
- *Hard part: taking sums of* Φ *(such as* Z*)*

Example

- *What is* $P(T, T, T, T, T)$*?*

Example

- *What is P(Rusty=T | Rains=T)?*
- *This is P(Rusty=T, Rains=T) / P(Rains=T)*
- *P(Rains) is a marginal of P(…)*
- *So is P(Rusty, Rains)*
- *Note: Z cancels, but still have to sum lots of entries to get each marginal*

Relationship to SAT, ILP

- *Easy to write a clause or a linear constraint as a factor: 1 if satisfied, 0 o/w*
- *Feasibility problem: is Z > 0?*
  - *more generally, count satisfying assignments (determine Z)*
  - *NP or #P complete (respectively)*
- *Sampling problem: return a satisfying assignment uniformly at random*