CS 460/560
Introduction to Computational Robotics
Fall 2019, Rutgers University

Lecture 18
Feedback & Potential Field Based Planners

Instructor: Jingjin Yu

Outline

Artificial potential field

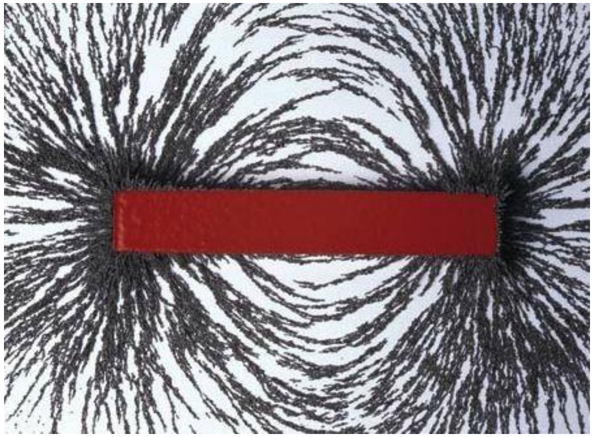
Feedback based planner

- ⇨Discrete state spaces
- ⇨Vector fields for continuous domains

Potential Field

Potential fields in the nature



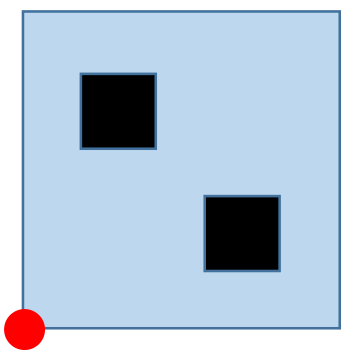


- ⇨ A dropped ball or free water will simply fall with gravity
- ⇨ Same applies to other types of potential fields
  - ⇨ E.g., magnets can be used to pick up small ferrous metals (iron, steel)
  - ⇨ E.g., the forces between gaseous molecules
  - ⇨ There can be attractive and repulsive forces
- ⇨ Recall in the case of gravity, the potential $U(r) = -\frac{GMm}{r}$
  - ⇨ M, m are the masses of the two attracting bodies, G is a constant, r is the distance
  - ⇨ $F = -\frac{dU}{dr} = -\frac{GMm}{r^2}$ ($\frac{GM}{r^2}$ is about $9.8\ m/s^2$ near earth's surface)

Artificial Potential Field for Motion Planning (I)

Potential fields inspired the development of **artificial potential field** for motion planning. General idea:

- ⇨Used to guide a robot to a goal region from anywhere in the environment
- ⇨To do so, builds a **navigation function** for the robot to follow
- ⇨The function can be thought of forces from an artificial potential field
- ⇨Includes an **attraction term** to pull the robot toward the goal
- ⇨Includes **repulsive terms** to push the robot away from obstacles
- ⇨Together, creating a desired potential field
- ⇨This yields a **function** $U(x)$



Goal

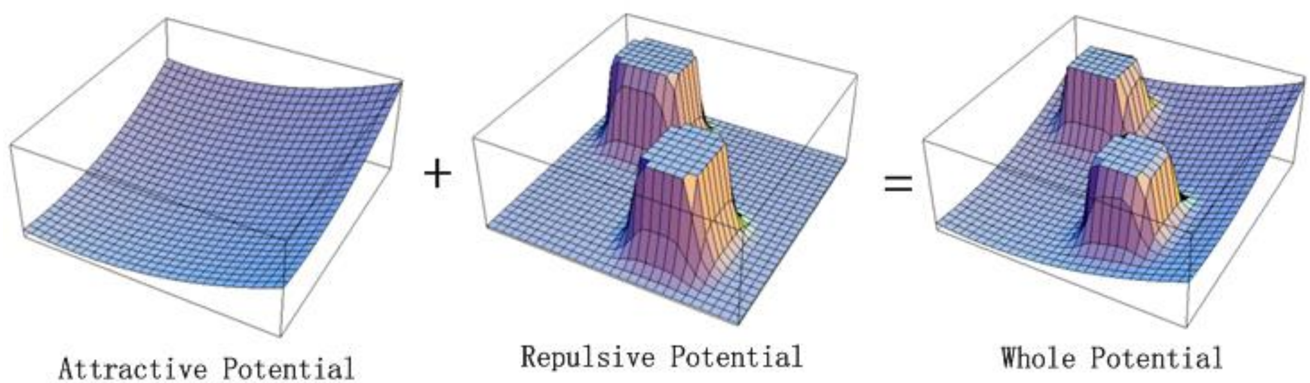
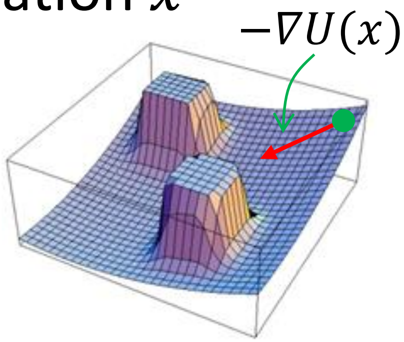


Image source: https://taylorwang.wordpress.com/

Artificial Potential Field for Motion Planning (II)

Given a potential field $U(x)$, to reach goal from configuration x

- ⇨Compute the negative gradient $-\nabla U = -[\frac{\partial U}{\partial x_1}, \ldots, \frac{\partial U}{\partial x_n}]$
- ⇨Simply follow it! Update as x changes
- ⇨$U(x)$ may be viewed as level sets



Constructing an artificial potential field can be tricky

- ⇨Generally easier to do for convex obstacles
- ⇨Can be challenging for non-convex obstacles
  - ⇨ May create local minima
  - ⇨ Traps the robot
- ⇨There are ways to avoid it for “star worlds”
- ⇨Applies to higher dimensions as well
- ⇨For details, see Rimon & Kodischeck, ’92

May be viewed a **feedback** based planner

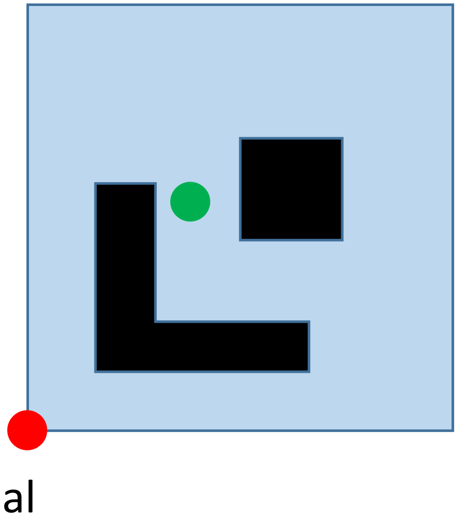


Image source: https://taylorwang.wordpress.com/

Problems with Classic Path Planners

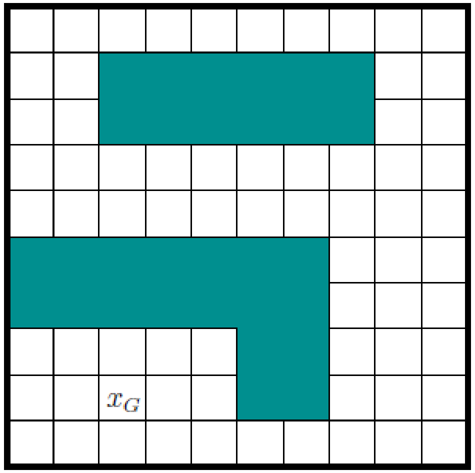
Most planners covered until now are **open loop**

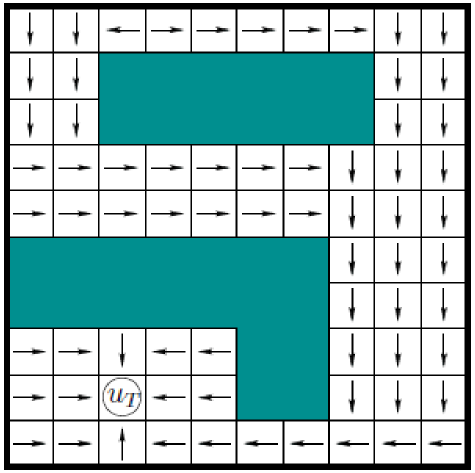
- ⇨ Paths generated by these planners are for ideal conditions
  - ⇨ E.g., applying same thrust to two wheels of a DDR, the robot will move straight
  - ⇨ Works fine for ideal models, e.g., simulation, games
  - ⇨ What happens if you do this to the our robot?
    - ⇨ Most likely not a straight line
    - ⇨ Otherwise, our lives will be much simpler!
    - ⇨ Perhaps boring though…
- ⇨ How to accommodate such variations?
- ⇨ Use **feedback**!
  - ⇨ Potential field is such a planner
- ⇨ Feedback allows the handling of many sources of uncertainties, e.g.
  - ⇨ Motor speed variations between different motors under the same input
  - ⇨ Motor speed changes as battery charging state changes
  - ⇨ Uneven terrain: carpet, wooden floor, gravel, …
  - ⇨ Air flow
  - ⇨ …

Feedback Based Planner

What does feedback based planner do?

- Assumptions
  - The robot has a good estimate of its current state $x \in X$
  - There is a specific goal set (may be multiple goals) to navigate to
- Feedback plan is actually a **policy** $\pi(x)$
  - At each $x \in X$, $\pi: X \rightarrow U, x \mapsto \pi(x)$ produces a control for the robot
  - All states in the free space must be covered
  - The policy is executed until a goal is reached





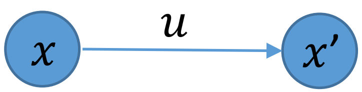
- Key: feedback based planner provides a (lookup) policy. In particular, **no search** is needed after the policy is created.

Image source: Planning Algorithms

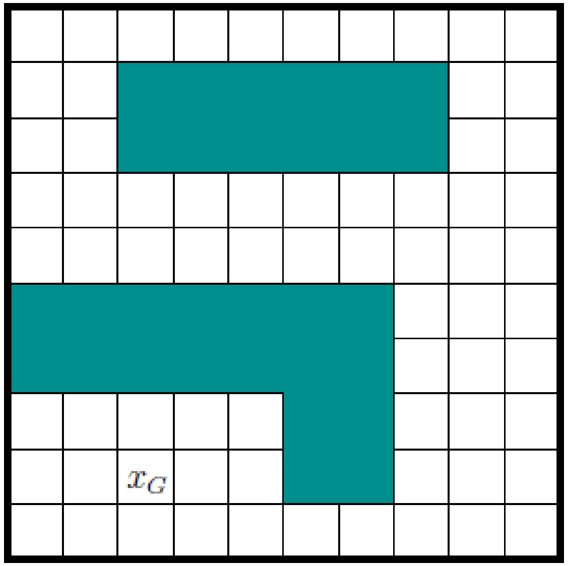
Finding Feasible Solutions

Feedback based problem uses transition function $f(x, u)$

- ⇨ For a finite control set U, this defines a directed graph. How?
  - ⇨ Basically, $x' = f(x, u)$ induces a directed edge



- ⇨ To find a policy, we need to grow a directed acyclic graph (DAG) from x_G
  - ⇨ To do this, first reverse all edge directions
  - ⇨ Then execute any **spanning tree** algorithm
  - ⇨ This yields a spanning tree from x_G to all other states in the free space
  - ⇨ Reversing the edge directions again then yields a feasible solution



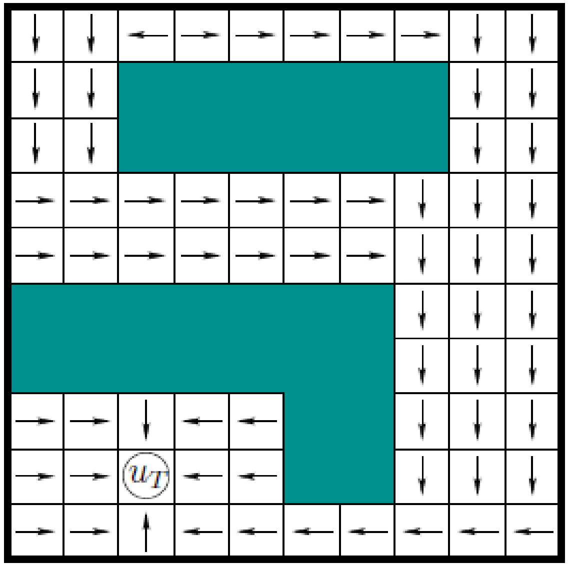
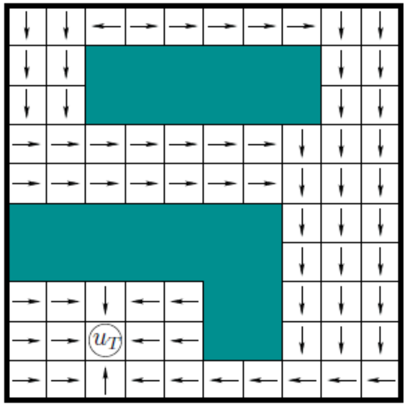


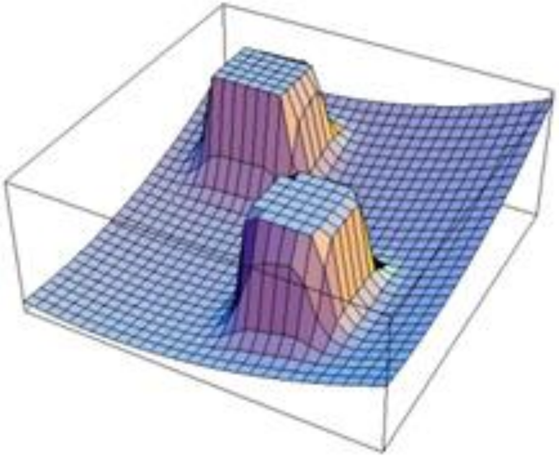
Image source: Planning Algorithms

Finding Optimal Solutions



What about optimal feedback policy?

- ⇨ We may simply run Dijkstra's algorithm!
  - ⇨ There are other faster algorithms, but with similar principle
- ⇨ Since we inverted the edges, how do we know this works?
  - ⇨ Can prove via contradiction
- ⇨ We can go one step further and build **navigation functions**
  - ⇨ Associate each state with a (cost-to-go) value
  - ⇨ For a non-goal node, there must exists a neighbor with lower value
  - ⇨ Essentially, this yields a level set (potential function)
  - ⇨ Can also take other forms, e.g., maximum clearance



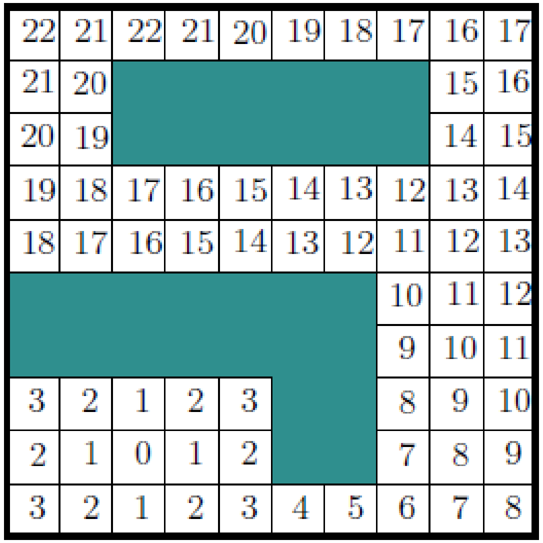
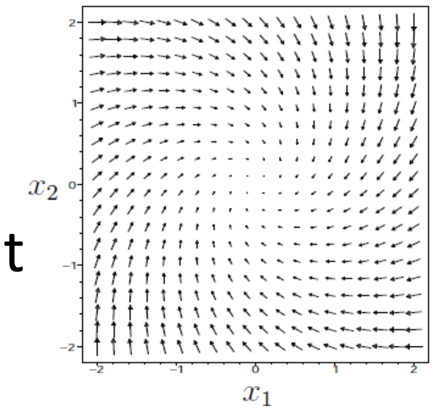


Image source: Planning Algorithms

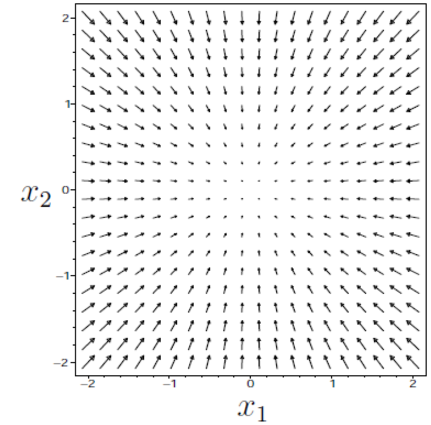
Continuous Domain: Vector Field



Continuous domain is similar to the discrete domain but

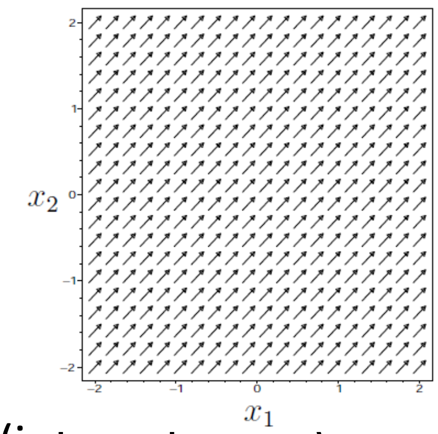
⇨Need to express the (control) policy differently

⇨For this we need vector fields



Basic properties of vector fields are:

1. **(Commutative Group Under Vector Addition)** The set V is a commutative group with respect to vector addition, $+$.
2. **(Associativity of Scalar Multiplication)** For any $v \in V$ and any $\alpha, \beta \in \mathbb{F}$, $\alpha(\beta v) = (\alpha\beta)v$.
3. **(Distributivity of Scalar Sums)** For any $v \in V$ and any $\alpha, \beta \in \mathbb{F}$, $(\alpha + \beta)v = \alpha v + \beta v$.
4. **(Distributivity of Vector Sums)** For any $v, w \in V$ and any $\alpha \in \mathbb{F}$, $\alpha(v + w) = \alpha v + \alpha w$.
5. **(Scalar Multiplication Identity)** For any $v \in V$, $1v = v$ for the multiplicative identity $1 \in \mathbb{F}$.



How does it work?

⇨Following smooth vector fields creates a smooth trajectory (integral curve)

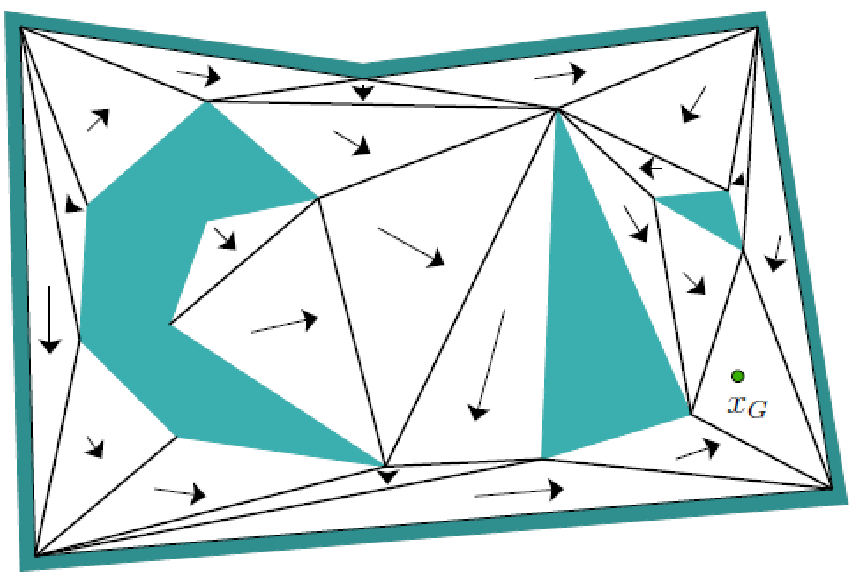
⇨If the vector field “flows” into the goal region, then we are done

Image source: Planning Algorithms

Vector Field Construction Methods (I)

Triangulation based construction

- ⇨First, triangulate the domain
- ⇨Build a graph of the triangles and then build a spanning tree from x_G
- ⇨For each triangle, generate a simple vector field pointing to the **exit edge**
- ⇨Works for high dimensions



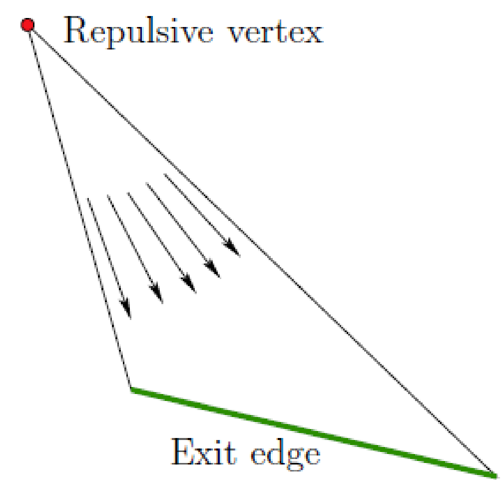
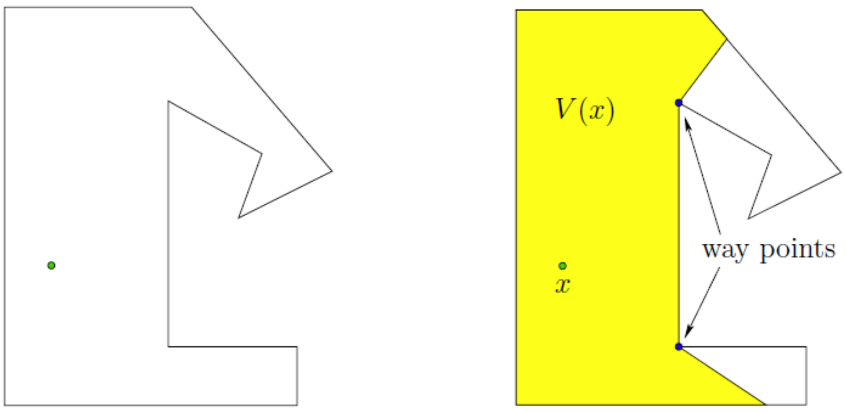


Image source: Planning Algorithms

Vector Field Construction Methods (II)

Visibility based construction

- ⇨Compute visibility region of goal
- ⇨Do wavefront propagation
- ⇨Iteratively applies to way points
- ⇨The method is optimal
- ⇨But difficult for high dimensions



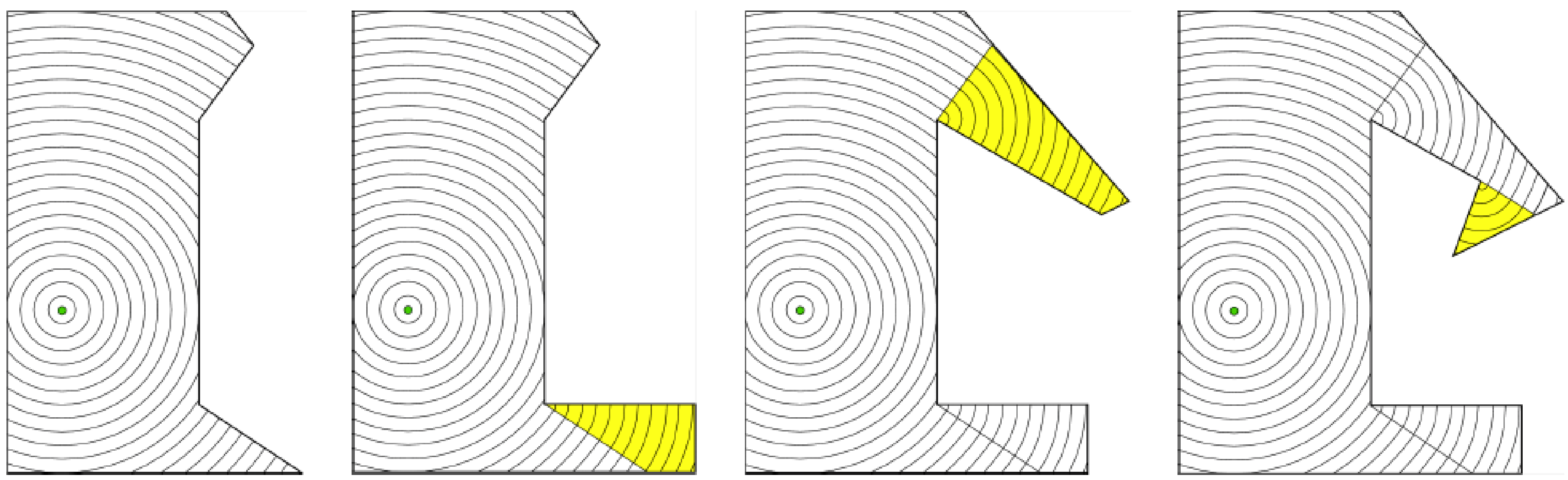
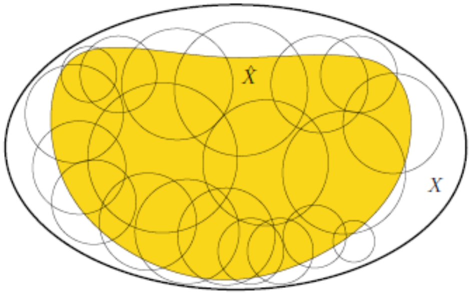


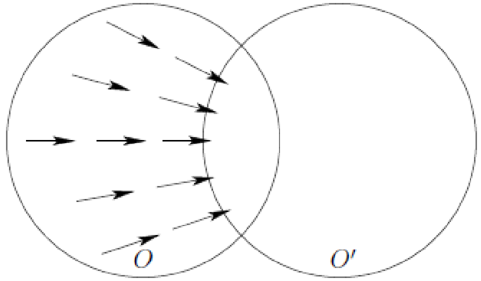
Image source: Planning Algorithms

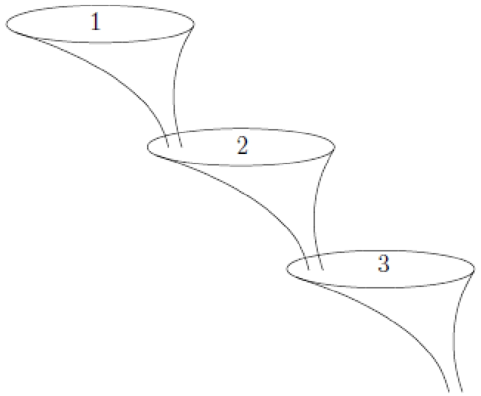
Vector Field Construction Methods (III)



Sampling-based funnel composition

- ⇨Sampling based methods also apply here!
- ⇨First, compute a cover via sampling
- ⇨Then, again build a graph based on cover adjacency
  - ⇨ I.e., two intersecting covers will share an edge
- ⇨Obtain a spanning tree from the graph
- ⇨For each pair of covers, construct a local vector field
- ⇨All together, these become a bunch of “funnels”





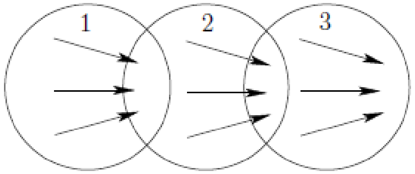


Image source: Planning Algorithms