Graph Search Algorithms

PDMAR - kmangold

Forward Search

Greedy BFS

```
Q ← ⟨s0⟩
V ← {s0}
Parent(s0) ← null
while Q is not empty do
    take 1st* element s from Q
    if s ∈ S_goal then
        return path ending at s
    for all a, s' such that (s,a,s') ∈ → do
        if s' is not in V then
            insert s' into Q
            add s' into V
            set Parent(s') ← s
return FAILURE
```

Depth-First Search (DFS)

new states added at **FRONT**

- sound but not complete

Time Complexity: ~ # visited nodes
↳ Worst Case: $O(b^m)$

Space Complexity: ~ max. size priority Q
↳ Worst Case: $O(b \cdot m)$

Breadth-First Search (BFS)

new states added at **BACK**

- sound and complete

Time Complexity: ~ # visited nodes
↳ Worst Case: $O(b^d)$

Space Complexity: ~ max. size priority Q
↳ Worst Case: $O(b^d)$

Tree: branching factor b, max. depth m, min. goal depth d

Iterative Deepening

Idea: explore in BFS order using DFS

```
d ← 1
repeat
    Run DFS up to depth d
    if DFS returns a path p then
        return p
    d ← d+1
until d ≥ m
return FAILURE
```

Shortest Path Problems (SPP)

given: Σ, s_0, $\{g_1, g_2, \dots\} = S_{Goal} \subset S$,
$w: S \times A, S \to \mathbb{R}_{>0}$

find: trajectory of $\sigma = (s_0, s_1, \dots, s_n)$ of Σ and corresponding $(a_1, \dots, a_n)$ s.t. $s_n \in S_{Goal}$ and min. cost $Cost(\sigma) := \sum_{i=1}^{n} w(s_{i-1}, a_i, s_i)$ among all such paths

Uniform-Cost Search (UCS)

```
Q ← ⟨s0⟩
costToReach(s0) = 0
Parent(s0) = 0
while Q is not empty do
    Take 1st (lowest costToReach) element s from Q
    if s ∈ S_Goal then
        return path ending at s
    for all a, s' such that (s,a,s') ∈ → do
        newCostToReach ← costToReach + w(s,a,s')
        if newCostToReach < costToReach(s') then
            costToReach(s') ← newCostToReach
            Parent(s') ← s
            insert (or update) s' in Q
return FAILURE
```

UCS = BFS if all edges same cost

- complete and optimal

Worst-case complexity in time and space $O(b^{W^*/\varepsilon})$ where W^* is the optimal cost and ε s.t. all edge weights no smaller than ε

Greedy BFS

Heuristic function $h: V \to \mathbb{R}_{\geq 0}$
↳ e.g. Euclidean distance

⇒ move s.t. distance minimized

* first ≙ lowest heuristic cost h

Worst-Complexity: $O(b^m)$

A-Search Algorithm

Idea: keep track of cost to reach state $c(s)$ and of heuristic function estimating cost to reach goal from s, $h(s)$

⇒ ranking function: $f(s) = c(s) + h(s)$

- complete but not optimal (depends on h); if $h=0$: $A \equiv UCS$

```
Q ← ⟨s0⟩
c(s0) = 0
Parent(s0) = 0
while Q is not empty do
    Take 1st (lowest f=c+h) elem. s from Q
    if s ∈ S_Goal then
        return path ending at s
    for all a, s' s.t. (s,a,s') ∈ → do
        newCostToReach ← c(s) + w(s,a,s')
        if newCostToReach < c(s') then
            c(s') ← newCostToReach
            Parent(s') ← s
            insert (or update) s' in Q
return FAILURE
```

A*-Search

choose admissible heuristic s.t. $h(s) \leq h^*(s)$ $\forall s$ where $h^*(s)$ true optimal cost to go

- heuristic that never overestimates

$h(v) = 0$: always works; not useful

$h(v) = distance(v, g)$: when vertices on graph are physical locations

$h(v) = \|v - g\|_p$: when vertices on graph are points in normed vector space

⇒ choose h as opt. cost to go for relaxed problem

partial order heuristics: h^* dominates $h_1(s) \geq h_2(s)$ and h_1 dominates h_2, which all dominate $h=0$

consistency: satisfy triangle inequality!
heuristic $h: S \to \mathbb{R}_{\geq 0}$ if for any $(s,a,s') \in \to$,
$h(s) \leq w(s,a,s') + h(s')$

Dynamic Programming

Optimality Principle

optimal paths made of optimal (sub-)paths

let $P = (s, \dots, v, \dots, g)$ be optimal path from s to g then $\forall v \in P$:

- subpath $P_v = (v, \dots, g)$ is itself optimal
- cost of P_v equal to $h^*(v)$, the opt. cost-to-go from v to g
- if v and v' are consecutive states on P then

$$h^*(v) = \min_{(v,a,v') \in \to} [w(v,a,v') + h^*(v')]$$ and optimal action at state v minimizes this

⇒ can always construct path backwards from goal since $h^*(g) = 0$ always

Dijkstra's Algorithm - Special Case of Dynamic Programming

```
for all s ∈ S do
    h(s) ← 0 if s ∈ S_g, +∞ otherwise
Insert s into Q
while Q is not empty do
    take 1st (lowest cost-to-go h) element s' from Q
    for all s, a, such that (s,a,s') ∈ → do
        newCostToGo ← w(s,a,s') + h(s')
        if newCostToGo < h(s) then
            h(s) ← newCostToGo
            Insert/update s in Q
```

Markov Decision Process (MDP)

defined by:
- countable set of states S and actions A
- transition probability function $T: S \times A \times S \to \mathbb{R}_{\geq 0}$
- reward function $R: S \times A \times S \to \mathbb{R}_+$

if action a is applied from state s, a transition to state s' will occur with probability $T(s,a,s')$ and with each made transition a reward $R(s,a,s')$ is collected

Markov property: T and R only depend on present state, not on history of the state

Transitions are stochastic, but state is known exactly → policy not path

policy $\pi: S \to A$ return action $a = \pi(s)$ to take at state s

Total Reward in MDP over infinite time

$$V = \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}), \quad \gamma \in (0, 1]$$

Value Function for given policy π at state s_0

$$V^\pi(s_0) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t), s_{t+1})\right]$$

$$= E\left[R(s_0, \pi(s_0), s_1) + \underbrace{\sum_{t=1}^{\infty} \gamma^t R(s_t, \pi(s_t), s_{t+1})}_{\gamma V^\pi(s_1)}\right]$$

Bellman's equation

$$V^*(s) = \max_a E[R(s,a,s') + \gamma V^*(s')]$$

optimal policy π^* yields max. expected reward V^*

Value Iteration

$O(card(S) \cdot card(A))$ time at each iteration

1. initialize with arbitrary $V_0(s)$
2. update: $V_1(s) \leftarrow \max_a E[R(s,a,s') + \gamma V_0(s')] = \max_a \sum_{s' \in S} T(s,a,s')[R(s,a,s') + \gamma V_0(s')]$
3. iterate until: $\max_s |V_{i+1}(s) - V_i(s)| < \varepsilon$

Value Iteration Convergence

Bellman operator B: $BV(s) := \max_a E[R(s,a,s') + \gamma V(s')]$
$= \max_a \sum_{s' \in S} T(s,a,s')[R(s,a,s') + \gamma V(s')]$

contraction for $\gamma < 1$: $\max_s \|BV_1(s) - BV_2(s)\| \leq \gamma \cdot \max_s \|V_1(s) - V_2(s)\|$

Optimal policy from $V^*(s)$

$\pi^*(s) = \arg\max_a E[R(s,a,s') + \gamma V^*(s')]$
↳ $\forall s \in S$

Policy evaluation

$$V^\pi(s) = E[R(s,\pi(s),s') + \gamma V^\pi(s')] = \sum_{s' \in S} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V^\pi(s')]$$

Roll-out policies

given baseline policy π_0 find at least as good π_1

$$\pi_1(s) = \arg\max_{a \in A} E[R(s,a,s') + \gamma V^{\pi_0}(s')] = \arg\max_{a \in A} \sum_{s' \in S} T(s,a,s')[R(s,\pi(s),s') + \gamma V^{\pi_0}(s')]$$

Policy Iteration

$O(card(S)^3)$ time at each iteration

1. Policy evaluation: solve linear system
$$V(\pi) = \sum_{s' \in S} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V(s')]$$
2. Policy improvement: for each $s \in S$
$$\pi(s) \leftarrow \arg\max_a \sum_{s' \in S} T(s,a,s')[R(s,a,s') + \gamma V(\pi')]$$

until π is unchanged

Modified Policy Iteration

approximation: $V_{i+1}(s) = \sum_{s' \in S} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V_i(s)]$

Game-Theoretic Planning

PDMAR - kmangold

Normal Form:
$a_{ij} = J_1(\gamma_i, \sigma_j)$, $b_{ij} = J_2(\gamma_i, \sigma_j)$ $\Leftrightarrow$ $\begin{bmatrix} (a_{11}, b_{11}) & (a_{12}, b_{12}) \\ (a_{21}, b_{21}) & (a_{22}, b_{22}) \end{bmatrix}$

Best response:
$R_1(\sigma) = \arg\min_{\gamma \in \Gamma} J_1(\gamma, \sigma)$
$R_2(\gamma) = \arg\min_{\sigma \in \Sigma} J_2(\gamma, \sigma)$

Nash Equilibrium

$\gamma^* \in R_1(\sigma^*) = \arg\min_{\gamma \in \Gamma} J_1(\gamma, \sigma^*)$ and $\sigma^* \in R_2(\gamma^*) = \arg\min_{\sigma \in \Sigma} J_2(\gamma^*, \sigma)$

Pure and Mixed Strategies

pure: deterministic vs. mixed: stochastic (each sums up to 1)

Randomized Play: $\Gamma = \{r_1, \dots, r_n\} \to y = [y_1, \dots, y_n]^T$; $\Sigma = \{\sigma_1, \dots, \sigma_m\} \to z = [z_1, \dots, z_m]^T$

Expected outcome: $J_1(y,z) = \sum_{i=1}^{m} \sum_{j=1}^{n} y_i a_{ij} z_j = y^T A z$

$J_2(y,z) = \sum_{i=1}^{m} \sum_{j=1}^{n} y_i b_{ij} z_j = y^T B z$

Zero-Sum Games

Definition: $J_1(y,z) = -J_2(y,z) =: J(y,z)$

Saddle point / Nash Equilibrium: $J(y^*, z) \le J(y^*, z^*) \le J(y, z^*) \quad \forall y \in \mathcal{Y}, z \in \mathcal{Z}$

Security Levels: P_2 could choose z knowing y in advance

P_1: $\min_{y, V} V$ s.t. $\sum_{i=1}^{m} y_i a_{ij} \le V$, $j = 1, \dots, n$, $y \in \mathcal{Y}$, $V \in \mathbb{R}$

strong duality

P_2: $\max_{z, V} V$ s.t. $\sum_{j=1}^{n} a_{ij} z_j \ge V$, $i = 1, \dots, m$, $z \in \mathcal{Z}$, $V \in \mathbb{R}$

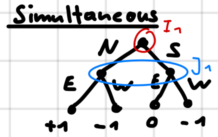
Backward Induction: ↳ subgame perfect Nash Equilibria

$\Rightarrow \bar{V} = \min_{y \in \mathcal{Y}} \max_{z \in \mathcal{Z}} J(y,z) = \max_{z \in \mathcal{Z}} \min_{y \in \mathcal{Y}} J(y,z) = \underline{V}$

$\Rightarrow$ saddle point strategies: $J(\bar{y}, z) \le J(\bar{y}, \bar{z}) \le J(y, \bar{z})$

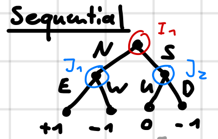
Multi-Stage Games

Simultaneous



$$\begin{array}{c|cc} & E & W \\ \hline N & +1 & -1 \\ S & 0 & -1 \end{array}$$

Sequential



$$\begin{array}{c|cccc} & E\,U & E\,D & W\,U & W\,D \\ \hline N & +1 & +1 & -1 & -1 \\ S & 0 & -1 & 0 & -1 \end{array}$$

Behavioral Strategy:
Map that assigns probability distribution
$I_n \mapsto \gamma^b(I_n) \in \mathcal{Y}_n \subset \mathbb{R}^{|U_n|}$
- decide at each vertex a prob. distr. on actions
- not restrictive in feedback games

Alpha-Beta Pruning (Branch and Bound)

1. visit vertices in DFS order
2. at 1^{st} visit of $\begin{Bmatrix} MAX \\ MIN \end{Bmatrix}$ node, set its $\begin{Bmatrix} \beta \\ \alpha \end{Bmatrix}$ value to $\begin{Bmatrix} \beta \\ \alpha \end{Bmatrix}$ value of parent
3. every time $\begin{Bmatrix} MAX \\ MIN \end{Bmatrix}$ node is revisited, update its $\begin{Bmatrix} \alpha \\ \beta \end{Bmatrix}$ value to $\begin{Bmatrix} \text{maximum} \\ \text{minimum} \end{Bmatrix}$ known value of children
4. if at any point $\alpha \ge \beta \Rightarrow$ PRUNE SUBTREE
5. when leaving vertex s for last time, set its value to $\begin{Bmatrix} \alpha \\ \beta \end{Bmatrix}$ if $\begin{Bmatrix} MAX \\ MIN \end{Bmatrix}$ node

α-β needs $\mathcal{O}(b^{d/2})$ time vs. minmax (DFS) $\mathcal{O}(b^d)$

Steering

$\dot{x} = \frac{d}{dt} x(t) = F(x(t), u(t))$, $x(0) = x_0$

Linear Systems

Zero-order hold: $F(x,u) = Ax + Bu \Rightarrow \dot{x}(t) = Ax(t) + Bu(t)$, $x(0) = x_0$

u_0 on $[0,T)$ $\Rightarrow x(T) = \underbrace{e^{AT}}_{A_d} x_0 + \underbrace{\left(\int_0^T e^{A\tau} d\tau\right) B}_{B_d} u_0 \Rightarrow x_1 = x(T) = A_d x_0 + B_d u_0$

N steps $[(N-1)T, N \cdot T)$ $\Rightarrow x_N = A_d^N x_0 + \underbrace{\left[A_d^{N-1} B_d \,|\, \cdots \,|\, A_d B_d \,|\, B_d\right]}_{R_N} \begin{bmatrix} u_{N-1} \\ \vdots \\ u_1 \\ u_0 \end{bmatrix} = R_N U_N$

Minimum-energy control: find control sequence $u_0, u_1, \dots, u_{N-1}$ that takes system from x_0 to x_f in N steps by solving:

$R_N U_N = x_N - A_d^N x_0 \Rightarrow$ at least 1 solution if controllable and $N \ge \dim(x)$

$\Rightarrow$ N-Step min.-energy ctrl: least-square solution of underdet. sys. of eqn.

$U_N = R_N^T (R_N R_N^T)^{-1} (x_N - A_d^N x_0)$

Differential Flatness

non-linear system $\dot{x} = f(x,u)$ is differentially flat if there is a set of flat outputs $y = h(x, u, \dot{u}, \ddot{u}, \dots)$ from which both state and inputs can be recovered without integration
$x = \xi(y, \dot{y}, \ddot{y}, \dots)$, $u = \eta(y, \dot{y}, \ddot{y}, \dots)$

$\Rightarrow$ diff. flat sys. can be steered s.t. outputs match any given trajectory

$\Rightarrow$ generate output trajectories: $y^{(\ell)}(t) = v(t)$ chain of integrators

Splines in $[t_i, t_{i+1}]$: $y(t) = a_i + b_i(t - t_i) + c_i(t - t_i)^2 + d_i(t - t_i)^3$
↳ $\dddot{y}(t) = 6 d_i$

Holonomic Systems

holonomic constraint: $h(x(t)) = h(x(0)) = \text{const.}$

with $m \le n$ indep. ctrls $\to$ reduces system to m-dim. subset of state space

$\Rightarrow$ every path can be followed

maximally non-holonomic: if there are no holonomic constraints

$\to$ not every path can be followed

$\to \exists$ path in n-dim. set that can be reached using only $m < n$ indep. ctrl.

$\to$ corresponds to controllability

Driftless, affine-in-control systems

$\dot{x} = \sum_{i=1}^{m} g_i(x) u_i$; if $m < n$: can only move in dir. in lin. span $\{g_1(x), \dots, g_m(x)\}$

Lie Brackets: apply only u_1 for time ε, then u_2 for ε, then close cycle by applying $-u_1$ and then $-u_2$ both for time ε

final point: $x(4\varepsilon) = x_0 + \varepsilon^2 \underbrace{\left(\frac{\partial g_2}{\partial x} g_1 - \frac{\partial g_1}{\partial x} g_2\right)\Big|_{x = x(0)}}_{:= [g_1, g_2] \text{ Lie-Bracket}} + \mathcal{O}(\varepsilon^3)$

Chow's theorem: consider above system. If involutive closure of vectors $g_i(x)$, $i = 1, \dots, m$ has dimension n $\forall x \in \mathbb{R}^n$ then the system is controllable on $\mathbb{R}^n$

Optimal Control and Dubins car

general: $\min_{x, u, t_f} \int_0^{t_f} g(x(t), u(t))\, dt$ s.t. $\dot{x}(t) = f(x(t), u(t))$, $u(t) \in \mathcal{U}$, $x(0) = x_0$, $\forall t \in [0, t_f]$, $x(t_f) = x_f$, $t_f > 0$

Pontryagin's principle

given general system above, introduce auxiliary variables λ and define Hamiltonian function $H(x, u, \lambda) = g(x, u) + \lambda^T f(x, u)$

$\Rightarrow u^*: [0, t_f] \to \mathcal{U}$ optimal if:
- $\exists$ non-zero $\lambda: [0, t_f] \to \mathbb{R}^n$ s.t. $\dot{\lambda} = -\frac{\partial H}{\partial x}$
- $u^*(t) = \arg\min_u H(x(t), u, \lambda(t)) \quad \forall t \in [0, t_f]$
- $H(x^*(t), u^*(t), \lambda^*(t)) = \begin{cases} 0, & \text{if } t_f \text{ is free} \\ \text{const.}, & \text{if } t_f \text{ is fixed} \end{cases}$

Dubins problem: $H = 1 + \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{pmatrix} \begin{pmatrix} \dot{p}_1(t) = \cos\theta(t) \\ \dot{p}_2(t) = \sin\theta(t) \\ \dot{\theta}(t) = u \end{pmatrix} \Rightarrow u = \begin{cases} -1/r_{min}, & R \\ 0, & S \\ 1/r_{min}, & L \end{cases}$

↳ $u^*(t) = -\text{sgn}(\lambda_3(t)) / r_{min}$, $\lambda_3 = \Lambda^{\perp T} \theta$, $\Lambda^T \theta \ge -1$

at least π [rad]

$\Rightarrow$ canonical paths: $\{RSR, LSL, RSL, LSR, RLR, LRL\}$

Reeds-Shepp problem: like Dubins but can reverse direction $\Rightarrow u_2$

$H = 1 + [\Lambda^T \theta(t) + \lambda_3(t) u_1(t)] u_2(t)$, $u_2(t) = -\text{sgn}(\Lambda^T \theta(t))$
↳ never 0 on optimal path

$\Rightarrow$ every time sign of u_2 changes, the sign of u_1 must change too (cusps) s.t. heading θ is orthogonal to direction Λ

Configuration space

- Topological Space if for set $\mathcal{X}$ there is a collection of open sub sets for which:
  - union of any number of open sets is an open set
  - intersection of finite number of open sets is an open set
  - both $\mathcal{X}$ and empty set $\emptyset$ are open sets
- closed sets: subset $C \subset \mathcal{X}$ is closed iff $\mathcal{X} \setminus C$ is open
- Closure: $cl(\mathcal{U}) = int(\mathcal{U}) \cup \partial\mathcal{U}$, points in $cl(\mathcal{U})$: accumulation / limit points
- homeomorphism: bijective function $f: \mathcal{X} \to \mathcal{Y}$ between two topological spaces if both f and f^{-1} are continuous
- manifold: $\mathcal{M} \subseteq \mathbb{R}^m$ if $\forall x \in \mathcal{M}$ there is an open set $O \subset \mathcal{M}$ s.t.
  - $x \in O$
  - O is homeomorphic to $\mathbb{R}^n$ for some $n \in \mathbb{N}$
  - n is fixed for all $x \in \mathcal{M}$ and is the dimension of $\mathcal{M}$

  $\to$ identification: $[0, 2\pi] \Rightarrow 0$ & 2π same point $\Rightarrow$ homeomorphic

$\Rightarrow$ Configuration space has structure of manifold and dimension equal to DoF of robot (car like robot: $\mathbb{R} \times \mathbb{R} \times \mathbb{S}^1$) circle

- Groups: Set $\mathcal{G}$ together with binary operation $\circ$ defined on elements, if:
  - closure: if $g_1, g_2 \in \mathcal{G}$ then $g_1 \circ g_2 \in \mathcal{G}$
  - associativity: $g_1 \circ (g_2 \circ g_3) = (g_1 \circ g_2) \circ g_3 = g_1 \circ g_2 \circ g_3$
  - identity: $\exists\, e \in \mathcal{G}$ s.t. $g \circ e = e \circ g = g$
  - inverse: $\exists\, g^{-1}$ s.t. $g \circ g^{-1} = g^{-1} \circ g = e$

$\Rightarrow$ non-singular $n \times n$ matrices with matrix multiplication $\mathcal{GL}(n)$

Rotations: $SO(n) \subset O(n) \subset \mathcal{GL}(n)$

Rigid body motions: $G = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix}$ with $R \in SO(n)$, $p \in \mathbb{R}^n \Rightarrow SE(n)$

twist (const.) — body-frame twist

$G(t) = \exp(\eta t) \cdot G(0)$ $\quad G(t) = G(0) \cdot \exp(\xi t)$

skew-sym. matrix

↳ $= \begin{bmatrix} \hat{\omega}_B & v_B \\ 0 & 0 \end{bmatrix} = G^{-1} \eta G$

Rodriguez' Formula:

$e^{\omega^\times} = e^{\theta u^\times} = \mathbb{I} + \sin\theta\, u^\times + (1 - \cos\theta)(u^\times)^2$ rotation by angle $\theta = \|\omega\|$ about u axis

$\Rightarrow R = e^{\omega^\times_{yaw}} \cdot e^{\omega^\times_{pitch}} \cdot e^{\omega^\times_{roll}}$, $\omega_{yaw} = \begin{pmatrix} 0 \\ 0 \\ \psi \end{pmatrix}$, $\omega_{pitch} = \begin{pmatrix} 0 \\ \theta \\ 0 \end{pmatrix}$, $\omega_{roll} = \begin{pmatrix} \varphi \\ 0 \\ 0 \end{pmatrix}$

Quaternion: $q = (\cos\frac{\theta}{2}, \sin\frac{\theta}{2} u)$

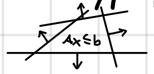
$\omega \neq 0$

$e^{\eta} = \begin{bmatrix} e^{\omega^\times} & Mv \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} e^{\omega^\times} & \frac{1}{\theta}(\mathbb{I} - e^{\omega^\times}) u \times v + u u^T v \\ 0 & 1 \end{bmatrix}$

Obstacles

Convex Polyhedra

represent obstacle as intersection of finite number of half-planes:
$a_r^T x \le b_r$, $r = 1,2,3,\dots$ or matrix $Ax \le b$, $A: r \times n$, r: num. hyperplane



Convex decomposition

non-convex shape represented as union of several convex polyhedra
⇒ obstacle predicate becomes disjunction of several predicates:
$\varphi(x) = \varphi_1(x) \vee \varphi_2(x) \vee \dots \vee \varphi_r(x)$ ($\vee \hat{=}$ OR)

Semi-Algebraic Models

use polynomials: $p(x) \le 0$ ⇒ e.g. $\begin{cases} \text{ellipsoid}: & x^T A x + b^T x + c \le 0 \\ \text{circle at } p: & x^T x - 2p^T x + p^T p - r^2 \le 0 \end{cases}$

Bounding Boxes for Clusters

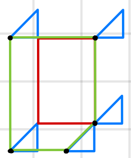
- Axis-aligned bounding box: smallest box containing all points
- Convex hull: smallest convex polyhedra containing all points
  1. draw line between points (min & max coord) ⇒ divide into two sub-groups/planes
  2. for each: - find farthest point from line & connect to previous ends
     - ignore points in triangle
     - divide into sub-groups & repeat

C-Space obstacles

in collision: $\mathcal{X}_{obs} \subset \mathcal{X} := \{x \in \mathcal{X} : \varphi(x) = true\}$
free space: $\mathcal{X}_{free} = cl(\mathcal{X} \setminus \mathcal{X}_{obs})$

Minkowski Sum

$\mathcal{A} \oplus \mathcal{B} := \{a + b : a \in \mathcal{A}, b \in \mathcal{B}\}$ $\quad \mathcal{O}_{conf} = \mathcal{O} \oplus (-\mathcal{R})$



Collision Detection and Obstacle Distance

check if distance between primitives is less than given value ε
⇔ check collision with inflated primitive

Axis Aligned Boxes intersect only if all corresponding intervals intersect

Spheres distance is distance between centers minus two radii

Polyhedra Feasibility LP: two polyhedra are disjoint only if the set $\begin{bmatrix} A_1 \\ A_2 \end{bmatrix} x \le \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$ is empty

Separating Axis Theorem: disjoint if $\exists$ line onto which the two sets projections are disjoint ⇒ at least 1 face normal is separating axis

Bounding Volume Hierarchies

trees of primitives that bound all of their children. The leafs are detailed primitives comparing the objects. Children of a node are only opened if node's bounding box is in collision

Virtual Potentials

Stabilization Problem

sys. $\dot{x} = f(x(t), u(t))$, find policy $\pi: x \mapsto u$ s.t. $u(t) = \pi(x(t))$ for each $\varepsilon > 0$ there is $\delta > 0$ s.t.:
- stab. in sense Lyapunov: $\|x(0)\| < \delta \Rightarrow \|x(t)\| < \varepsilon, \forall t \ge 0$
- asympt. stab.: stable i.s.L. and $\lim_{t \to \infty} x(t) = 0$
- exp. stab.: stable i.s.L. and $\exists\, \alpha, \beta > 0$ s.t. $\|x(t)\| < \beta e^{-\alpha t}$ $\forall t \ge 0$

Control Lyapunov Function

function $V: \mathcal{X} \to \mathbb{R}_{\ge 0}$ with following properties:
- V is pos. def.: $V(x) > 0$ $\forall x \ne$ and $V(0) = 0$
- V is radially unbounded: $\lim_{\|x\| \to \infty} V(x) = +\infty$
- $\forall x \in \mathcal{X}, x \ne 0$ $\exists\, u \in \mathcal{U}$ s.t.: $\dot{V}(x,u) = \frac{\partial V(x)}{\partial x} f(x,u) < 0$, CLF

⇒ V can be interpreted as potential function
⇒ stabilizable sys. $\dot{x} = Ax + Bu$, $V(x) = x^T P x$, P pos.def. sol. Riccati eqn.
$A^T P + PA - PBR^{-1}B^T P + Q = 0$, Q, R given pos. def. matrices ⇒ LQR

"Bug" Algorithms

Bug 0: follow obstacle CW or CCW until cleared not complete
Bug 1: go around obstacle, leave it at min. potential point > complete
Bug 2: draw line from start to goal, follow obstacle CW or CCW until line is reached again

Barrier Potential

$V(x) = x^T x + \frac{1}{k} \sum_i b_i(x)$, $b_i(x) = \begin{cases} 1/\varphi^p(x) & \text{OR} \\ -\log\left(\frac{\varphi(x)}{\varphi_{max}}\right) \end{cases}$
↳ not complete due to loc. minima

Koditschek-Rimon Navigation Function V

if:
- smooth (or at least twice differentiable) on $\mathcal{X}_{free}$
- attains bounded max. val. (1) on boundaries of $\mathcal{X}_{free}$
- attains bounded min. val. (0) at goal state
- has full-rank Hessian at all critical points where $\frac{\partial V}{\partial x} = 0$

Sphere World with spherical disjoint holes:
$\mathcal{X} = \{x \in \mathbb{R}^n : |x| \le r_0\}$ and $\mathcal{X}_{obs} = \bigcup_{i=1}^{n_{obs}} \{x \in \mathcal{X} : |x - c_i| \le r_i\}$ where $|c_i - c_j| > r_i + r_j$ for any $i \ne j$

⇒ $V(x) = \left(\frac{|x - x_{goal}|^{2k}}{|x - x_{goal}|^{2k} + (r_0^2 - |x|^2) \cdot \prod_{i=1}^{n_{obs}} (|x - c_i|^2 - r_i^2)} \right)^{1/k}$ for large enough $k \in \mathbb{N}$ on $\mathcal{X}_{free} = \mathcal{X} \setminus \mathcal{X}_{obs}$

Harmonic Functions

$\nabla^2 V(x) = \sum_{i=1}^{n} \frac{\partial^2 V(x)}{\partial x_i^2} = 0$

Hamilton-Jacobi-Bellman

$\min_{u \in \mathcal{U}} \left\{ g(x,u) + \frac{\partial V^*(x)}{\partial x} f(x,u) \right\} = 0$

Fast-Marching Methods

SPP: $\dot{x} = f(x,u) = u$, $|u| = 1$, $g(x,u) = 1$ optimal u^* anti-parallel to $\nabla V^*(x)$
HJB ⇒ Eikonal equation: $|\nabla V^*(x)| = 1$ ⇒ FMM: $\mathcal{O}(n \log n)$

Interior Point Methods

convex problems: $\min f_0(x)$ s.t. $f_i(x) \le 0$ $i = 1, \dots, m$; $Ax = b$

if $\lambda f(x_1) + (1-\lambda) f(x_2) \ge f(\lambda x_1 + (1-\lambda) x_2)$ for any $x_1, x_2 \in \mathbb{R}$ and $\lambda \in [0,1]$

⇒ solve
barrier functions: $\min f_0(x) - \frac{1}{t} \sum_{i=1}^{m} \log(-f_i(x))$
penalty functions: $\min f_0(x) + t \sum_{i=1}^{m} \max\{0, f_i(x)\}$
s.t. $Ax = b$

iteratively for $t = \mu^k t_0$, $t_0 > 0$, $\mu > 1$ using gradient descent or Newton

Motion Planning as Optimization

$\min_{x,u} \int_0^{t_f} g(x(t), u(t))\, dt$ s.t. $\dot{x}(t) = f(x(t), u(t))$, $\varphi(x) \le 0$ $\forall t \in [0, t_f]$ | $x(0) = x_0$, $x(t_f) = x_f$, $u(t) \in \mathcal{U}$ $\forall t \in [0, t_f]$

numerical discretization:
$\min_{x,u} \sum_{i=0}^{n} g_d(x_i, u_i)$ s.t. $\begin{cases} x_{i+1} = f_d(x_i, u_i) \\ \varphi(x_i) \le 0, \forall i = 1, \dots, n \\ x_n = x_f \\ u_i \in \mathcal{U}, \forall i = 1, \dots, n \end{cases}$

Mixed Integer Linear Programming

"OR" collision avoidance constraints as "AND" with large number M:
$a_1 x[i] \ge b_1 - M(1 - z_1[i])$, AND
$\vdots$
$a_p x[i] \ge b_p - M(1 - z_p[i])$
with $z_1[i] + \dots + z_p[i] = 1$, AND
$z_1[i], \dots, z_p[i] \in \{0,1\}$
$\forall i \in \{0, \dots, m\}$

general form:

MILP
$\min_x c^T x + d^T z$
s.t. $Ax + Bz \le b$
$x \ge 0$
$z \in \{0,1\}^{N_z}$

both exponential time to solve

BIP
$\min_x d^T z$
s.t. $Bz \le b$
$z \in \{0,1\}^{N_z}$

Sequential Convex Programming

1. initial guess of solution
2. convexify constraints (among "OR" constraints pick only least unfeasible)
3. stay within convex trust region
4. solve convex subproblem (using penalty or barrier functions)
5. iterate from 2. until convergence

Branch and Bound Algorithm

1. solve LP relaxation of MILP, call $\bar{J}$ optimal cost
2. if LP solution is integral, terminate, $\bar{J}$ is optimal
   if LP solution is infeasible, terminate, problem is not feasible
   else set $J^L \leftarrow \bar{J}$, $J^U \leftarrow \infty$
3. pick one of the z vars and create 2 subproblems: $\begin{cases} z = 0 \\ z = 1 \end{cases}$
4. for each sub-problem: solve LP-relaxation, call $\bar{J}$ optimal cost
5. if LP solution is integral, then it is candidate opt. sol.
   update: $J^U \leftarrow \min\{J^U, \bar{J}\}$
6. else if LP sol. is not integral but $\bar{J} > J^U$ then prune the branch
7. else if LP sol. is not integral but $\bar{J} < J^U$ then continue branching
8. if LP is infeasible prune the branch

Discretization

Cell Abstractions

1. Partition C-Space into openly disjoint cells

Delaunay Triangulation of a set of points P is a collection of openly disjoint triangles that cover the convex hull of P and s.t. no point in P is inside circumcircle of any triangle in DT
↳ edges of DT of points in 2D are edges of convex hull of projections of points to a paraboloid $(x, y, x^2 + z^2)$

2. design discrete transition system on the cells
3. solve motion planning problems (e.g. SPP) as problems on graph describing the discrete transition system
4. activate in each cell the control policy that generates des. behavior

Motion Primitive-Based Methods

Symmetry: group G that acts freely on $\mathcal{X}$ through map $\Psi: G \times \mathcal{X} \to \mathcal{X}$ is symmetry group if its dynamics are invariant w.r.t. action of G
$\Psi(g, \varphi_\pi(x_0, t)) = \varphi_\pi(\Psi(g, x_0), t)$ $\forall g \in G$

Motion Primitive: finite-time trajectory $\mu : [0,T] \to \mathcal{X} \times \mathcal{U}$

- are sequentially combinable if $\exists g \in G$ s.t. $\mu_1(T_1) = \Psi_g(\mu_2(0))$
- concatenation is motion primitive: $\mu_1\mu_2 = \begin{cases} \mu_1(t) & \text{if } t \leq T_1 \\ \Psi_g(\mu_2(t-T_1)) & \text{else} \end{cases}$
- repeatable if sequentially combined with itself
- strongly repeatable if: repeatable and control inputs continuous

$\Rightarrow$ invariance of cost and footprint

Lattice: given set of generator vectors $v_1, .., v_m \in \mathbb{R}^n$, a lattice is a subgroup of $(\mathbb{R}^n, +)$ given by all lin. combinations of generators (integer coefficients)

Visibility Roadmaps

reflex vertex = vertex of convex hull of obstacle

$\Rightarrow$ visibility roadmap: join each reflex vertex with all other visible reflex vertices.
↳ if line segment joining them does not intersect the interior of an obstacle

↑ contains shortest paths

Voronoi Diagram for max. clearance roadmaps

given set of generator points $P = \{p_1, .., p_n\}$ a VD is a partition of space into n regions $R_1, .., R_n$ such that
$R_i = \{x : |x - p_i| \leq |x - p_j|, i \neq j\}$

Sampling

Probabilistic RoadMaps (multi-query)

Offline pre-processing:
1. sample n points from $\mathcal{X}_{free} = [0,1]^d \setminus \mathcal{X}_{obs}$
2. try to connect points using fast local planner (ignore obstacles) (steering)
3. if connection is successful (= no collision), add an edge between points

at run-time:
1. connect start and goal to closest nodes in roadmap
2. find shortest path on roadmap

- Simplified PRM: attempted connections to all vertices within distance r
- Real PRM: attempted connections in increasing order of distance, only to other connected components in the graph

Rapidly-exploring Random Trees (single query)

build online tree, exploring region of state space from initial condition
at each step: sample one point from $\mathcal{X}_{free}$ and try to connect it to closest vertex in tree (nearest neighbor)

Voronoi bias: more "isolated" vertices of RRT have larger Voronoi regions and are more likely to be extended

Lp metrics $\quad g(x, x') = \left(\sum_{i=1}^{n} |x_i - x_i'|^p \right)^{1/p}$

Manhattan: $p = 1$ | Euclidean: $p = 2$ | Infinity: $p = \infty$: $g_\infty(x, x') = \max_{i=1}^{n} \{|x_i - x_i'|\}$

Dense Sequences $\Rightarrow$ e.g. s_i = rand(1)

dense if:
- for any open set $\mathcal{O} \subseteq \mathcal{X}$ $\exists \bar{n}$ s.t. S_n contains at least one point in $\mathcal{O}$ $\forall n \geq \bar{n}$
- $\lim_{n\to\infty} \Pr[\mathcal{O} \cap S_n = \emptyset] = 0$ (stochastic) • $cl(S_\infty) = \mathcal{X}$

van der Corput: sequence of 16 pts. as dense as possible on $[0,1]$
1. write numbers $0, 1/16, .., 15/16$ in binary notation
2. flip binary figures left/right

Haar measure: satisfies following:
assume $(\mathcal{X}, \circ)$ group if $x\mathcal{A} = \{x \circ a : a \in \mathcal{A}\}$ and $\mathcal{A}x = \{a \circ x : a \in \mathcal{A}\}$
then: $\Pr[s_i \in \mathcal{A}] = \Pr[s_i \in x\mathcal{A}] = \Pr[s_i \in \mathcal{A}x] \quad \forall x \in \mathcal{X}$

$\Rightarrow$ invariance w.r.t. group transformations $\Leftrightarrow$ uniformity of group

Low-dispersion: minimize biggest empty ball: $\delta_g(P) = \sup_{x \in \mathcal{X}} \left\{ \min_{p \in P} \{g(x,p)\} \right\}$

Sukharev grid: if $\mathcal{X} = [0,1]^d$ then opt. L_∞-dispersion for k pts.
1. divide each axis into $\lfloor k^{1/d} \rfloor$ intervals
2. place point at each of these $\lfloor k^{1/d} \rfloor^d$ cubes
3. remaining pts (if $\lfloor k^{1/d} \rfloor$ not int.) can be placed anywhere

$\delta_\infty(P) \geq \frac{1}{2\lfloor k^{1/d} \rfloor}$

Low-discrepancy: $D(P, \mathcal{R}) = \sup_{R \in \mathcal{R}} \left\{ \frac{\text{card}(P \cap R)}{\text{card}(P)} - \frac{\text{Vol}(R)}{\text{Vol}(\mathcal{X})} \right\}$

$\Rightarrow$ low disc. implies low disp. for $P \subset [0,1]^d$: $\delta_\infty(P) \leq D(P, \mathcal{R})^{1/d}$

Halton: $h_i = (s_i^{(b_1)}, .., s_i^{(b_d)})$ on $[0,1]^d$ co-primes (e.g. first d primes)

with $s_i^{(b)} = \sum_{k=0}^{\infty} d_k(n) \cdot b^{-k-1}$ where $\sum_{k=0}^{\infty} d_k(n)\, b^k = n$
↳ base $b \in \mathbb{N}$

Nearest-Neighbor Search

k-d tree construction:
- each node of tree associated to point and hyperplane splitting remaining into 2
- points recursively split into 2 along direction of axis, cycling at each level of tree
- pick median point in split direction

NN-search
- proceed down tree as if x was new point to add
- when leaf node reached: distance (squared) computed and stored as current nearest
- sibling searched if dist. to split smaller than curr. nearest
- else: search moves up tree, updating curr. nearest at parent nodes as appropriate

$\to$ inserting new point: $\mathcal{O}(\log n)$
$\to$ NN query: $\mathcal{O}(\log n)$
$\Rightarrow$ NN-search drives asympt. complexity when collision checking via safety certificates

Random Geometric Graphs $G^{rand}(n, r)$ in d dimensions

↳ n vertices indep., uniformly distributed random vars in $(0,1)^d$

Connectivity: $\lim_{n\to\infty} \mathbb{P}(G^{rand} \text{ connected}) = \begin{cases} 1, & \text{if } \zeta_d \cdot r^d > \log(n)/n \\ 0, & \text{if } \zeta_d \cdot r^d < \log(n)/n \end{cases}$

ζ_d: Volume unit ball

Probabilistic Completeness

if algorithm with feasible motion planning problem $\mathcal{P} = (\mathcal{X}_{free}, x_{init}, \mathcal{X}_{goal})$
$\lim_{n\to\infty} \Pr(\text{ALG returns solution to } \mathcal{P}) = 1$

$\Rightarrow$ robust if sol. still holds if obstacles dilated by δ

Asymptotic optimality $\quad \mathbb{P}\left(\left\{ \lim_{i\to\infty} Y_i^{ALG} = c^* \right\}\right) = 1$

with cost function c that admits robust optimal solution with finite cost c^*

PRM Algorithm

```
V ← ∅ ; E ← ∅
for all i = 0,...,n do
  x_rand ← SampleFree
  U ← Near(G=(V,E), x_rand, r)
  V ← V ∪ {x_rand}
  for all u ∈ U, in order of increasing ||u - x_rand|| do
    if x_rand and u not in same connected component of G=(V,E) then
      if CollisionFree(x_rand, u) then
        E ← E ∪ {(x_rand, u), (u, x_rand)}
return G = (V,E)
```

sPRM Algorithm

```
V ← {x_init} ∪ {SampleFree_i}_{i=1,..,N-1}
E ← ∅
for all v ∈ V do
  U ← Near(G=(V,E), v, r) \ {v}
  for all u ∈ U do
    if CollisionFree(v,u) then
      E ← E ∪ {(v,u), (u,v)}
return G = (V,E)
```

↳ probabilistic complete

PRM*

```
V ← {x_init} ∪ {SampleFree_i}_{i=1,..,n}
E ← ∅
r ← γ_PRM (log(n)/n)^{1/d}
for all v ∈ V do
  U ← Near(G=(V,E), v, r) \ {v}
  for all u ∈ U do
    if CollisionFree(u,v) then
      E ← E ∪ {(v,u), (u,v)}
return G = (V,E)
```

↳ $\gamma > 2\left(\frac{\mu_{free}}{\zeta_d}\right)^{1/d}\left(1 + \frac{1}{d}\right)^{1/d}$

RRT Algorithm

```
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_new)
  if ObstacleFree(x_nearest, x_new) then
  *  V ← V ∪ {x_new}
  *  E ← E ∪ {(x_nearest, x_new)}
return G = (V,E)
```

Rapidly-exploring Random Graphs (RRGs)

add
```
* X_near ← Near(G=(V,E), x_new, min{γ_RRG (log(card V)/card V)^{1/d}, η})
* for all x_near ∈ X_near do
    if CollisionFree(x_near, x_new) then
      E ← E ∪ {(x_near, x_new), (x_new, x_near)}
```

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_n., x_r.)
  if ObstacleFree(x_nearest, x_new) then
    X_near ← Near(G=(V,E), x_new, min{γ_RRG (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))
    for all x_near ∈ X_near do
      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)}
    for all x_near ∈ X_near do
      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)
```