Plánování pohybu – postupy založené na vzorkování (obsah)

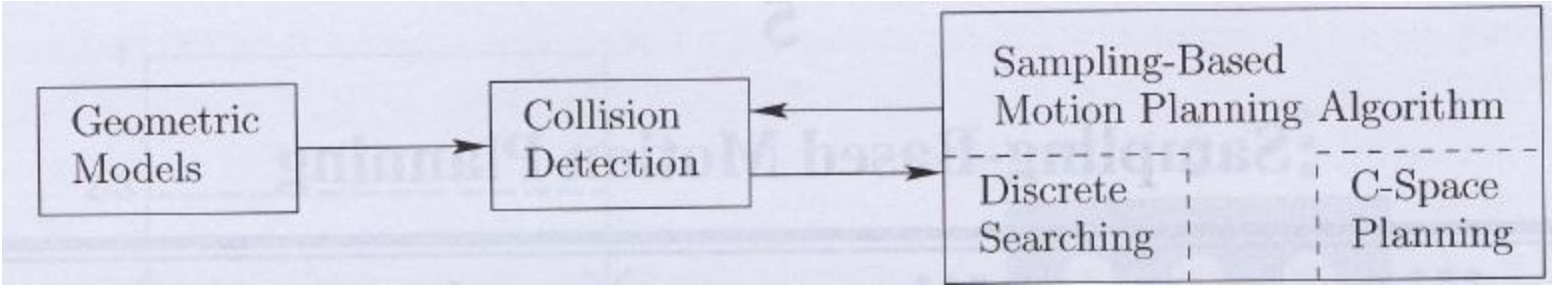
- Plánování - proč metody založené na vzorkování? Filosofie postupu.
- Vlastnosti vzorkovacích postupů pro plánování, srovnání s kombinatorickými postupy, souvislost s konfiguračním prostorem (*C-space*).
- Vzdálenost, metrika, pseudometrika a „hustota“ vzorkovaného prostoru
- Druhy vzorkování, hodncení náhodnosti výběru vzorku
- Detekce kolizí, hierarchické a inkrementální postupy
- Inkrementální plánování a hledání plánů
- Souvislost s diskrétním plánováním

- Metoda cest užitím vzorkování
- Náhodná procházka, randomizovaný potenciál
- Metoda rychle rostoucích stromů, princip a varianty (*RDT a RRT*).

- Reference

Proč vzorkovací postupy (Sampling-Based) plánování?

- Hlavní motivací je obejít nezbytnost rekonstrukce překážek v prostředí (reprezentováno $\mathcal{C}_{obst}$) a nahradit jí vhodným vzorkovacím mechanismem (který následně poskytne lokální potřebnou informaci pro plánování)
  - Vzorkovací postupy nevžadují pro svoji činnost nezbytně existenci úplného modelu prostředí (ale mohou při její znalosti pracovat dobře)
  - Způsob odběru vzorků z konfiguračního prostoru (*C-space*) de facto realizuje funkcionalitu „předcházení srážkám“, čímž metodě poskytuje potřebnou informaci k rozhodování o dalším vytváření plánu



Vzorkovací postupy plánování používají collision avoidance jako „černou skřinku“ oddělující vlastní plánování od daného geometrického a kinematického modelu situace. Na základě získaných vzorků je prováděno diskrétní plánování.

Vzorkovací postupy plánování – vlastnosti I

- Všechny algoritmy sampling-based plánování jsou *neúplné,* tj. negarantují vlastnost, že existující řešení úlohy je nalezeno v konečném čase (což je naopak vlastnost kombinatorických postupů)
- Nicméně, přijatelného výsledku může být splněno nalezením *přibližného řešení* - to umožňuje existence vlastnosti tzv. „hustoty vzorků“ (*denseness*), jejíž význam je, že jednotlivé vzorky se neomezeně blíží k libovolné konfiguraci, jestliže počet vzorků $\rightarrow\infty$.
- Při deterministickém pohledu (systematické prohledávání) a dostatečně hustém vzorkování (densely complete) tedy nastávají alternativy:
  1. Existuje-li přijatelné řešení, je nalezeno v konečném čase.
  2. Při neexistujícím řešení i přes provádění „hustého“ vzorkování algoritmus nikdy nezkončí

- Předchozí implikuje tzv. *pravděpodobmostně úplné (*probalistically complete) algoritmy, což značí, že při dostatečném počtu vzorků je pravděpodobnost, že bude nalezeno (přijatelné) řešení, se blíží $\rightarrow$1.
- Problémem ale zůstává rychlost konvergence, jejíž zajištění je obtížné... Proto, čím „lépe“ dokážu vzorkovat, tím rychlejší dosáhnu konvergence a méně vzorků musím zpracovat k dosažení přijatelného řešení

Vzorkovací postupy plánování – vlastnosti II

- Sampling-based algoritmy mohou být tzv. s jedním- nebo více- požadavky.
  - Jeden požadavek je reprezentován zadáním párem *(start, cíl),* přičemž se očekává, že algoritmus bude pracovat až do dosažení cíle (nebo ohlásí chybu)
  - Postupy schopné zpracovat více požadavků, tj. dodat najednou řešení pro více párů *(start, cíl),* investují značné úsilí do fáze předzpracování, tj, vytváření datových struktur, schopných následně jednoduchým výpočtem dodat vicero řešení (při předpokladu neměnných omezení pracovního prostoru - překážek).

Vzdálenost a metrika I

- Veškeré sampling-based postupy vyžadují funkci, která měří vzdálenost mezi dvěma body prostoru → metriku

Def.: metrický prostor (X, *ρ)* s metrikou (vzdáleností) *ρ: X* x *X* → *R* splňující pro libovolná *a,b,c* ∈ *X* následující axiomy:

1. *ρ(a,b)* ≥ *0* (nezáporná vzdálenost)
2. *ρ(a,b)* = *0* tehdy a jen tehdy, když *a=b* (vzdálenost od sebe samého je nulová, reflexivita)
3. *ρ(a,b)* = *ρ(b,a)* (symetrie)
4. *ρ(a,b)*+ *ρ(b,c)* ≥ *ρ(a,c)* (trojúhelníková nerovnost)

- Nejčastější tzv. L_P-metriky na R^n mají metriku: $\rho(a,b) = \left(\sum_{i=1}^{n} |a_i - b_i|^p \right)^{1/p}$

1. L_1 metrika – tzv. Městská, Manhattan, ortogonální.... (pravoúhlá vzdálenost, součet rozdílů souřadnic)
2. L_2 metrika – Euklidovská metrika (Euklidovská vzdálenost v R^n)
3. L_∞ metrika – Maximum metrika $L_\infty(a,b) = \max_{1 \le i \le n} \{|a_i - b_i|\}$

Vzdálenost a metrika II

- Další metriky pro plánování pohybu v robotice:

  1. Metrika komplexních čísel je vhodná pro libovolny C-space, jenž je podmnožinou R^2 , takovou, že:

     $$\{(a,b) \in R^2; a^2 + b^2 = 1\} \qquad \rho(a_1, b_1, a_2, b_2) = \sqrt{(a_1 - a_2)^2 + (b_1 - b_2)^2}$$

  2. Předchozí metrika neuvažuje vydálenost podél kružnic, což může být žádoucí → *metrika úhlů*, prostým uvažováním úhlů Θ_1 a Θ_2 (s ohledem na dosažitelnost Θ_2 z Θ_1 dvěma možnými směry)

     $$\rho(\Theta_1, \Theta_2) = \min\{|\Theta_1 - \Theta_2|, 2\Pi - |\Theta_1 - \Theta_2|\}$$

     Nebo v komplexním oboru (vyjádření bodu je: *a+jb*):

     $$\rho(a_1, b_1, a_2, b_2) = \cos^{-1}(a_1 a_2 + b_1 b_2)$$

- Pseudometrika – konstrukce hodnotící funkce / funkce „vzdálenosti“ je taková, že nezajišťuje splnění všech axiomů 1.-4. ale přesto je schopná udat směr pohybu robotu k cíli.

  Příklady:

  - Vyhodnocování *úspěšnosti* na základě spotřebované energie nemusí splnit podmínku symetrie (při kinematických omezeních nelze couvat)
  - *Potenciální funkce* (viz použití potenciálního pole k plánování, odpudivé složky potenciálu mohou generovat lokální extrémy, a porušují typicky podmínku trojúhelníkové nerovnosti)

Vzorkování prostoru I – hustota vzorkování

- Náhodné vzorkování/posloupnosti jsou „pravděpodobnostně husté“, čímž garantuje podmínku použitelnosti pro plánovací postupy:
  - Tedy pro pro interval délky *e* je při provádění náhodného výběru *k* vzorků přes celý prostor je pravděpodobnost, že žádný ze vzorků nepadne do tohoto intervalu je $(1-e)^k$.
  - Jestliže $k \to \infty$, jde hodnota $(1-e)^k \to 0$, což značí že interval *e* za těchto podmínek pravděpodobně obsahuje aspoň jeden vzorek, tedy taková nekonečná posloupnost náhodně a nezávisle vybraných vzorků je „hustá s pravděpodobností =1 “ (což nemá význam garance; pravděpodobnost výběru určitých konkrétnch vzorků je sice nulová ale přesto je jejich výběr možný)

- Náhodné vzorkování – nejjednodušší postup, je velmi vhodné pro C-space neboť lze vzorky vybírat „po souřadnicích“, tj, nezávisle pro jednotlivé dimenze prostoru a přesto dosahuje rovnoměrně náhodné rozložení pokrytí prostoru (daleko složitější je pokrytí kartézského prostoru pro deterministické metody)
- Generování (pseudo)náhodných čísel. Počítačové implementace nejsou korektně plně náhodné!

  Klasický lineární kongruenční generátor (pro *a, M nesoudělná a c takové, že: 0 ≤ c < M, M pokud možno »1*)

$$y_{i+1} = a y_i + c \mod M$$

  Kde pseudonáhodná čísla x_i z intervalu [0,1] lze obdržet jako: $x_i = y_i / M$ a jsou periodická.

Vzorkování prostoru II - jak testovat/hodnotit míru náhodnosti vzorkování?

- Počítačem realizované vzorkování je přes svoji „náhodnost“ do jisté míry deterministické → potřeba hodnocení uniformity/náhodnosti odebraných vzorků.

- Nejjednodušším je *Chi-kvadrát* test, který ověřuje, jak vzdálená je vypočtená statistika od očekávané.

  Příklad:

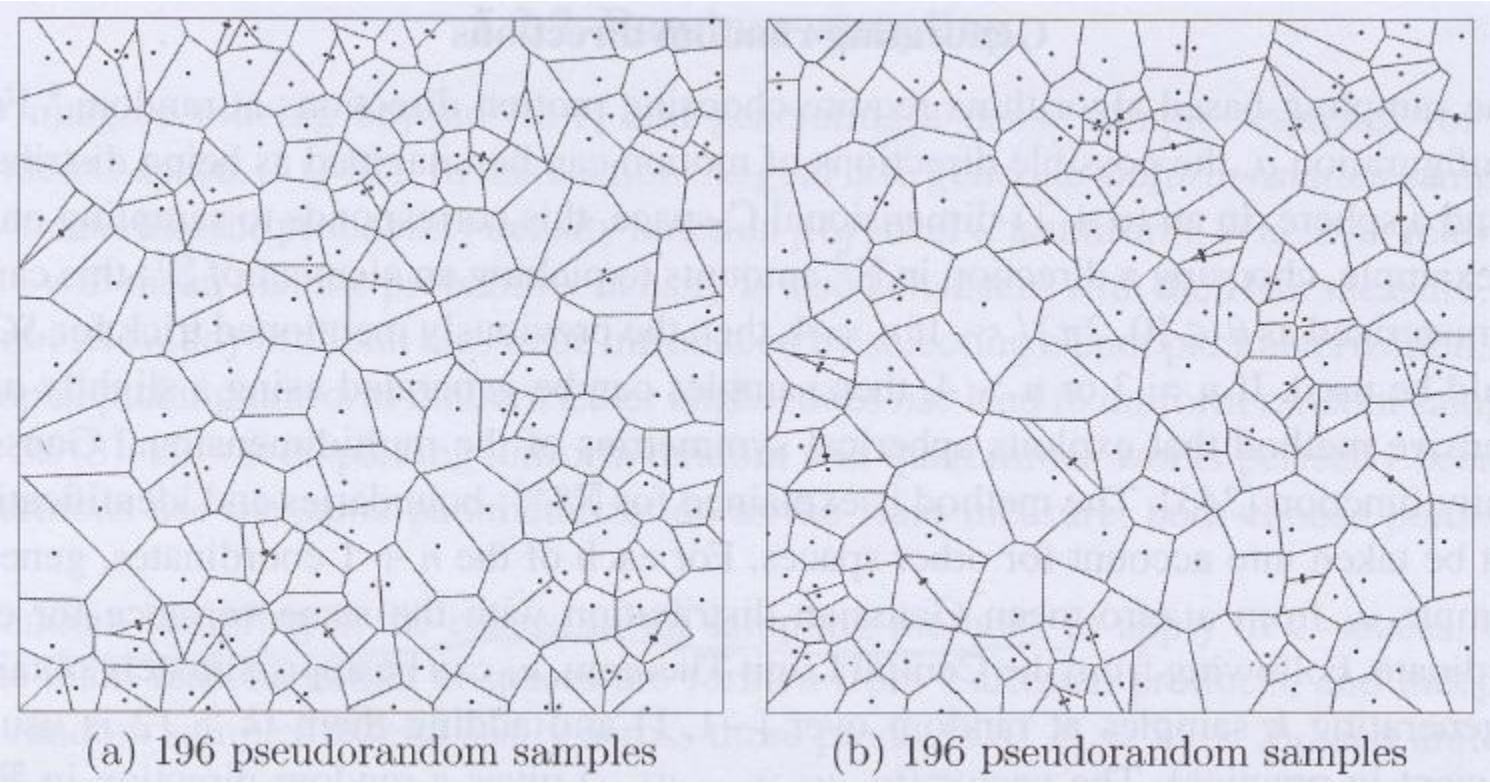
  Pracovní prostor nechť je diskretizován na *n* x *n* pixelů. Je-li množina *P* reprezentující *k* náhodně vybraných vzorků, každý pixel by intuitivně měl obsahovat k/n^2 vzorků. To umožní zavést chybovou funkci, reprezentující míru s jakou je předchozí předpoklad správný:

$$e(P) = \sum_{i=1}^{n^2} \left(b_i - k / n^2 \right)^2$$

  Kde *b* značí počet vzorků v pixelu *i* (chybová funkce *e(P)* reprezentuje Chi-kvadrát rozložení náhodné proměnné). *Hodnota e(P) = 0* má význam dokonalé uniformity rozdělení vzorků, tj. nejsou náhodné. Obecně hodnota musí být *e(P) » 0* (statist. tabulky).

- Hodnocení užitím Voroného diagramů – každý vzorek má přiřazenou Voroného oblast *Vor(x).* Pro každý bod $y \in Vor(x)$, je *x* nejbližším vzorkem k *y* při použití Euklidovské metriky. Hodnotí se diverzita velikosti a tvaru Voroného oblastí.

Vzorkování prostoru III



Příklad sad 196ti pseudonáodných vzorků, pro zviditelnění náhodnosti jsou doplněny Voroného oblasti.

Vzorkování prostoru IV – pseudonáhodné vzorkováni

- Odběr vzorků podle Van der Corputovy posloupnosti
  Založeno na principu binárního kódování délky intervalu a jeho následné modifikaci půlením binární reprezentace a prohozením (swap) bitů významově nižší a vyšší části. Má vlastnost, že každá otevřená podmnožina (z hlediska kvantizace intervalu binárním vyjádřením, obsahuje aspoň jeden vzorek).

i	Naive Sequence	Binary	Reverse Binary	Van der Corput	Points in $[0,1]/\sim$
1	0	.0000	.0000	0	
2	1/16	.0001	.1000	1/2	
3	1/8	.0010	.0100	1/4	
4	3/16	.0011	.1100	3/4	
5	1/4	.0100	.0010	1/8	
6	5/16	.0101	.1010	5/8	
7	3/8	.0110	.0110	3/8	
8	7/16	.0111	.1110	7/8	
9	1/2	.1000	.0001	1/16	
10	9/16	.1001	.1001	9/16	
11	5/8	.1010	.0101	5/16	
12	11/16	.1011	.1101	13/16	
13	3/4	.1100	.0011	3/16	
14	13/16	.1101	.1011	11/16	
15	7/8	.1110	.0111	7/16	
16	15/16	.1111	.1111	15/16	

Detekce kolizí

- Detekce kolizí je klíčovou součástí plánování s využitím vzorkování

- Určením místění vzorku je třeba rozhodnout, zda-li vzorek koliduje s prostředím a zda-li je možné jej dále využít pro vytvářený plán.
- Detekce kolizí je často řešena jako „černá skřínka“ a postačí, že poskytuje správná hodnocení kolizích situací (neboť nemá přímou vazbu k plánovacímu postupu). Nicméně její výpočetní náročnost může být velmi vysoká a ovlivní tak negativně činnost plánovače.
- Existuje množství postupů pro detekci kolizí (exaktní a heuristické)...
- Nejčastěji používanými jsou postupy založené na ověřování platnosti vhodné podmínky vzhledem k $\mathcal{C}_{obst}$... (kterou nejčastěji bývá test na přítomnost daného vzorku v modelu překážek). Výstupem je rozhodnutí, je-li vzorek (konfigurace) kolizní, či nikoliv (logická hodnota)
- V případě 2D světa s konvexními překážkami a robotem je odpovídající algoritmus lineárně náročný. Nicméně, pokud lze, vždy je výpočetně nejjednodušší určit, zda-li je konfigurace kolizní bez nutnosti úplné konstrukce $\mathcal{C}_{obst}$ (model překážek/prostředí).

Detekce kolizí II

- Určování kolizí využívá výpočet *vzdálenosti d mezi dvěma množinami,* $d: C \rightarrow \langle 0, \infty]$, jenž odpovídá nejmenší vzdálenosti v existujících párech bodů *e* a *f* mezi danými množinami E a F:

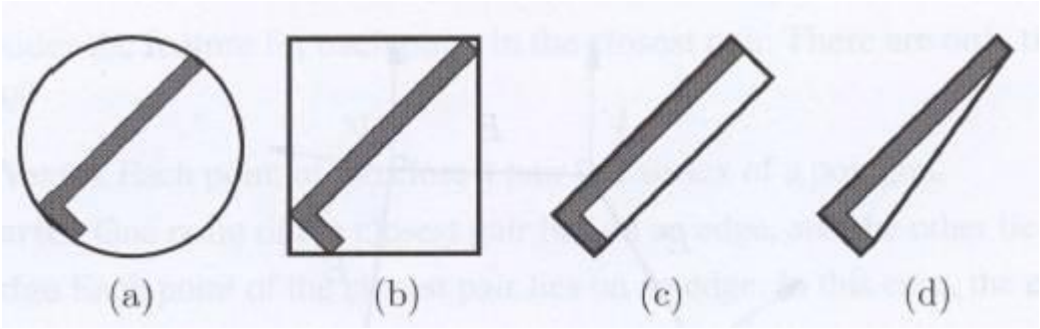
$$\rho(E,F) = \min_{e \in E} \{ \min_{f \in F} \{ \| e - f \| \} \}$$

kde $\| \cdot \|$ značí Euklidovskou normu a současně platí: $E \cap F = \varnothing \Rightarrow \rho(E,F) = 0$

- Ke zjednodušení výpočtu a úspoře výpočetního času se často detekce kolizí provádí ve 2 stupních, přibližně (na velkou vzdálenost) a přesně (v blízkém okolí):
  - **Přibližná detekce kolizí** – vyšetření vzájemné polohy (konvexních) obalů objektů a robotu, krajních bodů jim opsaných oken, atd. plus využití hashování ke snížení počtu možných kolizních kombinací
  - **Blízká detekce kolizí** – provádí se detailně na úrovni jednotlivých bodů objektů a robotu – hierarchické a inkrementální postupy - viz dále.

Hierarchické postupy I

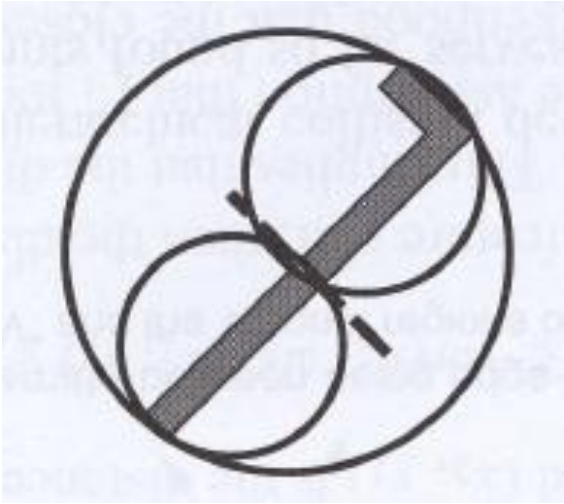
- Vhodné pro určování vzájemné kolize nebodových objektů (překážka v prostředí a robot).
- Postup provede dekompozici každého objektu na jeho základní primitiva (např. trojúhelníkovou dekompozicí) a uspořádá ji do *stromové struktury*, kde:
  - Každý vrchol ve stromu se váže na oblast ohraničující jeho část (podmnožinu).
  - Kořenový vrchol reprezentuje celý objekt
- Volba způsobu dekompozice je řízena 2-ma protichůdnými požadavky:
  - Ohraničující oblast přiléhá k objektu co možná nejtěsněji
  - Postup k testování průniku takových dvojic oblastí musí být co nejjednodušší



Různé druhy ohraničujících oblastí: (a)kruhová sféra, (b) ohraničující pravoúhelník podle souřadných os, (c) orientovaný pravoúhelník, (d) konvexní obal

Hierarchické postupy II

- Konstrukce stromu probíhá od shora dolů, tj. ohraničující oblast je vždy dělena na dceřiné oblasti o přibližně shodné velikosti (ploše). Jestliže model objektu je složen z primitiv (např. trojúhelníků) je dělení prováděno k dosažení shodného počtu primitiv v každém potomkovi.
- Dělení probíhá až k dosažení elementárních tvarů částí objektu, které je jednoduché testovat na kolize.



Kruhová sféra definuje vrchol, popisující celý objekt. Po rozdělelní (přerušovaná čára), 2 menší kruhy reprezentují popis jednotlivých polovin (2 vrcholy na nižší úrovni)

Hierarchické postupy III

- Postup detekce kolize ve stromě:
  1. Mějme objekty E a F s jim odpovídajícími stromy T_e a T_f, jejichž možnou kolizi zkoumáme
  2. Jestliže kořenové vrcholy T_e a T_f nekolidují, oba objekty E a F **nejsou v kolizní situaci** → **konec**.
  3. Jestliže kořenové vrcholy T_e a T_f kolidují, pak ohraničující oblasti všech potomků T_e jsou porovnány s oblastí T_f.
  4. Jestliže žádné z předchozích v bodě 3 nekolidují, ohraničující oblast T_f je nahrazena všemi oblastmi jeho bezprostředních potomků.
  5. Postup rekurzivně pokračuje na bod 2, pokud nebylo dosaženo koncových vrcholů (listů stromu), jinak se oba původní **objekty nalézají v kolizní situaci** → **konec**.

Pozn1.: Pokud byla dekompozice provedena na primitiva (trojúhelníky), pak je testování na kolizi prováděno vzájemně mezi jednotlivými primitivy.

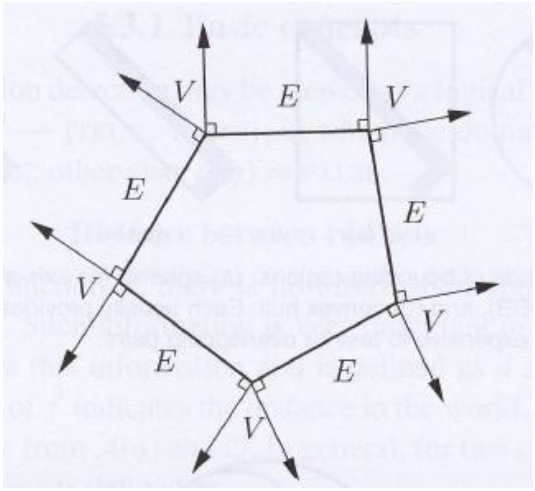
Pozn2.: Doplnění algoritmu o výpočet skutečné vzdálenosti ohraničujících oblastí umožní další prořezávání stromu → zrychlení výpočtu.

Inkrementální postupy I

- *Postup incremental distance computation* – předpokládá, že mezi dvěma požadavky na detekci kolizí nedojde k podstatné změně scény (objekty se přemístí jen málo).
- ***Výhoda:*** Předchozí předpoklad umožní dosáhnout téměř konstatního času výpočtu pro objekty typu konvexní mnohostěny. Nekonvexní objekty lze vždy dekomponovat na konvexní.
- ***Nevýhoda:*** Modely objektů/robotů musí být koherentní, tj. všechna jejich primitiva musí bezvadně navazovat, objekty musí být uzavřené... (nepřipouští se výskyt izolovaných segmentů hranic/stěn, těles s chybějící stranou/stěnou pro 2D, resp. 3D situace, model objektů nesmí být prostá sada primitiv) → náročné na předzpracování dat.

Inkrementální postupy II

- Princip detekce kolizí výpočtem na základě vzájemných relací popisů (příznaků) objektů (2D situace)
  - Každý objekt - polygon s *n* vrcholy je popsán *2n* příznaky (vrcholy plus strany), jimž odpovídají Voroného oblasti – viz obr.



  - Pro každý pár objektů s potenciálním výskytem kolize nastávají výhradně tyto kombinace:
    - *vrchol-vrchol*, kdy oba body z nejbližšího páru jsou vrcholy polygonu
    - *hrana-vrchol*, jeden z nejbližších bodů leží na hraně a druhý je ve vrcholu
    - *hrana-hrana*, oba nejbližší body leží na hranách (hrany jsou paralelní)

    Pro něž je možné provést snadno výpočet vzdálenosti obou objeků (množin)

Inkrementální vzorkování a hledání plánu I

- Algoritmy s unikátním zadáním (single query) mají dánu jedinou dvojici start, cíl: (q_I, q_G) pro každý robot a sadu překážek (model)

  → není třeba činit předběžné výpočty struktur pro plánování s více cíli

  → plánování pohybu lze pojmout jako úlohu *prohledávání s následujícími odlišnostmi, kdy:*

  - Realizace "akce" jej nahrazena generováním „segmentu cesty" (viz. bod 3 následujíccího alg.)
  - Prohledávaný graf je neorientovaný s hranami reprezentujícími cesty (oproti orientovanému grafu s hranami reprezentujícími akce)

- Základní postup – algoritmus plánování cesty s unikátním zadáním

  1. **Inicilalizace.** Mějme *G(V,E)* reprezentující neorientovaný graf, který je tvořen nejméně jedním vrcholem *V,* přičemž *E* neobsahuje žádnou hranu. Typicky, *V* obsahuje *start* a/nebo *cíl* a případně další body volného prostoru $\mathcal{C}_{free}$.
  2. **Výběr vrcholu.** (Vertex Selection Method, VSM) Zvolte k expanzi vrchol q_{cur}
  3. **Lokální plánování.** (Local Planning Method, LPM) Pro vhodný nový q_{new}, který nemusí nutně být z existující množiny vrcholů *V, se pokuste nalézt cestu z* q_{cur} *do* q_{new} takovou, aby nezpůsobila kolizi. Pokud tento krok selže, při hledání kolizeprosté cesty, jdi na krok 2. Jinak pokračuj.

Inkrementální vzorkování a hledání plánu II

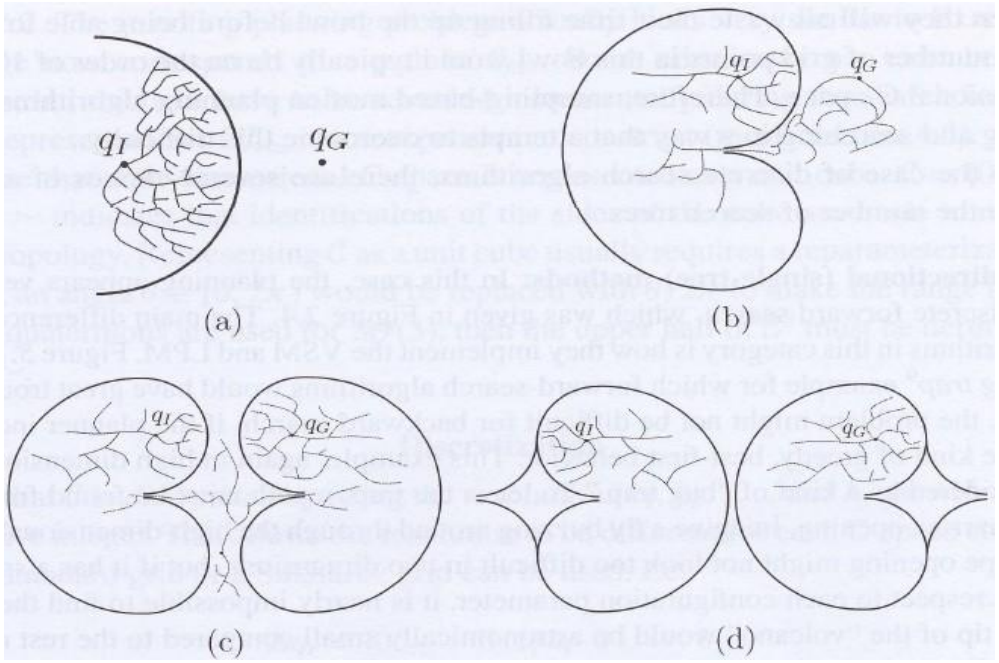
- Základní postup – algoritmus plánování cesty s unikátním zadáním (pokračování)

  4. **Vložení hrany do grafu.** *Pokud je q_{new} dosud není prvkem E,* vložte do množiny *E* v předchozím bodě nalezenou cestu (úsek cesty) *jako hranu q_{cur} do q_{new}.*
  5. **Bylo dosaženo řešení?** Ověřte, zda-li graf *G* již reprezentuje výslednou požadovanou cestu, řešení? (Krok je jednoduchý v diskrétních případech, kdy existuje jediný prohledávací strom, jinak může být jeho řešení složité a výpočetně náročné)
  6. **Návrat na krok 2.** Iterujte předchozí dokud není dosaženo řešení nebo není splněna vhodná ukončující podmínka (algorimus může v určitých situacích pracovat neomezeně, viz. předchozí)

Pozn. Výše užitý graf G je topologickým grafem, někdy též označovaným jako *graf cest (roadmap).*

Inkrementální vzorkování a hledání plánu III

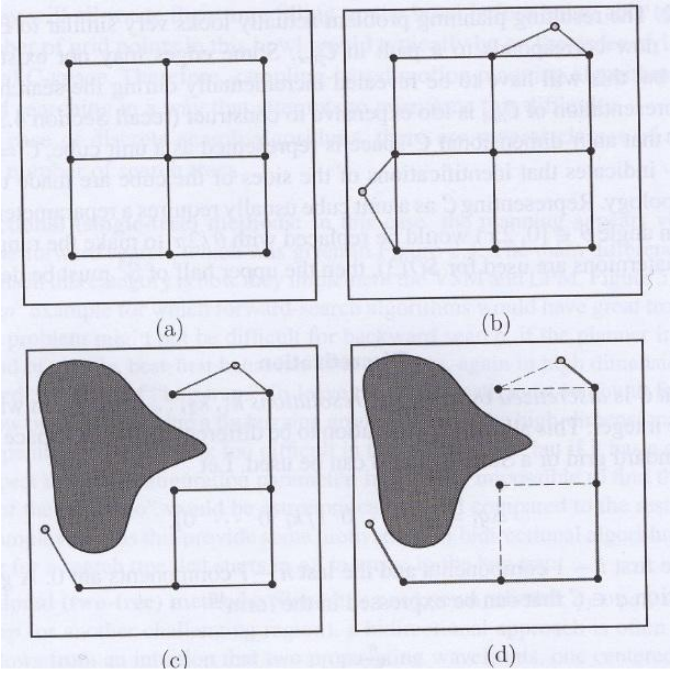
- V případech nepříznivé konfigurace překážek existuje možnost uvíznutí metody („bug-trap“ situace postup obtížně umisťuje vzorky do některých lokalit → vede ke zpomalení řešení, či úplnému uvíznutí)
- K řešení může být využita kombinace odlišných postupů prohledávání (LPM):
  - Jednosměrné prohledávání – expanze stromu hledání od počátku k cíli.
  - Dvousměrné prohledávání – narůstání 2 stromů prohledávání proti sobě, od počátku i cíle (většinou řeší bug-trap situace)
  - Mnohosměrové prohledávání – narůstání více prohledávacích stromů z různých lokalit scény (většinou řeší dougle bug-trap konfigurace)



Obtížné situace pro sampling-based plánovací postupy: (a) přednostní prohledávání prostoru v dutině, (b) bug-trap konfigurace s obtížným průnikem ven z oblasti, (c) dvojitá bug-trap vyžaduje mnohosměrové prohledávání s počátkem i mimo oblasti, (d) téměř neřešitelná situace...

Souvislost s diskrétním určováním plánu

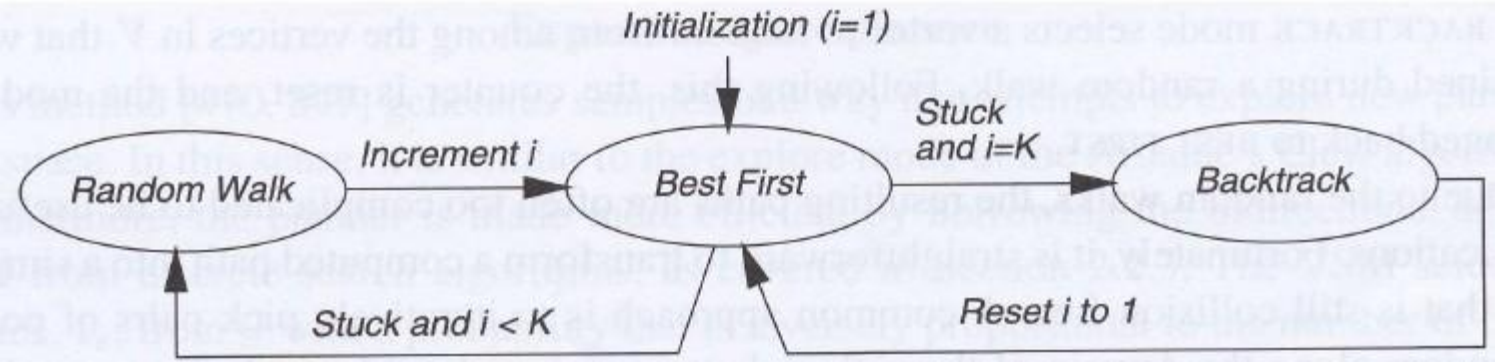
- Plánování s užitím vzorkovacích postupů je možné kombinovat s postupy pro diskrétní hledání plánů.
  - → Přináší značné zrychlení výpočtu plánu apriorním omezením prohledávacího prostoru, např. definováním mřížky nad konfiguračním prostorem $\mathcal{C}$, resp. příslušným topologickým grafem - grafem cest.
  - → Uzlové body je nutné vhodně zvolit s ohledem na pokrytí prostoru.
  - → Nad redukovaným prostorem lze spustit vzorkovací postup, který nalézá řešení v několika málo krocích.



Topologický graf (graf cest) může být zkonstruhován během prohledávání prostoru. Vrcholů, v nichž navazují cesty je nižší počet, což umožní nalézt řešení výběrem z mála vzorků poměrně rychleji.

Náhodná procházka a randomizovaný potenciál

- Srovnání náhodných a systematických postupů:
  - Náhodné postupy (random walk) strádají v případech výskytu nevhodných struktur v prostředí (obtížnost nalezení průchodu - viz předchozí „nálevkovité objekty“)
  - Systematické (neinformované) postupy prohledávání stavového prostoru jsou výpočetně náročné (např. 10 dimenzí, 50 vzorků/dimenze → 50^{10} alternativ)
  - Obtížná implementace informovaných postupů → použití pseudometriky (potenciálové pole) je obtížné k výpočtu a/nebo negarantuje neexistenci dalších extrémů (jiných, než cíle).
- Předchozí nedostatky lze obejít kombinací předchozího – informovaného náhodného prohledávání, kritériem přepínání postupů je „stav uvíznutí“ v lokálním extrému a/nebo počet provedených kroků jedním postupem, viz.:

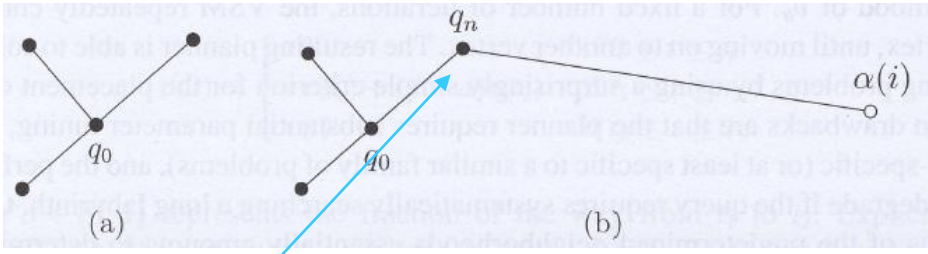


Rapidly Exploring Dense Trees (RDT) I

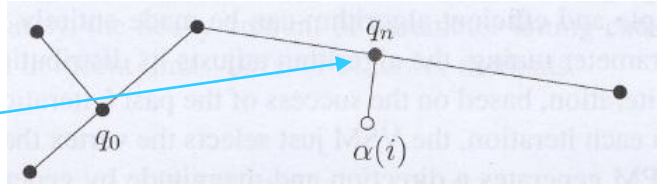
- Inkrementální vzorkovací/plánovací postup, který dosahuje dobrých výsledků bez nutnosti nastavování jakýchkoliv parametrů.
  - **Idea:** Postupná konstrukce prohledávacího stromu, který průběžně zvyšuje rozlišení (a nevyžaduje nastavování jakýchkoliv parametrů, které se vztahují k řízení rozlišení). Ve výsledku je chopen postupně pokrýt celý pracovní prostor „hustě“.
  - Konstrukce hustého stromu je deterministická nebo náhodná (RDT), spec. případem jsou *Rapidly Exploring Random Trees (RRT).*

Algoritmus konstrukce hustého stromu:

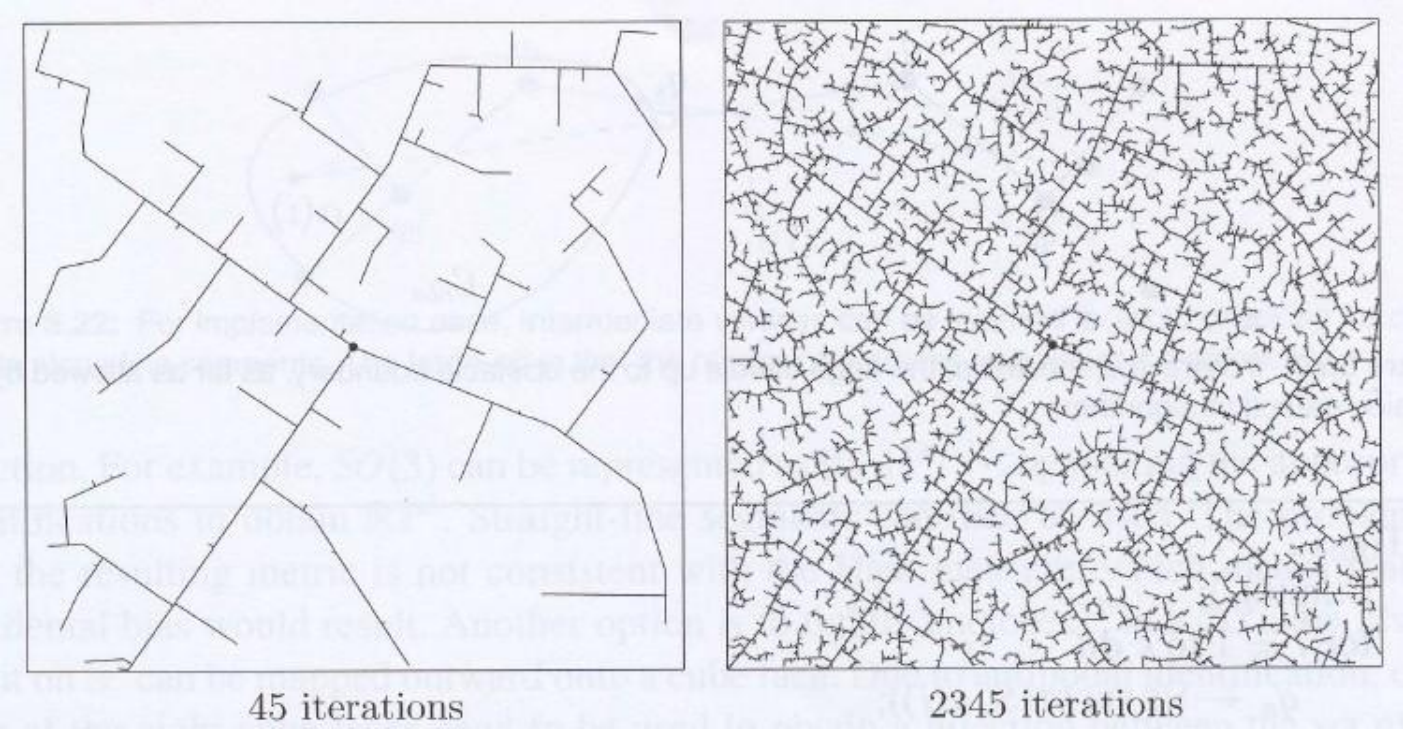
```
SIMPLE_RDT(q_0)
1  G.init(q_0);
2  for i = 1 to k do
3      G.add_vertex(α(i));
4      q_n ← NEAREST(S(G), α(i));
5      G.add_edge(q_n, α(i));
```



Funkce *NEAREST* určuje nejbližší vrchol dosud existujícího stromu *S(G)* k nově vygenerovanému vrcholu *α(i)*; s každým novým vrcholem vznikne ve stromě 1 *(nejbližším je vrchol)* nebo 2 *(nejbližším je hrana → půlení existující hrany)* nové hrany.



Rapidly Exploring Dense Trees (RDT) II



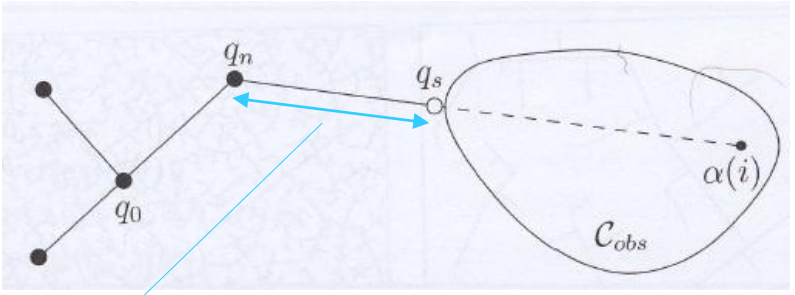
Chování RDT: v počátečních iteracích rychle dosahuje dosud nenavštívené oblasti, následně pak zpřesňuje pokrytí (v limitním případě zajistí husté pokrytí s pravděpodobností 1)

Rapidly Exploring Dense Trees (RDT) III

- Reprezentace C_{obst} v RDT je řešena ve fázi generování stromu → definicí koncové podmínky na „dosažení nejbližšího bodu u hranice objektu“ ve směru generovaného vrcholu *α(i),* nejbližší bod q_n je definován shodně (bez ohledu na existenci překážky) a jemu příšlušná nová hrana je pouze do q_s, *viz. obr.* :

Algoritmus RDT pro situaci s překážkami:

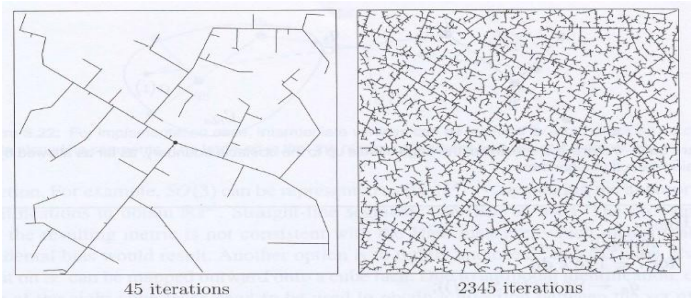
```
RDT(q_0)
1  G.init(q_0);
2  for i = 1 to k do
3      q_n ← NEAREST(S, α(i));
4      q_s ← STOPPING-CONFIGURATION(q_n, α(i));
5      if q_s ≠ q_n then
6          G.add_vertex(q_s);
7          G.add_edge(q_n, q_s);
```



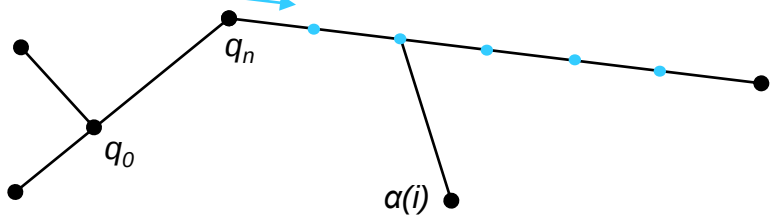
- Nejmenší přípustná vzdálenost bodu q_s od skutečné hranice překážky je dána užitým algoritmem pro collision avoidance; v některých případech malé vzdálenosti q_n od překážky nemusí být hrana q_n q_s vůbec generována.

Rapidly Exploring Dense Trees (RDT) IV

- Realizace funkce *NEAREST* (hledání nejbližšího bodu ve stromě) – dva alternativní postupy:
  - **Exaktní řešení:** Založeno na hierarchické přístupu. Výpočet vzdálenosti nového vrcholu *α(i)* od větví stromu, získaných v ranných fázích jeho generování (hlavní větve) poskytne přibližnou infomaci, která část stromu nese kandidáty na nejbližší bod. Výpočet postupy vektorového počtu je lineárně náročný.
  - Následně je hodnota *NEAREST* iterativně zpřeňována až k libovolné požadované přesnosti.



  - **Přibližné řešení:** Je založeno na převzorkování stromu. Do jednotlivých hran stromu jsou vloženy další přídavné vrcholy tak, že každá po sobě jdoucí dvojice vrcholů není vzdálena navzájem ne více než zvolené *Δq*. Ostatní vnitřní body hran jsou zanedbány a výpočet vzdálenosti je proveden mezi *α(i)* a všemi vrcholy stromu.
  - Přesnost výpočtu je dána zvolenou vzdáleností vrchlolů.



Reference:

- LaValle, S. M.: Planning Algorithms, Cambridge University Press, U.S.A, 2006, 826 pp. ISBN 0-521-86205-1