National Conference with International Participation
ENGINEERING MECHANICS 2006
Svratka, Czech Republic, May 15 – 18, 2006
paper no. 112

MOBILE ROBOT PATH PLANNING BY MEANS OF CASE-BASED REASONING AND RAPIDLY-EXPLORING RANDOM TREES

P. Krček*, J. Dvořák*

Summary: *The aim of the robot path planning is searching for a path from a robot start configuration to a goal configuration without collisions with known obstacles minimizing weight of the path. We consider a nonholonomic robot moving in a two-dimensional continuous space with known polygonal obstacles. When planning a path by Case-Based Reasoning, the most similar cases (already used paths or their parts) are searched for to be subsequently adapted to the new problem. Rapidly-exploring Random Trees are used to find the missing parts of the constructed paths or new paths if similar cases are not found or adapted solutions are not good enough.*

1. Úvod

Úkolem plánování cesty robotu je nalezení cesty z počáteční do cílové pozice bez kolize se známými překážkami tak, aby ohodnocení cesty bylo minimální. Ohodnocení cesty je určeno hlavně délkou cesty a může zahrnovat i další aspekty, jako např. obtížnost a riziko cesty. Prostředí robotu se uvažuje buď jako spojité, nebo se nějakým způsobem diskretizuje (např. pomocí mřížky). Jestliže jsou na pohyb robotu kladena nějaká dodatečná omezení (vyjma těch která jsou spojena se způsobem modelování prostředí), pak je robot charakterizován přívlastkem *neholonomický* (*nonholonomic*). V tomto článku je zkoumáno plánování cesty autonomního neholonomického mobilního robotu ve dvourozměrném spojitém prostoru.

V případě dynamického nebo částečně známého prostředí je důležitou schopností robotu schopnost učit se na základě zkoumání okolního prostředí. Tato schopnost je často zajišťována použitím neuronových sítí, posilovaného učení či evolučních algoritmů. Strojové učení může být také realizováno použitím případového usuzování. Případové usuzování řeší nový problém adaptací známých řešení podobných problémů, které byly již řešeny v minulosti. Zdá se, že případové usuzování je pro navigaci robotu vhodnou metodou, neboť v řadě aplikací robot často opakovaně řeší podobné úkoly. Základní principy případového usuzování jsou popsány např. v (Aamodt & Plaza 1994).

Případové usuzování pro plánování dráhy robotu ve spojitém prostředí použili Haigh & Shewchuk (1994). V uvedené práci bázi případů reprezentuje *případový graf*. Při ukládání cesty do báze případů je cesta uložena buďto celá jako jeden případ, nebo je rozdělena do několika částí, které jsou uloženy jako samostatné případy (pokud se ovšem ještě tyto případy nebo případy jim podobné v bázi případů nevyskytují). Segmenty cest (uložené případy) mohou mít společné pouze své krajní body. Pokud cesta protíná nějaký již existující případ,

* Ing. Petr Krček, RNDr. Jiří Dvořák, CSc.: Ústav automatizace a informatiky, Fakulta strojního inženýrství VUT v Brně; Technická 2896/2; 616 69 Brno; tel.: +420 541 142 435; e-mail: ykrcek00@stud.fme.vutbr.cz

pak tato cesta i případ jsou rozděleny do menších případů tak, aby byla splněna výše uvedená podmínka. Případy (segmenty) reprezentují hrany výsledného grafu a uzly tohoto grafu jsou krajními body těchto segmentů.

Případové usuzování pro plánování cest robotu se neobejde bez některé další metody hledání cest. Použití jiné metody je nezbytné v situacích, kdy systém případového usuzování začíná pracovat s prázdnou bází případů, nebo když získané řešení není dost dobré a musí být přepracováno. Přehled metod plánování cesty mobilního robotu je možno najít např. v práci Meyer & Filliat (2003) a v knize LaValle (2006). Pro plánování cesty ve spojitém prostředí lze použít např. metodu *potenciálů* (*potential field method*, Agirrebeitia, J. et al. 2005), metodu *pravděpodobnostní cestovní mapy* (*probabilistic roadmap*, Geraerts & Overmars 2006), algoritmy *rychle mapujících náhodných stromů* (*rapidly-exploring random trees*, LaValle & Kuffner 2001, Krejsa & Věchet 2005), metody založené na *grafech viditelnosti* (Priya & Sridharan 2006) a *Voronoiových diagramech* (Kobayashi & Sugihara 2002, Šeda 2005), *genetické algoritmy* (Homaifar et al. 2001) a *simulované žíhání* (Martínez-Alfaro & Gómez-García 1998). Haigh & Shewchuk (1994) kombinovali případový graf s Delaunyho triangulací. V tomto příspěvku je uvažována kombinace případového usuzování s algoritmy náhodných stromů. Navržená metoda vychází z práce (Dvořák & Krček 2005), kde bylo pro plánování cesty ve dvourozměrné mřížce použito případové usuzování ve spolupráci s Dijkstrovým algoritmem a algoritmem A^*.

2. Robot a okolní prostředí

Při plánování cesty robotu se často rozlišuje *pracovní prostor* (prostředí) W, v němž se robot pohybuje, a *konfigurační prostor* X tvořený *konfiguracemi* robotu. Konfigurace $\mathbf{x}$ představuje n-rozměrný vektor, jehož složky jednoznačně určují pozici a stav robotu v pracovním prostředí.

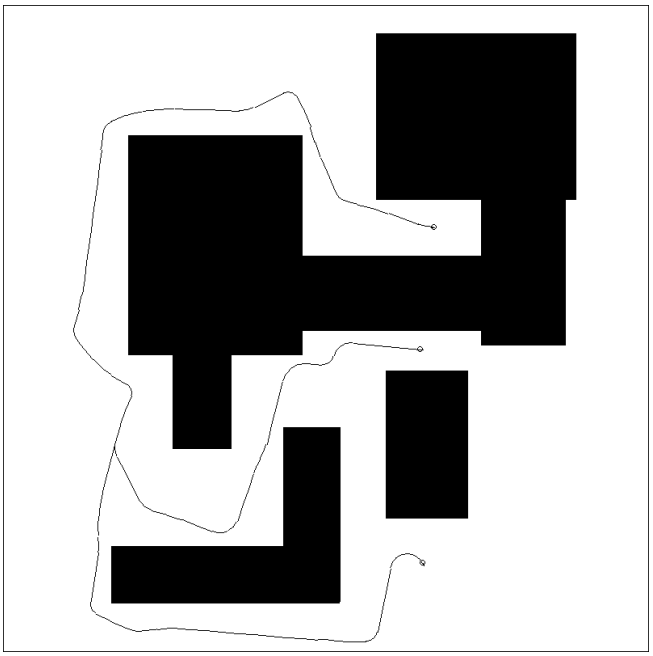
V tomto příspěvku uvažujeme 2D spojité pracovní prostředí robotu s polygonálními překážkami. V souvislosti s překážkami se zavádí pojem *volného konfiguračního prostoru* X_{free}, který je podmnožinou konfiguračního prostoru a obsahuje pouze takové konfigurace robotu, které nejsou v kolizi s žádnou překážkou. Podobně jako v (LaValle & Kuffner 2001) používáme funkci detekce kolize $D: X \rightarrow \{\text{pravda}, \text{nepravda}\}$, která určuje zda konfigurace $\mathbf{x} \notin X_{free}$.

Dále uvažujeme mobilní robot typu „neholonomická tříkolka s diferenciálním řízením“, který se může pohybovat pouze směrem dopředu a je pro něj dán minimální poloměr otáčení. Dimenze konfiguračního prostoru $n = 3$ a složkami konfigurace jsou souřadnice polohy geometrického středu robotu ve 2D pracovním prostředí a úhel natočení robotu.

3. Algoritmus RRT

Klasický algoritmus RRT (*Rapidly-exploring Random Trees*), popsaný např. v (LaValle & Kuffner 2001), modifikujeme tak, že umožňuje vyhledat cesty z počáteční konfigurace $\mathbf{x}_s \in X_{free}$ nejen do jedné, ale do více cílových konfigurací obsažených v množině $Q \subset X_{free}$. Tato úprava má za cíl nalezení cest z dané konfigurace do blízkých uzlů případového grafu. Součástí množiny Q může být i původně zadaná cílová konfigurace $\mathbf{x}_g$. Je nutno ale poznamenat, že zadaný počet iterací K by měl být při hledání cest do blízkých uzlů

případového grafu (a eventuálně konfigurace $\mathbf{x}_g$) podstatně nižší, než když hledáme cestu pouze do konfigurace $\mathbf{x}_g$.



Obr. 1. Cesty do dvou cílů vyhledané algoritmem RRT

Modifikovaný algoritmus RRT sestává z následujících kroků:

1. Jako kořen stromu T zvolíme $\mathbf{x}_s$ Potom v K iteracích opakujeme kroky 2 až 8.
2. Jako konfiguraci $\mathbf{x}_{rand}$ zvolíme se zadanou pravděpodobností p náhodně vygenerovanou konfiguraci z X nebo s pravděpodobností $1 - p$ náhodně vybranou konfiguraci z množiny cílů Q.
3. Jako konfiguraci $\mathbf{x}_{near}$ zvolíme jednu konfiguraci z T takovou, pro kterou je vzdálenost mezi $\mathbf{x}_{near}$ a $\mathbf{x}_{rand}$ minimální a pro kterou mezi konfiguracemi $\mathbf{x}_{near}$ a $\mathbf{x}_{rand}$ existuje triviální hrana. Vzdálenost počítáme jako euklidovskou vzdálenost mezi polohami středu robotu. Triviální hrana se skládá nejvýše ze dvou úseček a nejvýše z jednoho kruhového oblouku o velikosti úhlu menším než π a poloměru větším než minimální poloměr otáčení robotu (poloměr oblouku volíme co největší).
4. Nebyla-li konfigurace $\mathbf{x}_{near}$ v kroku 3 nalezena, pak jdeme na krok 2. Pokud $\mathbf{x}_{rand} \in Q$, pak jdeme na krok 7.
5. Novou konfiguraci $\mathbf{x}_{new}$ určíme jako $\mathbf{x}_{new} = \Phi(\mathbf{x}_{near}, \mathbf{x}_{rand}, \varepsilon)$. Výsledkem funkce Φ je konfigurace ležící na triviální hraně mezi $\mathbf{x}_{near}$ a $\mathbf{x}_{rand}$ taková, že délka úseku hrany mezi $\mathbf{x}_{near}$ a $\mathbf{x}_{new}$ je rovna ε. Pokud délka triviální hrany je menší než ε, pokračuje se krokem 2.
6. Pokud neplatí $D(\mathbf{x})$ pro všechny konfigurace $\mathbf{x}$ na triviální hraně mezi $\mathbf{x}_{near}$ a $\mathbf{x}_{new}$, pak strom T rozšíříme o uzel $\mathbf{x}_{new}$ a o triviální hranu mezi $\mathbf{x}_{near}$ a $\mathbf{x}_{new}$. Pokračujeme krokem 8.

7. Jestliže neplatí $D(\mathbf{x})$ pro všechny $\mathbf{x}$ na triviální hraně mezi $\mathbf{x}_{near}$ a $\mathbf{x}_{rand}$, pak strom T rozšíříme o uzel $\mathbf{x}_{rand}$ a o triviální hranu mezi $\mathbf{x}_{near}$ a $\mathbf{x}_{rand}$. Konfiguraci $\mathbf{x}_{rand}$ vyřadíme z množiny Q.

8. Pokud již proběhlo K iterací nebo je $Q = \varnothing$, pak algoritmus ukončíme, jinak jdeme na krok 2.

Jestliže algoritmus nalezl cestu ze startu do cílové konfigurace, snadno ji nyní získáme procházením stromu T od cílového uzlu směrem ke kořenu. Cíle, do kterých algoritmus cestu nenalezl, představují zbylé prvky v množině Q. Cesta $P(\mathbf{x}_0, \mathbf{x}_k)$ ze startovní konfigurace $\mathbf{x}_0 = \mathbf{x}_s$ do cílové konfigurace $\mathbf{x}_k$ je definována jako orientovaný sled konfigurací a hran $\mathbf{x}_0$, h_1, $\mathbf{x}_1$, h_2, $\mathbf{x}_2$, …, h_k, $\mathbf{x}_k$, kde každá hrana h_i představuje triviální hranu spojující konfigurace $\mathbf{x}_{i-1}$ a $\mathbf{x}_i$. Předpokládáme, že robot může absolvovat tutéž cestu i v opačném směru otočením všech jejích konfigurací o úhel π. Příklad použití modifikovaného algoritmu RRT ukazuje obr. 1.

4. Kombinace případového usuzování a pravděpodobnostních stromů

Báze případů je organizována jako *případový graf* $G = (V, E)$, kde V je množina uzlů, $V = \{0, 1, 2, \ldots, n\}$ a E je množina orientovaných hran $E \subseteq \{(i, j) \mid i, j \in V\}$, přičemž každý případ je reprezentován dvojicí opačně orientovaných hran. Při ukládání úspěšně absolvované cesty robotu do báze případů se jako samostatný případ ukládá každá triviální hrana cesty. K uložení jednotlivých hran ovšem může dojít pouze tehdy, pokud se v bázi případů ještě nevyskytují podobné případy. Uzly případového grafu odpovídají konfiguracím robotu ohraničujícím jednotlivé triviální hrany a jsou ohodnoceny souřadnicemi středu robotu. Pokud hrana v případovém grafu protíná jinou hranu, potom se snažíme propojit tyto hrany (pokud je to možné) doplněním dalších uzlů a triviálních hran.

Každá triviální hrana je v bázi případů uložená s hodnotou funkce $e(i, j) = f(l, r)$, která reprezentuje cenové ohodnocení daného případu. Parametr l je délka hrany a parametr r charakterizuje nebezpečnost (nebo obtížnost) hrany a měl by být upravován během každého skutečného průchodu touto hranou. Dále je hrana charakterizována dvojicí úhlů $\alpha_i(i, j)$ a $\alpha_j(i, j)$, které svírají směrové vektory tečny hrany (i, j) v uzlech i a j s osou x. Směrový vektor tečny v uzlu směřuje dovnitř hrany.

Vyhledávání podobných cest je založeno na okolí konfigurace robotu. Definujme okolí $V(\mathbf{x}_i, \delta)$ konfigurace $\mathbf{x}_i$ v grafu G jako množinu uzlů, jejichž euklidovská vzdálenost od geometrického středu robotu v konfiguraci $\mathbf{x}_i$ je menší než dané δ.

Námi navrhovaný algoritmus pracující nad případovým grafem lze popsat následujícími kroky ($\mathbf{x}_s$ je počáteční konfigurace, $\mathbf{x}_g$ je cílová konfigurace):

1. Určíme okolí konfigurací $\mathbf{x}_s$ a $\mathbf{x}_g$ v případovém grafu G:

$$V_s = V(\mathbf{x}_s, \delta),\ V_g = V(\mathbf{x}_g, \delta)$$

2. Pokud je některá z množin V_s nebo V_g prázdná, cestu z $\mathbf{x}_s$ do $\mathbf{x}_g$ najdeme pomocí RRT a algoritmus ukončíme.

3. Dočasně modifikujeme případový graf G takto: rozšíříme tento graf o uzel korespondující s $\mathbf{x}_s$ a přidáme nekolidující triviální hrany spojující tuto konfiguraci s nejbližšími konfiguracemi vzniklými z uzlů obsažených ve V_s. Z každého uzlu $i \in V_s$ vznikne tolik

konfigurací, kolik je v grafu G hran vycházejících z tohoto uzlu. Natočení robotu v konfiguraci odpovídající uzlu i a hraně (i, j) je rovno úhlu $\alpha_i(i, j)$. Všechny konfigurace z V_s, pro které neexistují nekolidující triviální hrany z $\mathbf{x}_s$, vložíme do množiny Q a do této množiny vložíme i konfiguraci $\mathbf{x}_g$. Nyní spustíme RRT algoritmus se startem $\mathbf{x}_s$ a množinou cílů Q. Všechny takto nalezené cesty přidáme do modifikovaného grafu G'. Všechny přidané hrany jsou ohodnoceny svými délkami.

4. Stejné rozšíření jako v kroku 3 provedeme pro cílovou konfiguraci $\mathbf{x}_g$. Protože předpokládáme možnost absolvování cesty v opačném směru, jako startovací konfiguraci volíme $\mathbf{x}_g$ otočenou o π.
5. V modifikovaném grafu G' použijeme pro hledání optimální cesty spojující $\mathbf{x}_s$ a $\mathbf{x}_g$ upravený Dijkstrův algoritmus., popsaný v následujícím odstavci. Nenašel-li tento algoritmus cestu, vyhledáme ji pomocí RRT a algoritmus ukončíme.

5. Upravený Dijkstrův algoritmus

Následující úprava Dijkstrova algoritmu je nutná z toho důvodu, že při hledání optimální cesty v grafu nemůžeme v důsledku kinematických omezení robotu pokračovat z aktuálního uzlu libovolnou hranou. Volba navazující hrany závisí na tom, po jaké hraně se robot do aktuálního uzlu dostal.

Symbolem $d_s(j, k)$ je označena vzdálenost z uzlu s do uzlu k přes hranu (j, k) v grafu $G = (V, E)$. Jestliže neexistuje cesta z uzlu s do uzlu k přes hranu (j, k), klademe $d_s(j, k) = \infty$. Označme symbolem $p_s(i, j)$ počáteční uzel hrany bezprostředně předcházející hraně (i, j) na nejkratší cestě z uzlu s do uzlu j přes hranu (i, j). Je-li $p_s(i, j) = -1$, znamená to, že cesta z uzlu s do uzlu j přes hranu (i, j) neexistuje. Označme symbolem D množinu hran, o nichž víme, že hodnota $d_s(j, k)$ je již definitivní. Dále značme symbolem s' fiktivního předchůdce uzlu s a položme $\alpha_s(s', s)$ rovno úhlu počátečního natočení robotu zvětšeného o π. Symbolem $E(i)$ označme množinu hran, které incidují s uzlem i.

Výpočet vzdáleností z uzlu s do uzlu g přes všechny jeho sousedy (při dané orientaci robotu v uzlu s) lze popsat následujícími kroky:

1. Polož $d_s(s', s) = 0$ a pro každou hranu (i, j) polož $d_s(i, j) = \infty$, $p_s(i, j) = -1$. Dále polož $D = \varnothing$, $(k, r) = (s', s)$.
2. Prozkoumej všechny hrany $(r, i) \in E - D$, pro něž $\alpha_r(r, i) = \alpha_r(k, r) + \pi$. Jestliže $d_s(r, i) > d_s(k, r) + e(r, i)$, pak polož $d_s(r, i) = d_s(k, r) + e(r, i)$, $p_s(r, i) = k$.
3. Je-li $E(g) \in D$, pak konec (byly nalezeny cesty do cíle přes všechny jeho sousedy). V opačném případě najdi hranu $(k, r) \notin D$ takovou, že $d_s(k, r) = \min\{d_s(i, j) \mid (i, j) \notin D\}$.
4. Je-li $d_s(k, r) = \infty$, pak konec (žádná další hrana již není dostupná z vrcholu s). V opačném případě zařaď hranu (k, r) do množiny D a pokračuj krokem 2.

Předpokládejme, že uzel g je takový, že existuje hrana $(i, g) \in D$. Cesta robotu z konfigurace $\mathbf{x}_s$ do konfigurace $\mathbf{x}_g$ bude určena jako posloupnost konfigurací a hran uložených v zásobníku Z, přičemž počáteční konfigurace se bude nacházet na vrcholu zásobníku. Určení nejkratší cesty z $\mathbf{x}_s$ do $\mathbf{x}_g$ lze popsat takto:

1. Polož $Z = \varnothing$. Urči hranu (j, g) takovou, že $d_s(j, g) = \min\{d_s(i, g) \mid (i, g) \in D\}$. Polož $k = g$.

2. Vlož konfiguraci $\mathbf{x}_k$ určenou uzlem k a úhlem $\alpha_k(j,k)+\pi$ do zásobníku Z.
3. Je-li $k = s$ a $j = s'$, pak konec.
4. Vlož hranu (j, k) do zásobníku Z.
5. Polož $i = p_s(j, k)$, $k = j$, $j = i$ a pokračuj krokem 2.

6. Simulační experimenty

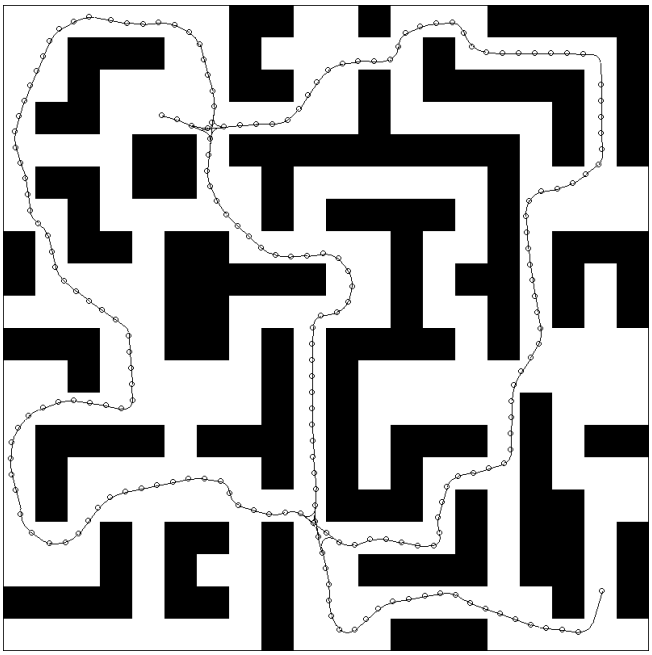
Pro ověření navržených algoritmů bylo vytvořeno simulační prostředí. Experimenty byly provedeny ve scéně 800×800 jednotek se zadanými překážkami. V této scéně byla náhodně vygenerována posloupnost 101 konfigurací taková, že vždy pro následující konfiguraci existuje cesta z předcházející konfigurace. Bylo tedy vyhledáváno 100 cest a pro každé hledání byla měřena doba výpočtu a délka nalezené cesty. Vyhledávání se provádělo jednak algoritmem RRT a jednak pomocí algoritmu kombinujícího případový graf a RRT (CGRRT). Pro experimenty s algoritmem CGRRT byly použity dvě uměle vytvořené báze případů (viz obr. 2 a 3). Tab. 1 obsahuje parametry pro algoritmus RRT. Výsledky spolu s nastavením CGRRT (týká se velikosti okolí δ a použité báze případů) jsou uvedeny v tab. 2. Pro simulace byl uvažován neholonomický mobilní tříkolový robot s diferenciálním řízením zadních kol. Rozměry robotu jsou 10×10 jednotek, robot má povolen pouze dopředný pohyb a jeho minimální poloměr otáčení je 10 jednotek. Výběr minima v upraveném Dijkstrově algoritmu byl implementován pomocí haldy. Experimenty byly prováděny na počítači PC s procesorem AMD Athlon XP1700+ a 768 MB RAM.

Tab. 1. Parametry pro algoritmus RRT

Použití RRT	K	ε [jednotky]	p
pro vyhledávání celých cest	30 000	20	0,95
pro spojení s případovým grafem	2 000	20	0,95

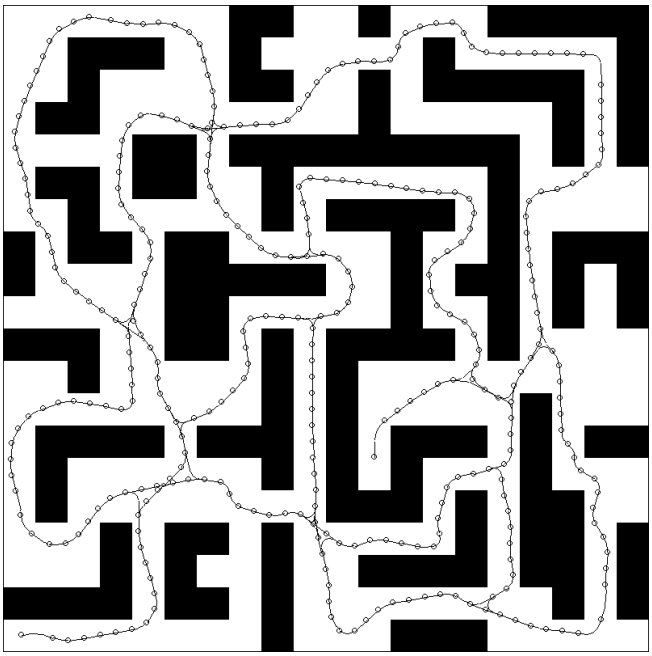
Tab. 2. Naměřené doby výpočtů a nastavení CGRRT

Algoritmus	Báze případů	δ [jednotky]	Průměrná doba výpočtu [s]	Průměrná délka trasy [jednotky]	Neúspěšnost CGRRT
RRT	-	-	6,493	1342,242	-
CGRRT	č. 1	100	3,441	1433,048	27 %
CGRRT	č. 2	100	2,612	1206,721	9 %
CGRRT	č. 2	50	2,678	1379,276	23 %

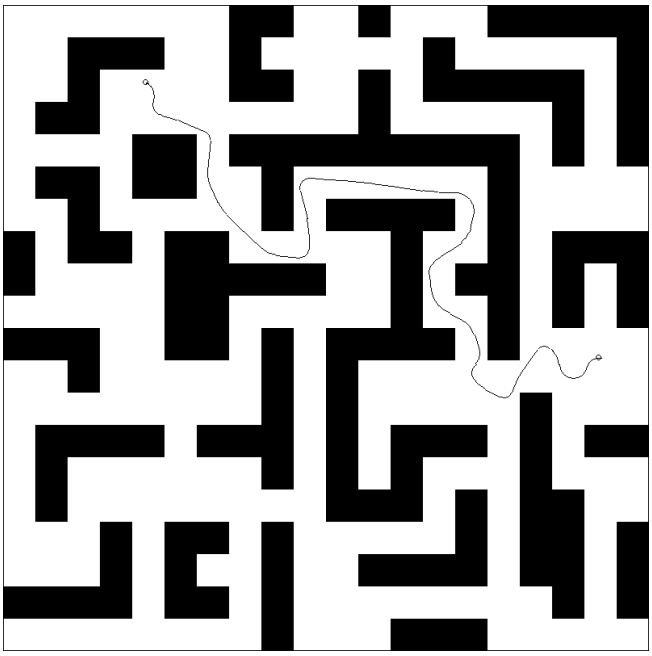


Obr. 2. Použitá případová báze č. 1

Z provedených experimentů vyplývá, že při vhodné volbě okolí a případové bázi dostatečně pokrývající svými případy scénu, lze dosáhnout vůči samostatnému používání RRT jak zrychlení doby výpočtu, tak i vyhledání kratších cest.



Obr. 3. Použitá případová báze č. 2



Obr. 4. Příklad cesty vyhledané pomocí algoritmu CGRRT s bází případů č. 2

7. Závěr

Tento článek se zabývá plánováním cesty autonomního neholonomického mobilního robotu ve dvourozměrném spojitém prostředí se zadanými překážkami polygonálního tvaru. Navrhujeme zde metodu kombinující případové usuzování s hledáním cesty pomocí náhodných stromů. Tato metoda je založena na struktuře zvané případový graf, která je sestavena z částí již dříve nalezených cest. Dosud provedené experimenty naznačují, že použití případového grafu může přinést významnou úsporu ve srovnání se samotným algoritmem RRT. Tento poznatek bude ovšem třeba ověřit rozsáhlejšími experimenty, při nichž bude případový graf postupně vytvářen a bude také zkoumán vliv velikosti tohoto grafu na rychlost hledání cest.

V dalším výzkumu budeme rovněž studovat možnosti adaptace nalezených cest jejich vyrovnáváním a problematiku související s uchováváním případů (volba podobnostní metriky pro doplňování nových případů) a udržováním báze případů (volba kritérií pro vypouštění případů). Bude také zkoumána možnost počátečního naplnění báze případů podmnožinou nějakého stromu nalezeného algoritmem RRT.

8. Poděkování

Tento článek byl zpracován v rámci vědecko-výzkumného záměru MSM 0021630518 "Simulační modelování mechatronických soustav".

9. Literatura

Aamodt, A. & Plaza, E. (1994) Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, No. 5, vol. 7, pp. 39-59.

Agirrebeitia, J., Avilés, R., de Bustos, I. F. & Ajuria, G. (2005) A New APF Strategy for Path Planning in Environments with Obstacles. *Mechanism and Machine Theory*, vol. 40, issue 6, pp. 645-658.

Dvořák, J. & Krček P. (2005) Mobile Robot Path Planning by Means of Case-Based Reasoning *Engineering Mechanics*, vol. 12, no. A1, pp. 219-226.

Geraerts , R. & Overmars, M. H. (2006) Sampling and Node Adding in Probabilistic Roadmap Planners. *Robotics and Autonomous Systems*, vol. 54, issue 2, pp. 165-173.

Haigh, K. & Shewchuk, J. (1994) Geometric Similarity Metrics for Case-Based Reasoning, in: *Case-Based Reasoning: Working Notes from the AAAI-94 Workshop*, Seattle, WA, August, AAAI Press, pp. 182-187.

Homaifar, A., Tunstel, E., Dozier, G. & Battle, D. (2001) Genetic and Evolutionary Methods for Mobile Robot Control and Path Planning. In Ali Zillouchian and Mo Jamshidi (eds.). *Intelligent Control Systems Using Soft Computing Methodologies*, CRC Press LLC, Boca Raton.

Krejsa, J. & Věchet, S. (2005) Rapidly Exploring Random Trees used for Mobile Robots Path Planning. *Engineering Mechanics*, vol. 12, no. 4. pp. 231-237.

Kobayashi, K. & Sugihara, K. (2002) Crystal Voronoi Diagram and its Applications. *Future Generation Computer Systems,* vol. 18, issue 5, pp. 681-692.

LaValle, S.M. (2006) *Planning Algorithms*. Cambridge University Press, http://planning.cs.uiuc.edu/

LaValle, S. M. & Kuffner, J. J. (2001) Rapidly-Exploring Random Trees: Progress and Prospects. In Donald, B. R., Lynch, K. M. & Rus, D. editors, *Algorithmic and Computational Robotics: New Directions*, A K Peters, Wellesley, MA, pp. 293-308.

Martínez-Alfaro, H. & Gómez-García, S. (1998) Mobile Robot Path Planning and Tracking Using Simulated Annealing and Fuzzy Logic Control. *Expert Systems with Applications*, vol. 15, issues 3-4, pp. 421-429.

Meyer, J.-A. & Filliat, D. (2003) Map-Based Navigation in Mobile Robots: II. A Review of Map-Learning and Path-Planning Strategies. *Cognitive Systems Research*, vol. 4, pp. 283-317.

Priya, T.K. & Sridharan, K. (2006) A parallel Algorithm, Architecture and FPGA Realization for High Speed Determination of the Complete Visibility Graph for Convex Objects. *Microprocessors and Microsystems*, vol. 30, issue 1, pp. 1-14.

Šeda, M. (2005) Motion Planning in the Plane with Polygonal Obstacles. *Engineering Mechanics*, vol. 12, no. 4. pp. 253-258.