SIMULACE AUTONOMNÍCH MOBILNÍCH ROBOTŮ

Jakub Hrabec, Bohumil Honzík

Ústav automatizace a měřicí techniky
Fakulta elektrotechniky a informatiky
Vysoké učení technické v Brně

1. ÚVOD

Robot - toto celosvětově známé a v dnešní době stále častěji používané slovo vymyslel a poprvé použil spisovatel Karel Čapek ve svém románu R.U.R. Odvodil jej od slova *robota* - práce. Robot má tedy být pomocníkem člověka, zastávat práci. To je již delší dobu skutečností, roboty jsou hojně používány v mnoha odvětvích průmyslu, svoje uplatnění najdou v případech provádění prací na místech ohrožujících lidské zdraví nebo při manipulaci s nebezpečnými předměty, ale také v jiných oborech.

Ve většině těchto případů však roboty provádějí předem naprogramovaný postup nebo jsou řízeny člověkem - operátorem. Cílem (a přáním nejen autorů science fiction) je však vytvořit samostatného (*autonomního*) inteligentního (*kognitivního*) robota.

Inteligence je vlastností některých *živých* organismů; vznikla a vyvíjela se v průběhu dlouhého časového intervalu a dnes umožňuje některým živým organismům efektivně reagovat na složité projevy prostředí a aktivně je využívat ve svůj prospěch, k dosažení svých cílů. Všechny postupy a algoritmy, které v konečném důsledku vedou k napodobování projevů inteligentního chování člověka jsou předmětem zkoumání vědní disciplíny *umělé inteligence*.

Kognitivní robot je elektromechanický systém řízený počítačem, který je schopen autonomní interakce s reálným fyzickým prostředím, aby dosahoval člověkem stanovené cíle.

Musí být schopen:

- *vnímat a rozpoznávat* prostředí
- *vytvářet vnitřní reprezentaci* prostředí
- *automaticky řešit problémy* a vytvářet plány své činnosti
- *vykonávat* ve vnějším prostředí tyto plány
- *komunikovat s člověkem*

Řídicí algoritmus -„mozek“ robota - je tedy jeho velmi důležitou součástí. Bez něj by robot byl jen kusem technologicky vyspělé, avšak mrtvé hmoty. Vhodným prostředkem k ověření řídicího algoritmu před implementací do skutečného robota je počítačová simulace.

2. POPIS SIMULÁTORU

Simulátor slouží k vývoji řídicích algoritmů autonomních mobilních robotů pro pohyb v neznámém prostředí. Informace o poloze překážek i ostatních robotů jsou získávány pomocí modelu ultrazvukových čidel nebo laserových dálkoměrů (resp. scannerů). Simulováno může být najednou více robotů (v tomtéž prostředí) – každý má svůj vlastní řídicí algoritmus. Jednotlivé roboty lze navzájem rozlišit – každý z nich pak může reagovat na přítomnost jiného robota.

Základní vlastnosti simulátoru jsou:

- maximální počet robotů v simulaci je 252
- každý robot je řízen vlastním algoritmem s možností využití různých toolboxů pro *MALTAB*
- informace o okolí robota jsou získávány pomocí modelu ultrazvukových a laserových čidel
- počet, parametry a umístění čidel na robotu je uživatelsky nastavitelné
- roboti se pohybují v prostředí tvořeném pevnými překážkami („zdmi“) a pohyblivými překážkami (ostatními roboty)
- při detekci čidlem lze jednotlivé roboty rozlišit (v praxi by bylo možné zajistit např. snímáním a rozpoznáváním čísel umístěných na robotech)
- barva a tvar každého robota jsou libovolně definovatelné (vizuální odlišení)
- po skončení simulace lze provést zpětné přehrání celého jejího průběhu – „replay“

Simulátor byl vytvořen v prostředí systému *MATLAB* z několika důvodů. Především je to relativní jednoduchost a přehlednost programů pro toto prostředí, dále pak otevřenost systému (umožňuje implementaci rutin napsaných v jazyce C), snadná práce s maticemi a v neposlední řadě dobré možnosti grafického zobrazování dat a vytváření uživatelského rozhraní. Možnost za členění podprogramů napsaných v jazyce C byla s výhodou použita u časově náročných a často používaných funkcí pro jejich velkou přednost - rychlost. Pro překlad byl použit kompilátor Borland C++ 5.01 a funkce *MATLABu mex*. Výsledkem je dynamicky linkovaná knihovna (DLL) pro operační systém Windows.

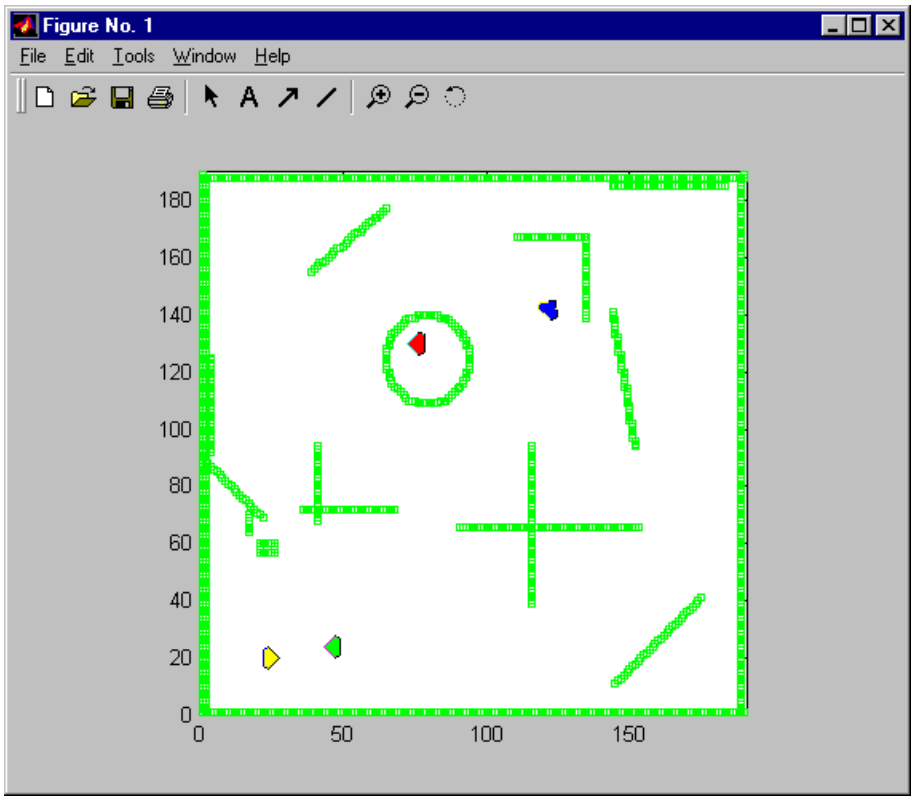
Funkční jádro simulátoru bylo vytvořeno za použití *objektů*. Výhoda tohoto přístupu spočívá v možnosti uchování všech potřebných informací o každém robotu v jedné proměnné. *Objekt* je proměnná (nebo instance) určité *třídy*. *Třída* je datový typ popisující strukturu proměnné a určující operace a funkce, které mohou být na danou proměnnou aplikovány. Tyto se nazývají *metody*. V prostředí *MATLABu* se jedná o soubory typu *m-file* (standardní funkce *MATLABu*), které jsou uloženy v adresáři s názvem *@<jméno_třídy>*. S jednotlivými položkami ve struktuře objektu je možné pracovat pouze za použití těchto metod. Třída vytvořená pro popisovaný simulátor byla nazvána *simrobot*.

Prostředí, ve kterém se roboti pohybují (model okolního světa), je reprezentováno maticí (použit datový typ *uint8* pro úsporu paměti a zrychlení operací oproti standardnímu typu *double*). Lze také použít import obrázku ve formátu *.bmp (černobílá bitmapa).

Pohybový (kinematický) model robota předpokládá tří- nebo čtyřkolový podvozek s řízenými zadními koly a otočným neřízeným předním kolem (v případě čtyřkolového podvozku zatáčí každé přední kolo samostatně) – jde tedy o model „kolečkové křeslo“ (viz [4]).

Simulace probíhá v krocích. V každém kroku je postupně pro všechny roboty provedeno spuštění řídicího algoritmu (je možné číst informace z čidel, měnit rychlost robota atd.), vypočtena a zobrazena aktuální pozice (včetně detekce kolize). Aktuální pozice je též uložena do „historie“ robota. Řídicí algoritmy jsou vyhodnocovány postupně, proto celková doba trvání povedení všech algoritmů je rovna součtu časů potřebných pro vyhodnocení jednotlivých algoritmů. Pokud tedy některý algoritmus obsahuje chybu (např. nekonečná smyčka), dojde k zastavení celé simulace.

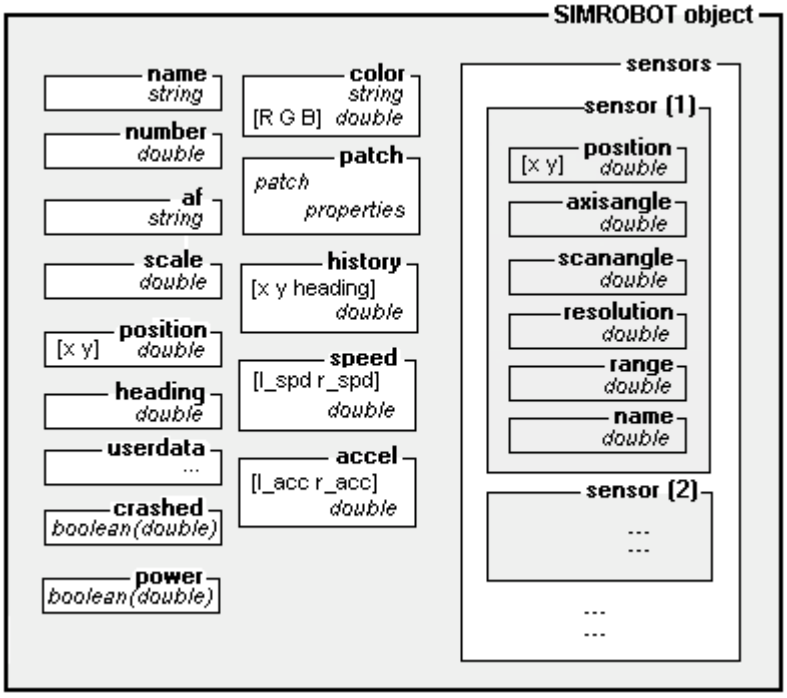
Minimální doba trvání jednoho kroku je 0,1 s (pro nezávislost na rychlosti počítače).



Obr. 1: Simulátor

2.1 TŘÍDA *SIMROBOT*

Třída *simrobot* umožňuje uchování všech potřebných informací o každém robotu. Pro připomenutí: proměnná, která má strukturu (je typu) dané třídy, nazýváme *objektem.*



Obr. 2: Struktura objektu třídy *SIMROBOT*

name	jméno robota - slouží k odlišení jednotlivých robotů
number	unikátní identifikační číslo robota – využití při běhu simulace k detekci robotů
af	jméno souboru (funkce) s řídicím algoritmem robota (bez přípony*.m). Při běhu simulace je v každém kroku volána.
color	barva robota - pro snadnou vizuální identifikaci robotů při simulaci – lze zadat buď jako text (např.'y' – yellow (žlutá) – kompletní seznam viz nápověda k příkazu *MATLABu plot*), nebo jako vektor složek barvy ve formátu RGB)
scale	měřítko robota - koeficient, kterým jsou násobeny zadané souřadnice (rozměry) robota - velikost tak lze upravit bez nové definice celého tvaru (volba velikosti robota relativně k rozměrům prostředí)
patch	*patch* (grafický objekt) znázorňující robota - při definici jsou použita tato pole:
	XData YData - tvar robota ZData
	FaceColor - barva robota
	EdgeColor - barva okraje (implicitně černá)
	Tag - identifikační jméno patche - přiřazena hodnota *simrobot.name* (bez využití při simulaci)
	EraseMode - způsob vykreslování patche - tato hodnota musí být 'xor' (nové vykreslení smaže předchozí)
position	pozice robota v absolutních souřadnicích v souřadném systému okna simulace - tento souřadný systém se shoduje s indexováním matice prostředí. Pozice však může být vyjádřena desetinným číslem.
heading	natočení robota ve stupních - absolutní hodnota.
speed	úhlová rychlost levého a pravého kola robota
accel	zrychlení levého a pravého kola robota

sensors	datová struktura obsahující informace o čidlech robota	
	position	umístění čidla na robotu,
	axisangle	úhel natočení osy čidla
	scanangle	celkový úhel scanování čidla (ve stupních)
	resolution	rozlišení - počet paprsků, na který je rozdělen úhel záběru čidla
	range	dosah čidla (v jednotkách souřadného systému prostředí)
	name	jméno - označení čidla - pomocí něj lze číst údaje z daného čidla. Každé čidlo musí mít v rámci daného robota unikátní označení.
history	obsahuje historii pohybu robota od počátku simulace - uchovávána je poloha a natočení (v souřadném systému prostředí) – aktualizováno každý krok.	
userdata	položka k dispozici uživateli, slouží jako „paměť“ robota	
crashed	logická proměnná, nastavena na hodnotu 1, pokud je detekována kolize robota s překážkou	
power	logická proměnná, indikuje stav robot zapnut/vypnut	

2.2 ČIDLA, DETEKCE PŘEKÁŽEK

2.2.1 Ultrazvuková čidla

Pro navigaci robota je nutné získávat informace o poloze překážek nacházejících se v okolí robota. Pro svou jednoduchost a nízkou cenu jsou často používána ultrazvuková čidla. Tato čidla pracují na principu odrazu emitovaných ultrazvukových vln (resp. pulsů) od překážek. Nevýhodou je, že poskytují pouze údaj o vzdálenosti překážky od čidla, nikoliv o jeho poloze - překážka se může nacházet kdekoliv v prostoru dosahu čidla. Problémy s detekcí mohou vznikat, pokud je překážka vyrobena z materiálu dobře tlumícího zvuk - překážka nemusí být vůbec zaregistrována - nebo dojde-li k několikanásobnému odrazu a teprve poté je signál zachycen čidlem - překážka se pak jeví ve větší vzdálenosti, než ve skutečnosti je. Tyto situace však vznikají pouze při určité konstelaci polohy robota a překážky (například míří-li čidlo do rohu místnosti). Protože se robot pohybuje, je po změně polohy detekce většinou provedena správně.

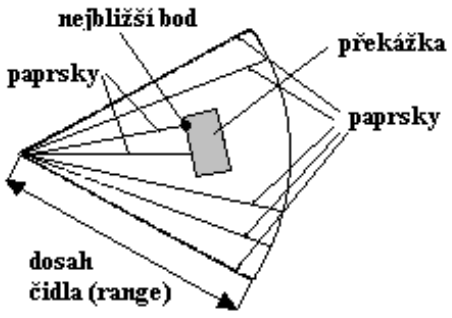
Dosah ultrazvukového čidla je typicky od 5 centimetrů až do několika metrů.

2.2.2 Laserová čidla (laserové scannery)

Jedná se o velmi přesná čidla pro měření vzdálenosti (citlivost měření je asi 10^{-7} m). Využívá se interferometrické měřicí metody, která je schopna měřit vzdálenosti až do 100 m. Laserové scannery mají úhel detekce většinou 180° s krokem asi 0,5°. Citlivost měření je asi 10^{-2} m. Nevýhodou těchto čidel je vyšší cena.

2.2.3 Model čidla

Detekce překážek v simulátoru znamená rozhodnout, zda se na daném místě v matici nachází nenulová hodnota, indikující překážku (pevnou či pohyblivou - robota). Překážky lze detekovat pomocí čidel umístěných na každém robotu.



Obr. 3 : Model čidla

Činnost čidla je simulována za použití Bresenhamova algoritmu - modifikace se středovým bodem (viz [2]). Prostor, ve kterém je čidlo schopno detekce objektů má tvar vějíře (resp. kužele ve 3D), viz obr.3, a je proložen „paprsky“. Body ležící na paprscích jsou vypočítávány s použitím výše zmíněného algoritmu. U každého čidla je možné nastavit úhel, dosah, a počet paprsků. Vhodnou volbou těchto parametrů lze zajistit, aby byl „paprsky“ pokryt rovnoměrně celý prostor dosahu čidla.

Je zřejmé, že se jedná o výpočetně náročné operace a proto byl algoritmus naprogramován v jazyce C. Do aplikace pro *MATLAB* byl implementován jako dynamicky linkovaná knihovna pro systém Windows.

3. ZÁVĚR

Simulátor byl vytvořen v prostředí MATLABu s použitím objektů. Ty umožňují přehledné uložení informací o každém robotu . Důraz byl kladen na jednoduchost (a rychlost) jednotlivých algoritmů. Proto bylo nutné provést při vytváření simulátoru některá zjednodušení. Jedná se především o použitý model ultrazvukového čidla, u kterého nejsou modelovány reálné fyzikální jevy spojené s šíření zvuku (odraz, pohltivost materiálu překážek apod.). V tomto případě se jedná o nutný kompromis mezi rychlostí a reálností simulace, který by však neměl mít vliv na funkčnost simulátoru.

LITERATURA

[1] Using MATLAB , The MathWorks Inc., 1999

[2] Wu, X. and Rokne, J.G. : Double-Step Incremental Generation of Lines and Circles”, Computer Vision, Graphics and Image Processing, (37), 1987, s. 331 - 334

[3] Herout, P. Učebnice jazyka C, Nakladatelství Kopp , 1996

[4] Muir, P. F. : Modelling and control of wheeled mobile robots, PhD. Dissertation, Carnegie Mellon University, Pittsburgh, 1988

[5] Miller, M.K. – Winkless, N. – Bosworth, J. : Personal robot navigator, Robot Press, Conifer, Colorado, 1998

Kontakt : Bohumil Honzík, ÚAMT FEI VUT, Božetěchova 2, 612 66 Brno,
e-mail : honzikb@dame.fee.vutbr.cz, url : http://www.fee.vutbr.cz/UAMT/robotics