OPTIMALIZÍCIA CHODU ROBOTA POMOCOU EVOLUČNÝCH METÓD

Ing. Stanislav Števo

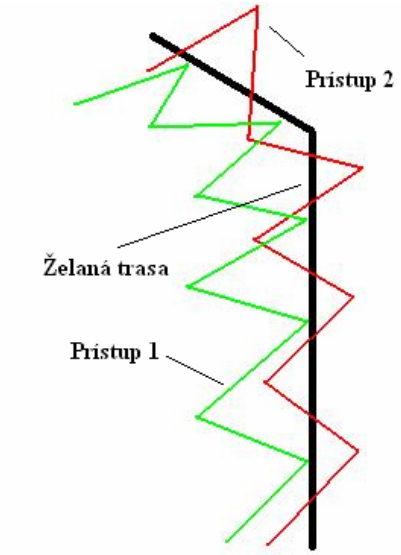
Section of Information and Communication Systems, Institute of Control and Industrial Informatics, Faculty of Electrical Engineering and Information Technology
Slovak University of Technology, Ilkovičova 3, 821 09 Bratislava, Slovak Republic
stanislav.stevo@stuba.sk http://www.fei.stuba.sk

Abstrakt
Príspevok ukazuje možnosti využitia genetického algoritmu pri návrhu konštrukcie robota sledujúceho zvolenú trajektóriu. Genetický algoritmom sa hľadajú optimálne parametre robota tak, aby sa dosiahlo čo najlepšie sledovanie trajektórie. Na konkrétnom príklade sú demonštrované možnosti a obmedzenia genetického algoritmu pre tieto typy úloh.

Úvod
Robotika zaznamenáva čoraz väčšiu popularitu nielen ako predmet na vysokých školách ale vďaka relatívne lacným stavebnicovým systémom sa rozšírila do povedomia stredných a základných škôl. Tieto mobilné robotické systémy sú určené na vykonanie transportných úloh spojených s riešením rozpoznávania prekážok, orientácie v prostredí s prekážkami a s plánovaním trajektórie podľa rôznych kritérií. Pri dostatočnom počte snímačov a akčných členov sú spomenuté úlohy relatívne jednoduché, avšak pri sprísnení kritérií sa aj zo zdanlivo jednoduchej úlohy stáva zložitý problém. V ďalšom opíšeme návrh robota (autíčka), ktorého úlohou je sledovanie trasy, s dvoma akčnými členmi a jedeným snímačom. Hlavná pozornosť je venovaná „plánovaniu“ trajektórie pomocou genetického algoritmu.

1 Stavba robota

1.1 Technické požiadavky



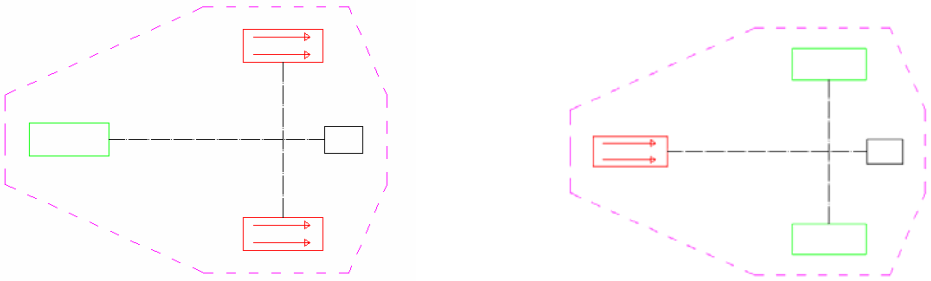
Obr. 1.1 Trajektórie robota

Hlavnou úlohou robota (autíčka) je sledovanie trasy, ktorá je istým spôsobom označená. Želaná cesta musí byť kontrastne odlíšená od podkladu, najčastejšie buď čiernou farbou na bielom podklade alebo naopak (pri predpoklade použitia svetelného senzora). Robot môže používať len 2 akčné členy (motory) a 1 snímač (zo zadania vyplýva, že sa bude jednať o foto-senzor). Podľa tohto predpokladu, máme vo všeobecnosti 2 možné trajektórie robota (viď. Obr. 1.1). Podľa prvého prístupu sa po „narazení“ na označenú trasu, robot vždy o „krok“ vzdiali (musí sa okamžite pri detekcii trasy zastaviť a zmeniť smer jazdy na opačný, aby nenastalo prekročenie trasy) a znovu priblíži (kým nenarazí znovu na želanú trasu). Druhý prístup je založený na detekcii „prekročenia“ želanej trasy. Vždy po jej prekročení sa zmení smer robota na opačný, aby mohlo nastať ďalšie prekročenie. V zásade iný postup pri zvolenej jednoduchej konfigurácii prvkov robota nie je možný.

1.2 Model robota

V zásade môžeme podľa zadaných požiadaviek uvažovať o dvoch modeloch robota. Prvým je model, ktorý má dve hnané (aktívne) kolesá s tretím voľne zaveseným kolesom. Druhý je klasický model auta, ktorý veľmi dobre popisuje tzv. *Ackermanov* model.

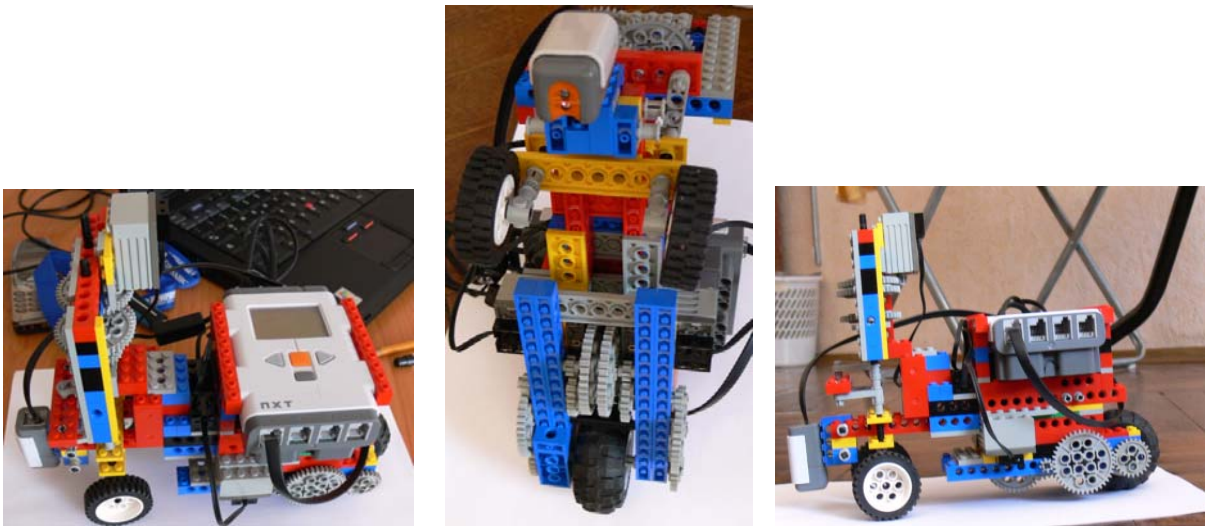
Keďže ani jeden zo spomenutých modelov nespĺňa základné požiadavky pre zadanú úlohu, musíme vytvoriť vlastný model, ktorý bude čo možno najviac využívať výhody jednotlivých prístupov. Keďže máme k dispozícii len dva motory, najjednoduchším riešením bude modifikácia pôvodného Ackermanovho modelu tak, že použijeme len jedno aktívne koleso (viď. Obr. 1.2 vpravo) Jedným motorom bude auto poháňané a druhým motorom bude menený jeho smer.



Obr. 1.2 Principiálna schéma modelu auta s dvoma poháňanými kolesami (vľavo), jedným hnaným kolesom (vpravo)

1.3 Fyzická stavba robota

Použili sme kombináciu prvkov zo staršieho a z novšieho setu LEGO Mindstorms (NXT). Na Obr. 1.4 je zobrazený model robota (autíčka). Na týchto obrázkoch je dobre vidieť novú Brick z LEGO Mindstorms NXT (ako aj svetelný senzor z tohto setu) a prvky zo staršieho setu (motory). Motory v tejto verzii nemajú vlastnú prevodovku, preto sme si museli zostaviť vlastnú, tak výsledná sila motorov bola dostatočná pre pohyb auta, rovnako aj pre otáčanie kolies.

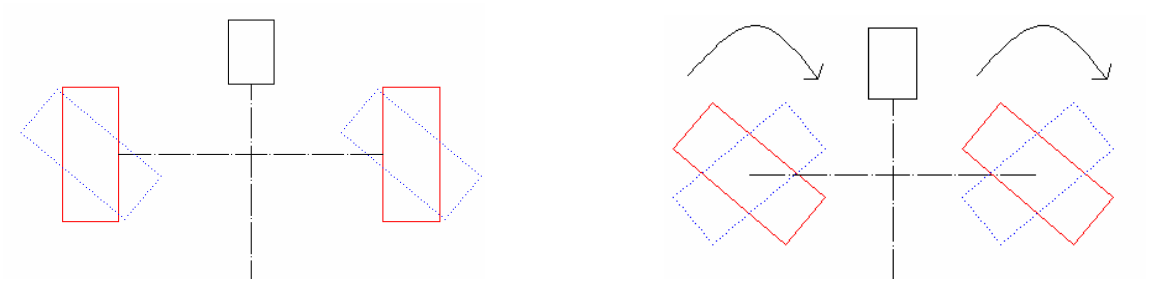


Obr. 1.4 Zostavenie auta pomocou LEGO Mindstorms (NXT)

Kombinácia oboch setov má výhodu v tom, že jednoduchou modifikáciou prevodovky zatáčania alebo pohonu, môžme meniť parametre auta v relatívne veľkých intervaloch.

2 Princíp riadenia robota

Podľa zvoleného modelu je zrejmé, že zmena smeru pohybu auta bude realizovaná otáčaním predných kolies (viď. Obr. 2.1 vľavo). Predpokladajme, že v štartovnej pozícii bude auto umiestnené zvolenú vzdialenosť od najbližšej čiary, ktorá určuje, želanú trasu (v ďalšom len ZT). Takže prvý krok bude natočenia kolies v smere ZT. Po dosiahnutí (detekcii) ZT sa zmení natočenie kolies na opačné, čím dosiahneme ďalšiu detekciu ZT. Počas času otočenia kolies z jednej hraničnej polohy na druhú (viď. Obr. 2.1 vpravo) musí byť snímač neaktívny a to z nasledujúcich dôvodov. Pri detekcii ZT môže byť táto podmienka splnená dlhší časový interval (napr. auto ide po čiare) čím by mohlo dôjsť k nesprávnemu zatočeniu. Ďalej musí byť podmienka neaktívneho snímača splnená aj z dôvodu požiadavky na ďalší snímač, pretože ak by sme menili smer otáčania kolies vždy pri detekcii ZT museli by sme poznať buď aktuálne natočenie kolies alebo by sme museli zistiť dosiahnutie hraničných polôh natočenia kolies. Ďalším faktorom, ktorý potvrdzuje podmienku snímača, je samotná hrúbka čiary označujúca ZT (napríklad ak by bola táto hrúbka väčšia ako vzdialenosť, ktorú vykoná snímač počas natáčania kolies).



Obr. 2.1 Schéma natáčania predných kolies auta

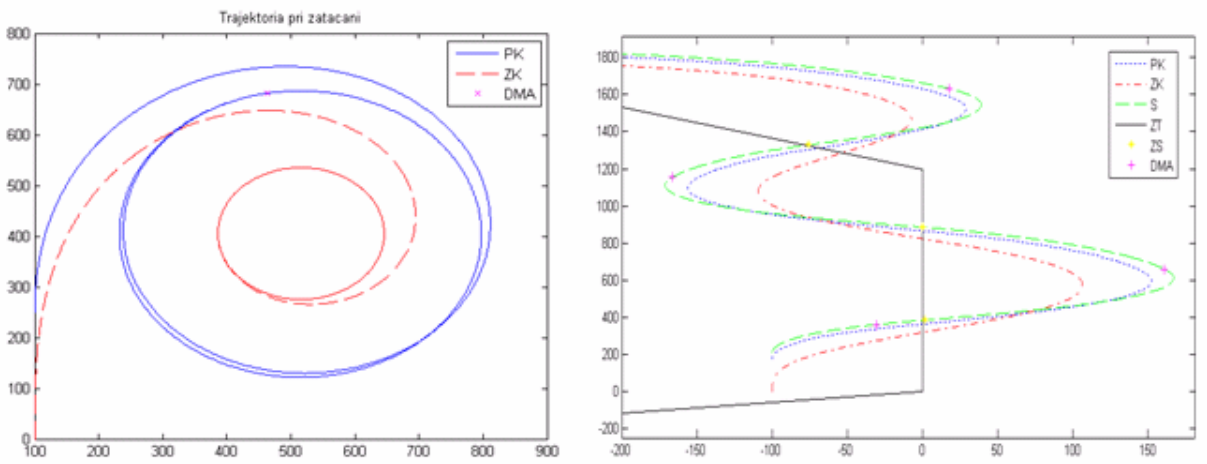
Z takto zadefinovaného algoritmu riadenia nám vyplýva nasledujúca podmienka:

(P) Počas doby natáčania kolies z jednej hraničnej polohy na druhú musí byť senzor neaktívny, respektíve auto nesmie prejsť cez ZT inak bude pokračovať iným smerom.

Označme maximálny uhol natočenia α_{max} a čas potrebný na otočenie kolies do tohto uhlu z štartovnej pozície (priamy smer) ako t_o . Označme časový element pre jednu iteráciu výpočtu trajektórie ako Δt. Potom platí:

$$t \in \langle 0, t_o \rangle \qquad N = \frac{t_0}{\Delta t} \qquad \alpha_N = \frac{\alpha_{\max}}{N} \qquad \alpha(i) = 0 + \alpha_N . i \qquad i \in \langle 1, N \rangle \qquad (2.1)$$

Podľa *2.1* je teda zrejmé, že od začatia zatáčania sa bude uhol zatáčania rovnomerne zväčšovať až kým za čas t_o dosiahne α_{max} . Trajektória bude mať tvar špirály, po dosiahnutí α_{max} sa tvar trajektória zmení na kružnicu (viď Obr. 2.2 vľavo). Pri prechode z -α_{max} do α_{max} (alebo naopak), t.j. pri prekročení TZ bude mať trajektória „esovitý“ tvar (viď. Obr. 2.2 vpravo)



Obr. 2.2 Trajektória robota pri zatáčaní, natočenie kolies z priameho smeru do α_{max} vľavo, natočenie kolies z -α_{max} do α_{max} vpravo. (PK trajektória predných kolies, ZK – trajektória zadných kolies, S – trajektória snímača, ZT – želaná trajektória, ZS – detekcia ZT, natáčanie kolies do opačného smeru, DMA- dosiahnutie α_{max} alebo -α_{max})

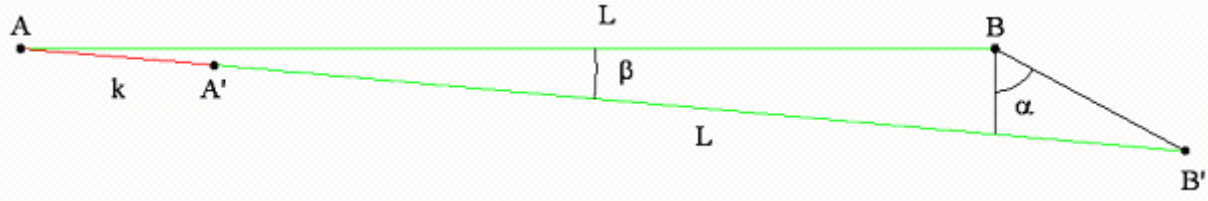
3 Matematický model robota

Matematický model z pohľadu trajektórie robota musí zohľadňovať všetky dôležité parametre, ktoré majú na ňu vplyv. Medzi tieto parametre patria:

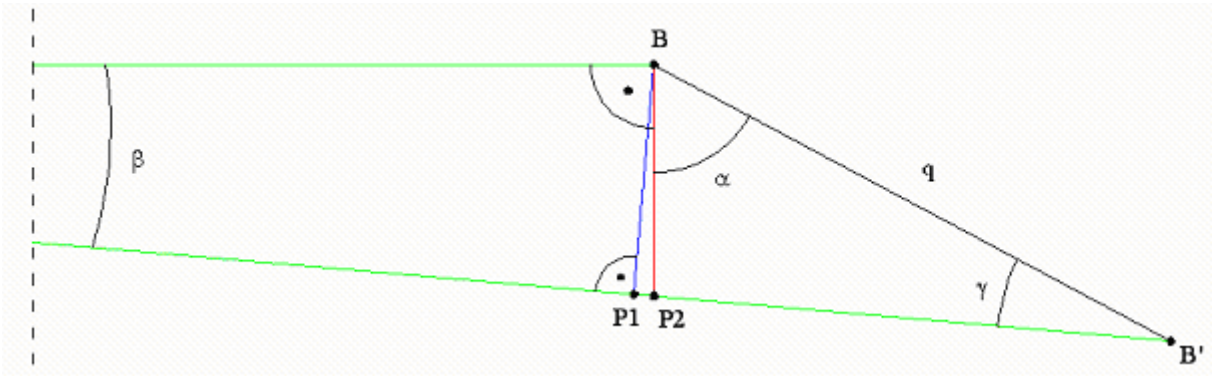
- α_{max} maximálny uhol natočenia kolies
- t_o čas potrebný k dosiahnutiu α_{max} od rovnovážnej polohy (priamy smer)
- L rázvor vozidla
- v priama rýchlosť vozidla

Ak Δt je časový element pre jednu iteráciu výpočtu trajektórie, potom označme k ako dráhu, ktorú za Δt prejde zadné koleso autíčka. ($k = v.\Delta t$)

Potom môžme pohyb zjednodušene popísať ako je znázornené na Obr. 3.1 a 3.2



Obr. 3.1 Princíp výpočtu trajektórie auta (jeden výpočtový krok)



Obr. 3.2 Detail jedného výpočtového kroku pre trajektóriu auta

Našou úlohou je nájsť rovnice opisujúce tento pohyb. Z trojuholníkov *ABP1* a *BB'P2* (Obr. 3.2) môžme jednoducho dovodiť nasledujúci vzťahy:

$$\alpha = 90 - \beta - arctg\frac{L.\sin\beta}{k + L - L.\cos\beta}$$
$$q = \sqrt{(L + k - L\cos\beta)^2 + L^2\sin\beta} \qquad (3.1)$$

Potom rovnice opisujúce pohyb auta v kartézkej sústave budú:

Zadné koleso:

$$X_{ZK}(n+1) = X_{ZK}(n) + \cos(\beta_A).k \qquad (3.2)$$
$$Y_{ZK}(n+1) = Y_{ZK}(n) + \sin(\beta_A).k$$

Predné kolesá (ich os):

$$X_{PK}(n+1) = X_{PK}(n) + \cos(\alpha_A).q \qquad (3.3)$$
$$Y_{PK}(n+1) = Y_{ZK}(n) + \sin(\alpha_A).q$$

α_A uhol natočenia kolies a vektora [0,1]
β_A uhol osi auta a vektora [0,1]

Označme vektor **S** ako $\vec{S} = [X_{PK} - X_{ZK}, Y_{PK} - Y_{ZK}]$
potom pohyb ľubovoľného bodu, ktorý sa nachádza na osi auta bude popísaný ako:

$$X_{LB}(n+1) = X_{ZK}(n+1) + S_x.c \qquad (3.4)$$
$$Y_{LB}(n+1) = Y_{ZK}(n+1) + S_y.c \qquad c\ldots parameter\ definujúci\ bod\ na\ osi$$

4 Genetický algoritmus

Genetický algoritmus (GA) je heuristický postup, ktorý sa snaží aplikáciou princípov evolučnej biológie nájsť riešenie zložitých optimalizačných alebo prehľadávacích problémov. Genetické algoritmy, resp. všetky postupy patriace medzi tzv. evolučné algoritmy používajú techniky napodobňujúce evolučné procesy známe z biológie – dedičnosť, mutácia, prirodzený výber a kríženie.[1]

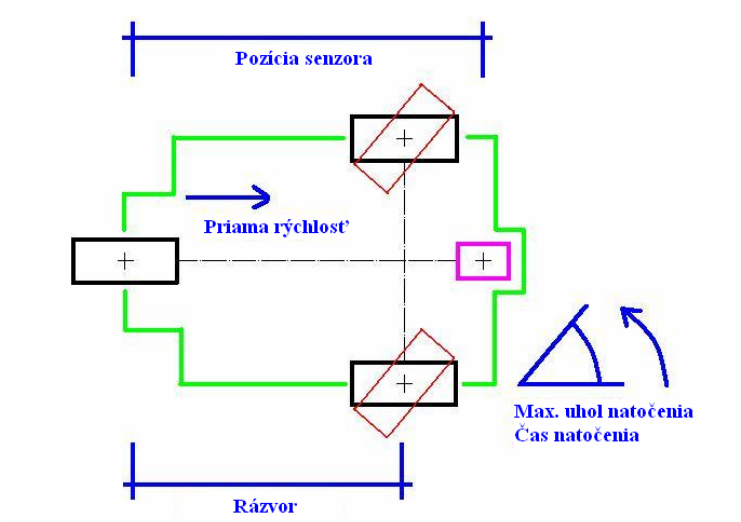
Princíp práce genetického algoritmu je postupná tvorba rôznych riešení daného problému. Pri riešení sa uchováva tzv. populácia, ktorej každý jedinec predstavuje jedno riešenie daného problému. Ako populácia prebieha evolúciou, riešenia sa zlepšujú. Tradične je riešenie reprezentované binárnymi číslami (reťazcami núl a jedničiek), ale aj iné reprezentácie (strom, pole, matice ...). Typicky je na začiatku GA (v prvej generácii) populácia zložená z úplne náhodných členov. Pri prechode do novej generácie je pre každého jedinca vypočítaná tzv. úspešnosť (*fitness)*, ktorá vyjadruje kvalitu riešenia reprezentovaného týmto jedincom. Podľa tejto kvality sú rôznymi spôsobmi (stochastický, náhodný, ...) vyberaný jedinci, ktorí sú modifikovaní (pomocou mutácie a kríženia), čím vznikne nová populácia. Tento postup sa iteratívne opakuje, čím sa kvalita riešenia v populácii postupne vylepšuje. Algoritmus sa obvykle zastaví pri dosiahnutí požadovanej kvality riešenia, prípadne po určenej dobe.

4.1 Vytvorenie genetického algoritmu

Ak uvažujeme model robota podľa *1.4.3* resp. jeho matematický model opísaný vzťahmi 3.1 až 3.4 je zrejmé, že základné parametre, ktoré majú vplyv na trajektóriu robota sú :

- priama rýchlosť robota
- maximálny uhol zatáčania α_{max}
- čas dosiahnutia α_{max}
- vzdialenosť kolies (rázvor)
- pozícia senzora (od pozície zadného kolesa)

Úlohou je nájsť také parametre robota, aby bolo dosiahnuté čo najlepšie sledovanie predpísanej trasy, inými slovami úlohou genetického algoritmu, je nájdenie takej kombinácie uvedených parametrov, aby bola dosiahnutý želaný cieľ (sledovanie trasy)



Obr. 4.1 Model auta a jeho parametre, ktoré budú optimalizované pomocou GA

Štruktúra jedinca

Ako sme už uviedli máme 5 konštrukčných parametrov auta, ktoré majú vplyv na trajektóriu, preto bude jedinec pozostávať práve z piatich génov. Intervaly, v ktorých sa môžu hodnoty génov nachádzať sú dané konštrukčnými obmedzeniami auta (motorov, pasívnych prvkov a pod.) a sú nasledovné:

- priama rýchlosť robota v ⟨30, 300⟩ $mm.s^{-1}$
- maximálny uhol natočenia α_{max} ⟨10, 80⟩ °
- čas dosiahnutia α_{max}, t_o ⟨0.5, 10⟩ s
- vzdialenosť kolies, rázvor ⟨80, 400⟩ mm
- vzdialenosť senzora od ZK ⟨-200, 400⟩ mm

Veľkosť populácie

Veľkosť populácie sme zvolili na 30 jedincov. Po uskutočnení niekoľkých testov a preskúmaní diverzity populácie sa ukázalo, že tento počet je dostatočný. Pri vytváraní novej populácie v novej generácii sme použili turnajový výber s elitazrizmom (t.j. najlepší jedinec postupuje priamo bez zmeny do ďalšej populácie).

Mutácie

Zvolili sme dva druhy mutácií. Mutáciu na náhodnú hodnotu zo zvoleného intervalu (pravdepodobnosť 0.25) a aditívnu mutáciu (pravd. 0.25). (Možnosť voľby jednotlivých pravdepodobností v rozsahu 0 až 1, pričom čím vyššia je táto miera, tým častejšie sa vyskytne mutácia)

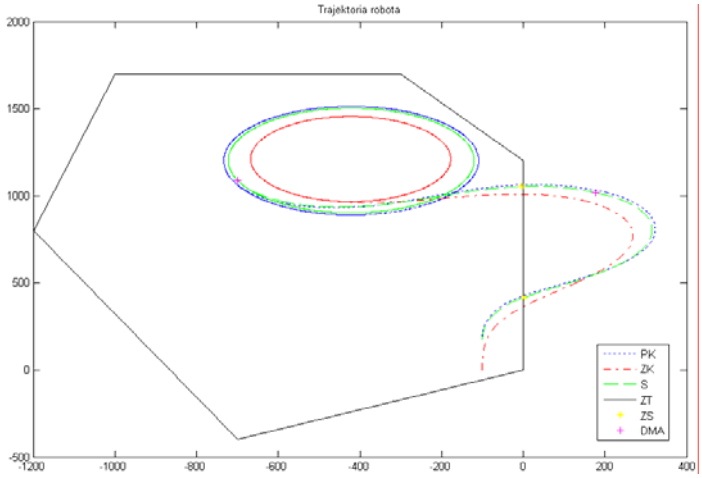
Kríženie

V algoritme je použité trojbodové kríženie, pričom krížené reťazce sú vyberané náhodne.

Voľba účelovej funkcie

Vo všeobecnosti máme viac možností ako posúdiť vhodnosť (nevhodnosť) daného jedinca (napr. počet prekročení ZT, min. čas pre prejdenie trasy, a pod.) My sme sa rozhodli zvoliť za rozhodujúce kritérium presnosť sledovania trasy. V algoritme bola táto presnosť (PS) vypočítaná ako súčet vzdialeností každého bodu trajektórie auta od ZT. Z tohto pohľadu máme tri možnosti voľby presnosti sledovania ZT (presnosť sledovanie ZT snímačom, prednými kolesami alebo zadným kolesom). Keďže jedným z optimalizovaných parametrov je aj rýchlosť, musí byť táto vzdialenosť (TZ – trajektória robota) normovaná vzhľadom na rýchlosť (pretože pri väčšej rýchlosti auto prejde dlhšiu trasu, preto aj PS bude väčšia). Aby sme rešpektovali podmienku ***P*** ak nastalo prekročenie skôr ako bola dosiahnutá α_{max}, dané riešenie bolo pokutované. Miesto veľmi veľkej pokuty sme stanovili jej hodnotu na polovicu hodnoty *fitness* najlepšieho jedinca. Rovnaké princíp sme použili aj v prípade, ak sa nachádzal snímač v čase dosiahnutia α_{max} na čiare ZT. Veľmi veľkú pokutu sme ale použili v prípade, že tvar TZ a parametre auta boli v takej kombinácii, že nastal prípad ako je zobrazený na obr. 4.2.

V algoritme sme vždy počítali koľko krokov po sebe sa auto pohybuje s natočením α_{max}, ($-\alpha_{max}$) ak je tento počet väčší ako $360/\beta_{max}$ je jasné, že auto uviazlo v kružnici, ktorá nepretína (dráha senzora) ZT, preto bude DP nastavené na hodnotu 1.



Obr. 4.2 Uviaznutie robota, (PK trajektória predných kolies, ZK – trajektória zadných kolies, S – trajektória snímača, ZT – želaná trajektória, ZS – detekcia ZT, natáčanie kolies do opačného smeru, DMA- dosiahnutie α_{max} alebo -α_{max})

Fitness jedinca je teda vyjadrená ako: $Fit = PS + SP.POK + BC.POK + DP.MPOK$

PS reprezentuje presnosť sledovania ZT, SP reprezentuje počet prekročení TZ pri $\alpha < \alpha_{max}$, BC reprezentuje počet výskytov stavu, keď sa dosiahne α_{max} a snímač je na čiare ZT, POK je hodnota pokuty definovaná ako polovica ZT, DP určuje či nastalo uviaznutie auta – krúženie na mieste (Obr. 4.2).

Ak chceme aby bolo schopné auto sledovať všeobecnú TZ, t.j. ľubovoľného tvaru (pri dodržaní určitých podmienok – min. polomer zatáčky a pod.) máme dve možnosti ako určiť parametre pomocou GA:

- vytvorenie jednej „dlhej“ zložitej ZT (fitness sa vypočíta len pre jednu ZT)
- vytvorenie množiny ZT (fitness bude súčtom jednotlivých pod-fitness pre jednotlivé ZT)

Ukončovacia podmienka

GA sa ukončí po dosiahnutí predpísaného počtu generácií.

4.2 Parametre ovplyvňujúce výpočtový čas GA

Výpočtový čas jednej generácie závisí od mnoho faktorov. V našom algoritme sa pre každý jedinec musí vypočítať trajektória, ktorú prejde za čas TS. Počas výpočtu trajektórie sa zisťuje či nenastali porušenia podmienok ktoré vyplývajú z algoritmu riadenia (prekročenie TZ ak je $\alpha < \alpha_{max}$ a pod.). Pri dosiahnutí α_{max} sa v každom kroku výpočtu trajektórie zisťuje, či nenastalo prekročenie ZT. V poslednom kroku výpočtu trajektórie sa určí či nastalo uviaznutie auta (točenie do kruhu) a výpočet vzdialenosti trajektórie od TZ (fitness). Čas potrebný pre výpočet fitness jedného jedinca (teda aj celého GA) závisí od nasledujúcich parametrov.

- veľkosť populácie GA
- predpísaný počet generácií
- čas simulácie (pre výpočet trajektórie autíčka)
- presnosť výpočtu trajektórie
- zložitosť ZT, resp. jej množstvo bodov ktoré definujú ZT

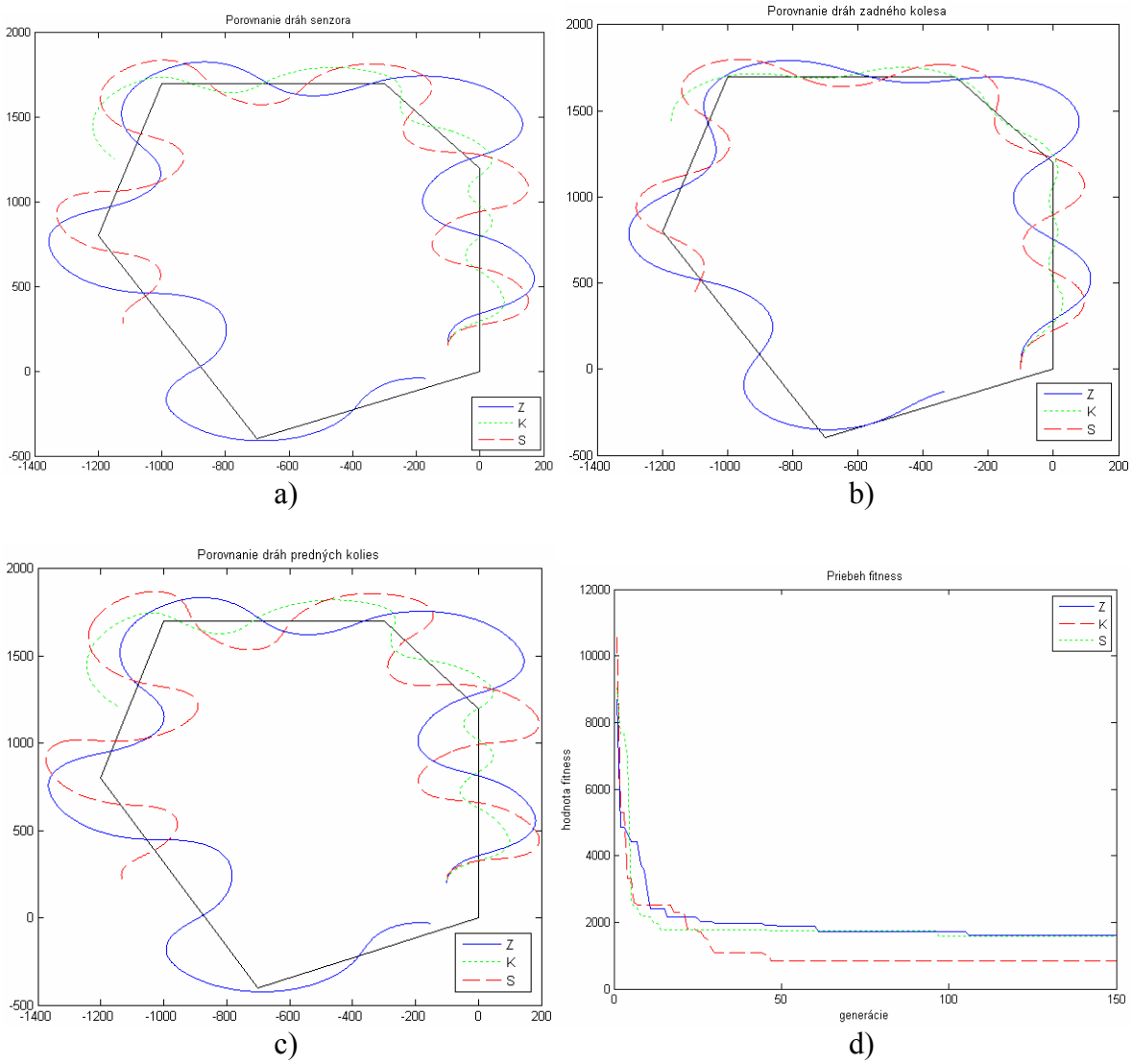
Pre všetky parametre okrem presnosti výpočtu trajektórie sa pre väčšie hodnoty predĺži čas behu GA.

4.2 Demo príklad

Pre zvolenú želanú trasu sme pomocou uvedeného GA navrhli optimálne parametre auta. Čas simulácie sme nastavili na 40 sekúnd, presnosť na 0.01 sekundy a ukončovaciu podmienku po vykonaní 150 generácií. GA sme spustili tri krát, resp. pre každú možnosť, t.j. ak sme brali do úvahy presnosť sledovania TZ snímačom (S), predným kolesom (PK) alebo zadným kolesom. Najlepšie riešenia sú zapísané v tabuľke 4.1.

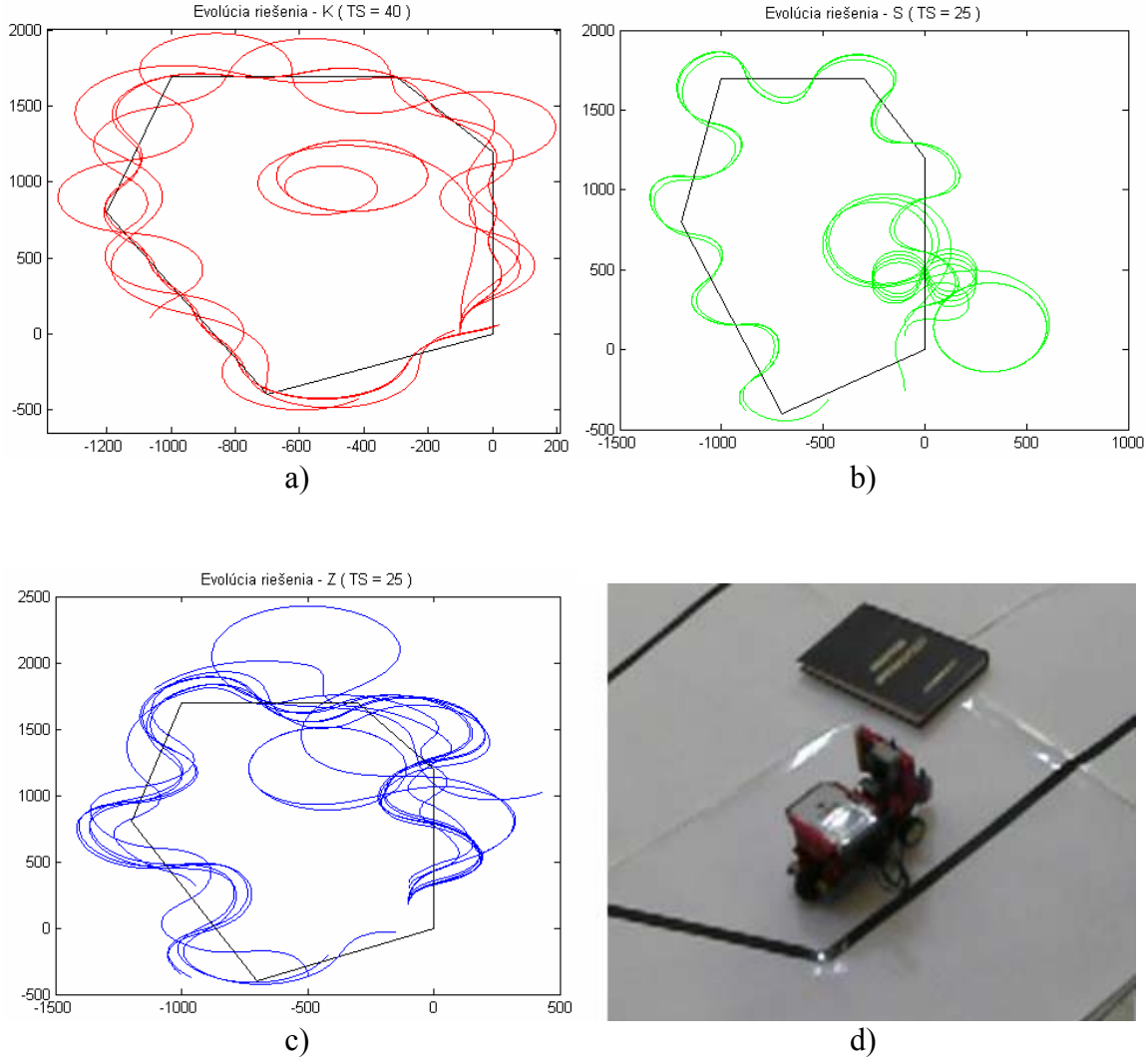
	Priama rýchlosť	α_{max}	t_o	Rázvor	pozícia senzora (c)
S	228.9874	52.1951	0.5000	221.7105	0.6988
PK	293.8896	38.3740	0.5000	202.6398	0.8958
ZK	145.7596	39.1215	0.5000	210.8309	0.7779

Tab. 4.1 Najlepšie riešenia, koef. c udáva vzdialenosť senzora od zadného kolesa c.Rázvor



Obr. 4.3 a) Porovnanie trajektórií senzora (S) pre PS(ZT,(S)(Z)(K)),
b) porovnanie trajektórií zadného kolesa (K) pre PS(ZT,(S)(Z)(K)),
c) porovnanie trajektórií predného kolesa (Z) pre PS(ZT,(S)(Z)(K)),
d) priebeh fitness počas evolúcie , Z – PS(ZT,PK), S - PS(ZT,S), PS(ZT,ZK)

Na Obr. 4.3 sú znázornené porovnania trajektórií pre prípady najlepšieho sledovania predpísanej trajektórie senzorom (a), zadným kolesom (b) alebo predným kolesom (c). Obr. 4.3 (d) znázorňuje priebeh *fitness* počas evolúcie. Z Obr. 4.3 (d) je zrejmé, že najlepšie sledovanie TZ je v prípade zadného kolesa. Z Obr. 4.3 a) b) c) je taktiež evidentné, že najďalej sa rovnaký časový úsek dostane auto, ktorého parametre boli optimalizované v prípade ak fitness bola *PS(ZT,(S)(Z)(K))* teda ak sa minimalizoval rozdiel vzdialeností trajektórie predného kolesa a ZT.



Obr. 4.4 a) Evolúcia trajektórií zadného kolesa (K) pre PS(ZT,K),
b) Evolúcia trajektórií senzora (S) pre PS(ZT,S),
c) Evolúcia trajektórií predného kolesa (Z) pre PS(ZT,Z),
d) reálny experiment

Na Obr. 4.4 a) b) c) sú znázornené priebehy trajektórií počas evolúcie. Vykreslená je najlepšie riešenie po desiatich generáciách. Keďže bol predpísaný počet generácií 150, máme 15 priebehov (niektoré splývajú). Obr. 4.4 d) znázorňuje reálny experiment, ktorého výsledok s dobrou zhodou zodpovedal simulácií.

Záver

Analyzovali sme možnosti stavby robota (autíčka), ktorého úlohou je sledovanie trasy, ktorá je kontrastne odlíšená od podkladu, za predpokladu použitia iba jedného senzora. Podobne boli analyzované možnosti stavby takéhoto robota pomocou stavebnicových setov. Pomocou systému LEGO Mindstorms a LEGO Mindstorms NXT sme postavili autíčko, ktoré bolo schopné meniť smer jazdy a spĺňalo požiadavky zadania (boli použité len 2 motory a 1 snímač). Vytvorili sme matematický model (pohybové rovnice), ktorý sme následne využili v genetickom algoritme, pomocou ktorého sme optimalizovali konštrukčné parametre auta tak, aby presnosť sledovania predpísanej dráhy bola čo najlepšia. Vytvorili sme 3 varianty účelovej funkcie, resp. fitness mohla byť počítaná ako presnosť sledovania želanej trasy zadným kolesom, predným kolesom auta alebo senzorom. Z uskutočneného príkladu sme zistili, že pre zvolenú trasu, bolo dosiahnuté najlepšie sledovanie zadným kolesom (čo je zrejmé pre všetky varianty) pri zvolenej fitness *K (4.2)*. Za rovnaký časový úsek, sa autíčko dostalo najďalej ak bola fitness zvolená ako Z. Na základe uskutočneného experimentu sa dá predpokladať, že podobným spôsobom je možné realizovať návrh konštrukcie aj zložitejších robotov.

Literatúra

[1] Ivan Sekaj, Evolučné výpočty a ich využitie v praxi, IRIS Bratislava, 2005

[2] Kvasnička, V., Pospíchal, J., Tiňo, P.: Evolučné algoritmy, Vydavateľstvo STU, Bratislava, (2000)

[3] James Floyd Kelly, Lego Mindstroms NXT – The Mayan Adventure, Apress, 2006

[4] James Floy Kelly, Lego Mindstroms NXT-G Programming Guide, Apress, 2007

[5] Michael Gasperi & Philippe Hurbain, Extreme NXT - Extending the Lego Mindstorms NXT to the next level, Apress, 2007

[6] J. S. Wang, S. N. Rong, B. P. Zhang: Researches in the walking principle and machanism of a omnidirectional walking machine. In: International Journal of Robotics Research, 1992.

[7] P. Štefaňák: Analýza a syntéza štruktúr mechanizmov pohonov a robotov. Veda, Bratislava 1989

[8] Joseph L. Jones and Anita M. Flynn, Mobile Robots: Inspiration to Implementation, 2nd edition, A K Peters, Wellesley, MA, 1999

[9] Algorithms for Robotic Motion and Manipulation: WAFR 1996 Jean-Paul Laumond, Mark Overmars, editors, 1997

[10] Behavior-Based Robotics (Intelligent Robotics and Autonomous Agents) by Ronald C. Arkin, MIT Press 1998.

[11] http://en.wikipedia.org/wiki/Lego_Mindstorms_NXT

[12] http://www.rctek.com/handling/ackerman_steering_principle.html