AT&P JOURNAL plus 1 2009

Intelligent motion control systems

reviewed slovak professional magazine for scientific and engineering issues

Inteligentné pohybové systémy

recenzované periodikum vedeckých a inžinierskych publikácií

ISSN 1336-5010

Inteligentné pohybové systémy

Intelligent motion control systems

Odborný garant

prof., Ing. Ladislav Jurišica, PhD.
Slovenská technická univerzita v Bratislave
Fakulta elektrotechniky a informatiky
Ústav riadenia a priemyselnej informatiky
Ilkovičova 3, 812 19 Bratislava, Slovensko
Tel.: +421 2 602 91 351
e-mail: ladislav.jurisica@stuba.sk

Technical guarantee

prof., Ing. Ladislav Jurišica, PhD.
Slovak University of Technology in Bratislava
Faculty of Electrical Engineering and Information Technology
Institute of Control and Industrial Informatics
Ilkovičova 3, 812 19 Bratislava, Slovak Republic
Tel.: +421 2 602 91 351
e-mail: ladislav.jurisica@stuba.sk

Vydavateľ Publisher

HMH s.r.o.
Tavarikova osada 39
841 02 Bratislava 42
IČO: 31356273

Spoluzakladateľ Co-founder

Katedra ASR, EF STU
Katedra automatizácie a regulácie, EF STU
Katedra automatizácie, ChtF STU
PPA CONTROLL, a.s.

IEEE Czechoslovakia Section
www.ieee.cz

IEEE
Celebrating 125 Years
of Engineering the Future
www.ieee125.org

www.atpjournal.sk
AT&P journal ***PLUS1*** 2009

Redakčná rada Draft committee

prof. Ing. Alexík Mikuláš, PhD., FRI ŽU, Žilina
doc. Ing. Dvoran Ján, CSc., FCHPT STU, Bratislava
prof. Dr. Ing. Fikar Miroslav, FCHPT STU, Bratislava
doc. Ing. Hantuch Igor, PhD., KAR FEI STU, Bratislava
doc. Ing. Hrádocký Ladislav, PhD., SjF TU, Košice
prof. Ing. Hulkó Gabriel, DrSc., SjF STU, Bratislava
prof. Ing. Jurišica Ladislav, PhD., FEI STU, Bratislava
doc. Ing. Kachaňák Anton, CSc., SjF STU, Bratislava
prof. Ing. Krokavec Dušan, PhD., KKUI FEI TU Košice
prof. Ing. Madarász Ladislav, Phd., FEI TU, Košice
prof. Ing. Malindžák Dušan, CSc., BERG TU, Košice
prof. Ing. Mészáros Alojz, CSc., FCHPT STU, Bratislava
prof. Ing. Mikleš Ján, DrSc., FCHPT STU, Bratislava
prof. Ing. Moravčík Oliver, CSc., MtF STU, Trnava
prof. Ing. Murgaš Ján, PhD., FEI STU, Bratislava
doc. Ing. Rástočný Karol, PhD. KRIS ŽU, Žilina
doc. Ing. Schreiber Peter, CSc., MtF STU, Trnava
prof. Ing. Skyva Ladislav, DrSc., FRI ŽU, Žilina
prof. Ing. Smieško Viktor, PhD., FEI STU, Bratislava
doc. Ing. Šturcel Ján, PhD., FEI STU, Bratislava
prof. Ing. Taufer Ivan, DrSc., Univerzita Pardubice
prof. Ing. Veselý Vojtech, DrSc., FEI STU, Bratislava
prof. Ing. Žalman Milan, PhD., FEI STU, Bratislava

Ing. Bartošovič Štefan,
generálny riaditeľ – president
ProCS, s.r.o.

Ing. Boďo Vladimír, CSc.,
riaditeľ – managing director
AXESS, spol. s r.o.

Ing. Csölle Attila,
riaditeľ – managing director
Emerson Process Management, s.r.o.

Ing. Horváth Tomáš,
riaditeľ – managing director
HMH, s.r.o.

Ing. Hrica Marián,
riaditeľ divízie A & D – head of A&D division
Siemens, s.r.o.

Ing. Murančan Ladislav,
PPA Controll a.s., Bratislava

Ing. Petergáč Štefan,
predseda predstavenstva – chairman of board director
Datalan, a.s.

Ing. Pilnan Branislav,
sales leader HPS
HONEYWELL s.r.o.

Ing. Tóth Andrej,
generálny riaditeľ – president
ABB, s.r.o.

Redakcia Editors office

AT&P journal
Evidenčné číslo: EV 3242/09
Košická 37, 821 09 Bratislava 2
tel.: 02/5026 1752 – 55
fax: 02/5026 1757
e-mail: info@atpjournal.sk
http://www.atpjournal.sk

Ing. Anton Gérer
šéfredaktor – editor in chief
sefredaktor@atpjournal.sk

Ing. Ildikó Csölleová
vedúca redakcie – editorial office manager
podklady@atpjournal.sk

Mgr. Zuzana Švecová
marketingová manažérka – marketing manager
marketing@atpjournal.sk

Ing. Branislav Bložon
odborný redaktor – editor
redaktor@atpjournal.sk

Ing. Martin Karbovanec
odborný redaktor – editor
karbovanec@atpjournal.sk

Zuzana Pettingerová
technická redaktorka – DTP
podklady@atpjournal.sk

Mgr. Bronislava Chocholová
jazyková redaktorka – text corrector

Datum vydania: jún 2009

Autorské práva HMH, s.r.o., Všetky práva vyhradené
Copyright HMH, Ltd., 2009, All Rights Reserved

ISSN 1336-5010

Obsah

Návrh systémov

Umelá inteligencia v riadení

Riadenie mobilných robotických systémov

Modelovanie systémov

Articles

System design

Artificial intelligence in control

Control of mobile robotic systems

System modeling

Metódy a prostriedky mechatronického návrhu

Michal Boršč

Abstrakt
Článok sa zaoberá metodologickými východiskami a metódami návrhu a vývoja mechatronických systémov. Prezentovaná je flexibilná metodika vývoja mechatronických systémov - VDI 2206. Osobitná pozornosť je venovaná metódam návrhu a vývoja mechatronických systémov na báze modelov. Koncepčný návrh systému, analýza dynamiky mechanických časti a tvorba softvéru pre riadenie mechatronckého systému sú charakterizované ako kľúčové problémy mechatronického návrhu a opísané sú prístupy k ich riešeniu.

Kľúčové slová: mechatronický návrh, mechatronický systém, V-model, simulačné prostredie, hybridný model.

Úvod

Vývoj mechatroniky sa vyznačuje kvalitatívnymi zmenami jej predmetu a metód. V počiatočnej fáze bolo cieľom zvýšenie úrovne automatizácie strojárskej výroby a strojárskych výrobkov. Podstata činnosti strojov a vykonávaných funkcií sa nemenila. Elektronické prostriedky slúžili na zvyšovanie kvality existujúcich strojov a boli nadstavbou strojov.

Kvalitatívna zmena nastala vtedy, keď prostredníctvom elektroniky a informačných technológií, hlavne prostredníctvom sofvéru riadiacich systémov sa začali do strojov implementovať nové *unikátne funkcie* [4]. Podstata týchto funkcií je v spojení energetických a informačných procesov. Elektronika a informačné technológie už nie sú len nadstavbou, ale sú vnútornou organickou súčasťou strojov, lebo neslúžia len na automatizáciu pôvodných funkcií stroja, ale pridávajú strojom nové unikátne funkcie, sú súčasťou podstaty ich funkčných vlastností. Ako príklad možno uviesť asistenčné systémy súčasných automobilov (ABS, ASR, ESP a pod).

Súčasným trendom v mechatronike je vývoj *inteligentných strojov*. Strojom sa pridávajú funkčné vlastností, ktoré sú realizované prostriedkami umelej inteligencie. Pod *inteligentným systémom* rozumieme súbor technických prostriedkov a softvéru, spojený informačným procesom, schopný na základe informácií a znalostí robiť samočinne syntézu cieľa, prijímať rozhodnutia o činnosti a nachádzať racionálne spôsoby dosiahnutia cieľa [11]. Tendencia je vytvárať také stroje a prístroje, ktoré sa dokážu prispôsobovať človekovi, alebo situácií. V tejto súvislosti sa zvykne mechatronika definovať ako veda o inteligentných strojoch. Do tejto skupiny patria, napríklad, adaptívne rýchlo rekonfigurovateľné stroje a výrobné systémy.

Mechatronické systémy, ktoré sú zložené z komponentov rôznej fyzikálnej podstaty a vyznačujú sa unikátnymi funkčnými vlastnosťami, môžeme považovať za *nový typ technického produktu*. Atribúty nového typu produktu zároveň ukazujú na veľký *inovačný potenciál mechatroniky*.

Prax ukázala, že navrhovanie, výroba a prevádzka zariadení, ktorých komponenty majú rôznu fyzikálnu podstatou, nie sú efektívne, ak sa zabezpečujú parciálne v rámci tradičných odborov, strojárstva, elektrotechniky, automatizácie a informatiky, ale vyžadujú nové metodologické prístupy k ich návrhu, syntéze a analýze, výrobe i prevádzke.

Postupy návrhu a vývoja mechatronických systémov opisované v literatúre sú kombináciou tradičných metód a nových prístupov. Osvedčené tradičné postupy mechatronika neneguje, ale stavia ich do inej polohy. Rozvoj metód návrhu a vývoja mechatronických systémov vyžaduje vyjasniť si predovšetkým osobitosti *mechatronických prístupov* a spôsob ich aplikácie v jednotlivých fázach návrhu.

V tomto článku chceme upriamiť pozornosť na kľúčové problémy, metodologické východiska a prístupy pri návrhu a vývoji mechatronických systémov. Stručne opíšeme tie prístupy a metódy, ktoré podľa nášho názoru odzrkadľujú špecifika návrhu mechatronických systémov.

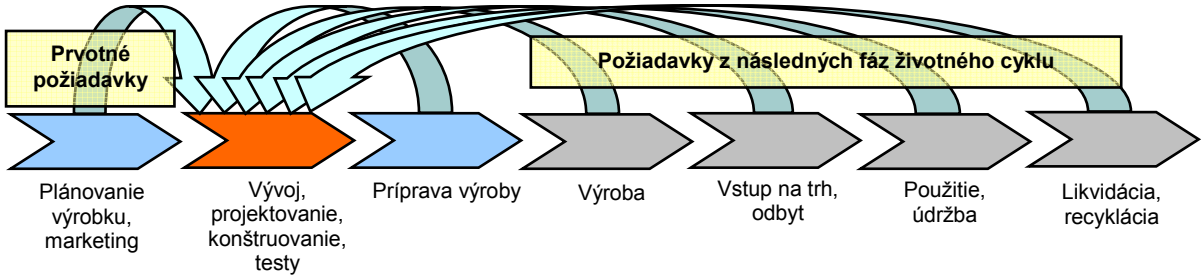
1. Ciele a metodologické východiska mechatronického návrhu

Rozhodujúcou fázou *životného cyklu výrobku* je jeho návrh a vývoj [3], [9] (Obr.1). Vyznačuje sa malou apriórnou informáciou, veľkou informačnou neurčitosťou a vysokým stupňom dôležitosti rozhodnutí. Úlohou projektovania nového výrobku je transformácia opisu požadovaných vlastností a technic-kých parametrov výrobku do štandardnej formy dokumentácie, podľa ktorej realizátor zhotoví reálny objekt. Súčasťou projektu je dokumentácia k technickým a prevádzkovým skúškam, prevádzke a servisu výrobku.

Ciele technického projektu treba zabezpečiť pri rešpektovaní troch obmedzení: *obsah projektu, náklady a čas*. Pri projektovaní nového výrobku nestačí zabezpečiť len technické parametre výrobku, ale aj jeho životaschopnosť v danom prostredí. Zohľadniť treba ekonomické podmienky, trhové a podnikateľské prostredie a environmentálne požiadavky.

Mechatronické systémy sa vyznačujú:

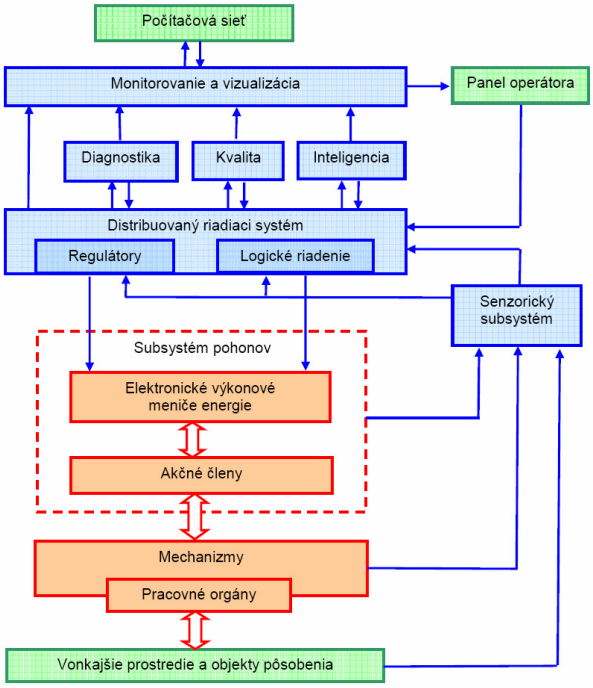
- komponentmi rôznej fyzikálnej podstaty - mechanické častí, elektromechanické meniče energie, elektronika, číslicové riadiace, informačné a komunikačné prostriedky,
- zložitými pohybovými operáciami a transformáciou rôznych foriem energie,
- unikátnymi funkčnými vlastnosťami a variabilitou funkcií.



Obr.1 Životný cyklus výrobku a požiadavky na výrobok
Fig.1 Product life-cycle and requerements to the product

Zložitejšie mechatronické systémy, tzv. *komplexné systémy,* sú zložené z komponentov, ktoré sami o sebe sú relatívne autonómnymi mechatronickými systémami. V tejto súvislosti sa okrem pojmu „*mechatronický systém*" používa tiež pojem a „*mechatronický uzol*", alebo *modul*.

Všeobecná štruktúra mechatronického systému je znázornená na Obr.2.



Obr.2 Štruktúra mechatronického systému
Fig.2 Mechatronic system structure

Cieľom mechatroniky je konzistentný ucelený integrovaný návrh mechatronického systému, pri ktorom sa dosiahne *synergický efekt* v technických, ekonomických, funkčných a prevádzkových vlastnostiach a parametroch výrobku. Ide najmä o tieto vlastnosti mechatronického systému:

- optimálna štruktúra a konštrukcia,
- kompatibilita subsystémov a optimálne interakcie medzi subsystémami a elementmi,
- variabilita funkčných vlastnosti a inteligencia správania,
- vysoká prevádzková spoľahlivosť,
- nízke investičné a prevádzkové náklady užívateľa.

Dôležitým cieľom mechatronických prístupov je skrátenie času potrebného na návrh a vývoj nového výrobku.

Postup návrhu mechatronického systému, ktorý sa vyznačuje novými špecifickými metódami sa zvykne nazývať *mechatronickým návrhom*. Tým sa zdôrazňuje, že nejde len o vecnú, ale aj o metodologickú stránku návrhu. Metodologické východiska mechatronického návrhu možno zhrnúť do nasledujúcich bodov:

- Súčasný *integrovaný interdisciplinárny návrh* mechanických, elektrických a informačno-riadiacich súčastí, sprevádzaný analýzou ich vlastností, optimalizáciou parametrov a vzájomných interakcií.
- *Komplexné rozhodovanie* vo všetkých fázach mechatronického návrhu, lebo zmeny jedného komponentu spravidla vyvolávajú nutnosť meniť aj ostané komponenty. Základom komplexného rozhodovania je *systémový prístup*, uvažovanie o celku s jeho vnútornými a vonkajšími väzbami. Súčasťou systémového prístupu je *dekompozícia* systému na subsystémy (moduly) a elementy, ich návrh a následná *integrácia* elementov a subsystémov do integrovaného mechatronického systému.
- Na overovanie vlastností systému a jeho komponentov je potrebný *reálny experiment*. Tendenciou v mechatronike je redukcia rozsahu experimentovania s reálnymi objektmi. Reálny experiment sa nahradzuje *simulačným experimentom*. Skúma sa virtuálny objekt – *matematický a počítačový model* mechatronického systému a jeho súčastí. Modelovaním a simuláciou sa zároveň zabezpečuje *integrita návrhu*. Model je spoločným integrovaným interdisciplinárnym priestorom mechatronického návrhu. Vytváranie modelov mechatronických systémov, zložených z modelov komponentov rôznej fyzikálnej podstaty vyžaduje „spoločný jazyk". Je ním *matematická teória systémov*. V teórií systémov sa abstrahuje od fyzikálnej podstaty objektov. Fyzikálne veličiny sú nahradené systémovými, ktorými sú: vstupné, stavové a výstupné veličiny. V prípade bránových modelov energetických interakcií sú to prietokové a spádové veličiny. Modely vyjadrujú vzťahy medzi systémovými veličinami.

2. Metody návrhu a vývoja mechatronické ho systému

2.1 Metodika VDI 2206

V literatúre sú prezentované rôzne všeobecné postupy návrhu mechatronických systémov i metódy riešenia konkrétnych špecifických problémov návrhu [4], [5], [6], [9], [10], [12]. Osobitnú pozornosť si zasluhuje metodika, ktorú vydal Spolok nemeckých inžinierov (Verein deutscher Ingenieure) pod názvom *VDI 2206 „Metodika vývoja mechatronických systémov*" [14]. Metodika je opísaná tiež v publikácii [8] a iných.

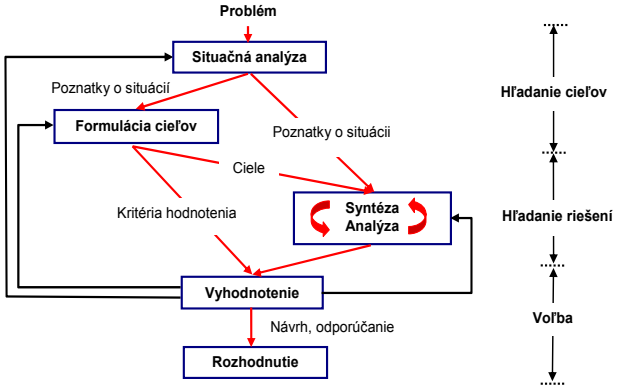
Metodika VDI 2206 je ucelenou flexibilnou metodikou návrhu a vývoja mechatronického systému. Je určená pre praktikov. Vývojovému pracovníkovi dáva k dispozícii flexibilný model postupu pri návrhu a vývoji mechatronického systému.

Metodika sa opiera o tri súčasti:

- všeobecný postup riešenia problému ako *mikrocyklus;*
- *V-model* ako *makrocyklus;*
- *metodiky pre čiastkové úlohy návrhu* - opakované pracovné kroky.

2.2 Riešenie problému ako mikrocyklus

Celý proces návrhu a vývoja mechatronického systému je štruktúrovaný ako postupnosť čiastkových úloh. Mikrocyklus má podporovať vývojového pracovníka pri riešení predvídateľných a tým plánovateľných čiastkových úloh. Metodika riešenia problému ako mikrocyklus má všeobecnú platnosť pre samostatné riešenie malých determinovaných úloh – elementov riešenia zložitejšieho problému.



Obr.3 Riešenie problému ako mikrocyklus

Fig.3 Problem-solving as a micro-cycle

Mikrocyklus sa skladá z troch krokov (Obr.3):

- *Hľadanie cieľov*, ktoré sa skladá z analýzy stavu a formulácie cieľov;
- *Hľadanie riešení*, ktoré spočíva v cyklickom opakovaní syntézy, analýzy a postupnej optimalizácie navrhovaných riešení;
- *Voľba riešenia*, ktorá spočíva v záverečnom vyhodnotení variantov riešenia problému a prijatí rozhodnutia.

2.3 V-model ako makro-cyklus

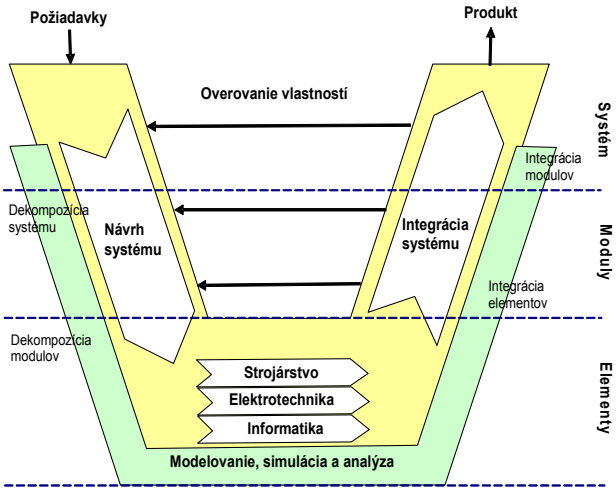
V-model je vyjadrením podstaty mechatronických prístupov a postupov pri navrhovaní a vývoji mechatronického systému. V-model reprezentuje logickú postupnosť hlavných fáz a krokov projektu. Vyznačuje sa *dekompozíciou funkcií* a technických prostriedkov na ich realizáciu na *moduly* a *elementy,* následnou *integráciou* elementov a modulov do celého systému a *cyklickou verifikáciou a optimalizáciou*. Pod pojmom modul tu chápeme mechatronický uzol, subsystém, ktorý sám o sebe je mechatronickým systémom, zloženým z elementov rôznej fyzikálnej podstaty.

V-model znázornený na obr.4 sa skladá z týchto hlavných fáz návrhu a vývoja mechatronického výrobku:

- formulácia požiadaviek
- návrh systému
- špecializovaný návrh elementov podľa odborov (špecifický doménový návrh)
- integrácia systému
- zabezpečenie vlastnosti

Požiadavky: Východiskovým bodom sú aktuálne požiadavky na vývoj systému. Požiadavky na funkcie a vlastnosti nového výrobku sa konkretizujú do formy *technického zadania.* Takto vyjadrené požiadavky sú meradlom, podľa ktorého sa bude hodnotiť výsledný produkt.

Formulácia požiadaviek na nový výrobok vychádza z analýzy potreby nového výrobku, potenciálnych cieľov, analýzy súčasného stavu vedecko-technického rozvoja, prieskumu trhu a vyhodnotenia technologických, ekonomických a časových možností podniku - budúceho výrobcu. Okrem týchto prvotných požiadaviek je potrebné zohľadniť požiadavky vyplývajúce z ostatných fáz životného cyklu výrobku (Obr.1). Konkrétnejšie poznatky a požiadavky, získané neskôr z následných fáz životného cyklu reálneho výrobku, sú informáciou pre ďalšiu inováciu výrobku. Výrobok preto má byť projektovaný tak, aby umožňoval inovácie a nevyžadoval zakaždým navrhovať a vyvíjať celý systém úplne od počiatku. Prostriedkom zabezpečenia tejto požiadavky je *modulárna stavba* mechatronického systému.



Obr.4 V-model návrhu a vývoja mechatronického systému

Fig.4 V-model of mechatronic system design and development

Návrh systému: Cieľom je vytvoriť *interdisciplinárnu koncepciu* riešenia, architektúru mechatronického systému, štruktúru jeho funkcií a prostriedkov technickej realizácie.

Špecializovaný odborový návrh (špecifický doménový návrh)**:** Po vypracovaní celkovej koncepcie nasleduje ďalšia konkretizácia riešenia, *detailné projektovanie, konštruovanie a dimenzovanie elementov,* spravidla oddelene podľa jednotlivých odborov: mechanické častí, elektrické a elektronické častí a informačno-riadiací subsystém. Cieľom je zabezpečiť spoľahlivú realizáciu funkcií komponentov.

Integrácia systému: Výsledky z jednotlivých odborných oblastí sú integrované do jedného spoločného systému, čo umožní vyšetrovať, preskúmať a porovnať skutočné a želané vlastností systému a interakcie komponentov.

Zabezpečenie vlastností: Výsledok každého kroku návrhu musí byť priebežne overený. K tomu slúži priebežný cyklický proces *verifikácie, validácie* a následnej *optimalizácie* parametrov systému. Cieľom je zabezpečiť zhodu skutočných vlastnosti elementov, subsystémov a celého systému s požadovanými.

Vývoj zložitejšieho mechatronického výrobku vyžaduje spravidla vykonať viaceré makrocykly podľa *stupňa zrelosti* navrhovaného systému. Z hľadiská stupňa zrelosti výrobok v jednotlivých cykloch reprezentuje: *model, laboratórna vzorka, funkčná vzorka, prototyp, finálny výrobok.* V každom nasledujúcom cykle je návrh detailnejší a presnejší. Počet a úprava makrocyklov závisí od charakteru problému a zložitosti produktu.

2.4 Metodiky pre čiastkové úlohy návrhu - opakované pracovné kroky

Jednotlivé fázy návrhu a vývoja mechatronického výrobku sa skladajú z krokov, v ktorých sa riešia čiastkové úlohy. Voľba metodík pre riešenie čiastkových úloh je závislá od špecifík navrhovaného systému a charakteru čiastkových úloh. Aj na tejto úrovni niektoré postupy možno odporúčať ako všeobecné.

Podstatou **návrhu systému** je jeho postupná *dekompozícia* do *štruktúry* komponentov. V-model mechatronického návrhu môžeme členiť vertikálne podľa stupňa dekompozície, resp. integrácie systému. Pri návrhu systému sú dve hlavné úrovne dekompozície:

- dekompozícia komplexného systému na *subsystémy - mechatronické uzly*, resp. *moduly*.
- dekompozícia modulov na *elementy podľa odborov.*

Tvorba štruktúry systému začína špecifikáciou *hlavných funkcií systému*. Pri špecifikácii hlavných funkcií sa vychádza z požiadaviek na pracovné charakteristiky budúceho výrobku, ktoré sú opísané v technickom zadaní. Opis funkcií je abstrahovaný od spôsobu ich technickej realizácie.

Nasleduje dekompozícia hlavných funkcií na *čiastkové funkcie* a stanovenie ich nadväznosti, teda *štruktúry funkcií*.

Ďalším krokom je hľadanie *princípov činností* a *prostriedkov technickej realizácie* čiastkových funkcií. Výsledkom je *štruktúra technických komponentov* - modulov a elementov systému.

Špecializovaný odborový návrh a vývoj elementov systému v rámci jednotlivých odborov sa uskutočňuje podľa osvedčených metód návrhu strojárskych, elektrotechnických, elektronických, elektromechanických, riadiacich, informačných a komunikačných systémov. V tejto fáze je návrh rozčlenený na úlohy pre špecialistov v jednotlivých odboroch. Charakter mechatronických systémov vyžaduje v tejto fáze návrhu *organizovanú kooperáciu* špecialistov z rôznych odborov a *koordináciu* ich činností. Ide teda o *koordinovaný* a tým aj *integrovaný návrh komponentov*.

Integrácia systému, podobne ako dekompozícia, prebieha na dvoch hlavných úrovniach:

- integrácia elementov do subsystémov - mechatronických uzlov, modulov,
- integrácia modulov do finálneho systému.

Vo fáze integrácie systému cieľom je prispôsobenie komponentov k celému systému.

Integrácia je ponímaná z viacerých hľadísk. Rozlišujeme nasledovné typy integrácie:

- integrácia funkcií,
- integrácia distribuovaných komponentov,
- modulárna integrácia,
- priestorová integrácia.

Integrácia je spojená so **zabezpečovaním vlastností** systému, *verifikáciou a validáciou* vlastností systému a jeho komponentov. Testujú sa *funkcie* a *systémová kompatibilita* elementov a subsystémov. Návrhy elementov a modulov sa hodnotia tiež z hľadiska nárokov na *technológiu výroby a spoľahlivosť*. Podľa výsledkov testov a ich analýzy robí sa *korekcia* a *optimalizácia* návrhu. V - model ako makrocyklus preto obsahuje aj menšie *vnútorné cykly*. Po verifikácií a validácií na určitej úrovni integrácie nasleduje návrat na príslušnú úroveň návrhu systému.

3. Postup návrhu s využitím modelov

Súčasné informačné technológie poskytujú širokú paletu nástrojov na podporu jednotlivých fáz a krokov návrhu mechatronických systémov. Možno ich zatriediť do viacerých skupín:

Nástroje na modelovanie funkcií a *správania sa* systémov. Sú to počítačové programy na modelovanie dynamiky procesov a logickej nadväznosti funkcií subsystémov. Sem patrí modelovanie a simulácia kinematiky a dynamiky viazaných telies, funkcií výkonovej elektroniky, funkcií a algoritmov riadiacich systémov.

Nástroje CAD (Computer Aided Design), FEM (Finite Element Method - metóda konečných prvkov) a BEM (Boundary Element Method) - metóda hraničných prvkov) slúžia hlavne pre navrhovanie tvarov, konštruovanie, dimenzovanie a analýzu priestorových vlastností mechanických častí.

Nástroje pre navrhovanie ďalších častí a elementov mechatronického systému podľa špecializovaných odborov. Sú to nástroje pre elektronický návrh, elektrický návrh, tekutinový návrh, návrh riadiaceho systému, simuláciu hardvéru a návrh softvéru.

Softvérové produkty na podporu riadenia projektu a archiváciu dokumentácie, napríklad nástroje opisu požiadaviek, nástroje na spravovanie požiadaviek, nástroje pre validáciu produktov v reálnom prostredí, systém spravovania dát produktov a pod.

Informačné technológie nie sú len nástrojom na podporu návrhu mechatronických systémov. Jednou z oblastí použitia informačných technológií pri návrhu mechatronických systémov je *modelovanie a simulácia*. Modelovanie a simulácia sú prostriedkom aplikácie špecifických mechatronických prístupov, preto sa stali *organickou súčasťou mechatronického návrhu*. V tejto súvislosti sa hovorí o *návrhu založenom na modeli* (*model based design*) a o *modelom integrovanej mechatronike (model integrated mechatronics)* ako novej paradigme mechatronického návrhu [13].

Modely používané pri návrhu mechatronických systémov sa vytvárajú na rôznych úrovniach abstrakcie. V závislosti od toho načo má model slúžiť volíme typ modelu. Základné *typy modelov* sú:

- topologický model,
- fyzikálny model,
- matematický model,
- numerický, resp. počítačový model.

Pri rozhodovaní o použití modelov je potrebné formulovať účel modelovania, odpovedať na otázku: prečo daný problém riešiť modelovaním? Ciele modelovania môžu byť:

- skúmanie funkčne orientovaného návrhu, analýza koncepcie návrhu, stanovenie požiadaviek na odborovo špecializovaný návrh;
- návrh a optimalizácia mechanických častí a akčných členov;
- návrh riadiaceho systému;
- skúmanie princípov inovácie mechatronického systému;
- úspora nákladov na reálne experimenty pri vývoj prototypu, najmä pri skúmaní hraničných režimov;
- úspora času pri skúmaní systémov s veľkými zotrvačnosťami;
- skúmanie a overovanie subsystémov, ich prepojenia a interakcie simuláciou na hybridných modeloch.

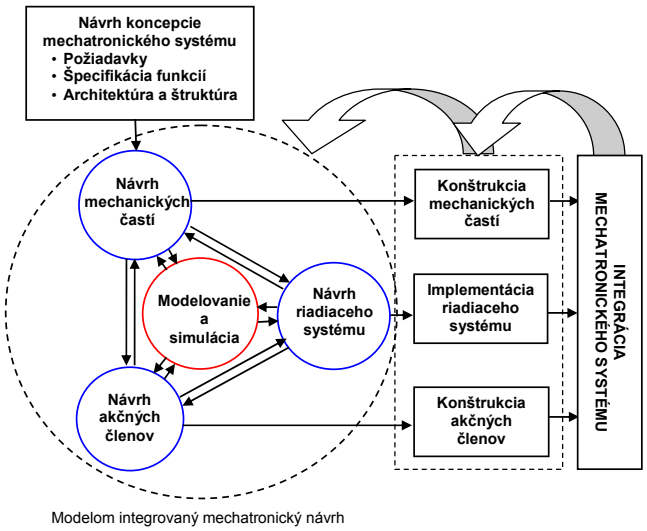
Osobitný význam majú *modely dynamiky procesov, modelovanie funkcií a simulácia správania sa* mechatronických systémov. V literatúre často sa uvádza, že *synergický efekt* pri mechatronickom návrhu sa dosahuje *súčasným integrovaným návrhom komponentov* rôznej fyzikálnej podstaty: mechanických častí, akčných členov a riadiaceho systému. Tézu o súčasnom návrhu nemožno však chápať doslovne. Fyzikálna podstata a kauzálne vzťahy medzi komponentmi mechatronického systému vyžadujú dodržať istú postupnosť návrhu jednotlivých častí.

Mechatronický prístup, v zmysle tézy o súčasnom návrhu komponentov, vyžaduje pri návrhu každej časti zohľadňovať jej *súčinnosť* s ostatnými súčasťami systému. Druhým dôsledkom uvedenej tézy je začlenenie *iteračných cyklov* do procesu návrhu. Cykly slúžia na verifikáciu, korekciu a postupnú optimalizáciu komponentov. Korekcie a optimalizácia návrhu iteračným postupom sa dajú efektívne robiť len pomocou simulačného experimentu na počítačových modeloch. Preto, po návrhu koncepcie a architektúry mechatronického systému, je vhodné ako ďalšiu fázu zaradiť návrh *virtuálneho mechatronického systému.* Modelujú a simulujú sa funkcie jednotlivých komponentov a následne funkcie komponentov integrovaných do mechatronického systému v *spoločnom simulačnom prostredí*.

Cieľom integrovaného interdisciplinárneho návrhu s použitím modelov v spoločnom simulačnom prostredí je postupne navrhnúť, overiť a optimalizovať mechanické časti, akčné členy, pohony, systém riadenia a nakoniec celý mechatronický systém. Ide o to, aby pri vykonávacom projekte, realizácií komponentov a kompletácií finálneho výrobku boli korekcie hlavných konštrukčných, výkonových a funkčných parametrov prakticky vylúčené.

Výsledky návrhu virtuálneho mechatronického systému a jeho komponentov sú východiskom pre odborovo orientovaný vykonávací projekt, realizáciu komponentov a kompletáciu mechatronického systému.

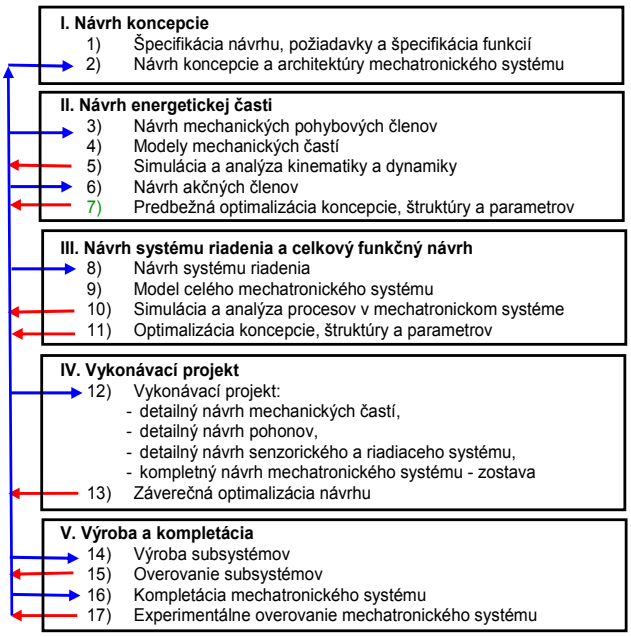
Postup integrovaného interdisciplinárneho návrhu s použitím modelov [2], ktorý sa osvedčil pri návrhu autonómnych mechatronických systémov, resp. mechatronických modulov zložitejších systémov, znázorňuje schéma na obr.5



Obr.5 Návrh a vývoj mechatronického systému založený na modeli

Fig.5 Model based mechatronic system design and development

Vývoj mechatronického systému podľa uvedeného postupu môžeme podrobnejšie opísať procedúrou podľa obr. 6. Každý krok analýzy výsledkov simulácie a optimalizácie parametrov vyžaduje korekciu parametrov, alebo aj štruktúry systému, teda návrat k niektorému z predchádzajúcich krokov návrhu. Ku ktorému kroku návrhu sa treba vrátiť a čo treba meniť, korigovať vyplýva z analýzy vlastností systému. Táto fáza návrhu, ktorej súčasťou je riešenie mnohokritériálnej úlohy optimalizácie, je ťažko formalizovateľná. Dôležitú úlohu tu zohrávajú znalostí, skúseností a tvorivé schopností projektanta.



Obr.6 Procedúra návrhu a vývoja mechatronického systému

Fig.6 Procedure of mechatronic system design and development

Efektívnym nástrojom skúmania a verifikácie variantov riešení je kombinácia modelov a reálnych funkčných vzoriek, alebo prototypov častí mechatronického systému. Kombináciou reálnych objektov a počítačových modelov vznikajú *hybridné modely*. Používajú dva typy kombinácie modelov:

a) **Software-in-the-Loop (SIL)** je integrácia modelov komponentov systému v jednom spoločnom simulačnom prostredí. Simulačné modely komponentov a celého systému vytvorené na základe matematických modelov nemusia pracovať v reálnom čase.

b) **Hardware-in-the-Loop (HIL)** je integrácia reálnych komponentov a modelov ostatných častí systému do jedného spoločného simulačného prostredia. Simulačné modely musia pracovať v reálnom čase.

Z reálnych objektov a modelov možno vytvoriť kombinácie na podporu riešenia nasledovných úloh (Obr.7):

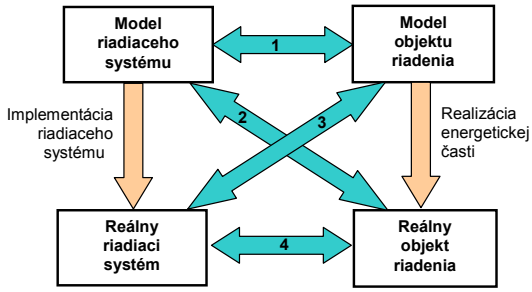
1. **Overenie funkcií:** Skúmanie, overovanie funkčných vlastností mechatronického systému na modeli zloženom, modelov energetickej častí a modelov riadiaceho systému a ich komponentov

2. **Aplikácia riadiaceho systému**: Spojenie modelu riadiaceho systému a reálneho objektu riadenia umožňuje overovať algoritmy riadenia a implementáciu nových funkcii na reálnom objekte.

3. **Overenie softvéru:** Spojenie reálneho riadiaceho systému a modelu objektu riadenia umožňuje overovať bez-

chybnosť softvéru a interfejsov reálneho riadiaceho systému.

4. **Integrácia:** Záverečnou fázou vývoja mechatronického systému je integrácia systému a overovanie vlastnosti mechatronického systému skúškami na reálnom systéme, ktorý vznikne spojením reálneho riadiaceho systému s reálnym objektom.



Obr.7 Kombinácia modelov a reálnych objektov

Fig.7 Kombination of models and real objects

SIL a HIL simulácie sa stali štandardnými súčasťami návrhu a vývoja mechatronického systému, pri overovaní vlastností mechatronického systému a jeho komponentov. K tomu účelu sú vyvinuté softvérové nástroje, ako napríklad program LabView, nadstavby programu Matlab-Simulink: SimMechanics, SimPowerSystems, Real-Time Workshop a iné.

4. Ťažiskové problémy mechatronického návrhu

Z úloh, ktoré treba riešiť pri návrhu mechatronického systému, za kľúčové problémy a ohniska mechatronického návrhu možno považovať nasledovné:

a) návrh koncepcie a architektúry - štruktúry a konštrukcie mechatronického systému,
b) analýza dynamiky mechanických častí a pohonov,
c) tvorba softvéru informačno-riadiaceho systému.

Tvorba koncepcie a architektúry mechatronického systému spočíva v návrhu štruktúry hlavných a čiastkových funkcií, výbere fyzikálnych princípov a prostriedkov na technickú realizáciu požadovaných funkcií systému. Riešiť treba najmä otázku, ako kombinovať energetické a informačné procesy, elektromechanické komponenty a informačné technológie pri realizácií jednotlivých funkcií. Dôležitým aspektom je priestorová integrácia elektronických a elektro-mechanických častí, teda konštrukcia stroja, alebo zariadenia zloženého z mechanických, elektrických a elektronických komponentov. Jednotlivé koncepcie je vhodné hodnotiť pomocou SWOT analýzy. Výsledky SWOT analýzy sú podkladom pre porovnávanie variantov a výber najvodnejšej koncepcie mechatronického systému.

Analýza kinematiky a dynamiky mechanických časti a pohonov, je hlavným článkom optimalizácie návrhu energetickej časti mechatronického systému, kľúčom k dosiahnutiu synergického efektu vo výkonových a pohybových parametroch mechatronického systému.

Pri návrhu mechanických štruktúr a pohonov treba mať na zreteli, že riadením sa dá dosiahnúť len taká dynamika procesov, akú umožňuje energetická časť systému so svojimi fyzikálnymi energetickými, konštrukčnými a funkčnými obmedzeniami. Z toho vyplýva aj iný pohľad na optimalizáciu dynamických procesov, než aký sa zvykne používať v teórií riadenia. Klasická úloha optimalizácie riadenia spočíva v dynamickej optimalizácií pohybových operácií a ich riadenia na danom objekte. Optimalizácia v mechatronike spočíva v riešení obrátenej úlohy: optimalizovať mechanický systém a akčné členy pre predpokladané, resp. požadované optimálne pohybové operácie.

Pri návrhu a optimalzácií mechanických časti a pohonov sa využíva *inverzný model* dynamiky mechanických časti. Rieši sa *obratená úloha dynamiky* [1], [2]. [7]. Pre zadané pohybové operácie počítajú sa časové priebehy síl a momentov, ktoré sú potrebné na realizáciu daných pohybov. Výsledky simulčného exerimentu na inverznom modeli sú východiskom pre návrh pohonov a overovanie správností dimenzovania mechanických častí. Umožňujú tiež odháliť kritické procesy, ktoré bude potrebné eliminovať zmenou charakteru pohybových operácií a tomu zodpovedajúcou úpravou algoritmov riadenia.

Ďalším ťažiskovvm problémom návrhu mechatronického systému je *tvorba softvéru*. Unikátnosť funkcií mechatronických systémov, nová kvalita funkčných a úžitkových vlastnosti sa dosahujú prostredníctvom softvéru. Súčasné informačné technológie z hľadiska aplikácií v mechatronických systémoch ponúkajú najmä tieto možností číslicového spracovania informácií:

- programová implementácia algoritmov riadenia a komunikácie,
- distribuované spracovanie informácie v reálnom čase,
- aplikácia vnorených (embedded) informačných a riadiacich systémov,
- variabilita softvéru,
- implementácia zložitých algoritmov nielen pre štandardnú reguláciu, logické a programové riadenie, ale aj optimalizáciu procesov, adaptáciu, učenie, rozpoznávanie a rozhodovanie,
- implementácia algoritmov pre monitorovanie, vizualizáciu, diagnostiku, vyhodnocovanie kvality a styk s operátorom v systémoch človek-stroj.

Stavba riadiacich systémov sa uberá cestou unifikácie, modularizácie a štandardizácie hardvérových a softvérových prostriedkov. Jeden zo smerov vývoja priemyselných riadiacich systémov reprezentujú stavebnicové modulárne systémy na báze PLC. Takéto systémy poskytujú určitý unifikovaný súbor funkcií. Návrh a implementácia riadenia na báze PLC sú pomerne rýchle, pružné a nevyžadujú veľké náklady. Ich funkčné možností sú však pomerne obmedzené.

Iným smerom je použitie univerzálnych riadiacich počítačov. Tento smer v súčasnosti sa sústreďuje na stavbu mikroprocesorových elektronických riadiacich jednotiek, ktoré prostredníctvom zberníc, napríklad CAN, vytvárajú distribuovaný riadiaci systém. Mikroprocesorové riadiace jednotky patria do skupiny vnorených (embedded) systémov. Poskytujú široké možností pre implementáciu rôznorodých i zložitejších funkcií do mechatronického systému. Elektronické riadiace jednotky tohto typu sa v súčasnosti široko využívajú v automobiloch, robotoch, rôznych automatoch, v systémoch technického zabezpečenia budov, medicínskej technike, zariadeniach pre experimentálny výskum a v spotrebných výrobkoch. Pre zefektívnenie návrhu, vývoja a výroby riadiacich systémov aj v tejto oblasti je tendencia unifikácie a modularizácie hardvéru a softvéru. Unifikované sú predovšetkým hardvérové platformy, operačné systémy a komunikačný softvér. Príkladom modulov aplikačného softvéru sú moduly pre zabezpečenie nasledovných funkcií:

- snímanie a spracovanie analógových veličín a signálov,
- snímanie a spracovanie diskrétnych signálov,
- vysielanie signálov pre akčné členy,
- riadenie mechatronického systému v režimoch: chod, porucha, servis ,
- monitorovanie a vizualizácia - HMI,
- komunikácia v distribuovaných systémoch,
- diagnostika.

Metodika návrhu priemyselného riadiaceho systému pre konkréné aplikácie, hlavne metodika návrhu architektúry a flexibilných modulov aplikačného softvéru riadiacich jednotiek a distribuovaných riadiacich systémov je osobitným problémom softvérového inžinierstva. Náročnosť tvorby aplikačného softvéru bola podnetom pre vývoj špeciálnych metodík a nástrojov pre zefektívnenie tvorby softvéru riadiacich systémov. Pri tvorbe modulárnej štruktúry aplikačného softvéru sa tiež používa V-model. Pre tento účel boli vyvinuté simulačné jazyky a programy, napríklad univerzálny modelovací jazyk UML. Okrem rôznych vývojových prostredí pre tvorbu aplikačného softvéru, boli vyvinuté aj prostriedky pre automatizáciu generovania kódu priamo z funkčného modelu riadiaceho systému, napríklad programy LabView, Real Time Workshop v prostredí Matlab-Simulink a iné. Problémom však zostáva optimalizácia takto automaticky generovaného kódu.

Záver

V článku sme chceli upriamiť pozornosť na kľúčové otázky a trendy vývoja metód navrhovania mechatronických systémov. V súčasnosti sú vykryštalizované metodologické princípy návrhu mechatronických systémov, ktoré sa premietajú aj do ucelených metodík. Vzhľadom na rozmanitosť mechatronických systémov metodiky majú charakter všeobecného modelu postupu pri návrhu a vývoji, ktorý treba prispôsobiť konkrétnemu typu mechatronických systémov a podmienkam ich vývoja a výroby. Ako vzor ucelenej flexibilnej metodiky sme uviedli metodiku VDI 2206.

Zároveň s rozvojom metód sa intenzívne rozvíjajú informačné technológie na podporu a zefektívnenie mechatronického návrhu. Praktické skúseností ukazujú, že osobitnú pozornosť si zasluhujú spôsoby použitia modelov a prostriedkov simulácie, ako nástroja integrovaného interdisciplinárneho návrhu a prostriedku dosiahnutia synergického efektu.

Charakterizovali sme ťažiskové problémy mechatronického návrhu, za ktoré považujeme: návrh koncepcie a architektúry mechatronického systému, analýzu dynamiky mechanických časti a tvorbu softvéru pre riadenie mechatronického systému.

Opísané princípy a metódy návrhu a vývoja mechatronických systémov sa osvedčili v našej výskumnej práci, pri návrhu a vývoji mechatronických systémov v spolupráci s výskumno-vývojovými a projektovými organizáciami a podnikmi. Boli tiež východiskovými tézami pri koncipovaní a stavbe nového predmetu na našej fakulte - „Projektovanie mechatronických systémov“.

Poďakovanie

Táto práca vznikla v rámci výskumnej úlohy podporovanej Vedeckou grantovou agentúrou Ministerstva školstva Slovenskej republiky: VEGA 1/4056/07 - Analýza a syntéza mechatronických systémov.

Literatúra

[1] BORŠČ, M., ŠTEFULA, J., SLIVKA, V., TOMEK, Ľ.: Dynamika mechanických častí výrobnej linky s gravitačním zásobníkom. AT&P journal 12/2006, ss. 73-79

[2] BORŠČ, M.: Modelling and simulation as means of mechatronic design. In State-ofthe-Art in Mechatronics, Editors: Maga, D., Bauer, P., Simulation Research Press, Alphe aan den Rijn, 2007, ISBN 978-90-807898-2-1, ss. 143-160

[3] COUTINHO, J. S.: Advanced Systems Development Management, John Wiley, New York 1977

[4] CRAIG, K.: Mechatronic System Design: Modeling, Design, and Control Integration. An Introduction. Rensselaer Polytechnic Institute, Department of Mechanical, Aerospace, & Nuclear Engineering, 2006, http://mechatronics.rpi.edu

[5] GMITERKO, A.: Prostriedky na podporu konštrukčného procesu v mechatronike vo fáze koncipovania AT&P journal PLUS6 2005 Str.79-86 ISSN 1336-5010

[6] JURIŠICA, L., VÍTKO, A.: Návrh mechatronických systémov. AT&P Journal. 1998, č. 12, s. 61. ISSN 1335-2237.

[7] KRUŤKO, P.D.: Obratnyje zadači dinamiki upravljajemych sistem. NAUKA, Moskava, 1987,

[8] MAIXNER, M. a kol.: Mechatronika, Computer Press, Brno, 2006, ISBN 80-251-1299-3.

[9] LINDEMANN, U.: Produktentwicklung und Konstruktion - Der Lebenszyklus mechatronischer Produkte, die Vorlesungen, Technische Universität, Munchen 2007.

[10] PODURAJEV, J.V.,KULEŠOV, V.S.: Pincipy postrojenija i sovremennyje tendenciji razvitija mechatronnych sistem. Mechatronika №1, 2000.

[11] PUPKOV, K.A.: Intelektuaľnyje sistemy v mechatronike. Moskovskij gosudarstvennyj techničeskij universitet imeni N.E. Baumana, Moskva
http://iu1.bmstu.ru/research/res20.htm

[12] SHETTY, D., KONDO, J., CAMPANA, C., KOLK, R.A.: Real Time Mechatronic Design Process for Research and Education. Proceedings of the 2002 American Society for Engineering Education Annual Conference & Exposition, Hartford, Farmington, 2002.
http://www.ni.com/pdf/academic/us/journals/lv02_53.pdf

[13] THRAMBOULIDIS, K.: Model Integrated Mechatronics –Towards a new Paradigm in the Development of Manufacturing Systéme. IEEE Transactions on Industrial Informatics, vol. 1, No. 1. February 2005, http://seg.ee.upatras.gr/MIM

[14] VDI Richtlinie 2206: Entwicklungsmethodik für mechatronische Systeme. Verein Deutscher Ingenieure, Düsseldorf, Beuth Verlag, GmbH, Berlin, Juni 2004.

Abstract

This article dealt of methodology basis and methods for mechatronic systems design and development. We presents flexible method for mechatronic system development - VDI 2206. A special attention is focused at the model based mechatronic system design and development. Conceptual system design, mechanical parts dynamics analyse and software creating for control of mechatronic system as key problems of mechatronic design are characterized and aproarchs to their solution are described.

Michal Boršč, prof. Ing., CSc.

Trenčianská univerzita Alexandra Dubčeka v Trenčíne
Fakulta mechatroniky
Katedra mechatronických systémov
Pri parku 19
911 06 Trenčín
Tel.: 00421 32 7417544
E-mail borsc@tnuni.sk

Mechatronické subsystémy vozidiel pre zlepšenie jazdných vlastností a bezpečnosti cestných vozidiel

Ľubica Miková, Rastislav Baláž, Mária Ádiová

Abstrakt
Súčasné automobily sa vyznačujú skvelými výkonmi, jazdnými vlastnosťami, komfortom, dizajnom ale aj hospodárnosťou prevádzky a to všetko s ohľadom na životné prostredie. Pri ich výrobe sú použité najmodernejšie technológie a materiály zo všetkých oblastí výskumu ako napr. elektrotechnika, chémia atď.

Kľúčové slová: jazdné vlastnosti, automobily, mechatronické subsystémy vozidiel

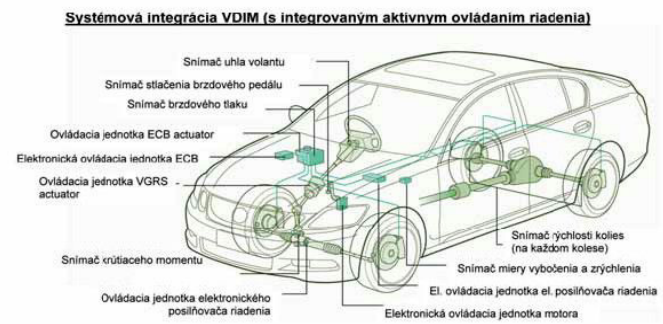
Úvod

Neustály rozvoj automobilového priemyslu vedie k implementácií stále pokrokovejších technológii do súčasných automobilov. Nároky spotrebiteľov na kvalitu a ponúkaný komfort čoraz viac narastajú. V dobách nedávno minulých vari hlavným kritériom k úplnej spokojnosti bola dokonalá funkčnosť počas doby prevádzky vozidla. Dnešný spotrebiteľ si pri výbere automobilu vo výrazne väčšej miere uvedomuje možné následky užívania automobilu. To vedie k zvýšeným nárokom, predovšetkým na bezpečnosť automobilu, či už pasívnu alebo aktívnu. V neposlednom rade narastá environmentálne cítenie spoločnoti, teda aj kritickejší pohľad na produkciu škodlivých látok zaťažujúcich životné prostredie.

Vrátením sa niekoľko desaťročí späť zistíme, že nielen v automobilovom priemysle pri riešení drvivej väčšiny funkčných celkov boli využívane výhradne komponenty, ktorých funkčnosť bola založená na mechanických princípoch. S postupnými nárokmi na bezpečnosť sú však mnohé problematické uzly týmto klasickým prístupom nezvládnuteľné k úplnej spokojnosti. V týchto prípadoch sa otvára priestor na skĺbenie mechaniky a elektroniky. Touto kombináciou sa nám otvárajú častokrát neobmedzené možnosti pri vývoji a zdokonaľovaní vozidiel.

1. Systémová integrácia

Automobily obsahujú veľké množstvo elektroniky a na zistenie chyby vyžadujú čoraz komplikovanejšie diagnostické prístroje. Človek ako omylný tvor je sledovaný (jeho reakcie) mnohými komplikovanými systémami, ktoré ho majú upozorniť na možné nebezpečenstvo poprípade čo najviac eliminovať alebo pomôcť zredukovať dôsledky jeho nesústredenosti. Okrem týchto systémov, ktoré vodičovi pomáhajú alebo zmenšujú následky dopravných nehôd sa v moderných automobiloch používajú systémy podporujúce prevádzku vozidla. Hlavnou výhodou používania elektroniky v automobiloch je zníženie celkovej nehodovosti. Jej hlavná nevýhoda je poruchovosť, ktorá však klesla niekoľkonásobne ako tomu bolo pred zhruba 20 rokmi.[1]



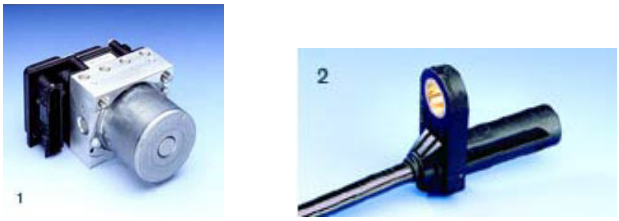
Obr.1 Systémová integrácia [1]
Fig.1 Integration system

1.1 ABS

ABS(Anti-skid Brake System) je historicky prvým zariadením pomáhajúcim riešiť problémy bezpečnosti automobilov.

Základná myšlienka protiblokovacej regulácie spočíva v trecích vlastnostiach pneumatiky na povrchu vozovky. Pri brzdení vznikajú v dotykovej ploche pneumatiky s vozovkou v dôsledku prenášanej pozdĺžnej trecej sily deformácie a prekĺzavanie jednotlivých častí behúňa pneumatiky. Výsledkom toho je, že sa koleso pri brzdení otáča pomalšie, ako by zodpovedalo rýchlosti jazdy vozidla. Koleso sa otáča so sklzom, presnejšie s merným sklzom. Systém ABS zabezpečuje, aby ani pri maximálnom brzdení (maximálnom tlaku vodiča na brzdový pedál) merný sklz kolesa nepresiahol uvedenú oblasť 15 až 30 %. Tým je zaručené, že vozidlo brzdí s maximálnou účinnosťou(najkratšia brzdná dráha), a navyše prenesiteľné bočné sily sú v tejto oblasti merného sklzu ešte dostatočne veľké(oproti stavu zablokovaného kolesa), aby umožňovali aj korekcie smeru jazdy volantom(pri nutnosti obísť počas brzdenia prekážku v dráhe vozidla). [1]

Obr.2 Jednotlivé časti systému ABS v automobile [1]
Fig.2 Individual parts of system ABS in car



a) **b)**

Obr.3 a) hydraulický modulátor s riadiacou jednotkou[1]
b) senzor rýchlosti kolesa [1]
Fig.3 a) hydraulic modulator with control unit
b) wheel speed sensor

1.2 ASR

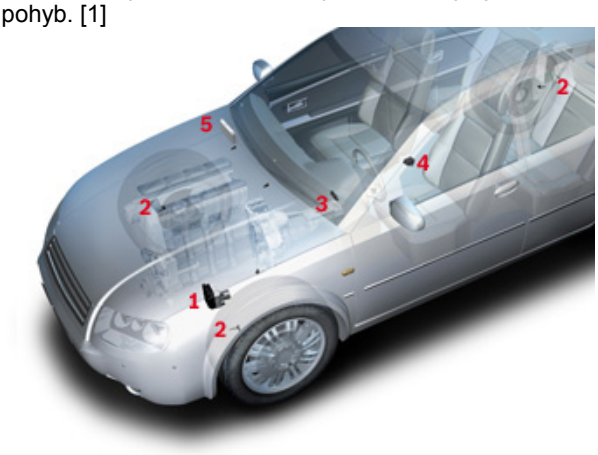
Ďalším vývojovým krokom je regulácia preklzu hnacích kolies pri akcelerácii vozidla, regulovaním veľkosti krútiaceho momentu motora, známa pod skratkou ASR (Anti Skid Regulation). Systém ASR musí zabrániť pretáčaniu hnacích kolies automobilu pri rozjazde do kopca (pri prednom pohone). Preklzujúce kolesá totiž, podobne ako pri brzdení zablokované kolesá, sú schopné prenášať len malé bočné sily, čím sa výrazne zhoršuje smerová stabilita vozidla. Preklzovanie kolies okrem toho spôsobuje väčšie opotrebenie pneumatík aj hnacieho ústrojenstva (napr. diferenciálu). Aby mohol systém ASR zasiahnuť a realizovať zmenšenie hnacieho momentu na preklzujúcich sa kolesách nezávisle na tom, akou silou vodič tlačí na plynový pedál, musí byť vozidlo vybavené tzv. elektronickým riadením výkonu motora EMS (Elektronische Motorleistung Steuerung), tiež nazývaným "elektronický plyn"(EGAS). [1]

1.3 AVB

Elektronický rozdelovač brzdnej sily EBV(Elektronische Bremskraftverteilung) je súčasťou niektorých novších vyhotovení systémov ABS. Preto sa tento veľmi efektívny prvok jazdnej bezpečnosti stáva prístupným aj pre osobné automobily nižších cenových kategórií, ktoré nie sú vybavené kompletným systémom ESP.
Rýchlym a jemným dávkovaním brzdného účinku na jednotlivých kolesách je možné vozidlo, ktoré by inak malo snahu pri brzdení v zákrute vybočovať, včas a tým bezpečne stabilizovať prakticky bez zmeny jeho pôvodnej dráhy. Regulačnou veličinou pre činnosť EBV je rozdiel sklzov jednotlivých kolies vozidla v konkrétnom jazdnom stave. Tento systém umožňuje vylúčiť známe deliče brzdného tlaku používané pri konvenčných hydraulických brzdových sústavách na potlačenie nebezpečenstva zablokovania zadných kolies. Ich nahradením presne dávkovanou brzdnou silou sa darí zväčšiť brzdný účinok zadných kolies vozidla a tým aj celkovú účinnosť brzdenia. [1]

1.4 ESP

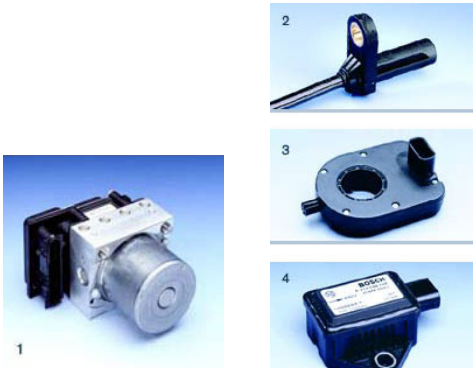
Systém dynamickej stabilizácie vozidla je známy pod skratkou ESP (Electronic Stability Program). Zatiaľ čo systémy ABS a ASR umožňujú regulovať sklz kolesa len v pozdĺžnom smere (pri brzdení alebo akcelerácii), ESP reguluje sklz kolesa aj v priečnom smere (uhol smerovej odchýlky kolesa pri pôsobení bočnej sily). Príliš veľký bočný sklz kolies spôsobí stratu bočného vedenia, vybočenie vozidla z požadovaného smeru a tým zhoršenie jeho smerovej stability. Systém ESP zlepšuje smerovú stabilitu vozidla pri prejazde zákrutou a súčasne ďalej zmenšuje nebezpečenstvo šmyku pri brzdení a zrýchľovaní. Stabilizácia jazdy vozidla sa realizuje samočinnými zásahmi do bŕzd jednotlivých kolies. Ak systém zaregistruje pomocou snímačov priečne dynamický kritický stav jazdy vozidla, nastane pribrzdenie príslušných kolies, čím sa vytvorí krútiaci moment okolo zvislej osi vozidla, ktorý kompenzuje jeho nežiaduci pohyb. [1]



Obr.4 Jednotlivé časti systému ESP v automobile [1]
Fig.4 Individual parts of system ESP in car

Prieskumy ukázali, že v Dánsku a Švédsku má najviac nových automobilov stabilizačný systém, pritom paradoxne sa v týchto krajinách jazdí najdisciplinovanejšie. Naopak Malta a Írsko skončili medzi poslednými s nízkym podielom ESP na štandardnej výbave hlavne v triede malých rodinných áut. [2]

ESC sa predáva na trhu pod rôznymi obchodnými názvami: môže sa nazývať ESP (Electronic Stability Programme, čiže elektronický stabilizačný program), DSC (Dynamic Stability Control, čiže dynamická kontrola stability), VSA (Vehicle Stability Assist, čiže podpora stability vozidla), alebo VSC (Vehicle Stability Control, čiže regulácia jazdnej stability). [3]



a) b), c), d)

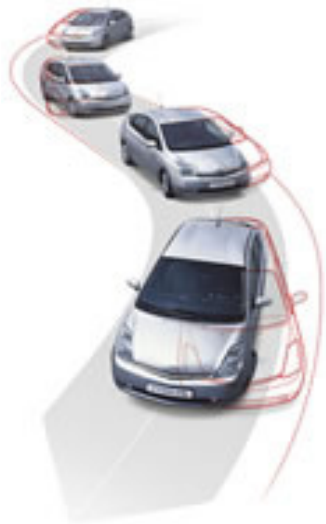
Obr.5 a) hydraulický modulátor s riadiacou jednotkou[1]
b) senzor rýchlosti kolesa [1]
c) senzor uhla natočenia volantu [1]
d) senzor priečneho zrýchlenia [1]
Fig.5 a) hydraulic modulator with control unit
b) wheel speed sensor
c) sensor of steer angle
d) sensor of side acceleration

1.5 MSR

Systém ASR je možné doplniť o reguláciu brzdného momentu motora MSR (Motorschleppmomentregelung). Pri zaradení nižšieho prevodového stupňa(v prípade ručne ovládanej , ale aj automatickej prevodovky), alebo pri prudkom uvoľnení brzdového pedála na šmykľavej vozovke môže totiž nastať vplyvom brzdného účinku motora veľký sklz hnacích kolies a tým zhoršenie smerovej stability vozidla. MSR musí ľahkým "pridaním plynu" zväčšiť krútiaci moment, aby sa brzdenie hnacích kolies zmenšilo na hodnotu zaručujúcu stabilitu jazdy. [1]

1.6 TRC

V prípade ostrej akcelerácie alebo prešmykávaniu kolies hnacej nápravy systém TRC (system riadenia trakčnej sily) túto situáciu zaregistruje. Automaticky reaguje výsledkom čoho je pribrzdenie príslušných kolies a zníženie krútiaceho momentu, aby pomohol obnoviť trakciu všetkých kolies. Systém dokáže nastaviť hnaciu silu každého hnacieho kolesa zvlášť tak, aby sa vozidlo dostalo čo najrýchlejšie pod kontrolu. [4]
Systém tak spolu s ABS a EBD pomáha riadiť činnosť vozidla pri zrýchľovaní a spomaľovaní. [5]



Obr.6 Dráha vozdila s použitím a bez použitia TRC [5]
Fig.6 Trajectory of a car with and without using TRC

Záver

Kľúčovými parametrami súčasných automobilov je predovšetkým bezpečnosť celej posádky vozidla. V počiatkoch zdokonaľovania automobilov v tomto smere sa vývojári sústreďovali predovšetkým na minimalizovanie následkov pri samotnej kolízii vozidla, či už s iným vozidlom alebo akoukoľvek prekážkou. Modernejší pohľad na danú problematiku, podobne ako v medicíne, vychádza z tvrdenia „najlepšou obranou je prevencia“. Následky nehôd znížime na najmenšiu možnú mieru predchádzaním a zabránením samotnej kolízie. Tento prístup sa nazýva aktívna bezpečnosť, bezpečnostné systémy vozidla aktívne pôsobia a vypomáhajú vodičovi kolízii zabrániť. Spomínané systémy sa dotýkajú riešenia pasívnej, ale aj aktívnej bezpečnosti.
Pri neustálom vývoji a zdokonaľovaní terajších systémov sa otvárajú neustále nove možnosti zlepšovania a vývoja nových, ktoré umožňujú terajšie nedostatky a problémy vyriešiť.

Poďakovanie

Autori týmto ďakujú Slovenskej grantovej agentúre pre vedu GU VEGA 1/0201/08 „Výskum štruktúr a správania sa modulov mechatronickej mobilnej technickej sústavy na úrovni orgánov a stavebných prvkov za účelom zlepšenia vlastnosti mobilnej technickej sústavy“ a GU VEGA 1/0464/09 „Výskum mechatronických sústav imitujúcich lokomóciu hada v obmedzenom a premenlivom priestore.“.

Literatúra

[1] ISA OWN [online] 2009 [cit. 2008-12-03] Dostupné na internete:<http://www.isa.own.cz/index.html>

[2] AUTO.SME.SK [online] 2009 [cit. 2008-12-15] Dostupné na internete:<http://auto.sme.sk/c/3352805/Na-co-je-v-aute-ESP-Testovali-sme.html>

[3] CHOOSE ESC! [online] 2009 [cit. 2008-12-01] Dostupné na internete:<http://www.chooseesc.eu/download/leaflets/Choose%20ESC%20leaflet_SK%20final.pdf>

[4] AWF SLOVAKIA [online] 2001-2008 [cit. 2008-12-01] Dostupné na internete:< http://www.awf.sk/?id=corolla_verso>

[5] AUTOGRUUP [online] 2008 [cit. 2008-11-12] Dostupné na internete:< http://www.autogrupp.sk>

Abstract

Present cars are characterizet by excellent performace, driving properties, comfort, design but also econnomy of running, all of it has regard for the enviroment. The most modern technology and materials of all study fields (electro-technics, chemistry etc.) are used at theyr manufacture.

Ľubica Miková, Ing.

Technická univerzita v Košiciach
Strojnícka fakulta, Ústav špeciálnych technických vied
Katedra aplikovanej mechaniky a mechatroniky
Letná 9, 042 00 Košice
00421 55 602 584
E-mail: lubica.mikova@tuke.sk

Rastislav Baláž, Ing.

Technická univerzita v Košiciach
Strojnícka fakulta, Ústav špeciálnych technických vied
Katedra aplikovanej mechaniky a mechatroniky
Letná 9
042 00 Košice
00421 55 602 2719
E-mail: rastislav.balaz@tuke.sk

Mária Ádiová, Ing.

Technická univerzita v Košiciach
Strojnícka fakulta, Ústav špeciálnych technických vied
Katedra aplikovanej mechaniky a mechatroniky
Letná 9
042 00 Košice
00421 55 602 2719
E-mail: maria.adiova@tuke.sk

Neuro-predictive controller design based on genetic algorithms

Slavomir Kajan, Ivan Sekaj, Zuzana Dideková

Abstract
Genetic algorithm based neural controller design of a non-linear dynamic system is described. The genetic algorithm represents an optimisation procedure, where the cost function to be minimized comprises the closed-loop simulation of the control process and a selected performance index evaluation. Using this approach the neural model of the process has been optimised from point of view its internal architecture. Next the parameters of the neural predictive controller were optimised in order to become the required behaviour of the control process.

Key words: neural network optimisation, genetic algorithm, neural controller

Introduction

For control of some classes of non-linear dynamic systems with advantage neural predictive controllers are used. This control structure consists from model of the controlled non-linear system and a predictive controller, which optimises the control value in the actual time period under consideration of the controlled system behaviour prediction. In the presented project the system model is realised using artificial neural network. The optimal performance of such control structure requires optimal parameters of the predictive controller and of the neural model. This all requires experience and expert knowledge. Our aim was to propose an approach, which is able to find the optimal (or suboptimal) control configuration of the neuro-predictive system. For that reason an evolution-based approach has been used. Evolutionary search/optimisation approaches are able to construct new control laws and non-intuitive solutions as well. One of the most frequently used evolutionary techniques is the genetic algorithm (GA). Recently, genetic algorithms have been applied in the area of process control for solving a wide spectrum of various optimisation problems in several ways and with several aims [2, 3, 4, 6]. In the proposed project the GA's are used for search for the optimal neural model architecture and for the predictive controller parameter optimisation.

1. Neural controller structure

The block scheme of the used controller structure is depicted in fig.1. Without loss of generality, in case of the controlled process model a MLP (multilayer perceptron) artificial neural network is considered. The number of layers is 3 and the number of neurons in the input and hidden layer and their interconnections are the objects of the design/optimisation process. As the learning rule the off-line version of the Error-back-propagation method with Levenberg–Marquardt modification is considered [1]. The model predictive control method is based on the receding horizon technique [8]. The neural network model predicts the plant response over a specified time horizon. The predictions are used by a numerical optimization routine to determine the control value u(k) in each control period k that minimizes the following performance criterion J over the specified horizon.

$$\Delta u(k) = ?; J \rightarrow \min \tag{1}$$

$$J = \sum_{i=N_1}^{N_2} \left[r(k+i) - \hat{y}(k+i) \right]^2 + \rho . \sum_{i=1}^{N_u} \left[\Delta u(k+i-1) \right]^2 \tag{2}$$

$$u(k) = u(k-1) + \Delta u(k) \tag{3}$$

Here the r is the reference signal, $\hat{y}$ is the controlled value prediction, Δu is the control value change, k is the control step, N_1 is the lower and N_2 the upper output prediction horizon, N_u is control horizon and ρ is a weight constant. The control performance depends on the values N_1, N_2, N_u and ρ. The block scheme of the neural model is in fig.2. The number of inputs y_{k-1}, y_{k-2},…,y_{k-n}, u_k, u_{k-1}, …, u_{k-m} (number of past samples of y and u) and the number of neurons in the hidden layer are normally set by the designer using experience. But often the chosen model architecture is not optimal because of the modelling accuracy and on the other hand from point of view computation time.

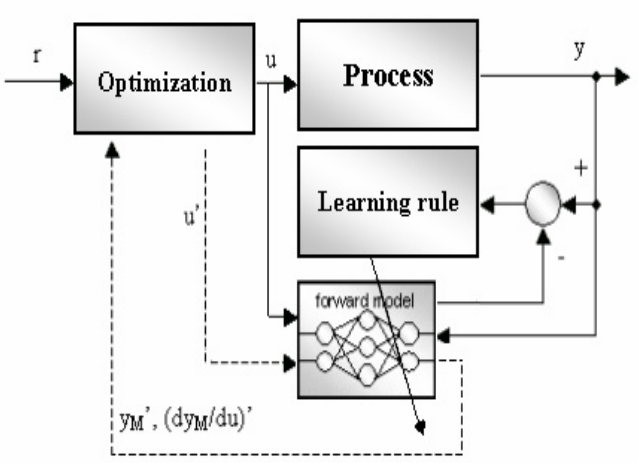


Fig.1 Block scheme of neuro-predictive controller

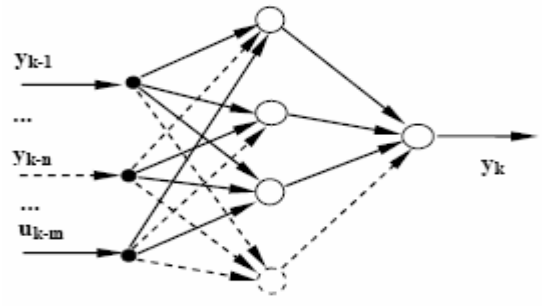


Fig.2 Neural network structure for process modelling

2. Genetic algorithm

Genetic algorithm (GA) is a powerful stochastic-based search/optimisation approach, which mimics the evolution in the nature. It is described in e.g. [2, 3, 4, 6, 7] and others. A general scheme of a GA can be described by following steps (Fig.3):

1. Initialisation of the population of chromosomes (set of randomly generated chromosomes).

2. Evaluation of the cost function (fitness) for all chromosomes.

3. Selection of parent chromosomes.

4. Crossover and mutation of the parents → children.

5. Completion of the new population from the new children and selected members of the old population. Jump to the step 2.

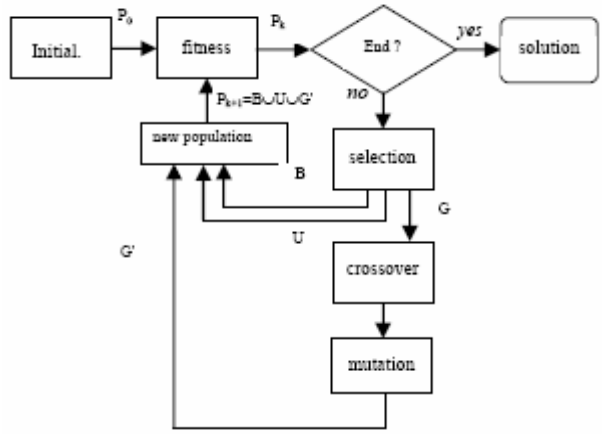


Fig.3 Block scheme of the used genetic algorithm

A block scheme of a GA-based design is in fig.4. Before each cost function evaluation, the corresponding chromosome (genotype) is decoded into network interconnections of the simulation neural model (phenotype) and after the simulation the performance index is evaluated.

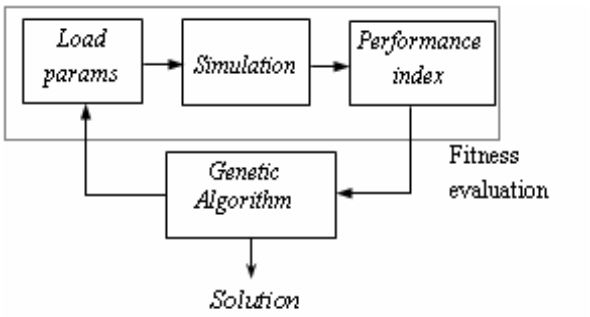


Fig.4 Block scheme of the GA-based neural model design

3. Optimisation of the neural-predictive controller

For the verification of the proposed methods consider the non-linear system, which is described by the differential equation

$$y'' + 0{,}7y' + 0{,}2y + 0{,}3y^3 - u = 0 \qquad (4)$$

The block scheme of the non-linear system (4) is depicted in fig. 5.

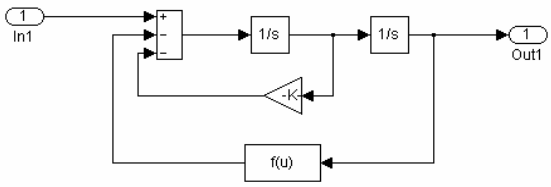


Fig.5 Simulation block scheme of non-linear system

3.1 Optimisation of neural model structure

For modelling of the non-linear system neural model structure with 6 inputs and 10 hidden neurons (Fig. 7) is used. In the case of the neural model the number of neurons in input and hidden layer, and their interconnections have been optimised [5]. The interconnection map of the net is coded into the chromosome of the GA (Tab.1). The length of the chromosome is 70 genes. The genes can be either values 0 or 1, where 1 represents connection between neurons (Tab.2) and 0 no connection.

	Input Neurons					Hidden Neurons					ON
	1	2	.	5	6	7	8	.	15	16	17
1	0	0	.	0	0	0	0	.	0	0	0
2	0	0	.	0	0	0	0	.	0	0	0
.	.	.	.	.	.	.	.	.	.	.	.
5	0	0	.	0	0	0	0	.	0	0	0
6	0	0	.	0	0	0	0	.	0	0	0
7	1	1	.	1	1	0	0	.	0	0	0
8	1	1	.	1	1	0	0	.	0	0	0
.	.	.	.	.	.	.	.	.	.	.	.
15	1	1	.	1	1	0	0	.	0	0	0
16	1	1	.	1	1	0	0	.	0	0	0
17	0	0	.	0	0	1	1	.	1	1	0

Tab.1 The interconnection map of the neural network between neurons

1-10	11-20	.	41-50	51-60	61-70
1 1 1 1	.	1 1	1	1	1 1

Tab.2 The chromosome representation

Over the chromosomes in the population genetic operations crossover and mutation are applied. After these operations unrealizable neural network structures can appear. Hence, in case of input neuron with no connection and hidden neuron with no connection to input or to output such neurons are omitted from the net structure.

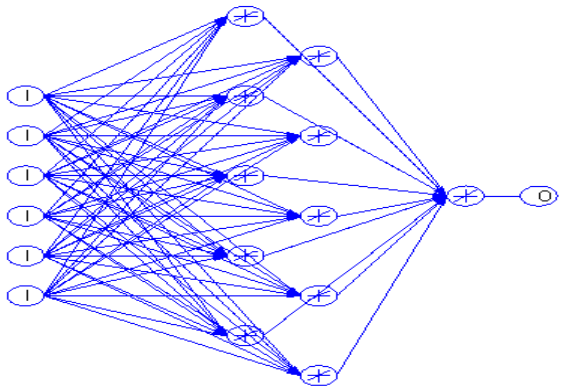


Fig.6 Neural model structure with full connections

For searching optimal neural model structure with GA the used cost function (fitness) was considered in the form

$$F = MSE + (1-\alpha)*(w/wn)*10^{-5} + \alpha*(n/nn)*10^{-5} \quad (5)$$

MSE – mean square error of the neural model (in comparison with the modelled object)

α – weight constant, which was set to 0.7

w – number of weighted interconnections between the input and hidden layer

w_n – maximal number of all interconnections

n – number of neurons in network

n_n – maximal number of neurons in network

10^{-5} – weight constant

The optimised neural network structure is shown in fig. 7. Using testing data the obtained error was MSE=0.4216e^{-6}.

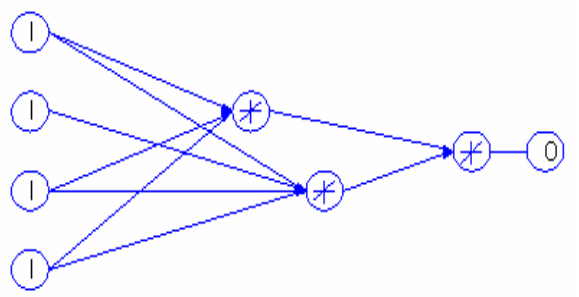


Fig.7 Neural model with optimal structure

3.2 Optimisation of neural controller parameters

The second design step was the search for the predictive controller cost function coefficients N_1, N_2, N_u and ρ. For this purpose another GA has been used, where the fitness consists of the closed-loop simulation and the performance index evaluation (6). An example of a simple performance index is as follows

$$I_{AE} = \int_0^T |e(t)|dt \quad (6)$$

where *e* is the control error. This performance index has been minimised, and it represents the controller performance. The obtained results after these two design steps, which were performed off-line are demonstrated in fig.8. It is the time-response of the controlled system (green) after the reference signal steps (blue). The control value time-response is in fig.9. In fig.10 the trend of the fitness function is depicted, which is the graph of the current best individual of the actual population vs. generation number.

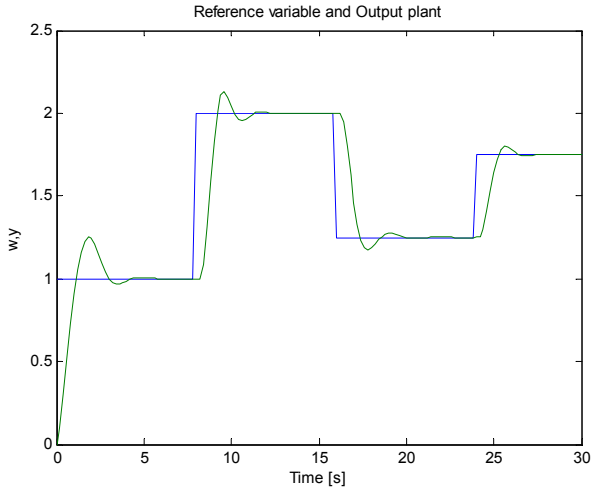


Fig.8 Time-response of the controlled system

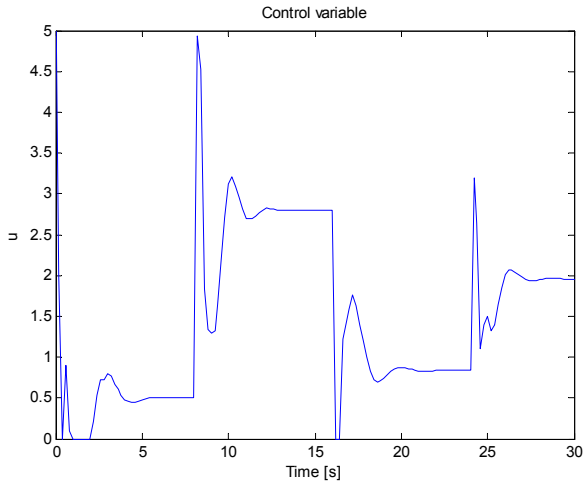


Fig.9 Control value of neural predictive controller

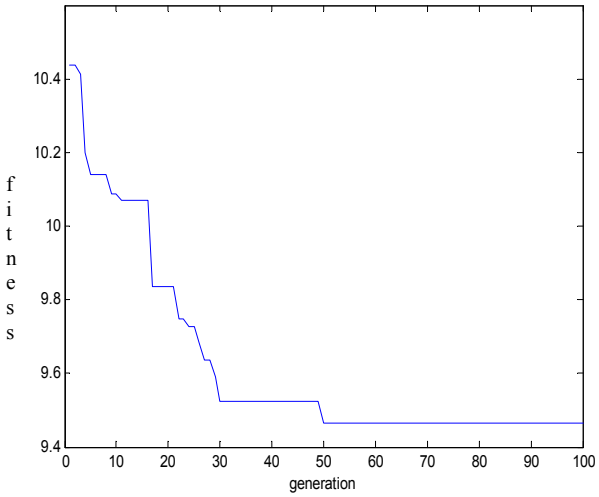


Fig.10 Evolution of the fitness function

Conclusion

Neuro-predictive control is an efficient mean for control of non-linear dynamic systems. To ensure the required performance, the controller has to fulfill some conditions. The first is that the neural-model has sufficient modelling accuracy in the entire working range. The second is the satisfactory parameter setting of the predictive controller. Both conditions require some knowledge or experience from the designer. The goal of the presented project is to replace the need for experience and knowledge and to design/optimize the neuro-predictive controller structure and its parameters using an automatic procedure, which is based on genetic algorithms. Evolutionary (genetic) algorithms can be used for the design and optimisation of various neuro-predictive controllers, which uses various types of neural networks and various types of predictive control algorithms.

Acknowledgement

The work has been supported by the grants agency VEGA no. 1/0544/09 and no. 1/0690/09. This support is very gratefully acknowledged.

References

[1] DEMUTH, H., BEALE, M.: Neural Network Toolbox, For use with Matlab, User's guide, 2003

[2] GOLDBERG, D.E.: Genetic Algorithms in Search, Optimisation and Machine Learning. Addisson-Wesley, 1989

[3] MAN, K.F., TANG, K.S., KWONG, S.: Genetic Algorithms, Concepts and Deign. Springer, 2001

[4] MICHALEWICZ, Z.: Genetic Algorithms + Data Structures = Evolutionary Programs. Springer, 1996

[5] SARTORIS, P.: Generovanie optimálnej štruktúry neurónových sietí pomocou genetických algoritmov, Diploma thesis, FEI STU Bratislava

[6] SEKAJ, I.: Evolučné výpočty a ich využitie v praxi, Iris 2005, Bratilava (in slovak)

[7] SEKAJ, I.: Genetic Algorithm Based Controller Design, In: 2nd IFAC conference Control System Design'03, Bratislava, 2003

[8] SOLOWAY, D., HALEY, P.J.: Neural Generalized Predictive Control, Proceedings of the 1996 IEEE International Symposium on Intelligent Control, 1996, pp. 277-281

Ing. Slavomír Kajan, PhD.

Slovak University of Technology in Bratislava
Faculty of Electrical Engineering
and Information Technology
Institute of Control and Industrial Informatics
Department of Robotics and Artificial Intelligence
Ilkovičova 3
812 19 Bratislava
E-mail: slavomir.kajan@stuba.sk

Doc. Ing. Ivan Sekaj, PhD.

Slovak University of Technology in Bratislava
Faculty of Electrical Engineering
and Information Technology
Institute of Control and Industrial Informatics
Department of Robotics and Artificial Intelligence
Ilkovičova 3
812 19 Bratislava
E-mail: ivan.sekaj@stuba.sk

Ing. Zuzana Dideková

Slovak University of Technology in Bratislava
Faculty of Electrical Engineering
and Information Technology
Institute of Control and Industrial Informatics
Department of Robotics and Artificial Intelligence
Ilkovičova 3
812 19 Bratislava
E-mail: zuzana.didekova@stuba.sk

Identifikácia parametrov asynchrónneho motora genetickým algoritmom

Marián Jančovič, Milan Žalman

Abstrakt
V článku je opísaný spôsob identifikácie parametrov asynchrónneho motora genetickým algoritmom. Genetické algoritmy predstavujú univerzálnu metódu, ktorou je možné riešiť rôzne typy optimalizačných úloh, medzi ktoré možno zaradiť aj identifikáciu parametrov. Identifikácia parametrov motora genetickým algoritmom je založená na generovaní nových riešení pre minimalizáciu zvolenej účelovej funkcie, ktorá porovnáva priebehy prúdov z motora a jeho simulačného modelu s nastaviteľnými parametrami. Asynchrónny motor predstavuje vysoko nelineárny systém, ktorého parametre sa môžu časom meniť, pričom hodnoty týchto parametrov sú dôležité pri návrhu riadenia, stavových pozorovateľov alebo pri konštrukcii modelu motora. Navrhnutá metóda umožňuje identifikovať elektrické a prípadne aj mechanické parametre motora s dostatočnou presnosťou pre návrh regulátorov vektorového riadenia asynchrónneho motora. Okrem simulačných experimentov sú v článku uvedené aj experimenty na reálnom zariadení.

Kľúčové slová: genetický algoritmus, asynchrónny motor, identifikácia parametrov

Úvod

V súčasnosti sa v priemyselných aplikáciách najviac využívajú striedavé asynchrónne motory s kotvou nakrátko (AM) a synchrónne motory s permanentnými magnetmi. Trendom je vývoj rýchlostných alebo polohových vektorovo riadených servopohonov bez mechanických snímačov, kde všetky požadované informácie sú získané z priebehov statorového napätia a prúdu [1].

Pri návrhu vektorového riadenia AM je potrebné poznať parametre AM, pričom kvôli svojej zložitosti je vektorové riadenie citlivé na zmenu týchto parametrov. K zvýšeniu kvality riadenia prispieva použitie identifikačných algoritmov, ktoré umožňuje implementovať adaptívne metódy riadenia. To je dôležité najmä v aplikáciách, kde sú kladené vysoké nároky na presnosť riadenia alebo robustnosť na zmeny parametrov AM. Preto sa nevyhnutnou súčasťou algoritmov riadenia polohových a rýchlostných servopohonov s AM stali pozorovatele stavových veličín a parametrov.

V prípade vektorového riadenia predstavujú stavové veličiny statorové prúdy, magnetický tok rotora a moment motora. Pozorovatele na báze prúdových, napäťových alebo prúdovonapäťových modelov AM potrebujú pre presné pozorovanie týchto veličín správne nastavenie parametrov, ktoré zodpovedajú elektrickým parametrom AM: odpor statora a rotora, indukčnosť statora a rotora, vzájomná indukčnosť. Pri nastavovaní regulátorov elektromagnetického systému AM – generátora momentu je možné využiť identifikované hodnoty uvedených elektrických parametrov. Hodnoty týchto parametrov je potrebné poznať aj pri použití pozorovateľov uhlovej rýchlosti v štruktúre riadenia bez snímača rýchlosti, ak sú navrhnuté na základe modelu AM. Pri nastavovaní parametrov regulátora rýchlosti v rýchlostnej štruktúre je vhodné poznať aj mechanické parametre AM, ako sú moment zotrvačnosti, moment záťaže, suché a viskózne trenie.

Štandardné metódy identifikácie slúžia na identifikáciu lineárnych systémov. Pri identifikácii nelineárneho systému je potrebné ho najskôr zlinearizovať. S identifikáciou nelineárnych systémov je spojená mimoriadna zložitosť, najmä keď neexistujú počiatočné informácie alebo podrobnosti o štruktúre modelu. V takomto prípade sa ukazuje ako vhodné použitie genetických algoritmov (GA), ktoré sú na rozdiel od iných metód univerzálne pre identifikáciu lineárneho aj nelineárneho modelu systému.

Pri parametrickej identifikácii vyberá genetický algoritmus skupinu parametrov, ktorá minimalizuje zvolené kritérium, t.j. chromozóm, ktorý najvernejšie kopíruje správanie sa identifikovaného systému. Postup identifikácie parametrov pomocou GA pozostáva z dosadenia jednotlivých kandidátov do modelu, so simulácie tohto modelu, vyčíslenia kriteriálnej funkcie na základe získaných údajov, aplikovania genetických operácií pre vytvorenie nových kandidátov a výberu najvhodnejšieho kandidáta s požadovanou hodnotou kriteriálnej funkcie alebo z konečného počtu kandidátov. Jednou z výhod GA je, že namiesto deterministických pravidiel sú na hľadanie riešenia použité stochastické operátory. Okrem toho GA berú do úvahy veľa bodov prehľadávaného priestoru súčasne a nie iba jeden bod, z čoho vyplýva, že sú schopné vyviaznuť z okolia lokálneho extrému a približovať sa ku globálnemu extrému [2]. Pre identifikáciu parametrov AM je dôležité, že GA dokážu identifikovať parametre s dobrou presnosťou aj pri zašumených a kvantovaných signáloch. Na druhej strane však zložitosť systému a počet hľadaných parametrov vplýva na počet potrebných simulácií systému a to spôsobuje časovú náročnosť algoritmu.

1. Identifikácia parametrov AM genetickým algoritmom

Úlohou GA je nájsť optimálne hodnoty vybraných parametrov AM. V našom prípade sme používali matematický model AM v súradnicovom systéme statora uvedený v [3], v ktorom pre zložky statorového prúdu platia vzťahy:

$$i_{s\alpha} = \frac{1}{R_1 + L'_s s}\left(u_{s\alpha} + \frac{k_r}{T_r}\psi_{r\alpha} + k_r\psi_{r\beta}\omega\right)$$

$$i_{s\beta} = \frac{1}{R_1 + L'_s s}\left(u_{s\beta} + \frac{k_r}{T_r}\psi_{r\beta} - k_r\psi_{r\alpha}\omega\right) \quad (1)$$

kde

$$R_1 = R_s + R_r\frac{L_m^2}{L_r^2},\ L'_s = \sigma L_s,\ \sigma = 1 - \frac{L_m^2}{L_s L_r},\ T_1 = \frac{L'_s}{R_1},$$

$$T_r = \frac{L_r}{R_r},\ k_r = \frac{L_m}{L_r} \quad (2)$$

Zložky magnetického toku rotora v statorovom súradnicovom systéme vyjadrujú nasledovné vzťahy:Rotorová rovnica má tvar:

$$\psi_{r\alpha} = \frac{L_m}{1 + sT_r}\left(i_{s\alpha} - \omega\psi_{r\beta}\frac{T_r}{L_m}\right)$$

$$\psi_{r\beta} = \frac{L_m}{1 + sT_r}\left(i_{s\beta} + \omega\psi_{r\alpha}\frac{T_r}{L_m}\right) \quad (3)$$

Pre moment motora platí vzťah:

$$M_m = \frac{3}{2}p'\frac{L_m}{L_r}\Im\left(\hat{\psi}_r{}^*\hat{i}_s\right) = \frac{3}{2}p'\frac{L_m}{L_r}\left(\psi_{r\alpha}i_{s\beta} - \psi_{r\beta}i_{s\alpha}\right) \quad (4)$$

Mechanický systém je opísaný pohybovou rovnicou:

$$\frac{d\omega_m}{dt} = \frac{1}{J}\left(M_m - M_z\right) \quad (5)$$

pričom

$$\omega = p'\omega_m \quad (6)$$

Význam jednotlivých symbolov je nasledovný:

$i_{s\alpha}$ – reálna zložka statorového prúdu,

$i_{s\beta}$ – imaginárna zložka statorového prúdu,

$u_{s\alpha}$ – reálna zložka statorového napätia,

$u_{s\beta}$ – imaginárna zložka statorového napätia,

$\psi_{r\alpha}$ – reálna zložka statorového magnetického toku rotora,

$\psi_{r\beta}$ – imaginárna zložka statorového magnetického toku rotora,

ω – elektrická uhlová rýchlosť,

ω_m – mechanická uhlová rýchlosť,

T_m – moment motora,

T_z – moment záťaže,

p' – počet pólových dvojíc,

R_s – odpor statora,

R_r – odpor rotora,

L_s – indukčnosť statora,

L_r – indukčnosť rotora,

L_m – vzájomná indukčnosť,

J – moment zotrvačnosti.

Hodnoty hľadaných parametrov AM predstavujú gény GA a spolu tvoria chromozóm genetického algoritmu (**r**). Čím väčší počet génov (parametrov) obsahuje chromozóm GA, tým ťažšiu úlohu predstavuje nájdenie správnych hodnôt týchto parametrov, z čoho vyplýva potreba väčšieho počtu chromozómov v populácii, väčšieho počtu generácií riešení a tým aj predĺženie výpočtového času. Na zredukovanie počtu identifikovaných parametrov budeme preto uvažovať zjednodušujúci predpoklad:

$$L_r = L_s \quad (7)$$

keďže tieto parametre sú lineárne závislé [4], [5] a rozdiely medzi ich hodnotami bývajú zvyčajne veľmi malé, resp. žiadne.

Nami navrhnutá metóda umožňuje identifikovať parametre v dvoch režimoch AM a to v zabrzdenom stave AM a pri rozbehu AM.

V zabrzdenom stave AM, po privedení jednosmerného napätia na zložku $u_{s\alpha}$ pričom zložka $u_{s\beta}$ je nulová, môžeme identifikovať jeho elektrické parametre. Chromozóm GA má nasledovný tvar:

$$\mathbf{r} = \left(R_s, R_r, L_s, L_m\right) \quad (8)$$

Pri rozbehu motora je možné identifikovať aj jeho mechanické parametre. Chromozóm v tomto prípade obsahuje nasledovné parametre:

$$\mathbf{r} = \left(R_s, R_r, L_s, L_m, J\right) \quad (9)$$

Na reprezentáciu parametrov AM je vhodné použiť reálnečíselné hodnoty. Pre tieto hodnoty je potrebné definovať prípustné intervaly hodnôt, t.j. určiť rozsahy jednotlivých parametrov. Čím užšie vieme ohraničiť prehľadávaný priestor, tým viac urýchlime riešenie. A naopak, príliš veľkým obmedzeným rozsahov môžeme vylúčiť optimálne riešenie.

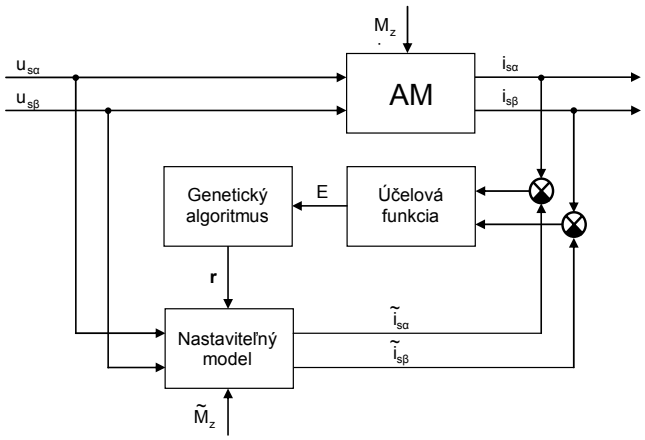
Rýchlosť riešenia ovplyvňujeme aj voľbou veľkosti populácie. Čím je populácia väčšia, tým viac vyčíslení účelovej funkcie je potrebných realizovať a tým dlhšie trvá realizácia GA. Avšak na druhej strane, zväčšovaním populácie sa zväčšuje počet potenciálnych riešení a dáva sa tým priestor pre väčšiu diverzitu medzi jedincami čím sa zabezpečuje širší záber prehľadávaného priestoru a znižuje riziko uviaznutia v lokálnom extréme. Vzhľadom na veľkosť chromozómov, sme pri navrhovaných metódach zvolili populácie s 50 chromozómami, t.j. s 50 kombináciami hľadaných parametrov AM.

Nastavenie hodnôt génov začiatočnej populácie chromozómov sa môže realizovať podľa apriórnych vedomostí o procese, čím by sa urýchlilo riešenie. Napríklad ak ide o identifikáciu zmeny hodnôt, môže sa v počiatočnej populácii nastaviť jeden alebo viac chromozómov pôvodnými hodnotami. Keďže však v prípade simulačných experimentov budeme poznať skutočné hodnoty parametrov, použijeme inicializáciu náhodnými hodnotami z definovanej oblasti parametrov.

Vhodnosť jednotlivých chromozómov sa vyhodnocuje účelovou funkciou, ktorá porovnáva výstupy identifikovaného systému a jeho modelu, do ktorého GA dosádza hodnoty z chromozómov (obr. 1). Minimalizáciou účelovej funkcie sa dosiahne priblíženie identifikovaných hodnôt parametrov systému k ich skutočným hodnotám. V našom prípade využívame pri identifikácii model AM, ktorého vstupnými veličinami sú zložky statorového napätia $u_{s\alpha}$ a $u_{s\beta}$ a výstupnými zložky statorového prúdu $i_{s\alpha}$ a $i_{s\beta}$ (obr. 1). V každej perióde vzorkovania sa zaznamenávajú ich hodnoty a na základe týchto hodnôt sa potom vypočítava kriteriálna funkcia, ktorá má tvar:

$$E = \sum_{k=1}^{N}\left(i_{s\alpha i} - \tilde{i}_{s\alpha i}\right)^2 + \sum_{k=1}^{N}\left(i_{s\beta i} - \tilde{i}_{s\beta i}\right)^2 \quad (10)$$

kde $i_{s\alpha i}$ a $i_{s\beta i}$ sú hodnoty *k*-tých vzoriek zložiek statorového prúdu z identifikovaného systému, $\tilde{i}_{s\alpha i}$ a $\tilde{i}_{s\beta i}$ sú hodnoty *k*-tých vzoriek zložiek statorového prúdu z modelu, *N* je počet vzoriek a *E* je hodnota účelovej funkcie.



Obr. 1 Bloková schéma identifikácie parametrov AM genetickým algoritmom

Fig. 1 Scheme of parameter identification of induction motor using GA

Po vyčíslení kriteriálnej funkcie všetkých chromozómov nasleduje na základe týchto hodnôt výber chromozómov do novej populácie a výber chromozómov do pracovnej skupiny, nad ktorou sú potom realizované genetické operácie kríženia a mutácie. Priamym skopírovaním najlepšieho chromozómu, prípadne niekoľkých chromozómov do novej populácie sa zabezpečuje konvergencia riešenia. Viacnásobným výberom najlepších chromozómov do pracovnej skupiny zabezpečíme kvalitné hodnoty pre následné kríženie a mutáciu a tým urýchlime konvergenciu k optimálnym hodnotám. To by však mohlo viesť k uviaznutiu v lokálnom minime, preto náhodným výberom viacerých chromozómov do pracovnej skupiny dávame priestor aj pre väčšiu rôznorodosť pri realizácii genetických operácií.

Nastavením výberu chromozómov do novej populácie a spôsobu kríženia a mutácie môžeme ovplyvniť rýchlosť a efektívnosť riešenia, ale aj riziko uviaznutia v lokálnom extréme. Existuje veľmi veľa možností a kombinácií týchto nastavení. Ak by sme použili iba operáciu kríženia, po vyčerpaní všetkých kombinácií by sme už nedostávali nové parametre a tým ani lepšie riešenia. Tie zabezpečuje mutácia, ktorá spočíva v pridaní náhodnej hodnoty do aktuálnej hodnoty parametra. Klasická mutácia nahrádza aktuálnu hodnotu hodnotou z ohraničeného intervalu, ktorý bol definovaný ako oblasť možných hodnôt pre daný parameter. Keď však vzdialenosť medzi aktuálnou hodnotou parametra a jeho optimálnou hodnotou je malá v porovnaní so vzdialenosťou medzi hornou a dolnou hranicou intervalu, možnosť pridať novú vhodnú hodnotu sa stane veľmi malou [6]. V tomto prípade by genetický algoritmus konvergoval rýchlo až do určitého okolia optimálneho riešenia a potom by trvalo veľa času urobiť zvyšok. Riešenie môže spočívať v zmene šírky tohto intervalu alebo v použití aditívnej alebo multiplikatívnej mutácie, ktorá k aktuálnej hodnote pripočítava, resp. ju vynásobí náhodnou hodnotou zo zvoleného rozsahu. Preto, aby sme sa vyhli problémom s rýchlosťou konvergencie najmä v záverečnej fáze algoritmu, keď sú už malé rozdiely medzi aktuálnymi a optimálnymi hodnotami, budeme používať multiplikatívnu mutáciu.

GA môžeme ukončiť buď pri dosiahnutí určitej minimálnej hodnoty účelovej funkcie alebo po dosiahnutí stanoveného počtu generácií. V našom prípade sme si zvolili ukončenie GA po konečnom počte 500 generácií.

Postupnosť jednotlivých krokov nami navrhnutého GA pre identifikáciu parametrov AM môžeme zhrnúť do nasledovných krokov:

1. Na začiatku sa náhodne vygeneruje populácia 50 chromozómov v danom intervale.
2. Postupne sa priradí všetkých 50 chromozómov do nastaviteľného modelu. Na vstup modelu je pripojené vstupné napätie. Pre každý chromozóm sa po pripojení statorového napätia zaznamenajú počas 2-sekundovej simulácie odozvy prúdu statora, pomocou ktorých sa vyhodnotí zvolená kriteriálna funkcia. Pri použití periódy vzorkovania 0,25 ms, to znamená použitie 8000 vzoriek signálov.
3. Do novej populácie sa prekopíruje chromozóm s najnižšou a chromozóm s druhou najnižšou hodnotou kriteriálnej funkcie.
4. Do pracovnej skupiny sa prekopíruje 5-krát chromozóm s najnižšou, 3-krát chromozóm s druhou najnižšou, 3-krát chromozóm s treťou najnižšou, 2-krát chromozóm so štvrtou najnižšou, 2-krát chromozóm s piatou najnižšou a 1-krát chromozóm so šiestou najnižšou hodnotou kriteriálnej funkcie.
5. Do pracovnej skupiny sa skopíruje 32 náhodne vybraných chromozómov.
6. Na pracovnú skupinu sa aplikuje operácia jednobodového kríženia, pričom výber párov pre kríženie je náhodný.
7. Na pracovnú skupinu sa aplikuje operácia multiplikatívnej mutácie s pravdepodobnosťou mutácie génu 0,1, ktorá hodnotu vybraného génu vynásobí náhodným číslom z rozsahu (0; 2).
8. Na pracovnú skupinu sa aplikuje operácia medziľahlého kríženia s náhodným výberom krížených dvojíc s parametrom $\alpha = 1{,}25$. Táto operácia môže byť použitá pri reálne-číselnom kódovaní, pričom predstavuje istú kombináciou kríženia a mutácie. Nové chromozómy sú vytvárané podľa rovnice:

$$p = r_1 + \alpha(r_2 - r_1) \qquad (11)$$

   kde *p* je nový chromozóm, r_1 a r_2 sú rodičovské chromozómy a α je náhodné číslo z intervalu $0{,}25 < \alpha < 1{,}25$.
9. Pracovná skupina je presunutá do novej populácie.

Algoritmus sa opakuje s novou populáciou až kým nie je dosiahnutých 500 generácií. Potom je chromozóm s najnižšou hodnotou kriteriálnej funkcie vyhodnotený ako najlepšie riešenie.

2. Experimenty na simulačnom modeli

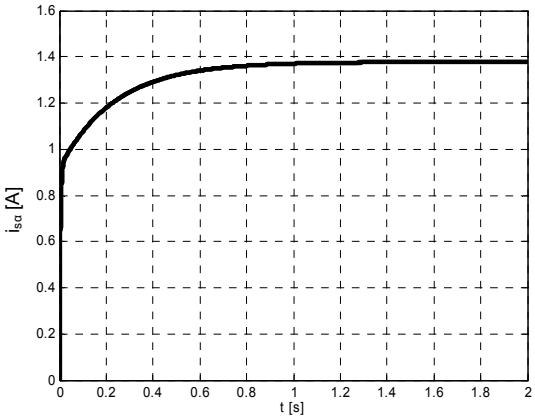
Simulačné overenie navrhnutého algoritmu sme realizovali v prostredí Matlab/Simulink. V tomto prípade bol nastaviteľný model AM zhodný s jeho simulačným modelom, čo nám umožňuje posúdiť schopnosť navrhnutého genetického algoritmu nájsť správne hodnoty parametrov tohto modelu.

Cieľom nášho experimentu bolo overiť použitie navrhnutého genetického algoritmu pri rôznych hodnotách amplitúdy vstupného napätia $u_{s\alpha}$ v režime so zabrzdeným AM. Identifikovali sme hodnoty parametrov chromozómu (8), pričom oblasť riešení jednotlivých parametrov sme definovali nasledovnými intervalmi:

$$R_s \in \langle 1;10 \rangle \ , \ R_r \in \langle 1;10 \rangle \ , \ L_s \in \langle 0{,}1;1 \rangle \ , \ L_m \in \langle 0{,}1;1 \rangle$$

Na obr. 2 je znázornená ukážka priebehu prúdu $i_{s\alpha}$ pre vstupné napätie $u_{s\alpha}$=10 V. Výsledky z identifikácií pre jednotlivé vstupné napätia spolu s vypočítanými odchýlkami od skutočných hodnôt sú uvedené v tab. 1.

Keďže pri tomto experimente sa identifikovali parametre lineárneho modelu AM, očakávali sme rovnakú presnosť identifikácie pri všetkých napätiach. Táto presnosť sa však zvyšovala so zvyšovaním použitého vstupného napätia, pričom pri najväčšom vstupnom napätí 30V sa opäť znížila. Z výsledných hodnôt účelovej funkcie možno vidieť, že pri vstupnom napätí 20V bola dosiahnutá nižšia hodnota ako pri použití menších napätí. Príčinou môže byť, že pri pevne stanovenom počte generácií nemusí byť dosiahnutá vždy rovnaká presnosť, najmä ak je tento počet nepostačujúci. Napriek tomu možno konštatovať, že pri zvolených napätiach bola dosiahnutá veľmi dobrá presnosť identifikácie.



Obr. 2 Priebeh prúdu $i_{s\alpha}$ pre vstupné napätie $u_{s\alpha}$=10V

Fig. 2 Stator current $i_{s\alpha}$ response for input voltage $u_{s\alpha}$=10V

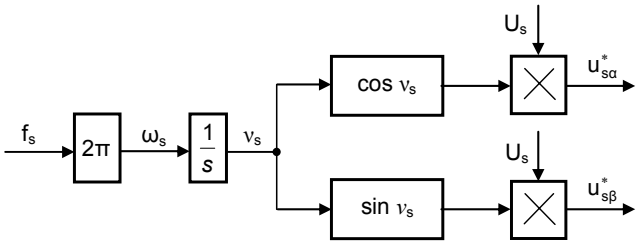
		R_s	R_r	L_s	L_m
	skutočná hodnota	**7,608**	**3,700**	**0,6015**	**0,5796**
$u_{s\alpha}$ =5V	$\tilde{\mathbf{r}}$	7,588	3,657	0,6223	0,6000
E=9,4.10^{-3}	δ	0,3%	1,2%	3,5%	3,5%
$u_{s\alpha}$ =10V	$\tilde{\mathbf{r}}$	7,611	3,701	0,5990	0,5773
E=6,5.10^{-4}	δ	0,0%	0,0%	0,4%	0,4%
$u_{s\alpha}$ =20V	$\tilde{\mathbf{r}}$	7,607	3,699	0,6024	0,5804
E=3,0.10^{-4}	δ	0,0%	0,0%	0,1%	0,1%
$u_{s\alpha}$ =30V	$\tilde{\mathbf{r}}$	7,613	3,706	0,5949	0,5731
E=3,5.10^{-2}	δ	0,1%	0,2%	1,1%	1,1%

Tab. 1 Porovnanie výsledkov identifikácie parametrov zabrzdeného AM genetickým algoritmom pri rôznych hodnotách vstupného napätia $u_{s\alpha}$ s ich skutočnými hodnotami

Ďalším experimentom bola identifikácia parametrov AM pri rozbehu motora, pri ktorej sme hľadali parametre chromozómu (9). V tomto prípade už priebehy prúdov ovplyvňujú aj mechanické parametre motora. Medzi hľadané parametre sme preto mohli zaradiť aj moment zotrvačnosti, pričom vplyv momentu záťaže sme zanedbávali. Oblasť riešení jednotlivých parametrov sme definovali nasledovnými intervalmi:

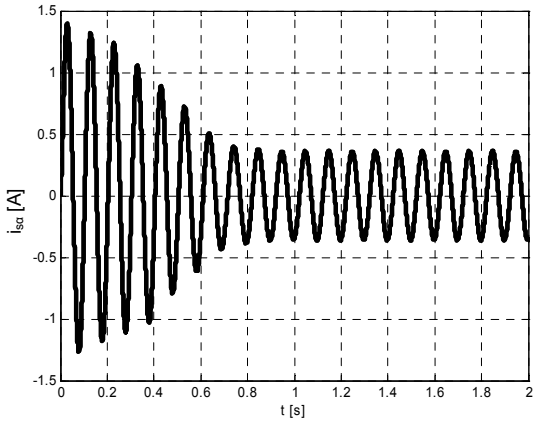
$$R_s \in \langle 1;10 \rangle \ , \ R_r \in \langle 1;5 \rangle , \ L_s \in \langle 0{,}1;1 \rangle , \ L_m \in \langle 0{,}1;1 \rangle ,$$
$$J \in \langle 0{,}0001;0{,}1 \rangle$$

Hodnoty zložiek prúdu pre účelovú funkciu boli zaznamenávané po privedení striedavých priebehov zložiek statorového napätia $u_{s\alpha}$ a $u_{s\beta}$ s rovnakou amplitúdou U_s a frekvenciou f_s ale s fázovým posunom -π/2 na vstup systému (obr. 3). Priebeh takýchto prúdov pri použití vstupného napätia s U_s=15 V a f_s=10 Hz je znázornený na obr. 4 a 5.



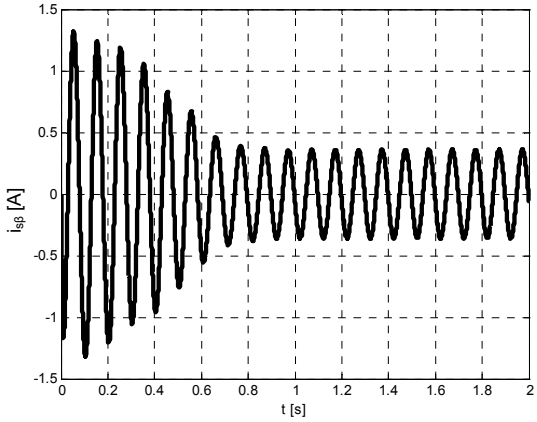
Obr. 3 Model dvojfázového generátora napätia

Fig. 3 Model of two-phase voltage generator



Obr. 4 Priebeh prúdu $i_{s\alpha}$ pre U_s=15V a f_s=10Hz

Fig. 4 Stator current $i_{s\alpha}$ response for U_s=15V a f_s=10Hz



Obr. 5 Priebeh prúdu $i_{s\beta}$ pre U_s=15V a f_s=10Hz

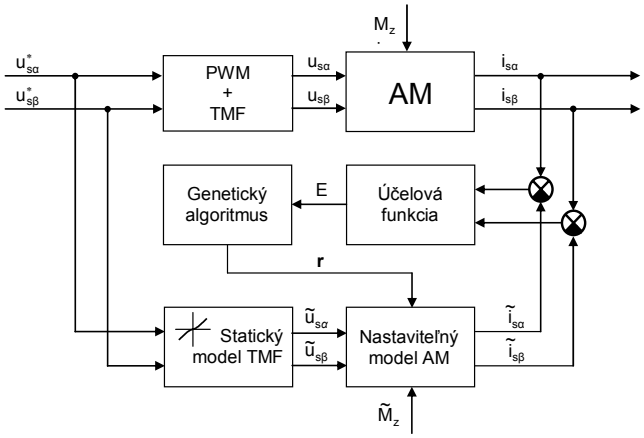
Fig. 5 Stator current $i_{s\beta}$ response for U_s=15V a f_s=10Hz

Výsledky z jednotlivých identifikácií pri použití rôznych hodnôt amplitúdy U_s a frekvencie f_s vstupných napätí sú uvedené v tab. 2, kde sú vyjadrené aj percentuálne odchýlky týchto výsledkov od skutočných parametrov.

Zvyšovaním frekvencie vstupného napätia sa pri použití tej istej periódy vzorkovania znižuje počet vzoriek harmonického signálu na jednu periódu. To má za následok znižovanie presnosti identifikácie parametrov. Preto by bolo potrebné so zvyšovaním frekvencie znižovať periódu vzorkovania. To je možné vidieť aj z výsledkov pri použití rôznych períód vzorkovania pre frekvenciu vstupného napätia 30Hz uvedených v tab. 3.

		R_s	R_r	L_s	L_m	J
	skutočná hodnota	**7,608**	**3,700**	**0,6015**	**0,5796**	**0,001700**
U_s=**7,5V** f_s=**5Hz** E=1,51.10^{-3}	$\tilde{\mathbf{r}}$	7,595	3,709	0,6023	0,5805	0,001706
	δ	0,2%	0,2%	0,1%	0,2%	0,4%
U_s=**15V** f_s=**10Hz** E=1,64.10^{-2}	$\tilde{\mathbf{r}}$	7,559	3,727	0,6020	0,5803	0,001717
	δ	0,6%	0,7%	0,1%	0,1%	1,0%
U_s=**30V** f_s=**20Hz** E=1,82	$\tilde{\mathbf{r}}$	7,407	3,812	0,6293	0,6074	0,001753
	δ	2,6%	3,0%	4,6%	4,8%	3,1%
U_s=**45V** f_s=**30Hz** E=15,39	$\tilde{\mathbf{r}}$	7,238	3,917	0,7063	0,6844	0,001807
	δ	4,9%	5,9%	17,4%	18,1%	6,3%

Tab. 2 Porovnanie výsledkov identifikácie parametrov AM genetickým algoritmom pri rôznych hodnotách amplitúdy a frekvencie vstupného napätia s ich skutočnými hodnotami



Obr. 6 Bloková schéma identifikácie parametrov AM genetickým algoritmom s kompenzáciou statickej chyby meniča

Fig. 6 Scheme of parameter identification of induction motor using GA with compensation of static inverter error

		R_s	R_r	L_s	L_m	J
	skutočná hodnota	**7,608**	**3,700**	**0,6015**	**0,5796**	**0,001700**
T_{vz}=**0,10ms** E=0,37	$\tilde{\mathbf{r}}$	7,645	3,675	0,6073	0,5853	0,001690
	δ	0,5%	0,7%	1,0%	1,0%	0,6%
T_{vz}=**0,20ms** E=2,83	$\tilde{\mathbf{r}}$	7,881	3,595	0,5922	0,5703	0,001633
	δ	3,6%	2,9%	1,5%	1,6%	3,9%
T_{vz}=**0,25ms** E=15,39	$\tilde{\mathbf{r}}$	7,238	3,917	0,7063	0,6844	0,001807
	δ	4,9%	5,9%	17,4%	18,1%	6,3%

Tab. 3 Výsledky identifikácie parametrov AM pri použití vstupného napätia s U_s=45V a f_s=30Hz a rôznych hodnôt periódy vzorkovania

3. Experimenty na reálnom systéme

Reálny systém obsahuje okrem AM aj menič frekvencie (obr. 6), ktorý má nelineárnu statickú prevodovú charakteristiku. Výsledkom jeho nelineárnosti je rozdiel medzi žiadanými hodnotami fázových napätí vstupujúcich do meniča a napätiami vstupujúcimi do motora. Ak pri identifikácii nepoužijeme skutočné napätia vstupujúce do motora, dopustíme sa chyby, ktorá bude mať vplyv na jej výsledky. Použitý laboratórny systém neumožňoval záznam týchto hodnôt, preto sme nepriamo zmerali prevodovú charakteristiku frekvenčného meniča, ktorú sme použili na kompenzáciu statickej chyby frekvenčného meniča.

Podobne ako pri simulačných pokusoch sme aj teraz realizovali identifikáciu parametrov v režime so zabrzdeným motorom pre rôzne hodnoty vstupného napätia. Keďže sme nepoznali skutočné hodnoty parametrov, podobnosť výsledkov z jednotlivých identifikácií sme overili porovnaním s ich spriemerovanými hodnotami v tab. 4.

Na rozdiel od simulácií sme pri rôznych napätiach nedosiahli rovnaké hodnoty jednotlivých parametrov. Najvýraznejší ja najmä rast hodnoty odporu rotora R_r so zvyšovaním napätia. Z tab. 4 možno vidieť, že pri všetkých vstupných napätiach by sme najlepšiu presnosť identifikácie dosiahli pre odpor statora. Po spriemerovaní všetkých identifikovaných hodnôt pre tento parameter dostaneme hodnotu 7,411. Maximálna nepresnosť identifikácie odporu statora je potom 2,8 % v porovnaní s touto hodnotou.

		R_s	R_r	L_s	L_m
	priemerná hodnota	**7,411**	**5,309**	**0,3540**	**0,3007**
$\tilde{u}_{s\alpha}$ =**5V** E=3,99	$\tilde{\mathbf{r}}$	7,233	1,984	0,3246	0,2603
	δ	2,4%	62,6%	8,3%	13,4%
$\tilde{u}_{s\alpha}$ =**10V** E=2,31	$\tilde{\mathbf{r}}$	7,206	3,726	0,4040	0,3477
	δ	2,8%	29,8%	14,1%	15,6%
$\tilde{u}_{s\alpha}$ =**15V** E=5,43	$\tilde{\mathbf{r}}$	7,489	4,789	0,4238	0,3692
	δ	1,1%	9,8%	19,7%	22,8%
$\tilde{u}_{s\alpha}$ =**20V** E=7,21	$\tilde{\mathbf{r}}$	7,344	6,092	0,3581	0,3088
	δ	0,9%	14,7%	1,2%	2,7%
$\tilde{u}_{s\alpha}$ =**25V** E=12,62	$\tilde{\mathbf{r}}$	7,611	7,120	0,3260	0,2782
	δ	2,7%	34,1%	7,9%	7,5%
$\tilde{u}_{s\alpha}$ =**30V** E=18,05	$\tilde{\mathbf{r}}$	7,581	8,145	0,2872	0,2398
	δ	2,3%	53,4%	18,9%	20,3%

Tab. 4 Porovnanie výsledkov identifikácie parametrov AM genetickým algoritmom na reálnom zariadení pri rôznych hodnotách vstupného napätia so spriemerovanými hodnotami

Keďže cieľom našej identifikácie parametrov reálneho systému bolo získať hodnoty parametrov vhodné pre návrh uzavretých regulačných štruktúr AM, ich presnosť sme posudzovali aj na základe kvality regulátorov navrhnutých použitím týchto hodnôt. Pre ohodnotenie správnosti jednotlivých výsledkov sme realizovali preskúmanie dynamických vlastností generátora momentu v štruktúre vektorového riadenia AM s PI regulátormi prúdu navrhnutými podľa identifikovaných hodnôt parametrov. Podľa vzťahu (2) je z parametrov R_r a L_r určovaná časová konštanta rotora T_r. Pomocou tejto konštanty je potom podľa nasledovného vzťahu nastavovaná konštanta $\tilde{K}_{sl}$:

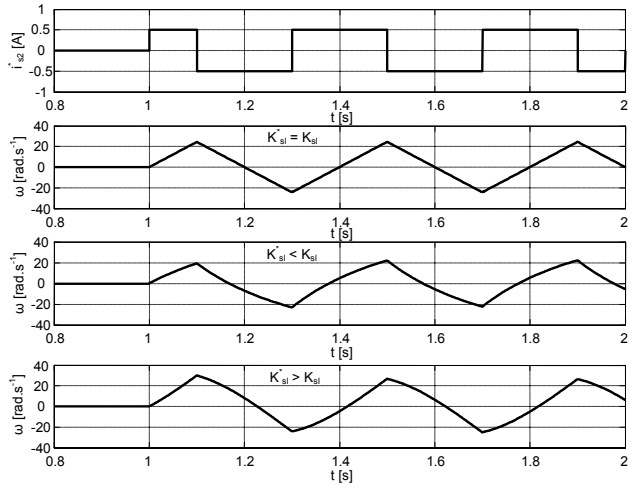
$$\tilde{K}_{sl} = \frac{\tilde{L}_m}{\tilde{T}_r \psi_r^*} = \frac{1}{\tilde{T}_r i_{s1}^*} \qquad (11)$$

V prípade ak nie je správne nastavená konštanta $\tilde{K}_{sl}$, prejaví sa to v zhoršenej dynamike generátora momentu.

Vzťah medzi želanou M_m^* a skutočnou hodnotou momentu motora M_m vyjadruje prenosová funkcia:

$$\frac{\Delta M_m(s)}{\Delta M_m^*(s)} = a\frac{1+T_r s}{1+T_r^* s}, \qquad a = \frac{T_r}{T_r^*} \tag{12}$$

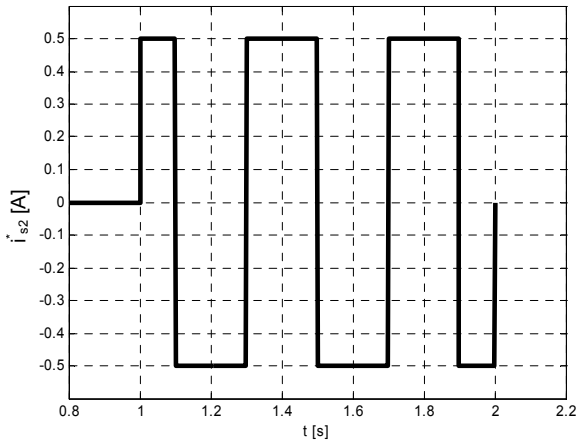
Táto prenosová funkcia kvalitatívne vyjadruje dynamické vlastnosti generátora momentu v okolí zvoleného pracovného bodu. Využíva sa pri experimentálnom nastavovaní $\widetilde{K}_{sl}$. Nastavovanie tejto konštanty je možné realizovať pomocou skokových zmien i_{s2}^*. Z pozorovania charakteru uhlovej rýchlosti môžeme potom jednoduchým experimentom nastaviť požadovanú hodnotu sklzovej konštanty $\widetilde{K}_{sl}$ (obr. 7).



Obr.7 Experimentálne nastavovanie dynamických vlastností generátora momentu pomocou konštanty $\widetilde{K}_{sl}$

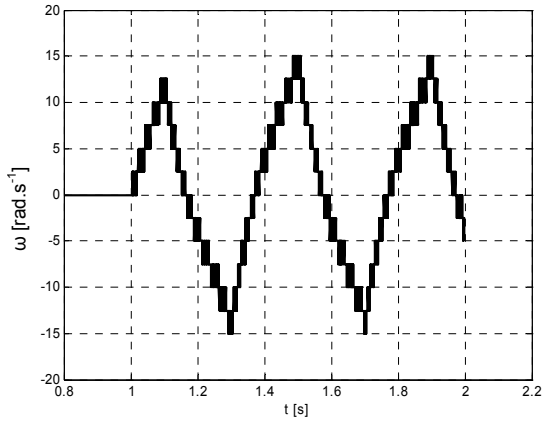
Fig. 7 Experimental setting of moment generator dynamical properties using by constant $\widetilde{K}_{sl}$

Na základe takýchto experimentov môžeme posúdiť kvalitu nastavenia konštanty $\widetilde{K}_{sl}$ a teda aj parametra T_r. Naše experimenty sme realizovali pri skokových zmenách i_{s2}^* podľa obr. 8 a pri konštantnej hodnote prúdu i_{s1}^* =1,4A. Na obr. 8 je znázornený priebeh uhlovej rýchlosti pri použití hodnôt parametrov zo simulačných experimentov. Tvar priebehu poukazuje na nepresnosť nastavenia parametrov. Použitie hodnôt identifikovaných pomocou GA (obr. 10 až 12) prinieslo zlepšenie dynamických vlastností GM. Lepšie priebehy uhlovej rýchlosti sme dosiahli použitím hodnôt získaných pri vyššom napätí. Dobrý priebeh uhlovej rýchlosti sme dosiahli aj pri použití spriemerovaných hodnôt parametrov (obr. 13).



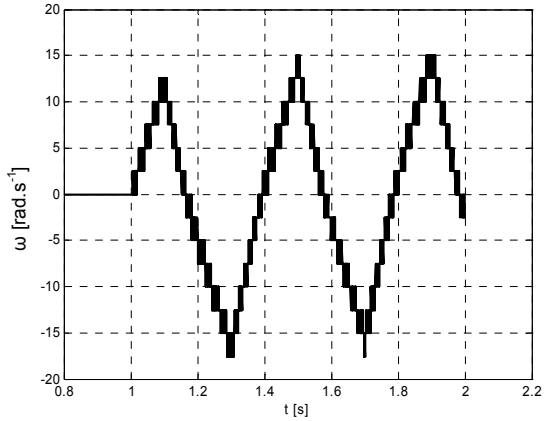
Obr. 8 Priebeh želanej hodnoty prúdu pre posúdenie dynamických vlastností generátora momentu

Fig. 8 Desired current course for the analysis of moment generator dynamical properties



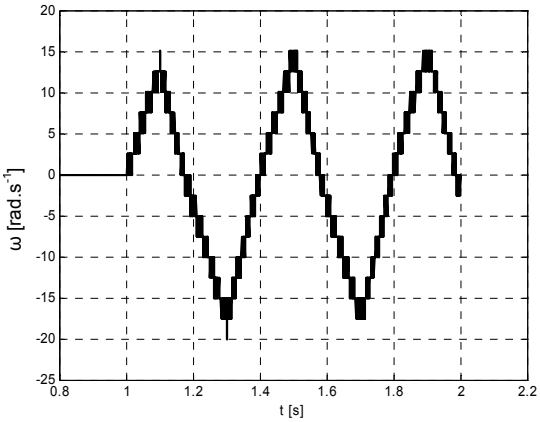
Obr. 9 Priebeh uhlovej rýchlosti pri použití hodnôt parametrov zo simulačných experimentov

Fig. 9 Angular speed response while using parameters from simulation experiments



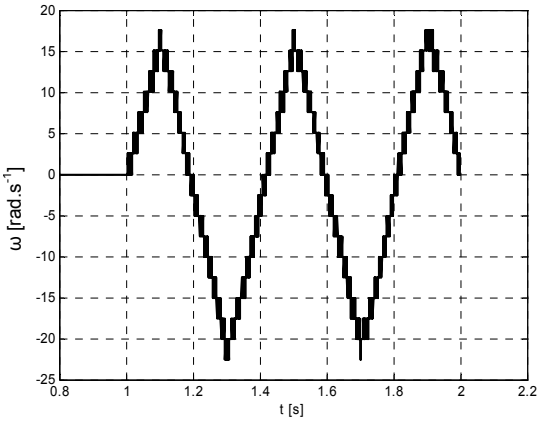
Obr. 10 Priebeh uhlovej rýchlosti pri použití hodnôt parametrov identifikovaných pri $\widetilde{u}_{s\alpha}$ =10V

Fig. 10 Angular speed response while using parameters from the identification with $\widetilde{u}_{s\alpha}$ =10V



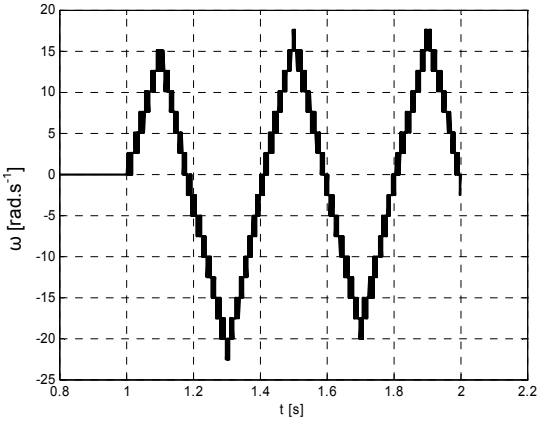
Obr. 11 Priebeh uhlovej rýchlosti pri použití hodnôt parametrov identifikovaných pri $\tilde{u}_{s\alpha}$ =15V

Fig. 11 Angular speed response while using parameters from the identification with $\tilde{u}_{s\alpha}$ =15V



Obr. 12 Priebeh uhlovej rýchlosti pri použití hodnôt parametrov identifikovaných pri $\tilde{u}_{s\alpha}$ =20V

Fig. 12 Angular speed response while using parameters from the identification with $\tilde{u}_{s\alpha}$ =20V



Obr. 13 Priebeh uhlovej rýchlosti pri použití spriemerovaných hodnôt parametrov

Fig. 13 Angular speed response while using averaged values of parameters

Presnosť identifikácie v režime s rozbehom motora je ovplyvňovaná voľbou počtu vzoriek na periódu signálu. Reálny systém nám umožňoval použiť najnižšiu periódu vzorkovania 0,25 ms. Preto zvyšovaním frekvencie vstupného napätia by mala klesať presnosť identifikácie. Keďže sme nepoznali skutočné hodnoty parametrov, opäť sme overili podobnosť výsledkov z jednotlivých identifikácií porovnaním s ich spriemerovanými hodnotami v tab. 5.

		R_s	R_r	L_s	L_m	J
	priemerná hodnota	**9,755**	**4,324**	**0,4113**	**0,3695**	**0,004283**
$\tilde{U}_s$ **=15V** f_s**=5Hz** E=54,86	$\tilde{\mathbf{r}}$	10,000	2,851	0,3727	0,3188	0,003250
	δ	2,5%	34,1%	9,4%	13,7%	24,1%
$\tilde{U}_s$ **=30V** f_s**=10Hz** E=313,19	$\tilde{\mathbf{r}}$	10,000	4,245	0,4261	0,3807	0,004115
	δ	2,5%	1,8%	3,6%	3,0%	3,9%
$\tilde{U}_s$ **=50V** f_s**=15Hz** E=431,51	$\tilde{\mathbf{r}}$	10,000	4,703	0,4389	0,4013	0,004403
	δ	2,5%	8,8%	6,7%	8,6%	2,8%
$\tilde{U}_s$ **=60V** f_s**=20Hz** E=540,21	$\tilde{\mathbf{r}}$	9,976	4,613	0,4177	0,3798	0,004436
	δ	2,3%	6,7%	1,6%	2,8%	3,5%
$\tilde{U}_s$ **=80V** f_s**=25Hz** E=559,50	$\tilde{\mathbf{r}}$	8,801	5,209	0,4012	0,3670	0,005211
	δ	9,8%	20,5%	2,5%	0,7%	21,7%

Tab. 5 Výsledky identifikácie parametrov AM pri rozbehu motora pre rôzne hodnoty amplitúdy a frekvencie vstupného napätia

Z výsledkov experimentu si možno ako prvé všimnúť, že hodnoty odporu statora sa dostávali na hranicu definovaného intervalu. Z toho vyplýva, že na identifikáciu tohto parametra je vhodnejšia identifikácia pri zabrzdenom motore. Odpor rotora sa až na frekvenciu vstupného napätia 20 Hz zvyšoval so zvyšovaním použitého napätia. Identifikované hodnoty indukčnosti statora a magnetizačnej indukčnosti sa pri jednotlivých pokusoch menili navzájom takmer lineárne. Hodnoty momentu zotrvačnosti sa postupne mierne zvyšujú so zvyšovaním použitej frekvencie vstupného napätia, t.j. uhlovej rýchlosti motora. To môže byť spôsobené viskóznym trením, ktoré sme pri týchto pokusoch zanedbávali. Preto sme sa rozhodli zistiť aký vplyv môžu mať na identifikáciu parametre suchého a viskózneho trenia. Celý postup identifikácie sme zmenili iba v tom, že medzi hľadané parametre (do chromozómu GA) sme doplnili aj parametre suchého trenia M_{z0} a viskózneho trenia B' a zároveň sme ich zakomponovali do nastaviteľného modelu podľa nasledovnej rovnice:

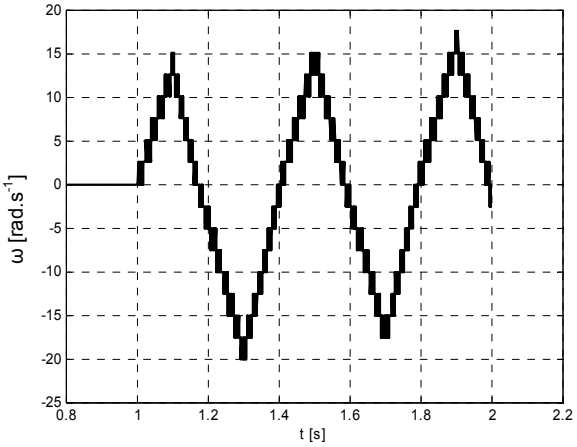
$$M_z = M_{z0}\,\mathrm{sgn}\,\omega + B'\omega \qquad (13)$$

V tab. 6 možno vidieť ako sa zmenili výsledky jednotlivých identifikácií. Ako prvé si je možné všimnúť, že prišlo k zmenšeniu hodnôt odporu statora, ktoré sa predtým nachádzali na hranici definovanej oblasti riešení. Výsledné hodnoty sa pri týchto pokusoch priblížili k hodnotám dosiahnutých identifikáciou pri zabrzdenom motore. Hodnoty ostatných parametrov sa naopak zvýšili. Avšak aj keď pridanie ďalších parametrov do reťazca GA prinieslo lepšiu identifikáciu niektorých parametrov, práve identifikované hodnoty týchto dvoch nových parametrov výrazne kolísali.

		R_s	R_r	L_s	L_m	J	M_{z0}	B'
	priemerná hodnota	**7,759**	**4,965**	**0,4312**	**0,3953**	**0,005211**	**0,01470**	**0,01317**
$\widetilde{U}_s$ =**15V** f_s=**5Hz** E=26,35	$\tilde{\mathbf{r}}$	7,770	4,192	0,4021	0,3662	0,005686	0,00484	0,02276
	δ	0,1%	15,6%	6,7%	7,4%	9,1%	67,1%	72,8%
$\widetilde{U}_s$ =**30V** f_s=**10Hz** E=44,17	$\tilde{\mathbf{r}}$	8,201	4,724	0,4473	0,4080	0,004706	0,02623	0,01077
	δ	5,7%	4,9%	3,7%	3,2%	9,7%	78,4%	18,2%
$\widetilde{U}_s$ =**50V** f_s=**15Hz** E=42,08	$\tilde{\mathbf{r}}$	7,852	5,068	0,4537	0,4188	0,005011	0,04034	0,00805
	δ	1,2%	2,1%	5,2%	5,9%	3,8%	174,4%	38,9%
$\widetilde{U}_s$ =**60V** f_s=**20Hz** E=51,43	$\tilde{\mathbf{r}}$	7,671	5,179	0,4363	0,4004	0,005083	0,00151	0,00636
	δ	1,1%	4,3%	1,2%	1,3%	2,5%	89,7%	51,7%
$\widetilde{U}_s$ =**80V** f_s=**25Hz** E=72,92	$\tilde{\mathbf{r}}$	7,301	5,660	0,4165	0,3832	0,005567	0,00057	0,00528
	δ	5,9%	14,0%	3,4%	3,1%	6,8%	96,1%	59,9%

Tab. 6 Výsledky identifikácie parametrov AM spolu s parametrami trenia pri rozbehu motora pre rôzne hodnoty amplitúdy a frekvencie vstupného napätia

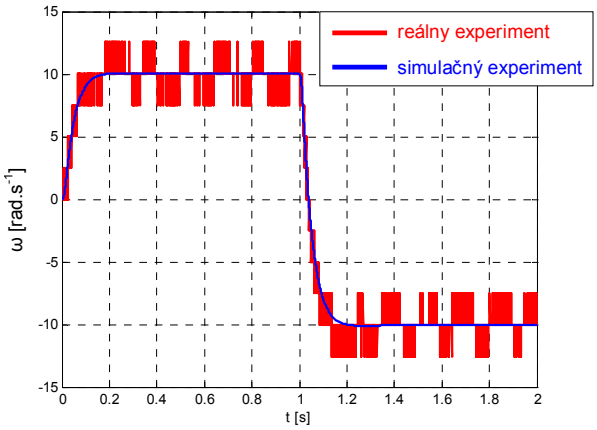
Správnosť jednotlivých výsledkov môžeme opäť ohodnotiť preskúmaním dynamických vlastností generátora momentu v štruktúre vektorového riadenia AM s PI regulátormi prúdu navrhnutými podľa identifikovaných hodnôt parametrov. Opäť sme kvalitu nastavenia jednotlivých parametrov posudzovali na základe odozvy uhlovej rýchlosti na skokové zmeny prúdu i_{s2}^* podľa obr. 8. Použitie hodnôt parametrov získaných spriemerovaním výsledkov z predchádzajúceho experimentu (obr. 14) prinieslo podobný priebeh uhlovej rýchlosti ako použitie hodnôt z identifikácie so zabrzdeným motorom.



Obr. 14 Priebeh uhlovej rýchlosti pri použití spriemerovaných hodnôt parametrov z identifikácie z rozbehom motora

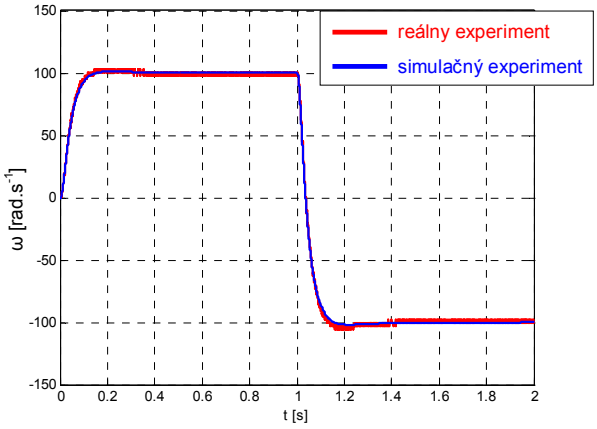
Fig. 14 Angular speed response while using averaged values of parameters from the identification with motor start

Správnosť nastavenia mechanických parametrov AM môžeme overiť na základe správania sa regulačného obvodu s regulátorom rýchlosti navrhnutým použitím týchto parametrov. Na reguláciu uhlovej rýchlosti reálneho systému boli použité PI regulátory navrhnuté metódou Whiteley-ho štandardných tvarov podľa spriemerovaných hodnôt parametrov z tab. 6. Na obr. 15 a 16 sú priebehy uhlovej rýchlosti pri skokovej zmene z 0 na 10 a reverzácii na -10 rad.s^{-1} a pri skokovej zmene z 0 na 100 a reverzácii na -100 rad.s^{-1}. Z obrázkov možno vidieť, že pre reálnych experimentoch boli priebehy uhlovej rýchlosti podobné ako pri simulačných experimentoch a potvrdzujú, že identifikované hodnoty môžeme použiť pre návrh vektorového riadenia AM.



Obr. 15 PI regulácia uhlovej rýchlosti z 0 na 10 a s reverzáciou na -10 rad.s^{-1}, pri použití spriemerovaných hodnôt parametrov z tab. 6

Fig. 15 PI control of angular speed from 0 to 10 and with reversing to -10 rad.s^{-1} while using the averaged values of parameters from tab. 6



Obr. 16 PI regulácia uhlovej rýchlosti z 0 na 100 a s reverzáciou na -100 rad.s^{-1}, pri použití spriemerovaných hodnôt parametrov z tab. 6

Fig. 16 PI control of angular speed from 0 to 100 and with reversing to -100 rad.s^{-1} while using the averaged values of parameters from tab. 6

Záver

V článku sme predstavili metódu identifikácie parametrov AM pomocou genetického algoritmu. Identifikáciu sme realizovali v dvoch režimoch. Prvým je identifikácia elektrických parametrov AM pri zabrzdenom motore. Simulačné experimenty ukázali pri tomto spôsobe veľmi dobrú presnosť identifikácie. Pri aplikácii na reálny systém sa však prejavil rozdiel medzi reálnym systémom a modelom AM, ktorý bol využívaný pri identifikácii. Išlo najmä o vplyv nelinearity frekvenčného meniča na napätia vstupujúce do motora. Keďže náš systém nemal realizovaný záznam týchto napätí, nepriamo sme zmerali prevodovú charakteristiku frekvenčného meniča pre kompenzáciu jeho statickej chyby. Navrhnutým GA sa potom podarilo pri rôznych napätiach najpresnejšie identifikovať odpor statora.

Druhá navrhnutá metóda identifikácie parametrov AM pomocou GA využíva striedavé priebehy zložiek statorového prúdu po rozbehu motora privedením statorového napätia so zvolenou amplitúdou a frekvenciou na vstup motora. Pri tejto metóde sme okrem elektrických parametrov identifikovali aj moment zotrvačnosti motora. Presnosť riešenia závisela od použitej frekvencie vstupného napätia, resp. od počtu použitých vzoriek na periódu signálu, ktorý môžeme ovplyvniť voľbou periódy vzorkovania pri danej frekvencii. Ako logicky vyplýva, zvyšovaním frekvencie napätia je potrebné zvyšovať počet vzoriek na periódu signálu pre presnejší opis referenčných signálov prúdu. Vhodným nastavením genetického algoritmu a výberom referenčných signálov sme boli pri simuláciách schopný dosiahnuť nepresnosť identifikácie menšiu ako 1 %. Pri identifikácii parametrov reálneho AM sa však prejavil rovnaký problém ako pri predchádzajúcej metóde. Nemožnosť merať skutočné hodnoty statorového napätia sa tak ako každý rozdiel medzi reálnym systémom a jeho modelom prejavil v rozdielnosti výsledkov jednotlivých reálnych experimentov. Na rozdiel od predchádzajúcej metódy sa hodnoty odporu statora dostali na hranicu definovaného intervalu a vyrovnanejšie výsledky sme dosiahli pre zvyšné parametre. Avšak pridaním parametrov suchého a viskózneho trenia medzi hľadané parametre sme dosiahli zníženie hodnôt odporu statora a ich priblíženie sa k hodnotám získaných pri zabrzdenom motore.

Najlepším ohodnotením presnosti výsledkov identifikácie a jediným, ktoré sme mohli v našom prípade realizovať je posúdenie kvality regulácie, ktorú vieme na základe týchto hodnôt navrhnúť. Na základe dynamických vlastností regulátorov generátora momentu, navrhnutých pomocou identifikovaných hodnôt parametrov AM, môžeme povedať, že identifikácia pomocou GA poskytuje dostatočne presné parametre pre návrh regulátorov vektorového riadenia AM.

Literatúra

[1] VAS, P.: Artificial Intelligence Based Electrical Machines and Drives. New York, Oxford University Press, 1999.

[2] SEKAJ, I.: Evolučné výpočty a ich využitie v praxi. Vydavateľstvo IRIS, Bratislava, 2005.

[3] ŽALMAN, M.: Akčné členy. Vydavateľstvo STU, Bratislava, 2003.

[4] URSEM, R. K., VADSTRUP, P.: Parameter Identification of Induction Motors using Differential Evolution. Proceedings of the Fifth Congress on Evolutionary Computation (CEC-2003), pp. 790 - 796.

[5] URSEM, R. K., VADSTRUP, P.: Parameter identification of induction motors using stochastic optimization algorithms. Applied Soft Computing, 4, 2004, pp. 49 – 64.

[6] JOUKHADAR, A., GARAT, F., LAUGIER, C.: Parameter Identification for Dynamic Simulation. Conference Paper, In Proc. of the IEEE Int. Conf. on Robotics and Automation, Vol. 3, Albuquerque, NM (US), April 1997, pp. 1928 - 1933.

[7] CHUNG, P. Y., DÖLEN, M., LORENZ, R. D.: Parameter Identification for Induction Machines by Continuous Genetic Algorithms. ANNIE 2000 Conference, St. Louis, November 2000.

Abstract

This paper describes the parameter identification of induction motor using genetic algorithm. Genetic Algorithms present universal methods, which can be used to solve a wide scope of optimization problems, between which the parameter identification may be classified. Parameter identification by genetic algorithm is based on generating new solutions for the minimization of specific objective function which compare current courses from induction motor and its simulation model with adjustable parameters. Induction motor represents the high nonlinear system which parameters can vary and their values are important for control design, state observers or construction of the motor model. The designed method makes it possible to identify the electricity and eventually also mechanical parameters with adequate accuracy for induction motor vector control design. The paper presents simulation and also real experiments.

Ing. Marián Jančovič, PhD.

Prof. Ing. Milan Žalman, PhD.

Slovenská technická univerzita
Fakulta elektrotechniky a informatiky
Ústav riadenia a priemyselnej informatiky
Ilkovičova 3
812 19 Bratislava
mjancovic@gmail.com
milan.zalman@stuba.sk

Bezsnímačový rýchlostný servosystém s AM s Luenbergerovým pozorovateľom

Juraj Gacho, Milan Žalman

Abstrakt
Článok sa zaoberá pozorovaním stavov AM použitím Luenbergerovho pozorovateľa v štruktúre rýchlostného servosystému. Pre póly pozorovateľa zvolené ako $p_{LO} = \alpha e^{j\varphi} p_{AM}$ je analyzovaný pohyb pólov motora a pozorovateľa pri zmene uhlovej rýchlosti motora a je navrhnutý nový spôsob výpočtu zosilnení pozorovateľa pre daný systém a nový spôsob rozmiestnenia pólov pozorovateľa pre pozorovanie magnetického toku AM. Štruktúra systému je doplnená o adaptívne pozorovanie uhlovej rýchlosti. Funkčnosť navrhnutého systému je simulačne overená v prostredí MATLAB Simulink.

Kľúčové slová: asynchrónny motor, bezsnímačový rýchlostný servosystém, Luenbergerov pozorovateľ

Úvod

Dynamické riadenie asynchrónnych motorov patrí medzi dominantné aplikácie v oblasti riadenia pohybu. V oblasti riadenia AM možno sledovať dva trendy. Prvým je riadenie s obmedzenou merateľnosťou stavových veličín (bez snímania mechanických veličín, riadenie – Sensorless Control) najmä v rýchlostných servosystémoch, pri ktorom je kladený dôraz na kvalitné pozorovanie stavových veličín. Pojem bezsnímačové (bezsenzorové, sensorless) v tomto kontexte znamená, že sa nepoužívajú snímače mechanických veličín (polohy, rýchlosti, zrýchlenia, momentu), ale pokiaľ daná riadiaca štruktúra vyžaduje informácie o týchto veličinách, sú získané napr. z pozorovateľov. Elektrické veličiny, ako sú prúdy a napätia sú v týchto systémoch merané. Dôvodom požiadavky implementácie bezsnímačového riadenia je to, že nie je potrebné montovať na motor snímač daných mechanických veličín, s ktorým sú spojené ďalšie náklady. Mechanický snímač navyše predstavuje ďalší zdroj chýb (najmä v nehostinných prostrediach). Vďaka pozorovateľom je možné nahradiť staršie typy riadenia AM (skalárne riadenie) kvalitnejším (napr. vektorovým) riadením a to iba zmenou riadiaceho systému.

Druhým trendom je „uzavreté“ dynamické riadenie s dôrazom na rýchlosť a kvalitu prechodových dejov s využitím predovšetkým mechanických veličín AM (poloha, rýchlosť). Tento prístup sa používa najmä v polohových systémoch.

V obidvoch prípadoch sa však stáva, že pre kvalitné riadenie potrebujeme stavové veličiny, ktoré v danej štruktúre nemožno jednoducho zmerať a preto je potrebné tieto veličiny pozorovať.

Existuje veľa metód pozorovania uhlovej rýchlosti, ale aj ďalších veličín potrebných pri riadení AM (napr. magnetický tok) a tiež pozorovania parametrov. Tieto metódy by bolo možné rozdeliť do niekoľkých skupín. Sú to tzv. otvorené pozorovatele, ktoré vychádzajú iba z modelu systému a zo znalosti parametrov, teda zmena parametrov má výrazný vplyv na presnosť pozorovania. Z tohto dôvodu sa v praktických aplikáciách vyskytujú len zriedka. Niekoľko takýchto pozorovateľov je uvedených napr. v [1], [16]. Ďalšiu, azda najväčšiu skupinu tvoria pozorovatele založené na MRAS systémoch, ktoré používajú referenčný a adaptívny model. Popis a aplikácie MRAS pozorovateľov je možné nájsť v [1], [2], [13], [14], [16] a v mnohých ďalších. Aplikácie rozšíreného Kalmanovho filtra (EKF) sú na pozorovanie stavov AM tiež veľmi časté. Rozšírený Kalmanov filter na pozorovanie uhlovej rýchlosti a magnetického toku možno nájsť v prácach [3], [15], [16]. Použite rozšíreného Luenbergerovho pozorovateľa na pozorovanie veličín AM možno nájsť v [6], [9], [10], [16]. Tento článok sa zaoberá použitím rozšíreného Luenbergerovho pozorovateľa na pozorovanie rotorového magnetického toku IM v štruktúre rýchlostného servosystém s adaptívnym (MRAS) pozorovateľom uhlovej rýchlosti. V článku je navrhnutá nová metóda rozmiestnenia pólov pozorovateľa pre pozorovanie rotorového magnetického toku AM a je odvodený výpočet zosilnení pozorovateľa pre túto polohu pólov.

Štruktúra bezsnímačového rýchlostného servopohonu

Základom tohto servosystému bude asynchrónny motor s kotvou nakrátko. Bude použitý model AM v statorovom súradnicovom systéme so stavovými veličinami $\hat{i}_s$, $\hat{\psi}_r$, ktorý možno v komplexnom tvare vyjadriť ako:

$$\frac{d}{dt}\begin{bmatrix}\hat{i}_s\\ \hat{\psi}_r\end{bmatrix} = \begin{bmatrix}-\frac{1}{T_1} & \left(\frac{k_r}{T_r L'_s} - j\frac{k_r\omega}{L'_s}\right)\\ \frac{L_m}{T_r} & \left(-\frac{1}{T_r} + j\omega\right)\end{bmatrix}\begin{bmatrix}\hat{i}_s\\ \hat{\psi}_r\end{bmatrix} + \begin{bmatrix}\frac{1}{L'_s}\\ 0\end{bmatrix}\hat{u}_s \qquad (1)$$

$$\hat{y} = \hat{i}_s = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}\hat{i}_s\\ \hat{\psi}_r\end{bmatrix}$$

Teda jednotlivé matice možno označiť ako:

$$\mathbf{x}=\begin{bmatrix}\hat{i}_s\\ \hat{\psi}_r\end{bmatrix};\qquad \mathbf{A}=\begin{bmatrix}-\frac{1}{T_1} & \left(\frac{k_r}{T_r L'_s}-j\frac{k_r\omega}{L'_s}\right)\\ \frac{L_m}{T_r} & \left(-\frac{1}{T_r}+j\omega\right)\end{bmatrix};\qquad \mathbf{B}=\begin{bmatrix}\frac{1}{L'_s}\\ 0\end{bmatrix};\qquad \mathbf{C}=\begin{bmatrix}1 & 0\end{bmatrix} \tag{2}$$

kde:

$$R_1=R_s+R_r\frac{L_m^2}{L_r^2},\quad T_1=\frac{L'_s}{R_1},\quad T_r=\frac{L_r}{R_r},\quad k_r=\frac{L_m}{L_r} \tag{3}$$

$$\sigma=1-\frac{L_m^2}{L_s L_r} \tag{4}$$

$$L'_s=\sigma L_s \tag{5}$$

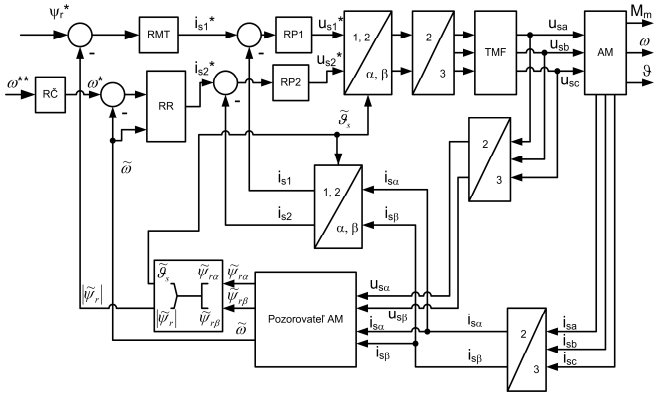
V nasledujúcej tabuľke je popis symbolov použitých v uvedenom modeli AM

u_s	Statorové napätie
i_s	Statorový prúd
R_s	Odpor statorového vinutia
L_s	Indukčnosť statorového vinutia
ψ_f	Rotorový magnetický tok
R_r	Odpor rotorového vinutia
L_r	Indukčnosť rotorového vinutia
L_m	Hlavná magnetizačná indukčnosť
ω	Elektrická uhlová rýchlosť
σ	Koeficient rozptylu

Tab.1 Symboly použité v modeli AM

Tento asynchrónny motor bude riadený v štruktúre priameho vektorového riadenia orientovanej na magnetický tok rotora. V štruktúre rýchlostného servosystému bude použitý prepočet trojfázového systému na ekvivalentný dvojfázový a naopak, tzv. priama a spätná Clarkovej transformácia (transformácia 3/2, resp. 2/3). Pre transformáciu veličín medzi statorovým súradnicovým systémom a súradnicovým systémom spojeným s vektorom rotorového magnetického toku bude použitá Parkova transformácia.

Pre rýchlostný systém s AM môžeme potom zvoliť nasledujúcu štruktúru:



Obr.1 Štruktúra rýchlostného servosystému s pozorovateľom stavových veličín AM

Fig.1 Structure of IM speed servodrive with state variable observer

RČ – rozbehový člen zabezpečujúci plynulý nábeh žiadanej uhovej rýchlosti

RP1, RP2 – diskrétne PI regulátory navrhnuté metódou inverznej dynamiky

RMT – diskrétny PI regulátor magnetického toku navrhnutý metódou inverznej dynamiky

RR – diskrétny IPD regulátor rýchlosti

V tomto článku sa budeme zaoberať najmä blokom Pozorovateľ AM

Luenbergerov pozorovateľ

Luenbergerov pozorovateľ (Luenberger observer - LO) patrí do skupiny pozorovateľov so spätnou väzbou (closed loop). Je to deterministický typ pozorovateľa pretože vychádza z deterministického modelu systému. Základný LO môže byť použitý iba na lineárne časovo invariantné systémy opísané nasledujúcimi rovnicami.

$$\frac{d\boldsymbol{x}}{dt}=\boldsymbol{Ax}+\boldsymbol{Bu} \tag{6}$$
$$\boldsymbol{y}=\boldsymbol{Cx}$$

V takomto prípade bude LO vyjadrený rovnicou

$$\frac{d\tilde{\boldsymbol{x}}}{dt}=\boldsymbol{A}\tilde{\boldsymbol{x}}+\boldsymbol{Bu}+\boldsymbol{K}\left(\boldsymbol{y}-\boldsymbol{C}\tilde{\boldsymbol{x}}\right) \tag{7}$$

Kde $\tilde{\boldsymbol{x}}$ je vektor pozorovaných stavových veličín a **K** je matica zosilnení pozorovateľa.

Ak uvažujeme asynchrónny motor v statorovom súradnicovom systéme a keď ako stavové veličiny zvolíme statorový prúd a rotorový magnetický tok (rovnica (1)), tak môžeme hovoriť o časovo premenlivom modeli, lebo parametre sa môžu meniť s časom (zmeny spôsobené zmenou uhlovej rýchlosti). Keďže základný LO nie je vhodný na uvedený typ systému, je potrebné použiť rozšírený Luenbergerov pozorovateľ (Extended Luenberger observer - ELO). Na rozdiel od základného LO, ELO je vhodný aj na nelineárne aj na časovo premenlivé systémy ako napr. aj systém opísaný nasledujúcimi rovnicami.

$$\frac{d\boldsymbol{x}}{dt}=f(\boldsymbol{x})+\boldsymbol{Bu} \tag{8}$$
$$\boldsymbol{y}=\boldsymbol{Cx}$$

Pre takýto systém môžeme napísať rovnice ELO:

$$\frac{d\tilde{\boldsymbol{x}}}{dt}=\tilde{\boldsymbol{A}}\tilde{\boldsymbol{x}}+\boldsymbol{Bu}+\boldsymbol{K}\left(\boldsymbol{y}-\boldsymbol{C}\tilde{\boldsymbol{x}}\right) \tag{9}$$

Kde

$$\tilde{\boldsymbol{A}}=\frac{df(\tilde{\boldsymbol{x}})}{d\tilde{\boldsymbol{x}}} \tag{10}$$

Maticu dynamiky pozorovateľa môžeme napísať ako

$$\boldsymbol{A}_o=\tilde{\boldsymbol{A}}-\boldsymbol{KC} \tag{11}$$

Ťažisko pozorovania stavov je vo vhodnom zvolení pólov pozorovateľa a následnom výpočte matice zosilnení pozorovateľa **K**.

Pre pozorovanie zložiek vektora magnetického toku rotora ELO algoritmom použijeme model AM vyjadrený rovnicami (1). Potom môžeme pre maticu pozorovateľa písať

$$\boldsymbol{A_o} = \boldsymbol{A} - \boldsymbol{KC} = \begin{bmatrix} -\frac{1}{T_1} & \left(\frac{k_r}{T_r L'_s} - j\frac{k_r \omega}{L'_s}\right) \\ \frac{L_m}{T_r} & \left(-\frac{1}{T_r} + j\omega\right) \end{bmatrix} - \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} = \tag{12}$$

$$= \begin{bmatrix} \left(-\frac{1}{T_1} - k_1\right) & \left(\frac{k_r}{T_r L'_s} - j\frac{k_r \omega}{L'_s}\right) \\ \left(\frac{L_m}{T_r} - k_2\right) & \left(-\frac{1}{T_r} + j\omega\right) \end{bmatrix}$$

Špeciálnym prípadom takéhoto pozorovateľa je prípad, kedy sa koeficienty k_1 a k_2 zvolia nulové a teda matica pozorovateľa bude zhodná s maticou systému. Takýto prístup bol uvedený napr. v práci [8]. Kvalita takéhoto pozorovateľa však nie je veľmi vysoká, lebo v tomto prípade nie je v pozorovateli zavedená spätná väzba od rozdielu výstupov modelu a systému ((9), (40)). A preto je potrebné venovať náležitú pozornosť návrhu spomínaných koeficientov.

Pre odvodenie koeficientov k_1 a k_2 je vhodné spraviť nasledujúcu substitúciu prvkov matice systému

$$\boldsymbol{A} = \begin{bmatrix} -\frac{1}{T_1} & \left(\frac{k_r}{T_r L'_s} - j\frac{k_r \omega}{L'_s}\right) \\ \frac{L_m}{T_r} & \left(-\frac{1}{T_r} + j\omega\right) \end{bmatrix} = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \tag{13}$$

Potom pre maticu pozorovateľa môžeme v našom prípade písať

$$\boldsymbol{A_o} = \boldsymbol{A} - \boldsymbol{KC} = \begin{bmatrix} (a_1 - k_1) & a_2 \\ (a_3 - k_2) & a_4 \end{bmatrix} \tag{14}$$

Nech p_1 a p_2 sú póly systému. Potom pre charakteristický polynóm systému platí

$$\begin{aligned} &(a_1 - s)(a_4 - s) - a_2 a_3 = s^2 - (a_1 + a_4)s + a_1 a_4 - a_2 a_3 = \\ &= (s - p_1)(s - p_2) = s^2 - (p_1 + p_2)s + p_1 p_2 = 0 \end{aligned} \tag{15}$$

V literatúre je opísaných niekoľko spôsobov, ako rozmiestniť póly pozorovateľa. Jedným z najčastejšie používaných spôsobov umiestnenia pólov pozorovateľa je, že póly pozorovateľa budú α násobkom pôvodných pólov motora, teda

$$p_{LO} = \alpha p_{AM} \tag{16}$$

Kde p_{LO} predstavuje póly pozorovateľa a p_{AM} póly motora a α je reálne číslo.

Takýto spôsob je uvedený napr. v [6], [7], [10], [11].

Predpokladajme teda, že póly pozorovateľa chceme umiestniť tak, aby boli α násobkom pólov motora (p_1, p_2). Potom pre takéto póly bude platiť

$$(s - \alpha p_1)(s - \alpha p_2) = s^2 - \alpha(p_1 + p_2)s + \alpha^2 p_1 p_2 = 0 \tag{17}$$

Ale pre póly pozorovateľa podľa (14) bude tiež platiť

$$\begin{aligned} &(a_1 - k_1 - s)(a_4 - s) - a_2(a_3 - k_2) = \\ &= s^2 - (a_1 - k_1 + a_4)s + (a_1 - k_1)a_4 - a_2(a_3 - k_2) = 0 \end{aligned} \tag{18}$$

Podľa (15) teda platí

$$\begin{aligned} &(p_1 + p_2) = (a_1 + a_4) \\ &p_1 p_2 = a_1 a_4 - a_2 a_3 \end{aligned} \tag{19}$$

Pre póly pozorovateľa platí (17) a (18), môžeme teda napísať nasledujúce vzťahy

$$\begin{aligned} &\alpha(p_1 + p_2) = (a_1 - k_1 + a_4) \\ &\alpha^2 p_1 p_2 = (a_1 - k_1)a_4 - a_2(a_3 - k_2) \end{aligned} \tag{20}$$

Po aplikácii (19)

$$\begin{aligned} &\alpha(a_1 + a_4) = (a_1 - k_1 + a_4) \\ &\alpha^2(a_1 a_4 - a_2 a_3) = (a_1 - k_1)a_4 - a_2(a_3 - k_2) \end{aligned} \tag{21}$$

Z týchto vzťahov je možné vypočítať hľadané koeficienty. Po spätnej substitúcii a drobných úpravách dostaneme nasledujúce vzťahy:

$$\begin{aligned} k_1 &= (\alpha - 1)\left(\frac{1}{T_1} + \frac{1}{T_r} - j\omega\right) \\ k_2 &= (\alpha^2 - 1)\left(\frac{L'_s}{k_r T_1} - \frac{L_m}{T_r}\right) - \frac{L'_s}{k_r} k_1 \end{aligned} \tag{22}$$

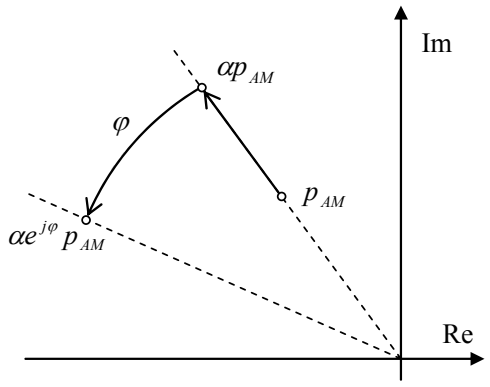
Alebo v zložkovom tvare

$$\begin{aligned} k_{11} &= (\alpha - 1)\left(\frac{1}{T_1} + \frac{1}{T_r}\right) \\ k_{12} &= (1 - \alpha)\omega \\ k_{21} &= (\alpha^2 - 1)\left(\frac{L'_s}{k_r T_1} - \frac{L_m}{T_r}\right) - \frac{L'_s}{k_r} k_{11} \\ k_{22} &= -\frac{L'_s}{k_r} k_{12} \end{aligned} \tag{23}$$

Ďalšou možnosťou ako navrhnúť umiestnenie pólov pozorovateľa je umiestniť ich tak, aby póly pozorovateľa boli α násobkom pólov motora a oproti pólom motora boli ešte otočené o fixný uhol smerom k zápornej časti reálnej osi. Póly pozorovateľa potom možno napísať ako

$$p_{LO} = \alpha e^{j\varphi} p_{AM} \tag{24}$$

Umiestnenie pólov pozorovateľa vzhľadom na póly motora podľa vzťahov (16) a (24) je vidieť na nasledujúcom obrázku.



Obr.2 Umiestnenie pólov pozorovateľa a motora
Fig.2 Observer and motor pole placement

Experimenty s takto umiestnenými pólmi pozorovateľa boli vykonané napr. v práci [6].

Pre odvodenie vzťahov na výpočet koeficientov matice zosilnení k_1 a k_2 môžeme použiť vzťahy (22) s tým, že miesto reálneho čísla α použijeme komplexné číslo g.

$$g = |g|e^{j\varphi} = |g|(\cos\varphi + j\sin\varphi) = g_r + jg_i \tag{25}$$

Ak by sme pôvodné póly motora vyjadrili v tvare

$$p_{AM} = |p_{AM}|e^{j\xi} \tag{26}$$

Potom póly pozorovateľa by sme mohli napísať ako

$$p_{LO} = gp_{AM} = |g|e^{j\varphi}|p_{AM}|e^{j\xi} = |g||p_{AM}|e^{j(\xi+\varphi)} \tag{27}$$

Z čoho vyplýva, že by sme póly pozorovateľa mohli oproti pólom motora otočiť o želaný uhol a tým zlepšiť vlastnosti pozorovateľa.

V tomto prípade by pre koeficienty k_1 a k_2 platili nasledujúce vzťahy

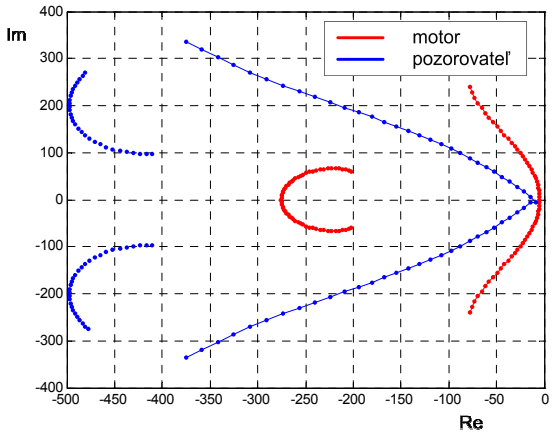
$$k_1 = (g-1)\left(\frac{1}{T_1}+\frac{1}{T_r}-j\omega\right)$$
$$k_2 = (g^2-1)\left(\frac{L'_s}{k_r T_1}-\frac{L_m}{T_r}\right)-\frac{L'_s}{k_r}k_1 \tag{28}$$

Alebo v zložkovom tvare (pre $g = g_r + jg_i$)

$$k_{11} = (g_r-1)\left(\frac{1}{T_1}+\frac{1}{T_r}\right)+g_i\omega$$
$$k_{12} = (1-g_r)\omega+g_i\left(\frac{1}{T_1}+\frac{1}{T_r}\right)$$
$$k_{21} = (1-g_r^2+g_i^2)\left(\frac{L_m}{T_r}-\frac{L'_s}{k_r T_1}\right)-\frac{L'_s}{k_r}k_{11} \tag{29}$$
$$k_{22} = 2g_r g_i\left(\frac{L'_s}{k_r T_1}-\frac{L_m}{T_r}\right)-\frac{L'_s}{k_r}k_{12}$$

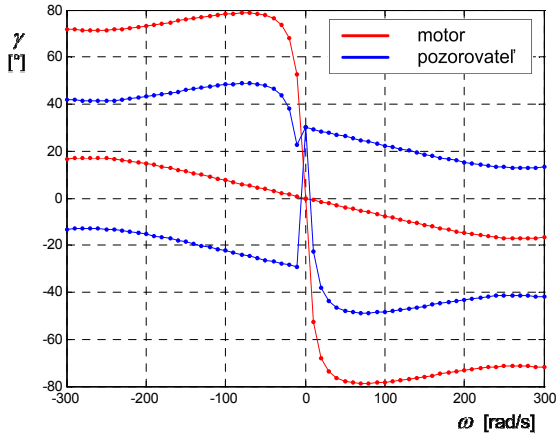
Je vhodné spomenúť, že pri zmene znamienka uhlovej rýchlosti je potrebné zmeniť aj znamienko uhla o ktorý majú byť póly pozorovateľa pootočené oproti pólom motora (to sa prejaví zmenou znamienka parametra g_i vo vzťahoch (29))

V nasledujúcich obrázkoch je naznačené, ako sa mení poloha pólov motora a pozorovateľa pri zmene uhlovej rýchlosti v intervale od -300 do 300 rad/s (krok 10 rad/s je v grafoch vyznačený bodkami). V grafoch sú vykreslené iba dva póly, ďalšie dva póly sú k nim komplexne združené. Uhol pólov γ je do grafu na Obr.4 vynášaný ako uhol od zápornej reálnej osi.



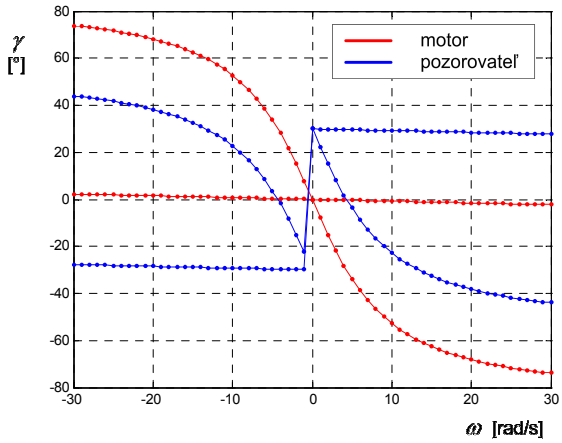
Obr.3 **Poloha pólov motora a pozorovateľa pri zmene uhlovej rýchlosti v intervale od -300 do 300 rad/s v spojitej oblasti**

Fig.3 **Motor and observer root loci for motor speed in interval -300 to 300 rad/s in continuous-time space**



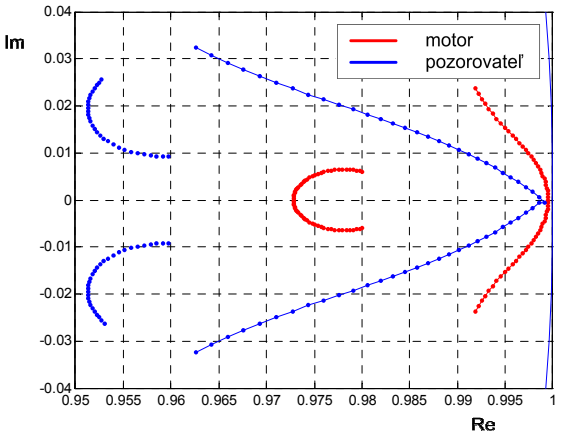
Obr.4 **Uhol pólov motora a pozorovateľa pri zmene uhlovej rýchlosti v intervale od -300 do 300 rad/s**

Fig.4 **Motor and observer poles angle for motor speed in interval -300 to 300 rad/s**



Obr.5 **Detail uhla pólov motora a pozorovateľa pri zmene uhlovej rýchlosti v intervale od -30 do 30 rad/s s krokom 1rad/s**

Fig.5 **Detail of the motor and observer poles angle for motor speed in interval -30 to 30 rad/s with step 1rad/s**



Obr.6 **Poloha pólov motora a pozorovateľa pri zmene uhlovej rýchlosti v intervale od -300 do 300 rad/s v diskrétnej oblasti**

Fig.6 **Motor and observer root loci for motor speed in interval -300 to 300 rad/s in discrete-time space**

Na obrázkoch Obr.3 – Obr.6 si môžeme všimnúť, že póly pozorovateľa sa v porovnaní s pólmi motora stávajú rýchlej-

šie a dominantný pól má relatívne menšiu imaginárnu časť, čím sa stáva stabilnejším. Ďalej si môžeme všimnúť, že v oblasti veľmi malých uhlových rýchlostí, kedy majú póly motora zanedbateľnú imaginárnu časť, tak v tomto prípade sa imaginárna časť pólov pozorovateľa v porovnaní s pólmi motora zväčšuje a navyše pri prechode uhlovej rýchlosti nulou sa skokovo zmení uhol pólov pozorovateľa. Toto môže v oblasti nulových otáčok nepriaznivo vplývať na systém v prípade väčších parametrických nepresností modelu motora alebo v prípade väčších šumov v meracích kanáloch. Na uvedených grafoch si môžeme všimnúť aj zvláštnu polohu rýchlejšieho pólu (v porovnaní so zodpovedajúcim pólom motora). Keďže je tento pól dostatočne rýchly (v porovnaní s druhým pólom pozorovateľa), nemuselo by jeho umiestnenie mať negatívny vplyv na kvalitu pozorovania.

Umiestnenie pólov pozorovateľa ako $p_{LO} = \alpha e^{j\varphi} p_{AM}$ s premenlivým uhlom natočenia

Z priebehov uvedených v predchádzajúcej kapitole vyplýva, že pri umiestnení pólov pozorovateľa podľa predpisu $p_{LO} = \alpha e^{j\varphi} p_{AM}$ vznikajú v oblasti nulovej uhlovej rýchlosti neželané javy, ktoré môžu spôsobiť nižšiu kvalitu pozorovania. Otázkou teda je, ako navrhnúť póly pozorovateľa, aby sa pri malej zmene uhlovej rýchlosti skokovo nemenili (počas prechodu uhlovej rýchlosti nulou) a aby zostali póly pozorovateľa rýchlejšie a s nezväčšenou kmitavosťou v porovnaní s pólmi motora aj pri veľmi nízkych rýchlostiach.

Ak p_{o1} a p_{o2} sú póly pozorovateľa, potom charakteristická rovnica pozorovateľa bude

$$(s - p_{o1})(s - p_{o2}) = s^2 - (p_{o1} + p_{o2})s + p_{o1}p_{o2} = 0 \qquad (30)$$

Ak matica pozorovateľa bude (14), potom (18) je taktiež charakteristická rovnica pozorovateľa. Porovnaním rovníc (18) a (30) môžeme napísať vzťahy pre výpočet parametrov k_1 a k_2 pre ľubovoľné póly pozorovateľa:

$$\begin{aligned} k_1 &= a_1 + a_4 - (p_{o1} + p_{o2}) \\ k_2 &= \frac{p_{o1}p_{o2} + a_2 a_3 - a_4(a_1 - k_1)}{a_2} \end{aligned} \qquad (31)$$

A po spätnej substitúcii

$$\begin{aligned} k_1 &= -\left(\frac{1}{T_1} + \frac{1}{T_r} + p_{o1} + p_{o2} - j\omega\right) \\ k_2 &= \frac{p_{o1}p_{o2}}{\dfrac{k_r}{T_r L'_s} - j\dfrac{k_r\omega}{L'_s}} + \frac{L_m}{T_r} - \frac{L'_s}{k_r}\left(\frac{1}{T_1} + k_1\right) \end{aligned} \qquad (32)$$

Vzťahy (32) nám umožnia ľubovoľné umiestnenie dvoch pólov pozorovateľa, ďalšie dva póly budú k nim komplexne združené.

Ak by sme teda navrhli póly pozorovateľa tak, že modul pólu pozorovateľa bude α násobkom pólu motora, získame tým, že pozorovateľ bude dostatočne rýchly. Ak póly pozorovateľa navyše natočíme o definovaný uhol, môžeme dosiahnuť aj dostatočné tlmenie systému. Tento uhol však nemôže byť fixný, ale bude závisieť od uhlovej rýchlosti – alebo inak povedané, bude závisieť od pólov motora. Cieľom bude natočiť póly pozorovateľa tak, aby sa natočili bližšie k zápornej reálnej osi (v porovnaní s pólmi motora), ale aby sa ani pri nízkych rýchlostiach nedostali až za reálnu os – teda aby zostali v rovnakom kvadrante ako zodpovedajúce póly motora. Ak póly motora budú

$$p_{AM} = |p_{AM}| e^{j\xi} \qquad (33)$$

potom pre póly pozorovateľa môžeme napísať

$$p_{LO} = p_{AM}\alpha e^{j\varphi} \qquad (34)$$

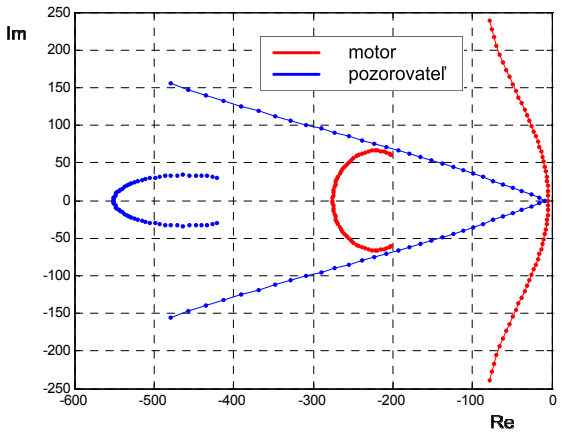
Kde uhol φ vypočítame ako

$$\varphi = \lambda * nrm(\pi - \xi) \qquad (35)$$

Funkcia *nrm*() normuje uhol do rozsahu $\langle -\pi, \pi \rangle$ a λ je číslo z intervalu $\langle 0,1 \rangle$ ktoré určuje ako veľmi sa budú póly pozorovateľa blížiť k zápornej časti reálnej osi. Pre $\lambda = 0$ nebudú póly pozorovateľa vzhľadom na póly motora otočené vôbec, pre $\lambda = 1$ budú póly pozorovateľa natočené tak, že budú ležať na reálnej osi. Tento prístup by sa dal graficky znázorniť Obr.2 s tým, že uhol φ sa bude meniť v závislosti od uhla zodpovedajúceho pólu motora.

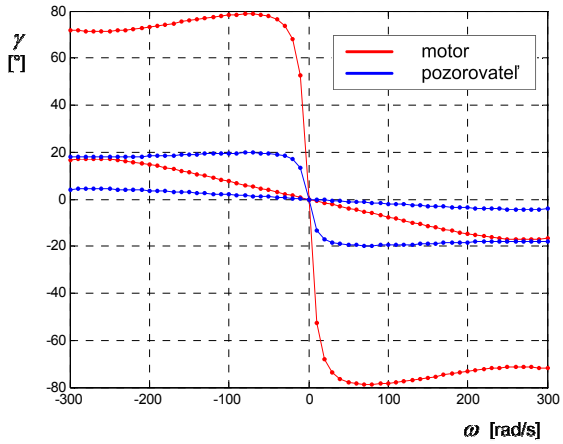
V nasledujúcich obrázkoch je naznačené, ako sa pri tejto metóde mení poloha pólov motora a pozorovateľa pri zmene uhlovej rýchlosti v intervale od -300 do 300 rad/s (krok 10 rad/s je v grafoch vyznačený bodkami). V grafoch sú opäť vykreslené iba dva póly, ďalšie dva póly sú k nim komplexne združené. Uhol pólov γ je do grafu na Obr.8 vynášaný ako uhol od zápornej reálnej osi.

Pozorovateľ bol navrhnutý pre $\alpha = 2, \lambda = 0{,}75$



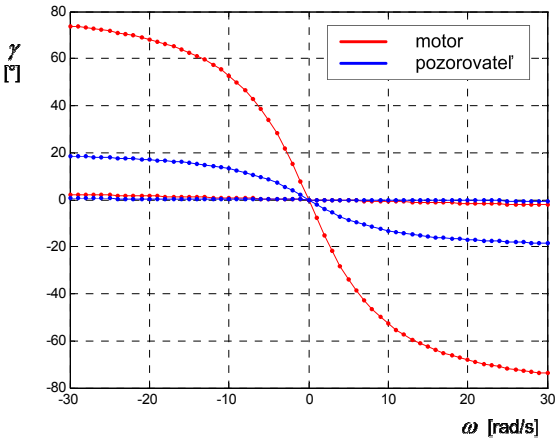
Obr.7 **Poloha pólov motora a pozorovateľa pri zmene uhlovej rýchlosti v intervale od -300 do 300 rad/s v spojitej oblasti**

Fig.7 **Motor and observer root loci for motor speed in interval -300 to 300 rad/s in continuous-time space**



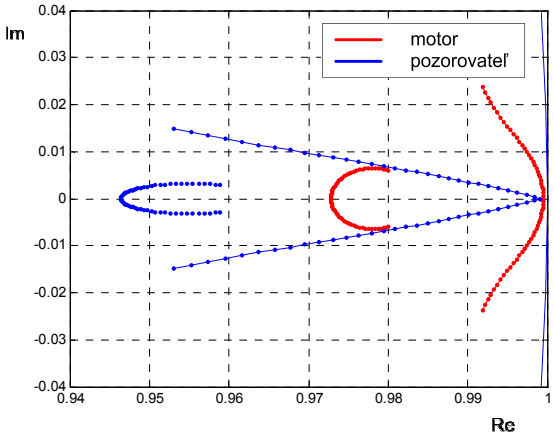
Obr.8 **Uhol pólov motora a pozorovateľa pri zmene uhlovej rýchlosti v intervale od -300 do 300 rad/s**

Fig.8 **Motor and observer poles angle for motor speed in interval -300 to 300 rad/s**



Obr.9 **Detail uhla pólov motora a pozorovateľa pri zmene uhlovej rýchlosti v intervale od -30 do 30 rad/s s krokom 1rad/s**

Fig.9 **Detail of the motor and observer poles angle for motor speed in interval -30 to 30 rad/s with step 1rad/s**



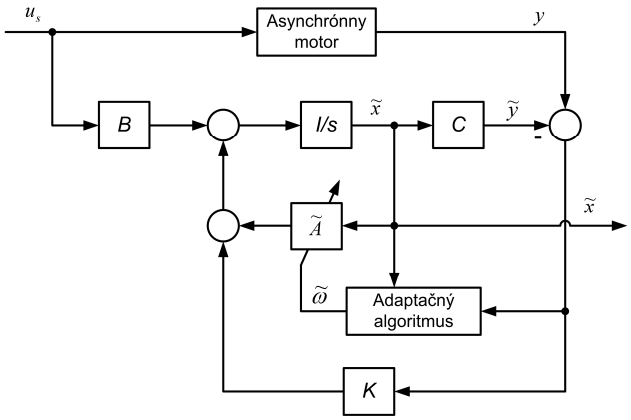
Obr.10 **Poloha pólov motora a pozorovateľa pri zmene uhlovej rýchlosti v intervale od -300 do 300 rad/s v diskrétnej oblasti**

Fig.10 **Motor and observer root loci for motor speed in interval -300 to 300 rad/s in discrete-time space**

Na Obr.7 – Obr.10 je vidieť, že póly pozorovateľa sú rozložené podľa očakávania, aj v oblastiach s veľmi nízkymi uhlovými rýchlosťami sa poloha pólov mení spojito.

Pozorovanie zložiek magnetického toku ELO pozorovateľom s adaptívnym pozorovaním uhlovej rýchlosti

Pre pozorovanie uhlovej rýchlosti v štruktúre rýchlostného servopohonu s Luenbergerovým pozorovateľom máme niekoľko možností. Jednou z nich je rozšíriť vektor pozorovaných stavových veličín o uhlovú rýchlosť. Tým sa však zvýši rád systému a model, ktorý by mal byť použitý pre návrh ELO, sa tým pádom stane zložitejší a teda aj voľba matice zosilnení **K** bude komplikovanejšia. Preto sa v takýchto štruktúrach väčšinou používajú iné metódy na pozorovanie uhlovej rýchlosti. Často sa možno stretnúť s MRAS štruktúrou uvedenou na nasledujúcom obrázku.



Obr.11 **Bloková schéma MRAS pozorovateľa uhlovej rýchlosti AM**

Fig.11 **Block diagram of the MRAS-based IM speed observer**

Na pozorovanie uhlovej rýchlosti bola takáto štruktúra použitá napr. v práci [7]. V práci [12] bola takáto štruktúra doplnená o pozorovanie statorového odporu R_s a použitá v kombinácii s Kalmanovým filtrom (ktorým boli pozorované zložky magnetického toku). V práci [9], [10], [11] bola uvedená štruktúra použitá s modelom motora v rotorovom súradnicovom systéme, čo viedlo k potrebe prepočítavať namerané veličiny do tohto súradnicového systému. V práci [11] bol taktiež tento adaptívny systém doplnený aj o pozorovanie statorového odporu R_s.

Uvedené práce používajú adaptačný zákon odvodený pomocou Ljapunovovej teórie stability.

Ako kandidát na Ljapunovovu funkciu bola zvolená funkcia

$$V = e^T e + \left(\tilde{\omega}_r - \omega_r\right)^2 / \lambda \qquad (36)$$

kde $e = x - \tilde{x}$ a λ je kladná konštanta.

Po derivácii tejto funkcie a úpravách bol pre odhad uhlovej rýchlosti odvodený vzťah

$$\tilde{\omega}_r = K_{I\omega} \int \left(e_{is\alpha}\tilde{\psi}_{r\beta} - e_{is\beta}\tilde{\psi}_{r\alpha}\right) dt \qquad (37)$$

Pre zlepšenie dynamických vlastností pozorovania uhlovej rýchlosti sa odporúča doplniť tento vzťah o proporcionálnu zložku a teda výsledný vzťah pre odhad uhlovej rýchlosti bude

$$\tilde{\omega}_r = K_{P\omega}\left(e_{is\alpha}\tilde{\psi}_{r\beta} - e_{is\beta}\tilde{\psi}_{r\alpha}\right) + K_{I\omega} \int \left(e_{is\alpha}\tilde{\psi}_{r\beta} - e_{is\beta}\tilde{\psi}_{r\alpha}\right) dt \qquad (38)$$

Vzťah je uvedený v statorovom súradnicovom systéme (tak, ako sa používa v prácach [7] a [12]). V prácach, v ktorých bol použitý model systému v rotorovom súradnicovom systéme sa používa rovnaký vzťah, avšak všetky stavové veličiny sú samozrejme v rotorovom súradnicovom systéme. Je treba povedať, že pri derivácii funkcie V pre model systému v statorovom súradnicovom systéme sa vo výrazoch objavili taktiež členy $e_{\psi r\alpha}$, $e_{\psi r\beta}$, ktoré boli zanedbané. Na druhej strane pre model motora v rotorovom súradnicovom systéme treba merané veličiny prepočítavať do tohto súradnicového systému a teda je potrebné poznať uhol natočenia rotora.

V uvedených vzťahoch vystupujú koeficienty $K_{P\omega}, K_{I\omega}$. Tieto koeficienty sú kladné čísla a ich vhodnou voľbou môžeme ladiť rýchlosť a kvalitu pozorovania uhlovej rýchlosti.

Pre parametre $K_{P\omega}, K_{I\omega}$ sme zvolili tieto hodnoty:

$$K_{P\omega} = 0{,}3 \qquad K_{I\omega} = 3.10^4\ s^{-1} \tag{39}$$

Simulačné experimenty

Keďže riadenie AM a taktiež pozorovanie stavových veličín je realizované v diskrétnej oblasti, je potrebné aj pozorovateľ prepísať do diskrétneho tvaru. Pre Luenbergerov pozorovateľ bude v diskrétnej oblasti platiť

$$\begin{aligned}\tilde{\boldsymbol{x}}(k+1) &= \tilde{\boldsymbol{A}}_d\tilde{\boldsymbol{x}}(k) + \boldsymbol{B}_d\boldsymbol{u}(k) + \boldsymbol{K}_d\left(\boldsymbol{y}(k) - \boldsymbol{C}_d\tilde{\boldsymbol{x}}(k)\right) = \\ &= \boldsymbol{A}_{Od}\tilde{\boldsymbol{x}}(k) + \boldsymbol{B}_d\boldsymbol{u}(k) + \boldsymbol{K}_d\boldsymbol{y}(k) \\ \tilde{\boldsymbol{y}}(k) &= \boldsymbol{C}_d\tilde{\boldsymbol{x}}(k)\end{aligned} \tag{40}$$

Diskretizované matice $\mathbf{A_d}$, $\mathbf{B_d}$, $\mathbf{C_d}$ a $\mathbf{K_d}$ možno vypočítať podľa vzťahov:

$$\boldsymbol{A}_{Od} = e^{AT} \approx \boldsymbol{I} + \boldsymbol{A}_O T + \frac{(\boldsymbol{A}_O T)^2}{2}\ldots \tag{41}$$

$$\boldsymbol{B}_d = \int_0^T e^{A\tau}\boldsymbol{B}\,d\tau \approx \boldsymbol{B}T + \frac{\boldsymbol{AB}T^2}{2}\ldots \tag{42}$$

$$\boldsymbol{K}_d = \int_0^T e^{A\tau}\boldsymbol{K}\,d\tau \approx \boldsymbol{K}T + \frac{\boldsymbol{AK}T^2}{2}\ldots \tag{43}$$

$$\boldsymbol{C}_d = \boldsymbol{C} \tag{44}$$

kde T je perióda vzorkovania.

Pre aproximáciu derivácie diferenciou prvého rádu bude pre diskretizované matice platiť:

$$\boldsymbol{A}_{Od} = \boldsymbol{I} + \boldsymbol{A}_O\boldsymbol{T} = \boldsymbol{I} + \left(\tilde{\boldsymbol{A}} - \boldsymbol{KC}\right)T = \boldsymbol{I} + \tilde{\boldsymbol{A}}T - T\boldsymbol{KC} = \tilde{\boldsymbol{A}}_d - \boldsymbol{K}_d\boldsymbol{C} \tag{45}$$

$$\tilde{\boldsymbol{A}}_d = \boldsymbol{I} + \tilde{\boldsymbol{A}}T \tag{46}$$

$$\boldsymbol{K}_d = \boldsymbol{K}T \tag{47}$$

$$\boldsymbol{B}_d = \boldsymbol{B}T \tag{48}$$

$$\boldsymbol{C}_d = \boldsymbol{C} \tag{49}$$

Matice modelu motora definované vzťahmi (2) je možné v diskrétnej oblasti napísať ako

$$\mathbf{A_d} = \begin{bmatrix} 1-\dfrac{T}{T_1} & T\left(\dfrac{k_r}{T_r L'_s} - j\dfrac{k_r\omega}{L'_s}\right) \\ \dfrac{TL_m}{T_r} & 1+T\left(-\dfrac{1}{T_r} + j\omega\right) \end{bmatrix}; \tag{50}$$

$$\mathbf{B_d} = \begin{bmatrix} \dfrac{T}{L'_s} \\ 0 \end{bmatrix}; \quad \mathbf{C_d} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

V simulačných experimentoch bol použitý model motora s týmito parametrami

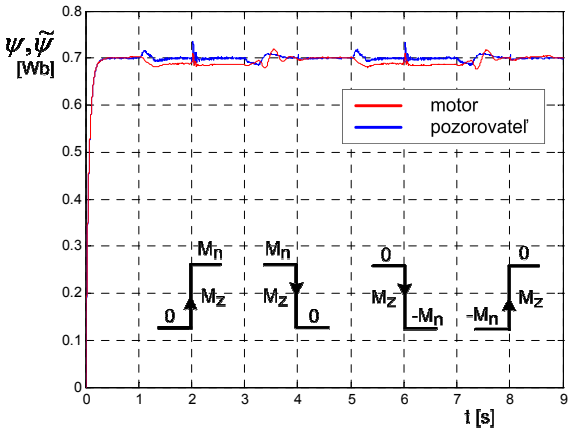
Parametre modelu AM:

Pn = 1,1 kW	Mn = 3,7Nm
n = 2840 min^{-1}	J = 0,002 kgm^2
p‘ = 1	Rs = 7,6 Ω
cos φ = 0,86	Rr = 3,7 Ω
η = 0,71	Ls = 0,6015 H
Un = 380/220 V	Lr = 0,6015 H
In = 2,6/4,5 A	Lm = 0,5796 H

Želaná hodnota modulu magnetického toku bola nastavená na hodnotu 0,7 Wb (čo je nominálna hodnota). Skoky záťažného momentu boli na hodnotu +/- 3,7 Nm, čo je rovnaká hodnota ako hodnota nominálneho momentu použitého motora. K jednotlivým zložkám vektora statorového prúdu bol pripočítaný biely šum s amplitúdou približne 5% menovitého prúdu. Perióda vzorkovania bola T=0.1 ms. Simulácie boli vykonávané pre systém riadený v štruktúre priameho vektorového riadenia.

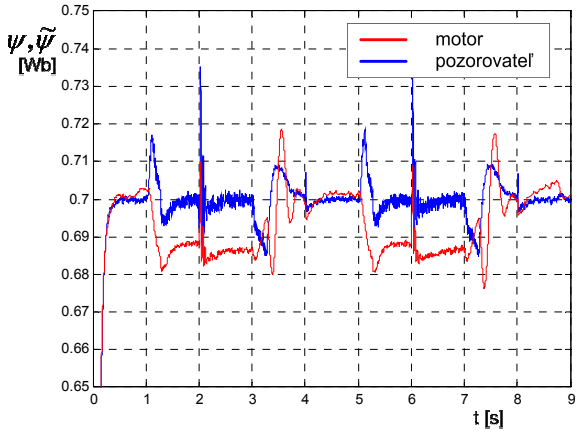
V nasledujúcich simuláciách budeme pojmom skutočná uhlová rýchlosť rozumieť uhlovú rýchlosť vychádzajúcu priamo z modelu asynchrónneho motora.

Ak uvedený algoritmus pozorovania uhlovej rýchlosti použijeme v danej v štruktúre s pozorovateľom pre $\alpha = 2, \lambda = 0{,}75$, dosiahneme nasledujúce výsledky:



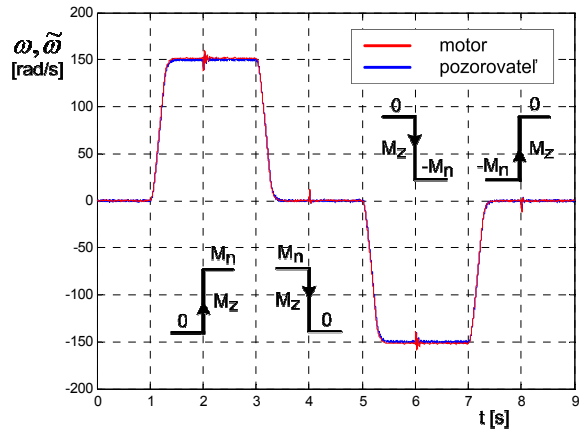
Obr.12 Priebeh modulu pozorovaného a skutočného magnetického toku

Fig.12 Observed and real (from IM-model) magnetic flux magnitude



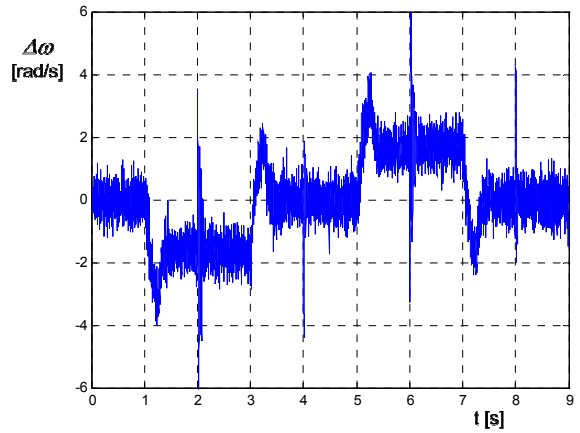
Obr.13 Detail priebehu modulu skutočného a pozorovaného magnetického toku

Fig.13 Detail of the observed and real (from IM-model) magnetic flux magnitude



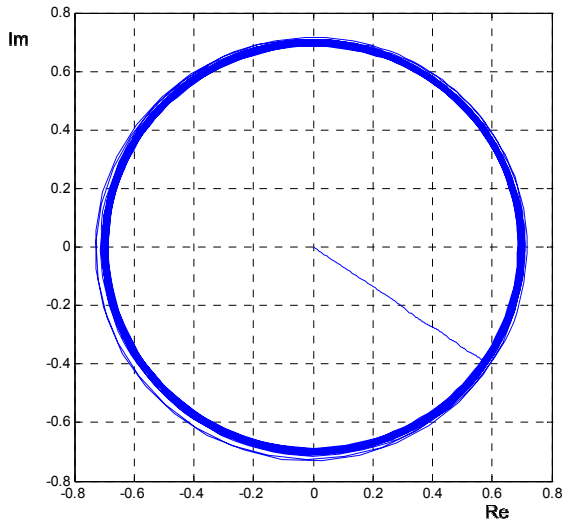
Obr.14 Priebeh pozorovanej a skutočnej uhlovej rýchlosti

Fig.14 Observed and real (from IM-model) speed



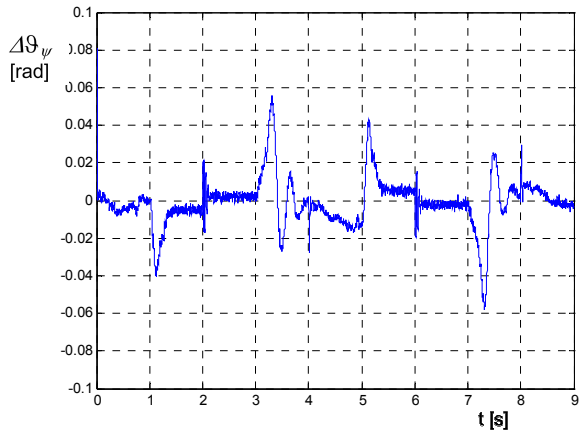
Obr.15 Rozdiel medzi pozorovanou a skutočnou uhlovou rýchlosťou

Fig.15 Observed and real (from IM-model) speed difference



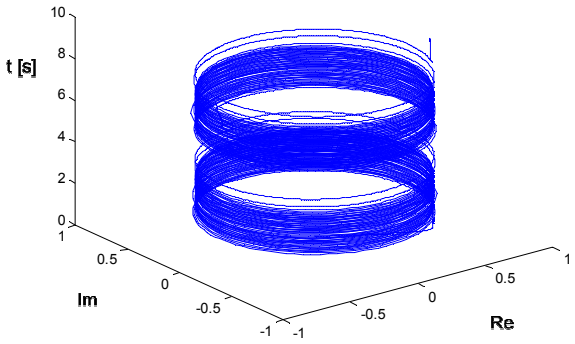
Obr.16 Vektor rotorového magnetického toku

Fig.16 Rotor magnetic flux vector



Obr.17 Rozdiel medzi pozorovaným a skutočným uhlom rotorového magnetického toku

Fig.17 Difference of the observed and real (from IM-model) rotor magnetic flux angle



Obr.18 vektor rotorového magnetického toku

Fig.18 Rotor magnetic flux angle

Ako vidieť z uvedených simulačných experimentov pozorovateľ dokáže dostatočne presne pozorovať rotorový magnetický tok a aj uhlovú rýchlosť.

Záver

Predložený článok sa zaoberal aplikáciou Luenbergerovho pozorovateľa na pozorovanie rotorového magnetického toku AM. Tento pozorovateľ bol aplikovaný na model AM v komplexnom tvare. Pre pozorovateľ s parametrami $\alpha = 2, \varphi = 30^\circ$ (kedy bol modul pólov pozorovateľa dvakrát väčší ako modul pólov motora a póly pozorovateľa boli oproti pólom motora otočené o uhol 30° smerom k zápornej časti reálnej osi) bol analyzovaný pohyb pólov pozorovateľa pri zmene uhlovej rýchlosti a bolo zistené, že v oblastiach s nízkymi uhlovými rýchlosťami, kedy majú póly motora zanedbateľnú imaginárnu časť, sa pólom pozorovateľa zväčšuje imaginárna časť. Taktiež pri prechode uhlovej rýchlosti nulou sa uhol pólov pozorovateľa nespojito mení. Na základe uvedených zistení bola navrhnutá nová metóda rozmiestnenia pólov pozorovateľa označená ako rozmiestnenie s parametrami $\alpha = 2, \lambda = 0{,}75$, kedy α vyjadruje (rovnako ako v predchádzajúcom prípade) pomer modulu pólu pozorovateľa a pólu motora a λ vyjadruje relatívnu zmenu fázy pólov pozorovateľa oproti fáze pólov motora (pre $\lambda = 0$ nebudú póly pozorovateľa vzhľadom na póly motora otočené vôbec, pre $\lambda = 1$ budú póly pozorovateľa natočené tak, že budú ležať na reálnej osi). Takto zvolené póly pozorovateľa majú fázu vždy menšiu alebo rovnú ako je fáza pólov motora a poloha pólov pozorovateľa sa bude meniť spojito aj počas prechodu uhlovej rýchlosti nulou.

Simulačné experimenty boli vykonané s takto navrhnutým pozorovateľom magnetického toku doplneným o adaptívne pozorovanie uhlovej rýchlosti v štruktúre priameho vektorového riadenia. Simulácie ukazujú, že takto navrhnutý pozorovateľ môže byť použitý v štruktúrach rýchlostných servosystémov pracujúcich aj v oblastiach nízkych uhlových rýchlostí.

Literatúra

[1] Abelovský, M.: Pozorovatele stavových veličín bezsnímačových servopohonov s AM, Dizertačná práca, Katedra automatizácie a regulácie, FEI STU, Bratislava, 2003

[2] Armstrong, G., Atkinson, D.: A Comparison of Model Reference Adaptive System and Extended Kalman Filter Estimators for Sensorless Vector Drives, EPE '97, Pages 1424-1429, Trondhaim 1997

[3] Beierke, S., Vas, P., Simor, B., Stroanch, A.: DSP-Controlled Sensorless AC Vector Drives Using The Extended Kalman Filter, Intelligent Motion, Pages 31-41, June 1997

[4] Gacho, J.:, Bezsnímačový rýchlostný servopohon s AM s pozorovaním stavových veličín spätnoväzbovými pozorovateľmi, Dizertačná práca, Ústav riadenia a priemyselnej informatiky, FEI STU, Bratislava, 2007

[5] Gacho, J., Žalman, M.: Realizácia rýchlostného servosystému s asynchrónnym motorom s rozšíreným Kalmanovým filtrom, AT&P Journal, roč.9, 2002, č.2, s. 67-70

[6] Griva, G., Ferraris, P., Profumo, F., Bojoi, R.: Luenberger Observer for High Speed Induction Machine Drives based on a New Pole Placement Method, EPE 2001, Graz, 2001

[7] Kubota, H., Matsuse, K., Nakano, T.: DSP-Based Speed Adaptive Flux Observer of Induction Motor, IEEE Trans. on Industry Applications, Vol. 29, No. 2, Pages 344-348 March/April 1993

[8] Lee, C.,M., Chen, C., L.: Observer-based speed estimation method for sensorless vector control of induction motors, IEE Proc. Control Theory Appl., Vol. 145, No. 3, May 1998

[9] Maes, J., Melkebeek, J.: Adaptive Flux Observer for Sensorless Induction Motor Drives with Enhanced Dynamic Performance, EPE '99, Lausanne, 1999

[10] Maes, J., Melkebeek, J.: Improved Adaptive Flux Observer for Wide Speed Range Sensorless Induction Motor Drives, Proc. Electromotion 6, Pages 49-54, 1999

[11] Maes, J., Melkebeek, J.: Speed-Sensorless Direct Torque Control of Induction Motors Using an Adaptive Flux Obsetver, IEEE Trans. on Industry Applications, Vol. 36, No. 3, Pages 778-785 May/June 1993

[12] Mora, J. L., Torralba, A., Franquelo, L. G.: A Speed Adaptive Kalman Filter Observer For Induction Motors, EPE 2001, Graz, 2001

[13] Ohyama, K., Asher, G., Sumner, M.: Comparison of the Practical Performance and Operating Limits of Sensorless Induction Motor Drive using a Closed Loop Flux Observer and a Full Order Observer, EPE '99, Lausanne 1999

[14] Schauder, C.: Adaptive Speed Identification for Vector Control of Induction Motors without Rotational Transducers, IEEE Trans. Ind. Appl., Volume 28, No. 5, September/October 1992

[15] Texas Instruments: Sensorless Field Oriented Speed Control of Three Phase AC Induction Motor using TMS320F240, Texas Instruments Europe, France, 1998

[16] Vas, P.: Sensorless Vector and Direct Torque Control, Oxford University Press, New York 1998

[17] Žalman, M., Abelovský, M., Jovankovič, J.: Otvorené rýchlostné servosystémy s AM, Riadenie v energetike, s. 137-142, Bratislava 2000

Abstract

This paper deals with Luenberger observer to observe state varialbles values of IM in speed servodrive structure. In the first part was analysed the root loci of the motor and obsrver for variable motor speed with observer poles chosen as $p_{LO} = \alpha e^{j\varphi} p_{AM}$. Then a new method was presented for the observer roots position also with method for the Luenberger gain matrix calculation for observing the rotor magnetic flux. MRAS-based adaptive scheme was used for motor speed estimation. The functionality of the system was tested in MATLAB Simulink program.

prof. Ing. Milan Žalman, PhD.
Ing. Juraj Gacho, PhD.

Slovenská technická univerzita
Fakulta elektrotechniky a informatiky
Ústav riadenia a priemyselnej informatiky
Ilkovičova 3, 812 19 Bratislava
milan.zalman@stuba.sk
juraj@amit.sk

Mobilné servisné roboty

Rastislav Baláž, Mária Ádiová, Ľubica Miková

Abstrakt
Článok sa zaoberá mobilnými servisnými robotmi a rozdelením ich mobilným subsystémov. Sú charakterizované jednotlivé typy a druhy podvozkov.

Kľúčové slová: robot, subsystém mobility, kolesový podvozok, kráčajúci podvozok,

Úvod

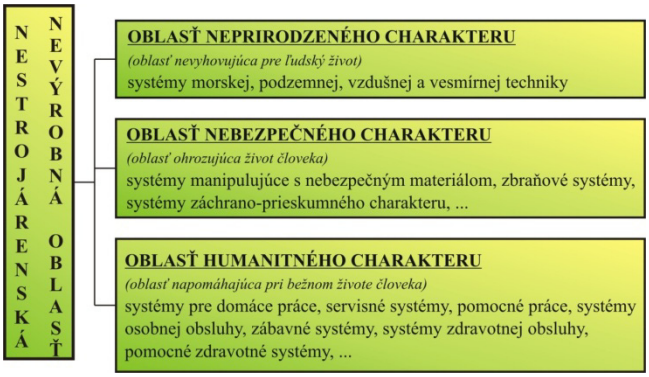
Mechatronika a robotika má silný interdisciplinárny charakter. Opiera sa o rad technických, humanitných a spoločenských vied. Majú obrovský dopad na všetky priemyselné odvetvia, má darvinovský vplyv na rozvoj a štruktúru technických vied, pričom neustále kladie nové nároky a požiadavky na pohybové systémy, meranie, senzoriku a umelú inteligenciu, t.j. aplikujú sa mechatronické prístupy a princípy.

Významnou oblasťou robotiky je mobilná robotika, ktorej značný podiel tvorí oblasť servisnej robotiky. Úlohou mobilných servisných robotov (MSR) sú vo všeobecnosti inšpekčné a servisné operácie, ktoré majú zabezpečiť uľahčenie práce človeku a riskovanie vlastného "života" v zdraviu škodlivom a nebezpečnom prostredí, čiže v prostredí ohrozujúci život človeka.

Spektrum využitia mobilných servisných robotov je rozsiahli, ako aj ich konštrukčné riešenie podvozkov. Najpoužívanejším typom mobilných robotov sú roboty s kolesovým podvozkom a ich modifikácie. Druhým najpoužívanejším podvozkom sú mechanizmy napodobňujúce pohyb zvierat v prírode tzv. "kráčajúce roboty". Výhodou takéhoto druhu podvozku je jeho využitie v členitejšom teréne.

Rozdelenie MSR podľa ich oblasti využitia

Mobilné servisné roboty majú širokú oblasť využitia. Predovšetkým svoje uplatnenie nachádzajú v oblasti ohrozujúcej život človeka, v oblasti, ktorá je ťažko alebo vôbec dostupná, ale i v oblasti humanitného charakteru – vykonávanie činnosti úzko spojených s činnosťou človeka (pomocné, opatrovateľské úkony) a v oblasti zábavy.



Obr.1 Rozdelenie MSR podľa ich oblasti použitia

Fig.1 Distribution of MSR following their field of application

Rozdelenie MSR podľa pracovného prostredia

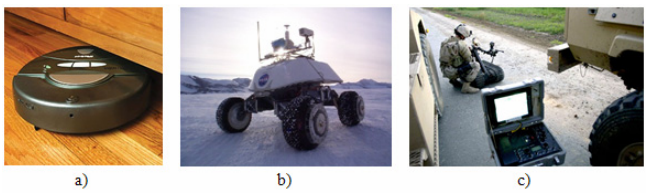
Delíme ich na:

- indoor – prostredie jednoznačne definované (interiér budov a hál, nádrže, potrubia, kde poloha a rozmery prekážok sú nám známe), bez uvažovanie vplyvu poveternostných podmienok a rušivých faktorov.
- outdoor – prostredie nie je vôbec jednoznačne definované alebo definované len na základe minimálnych informácií, či dohadov (okolie budov, pole, lesy, krátery, vesmír,...) s uvažovaním vplyvu rušivých faktorov a vplyvu poveternostných podmienok.

Obr.1 a) robot vo vonkajšom pracovnom prostredí [1], b) robot vo vnútornom pracovnom prostredí [2]

Fig.1 a) robot in outdoor working environment, b) robot in indoor working environment

Rozdelenie MSR podľa spôsobu riadenia



Obr.2 a) robot s plne autonómnym riadením[3], b) robot s čiastočne autonómnym riadením[4], c) robot riadení teleoperátorom[5]

Fig.2 a) robot with full autonomous regulated, b) robot with the in part autonomous regulated c) robot regulated tele-operator

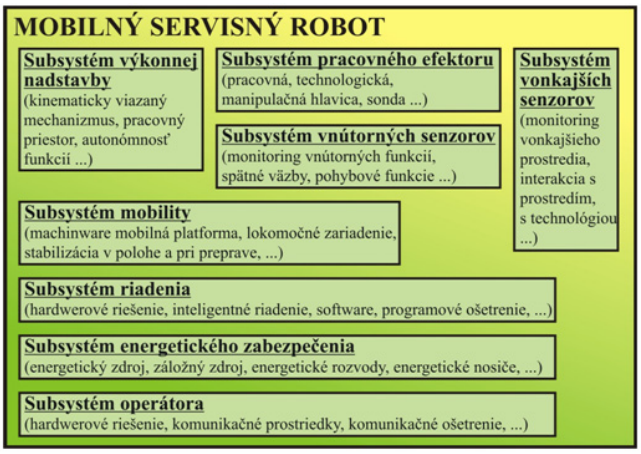
Delíme ich na:

- plne autonómne riadenie – riadenie, pod ktorým robot vykonáva pracovné úlohy samostatne, bez zásahu obsluhy, podľa vopred zadaných inštrukcií. V nepredvída-

nej situácii má operátor možnosť zmeny pracovnej úlohy alebo k jej predčasnému ukončeniu.
- čiastočne autonómne riadenie – riadenie robota operátorom je na primárnej úrovni, sekundárne riadenie vykonáva riadiaci systém robota.
- teleriadenie – robot je riadený operátorom, ktorý využíva pre riadenie informácie so snímačov robota.

Základná architektúra MSR

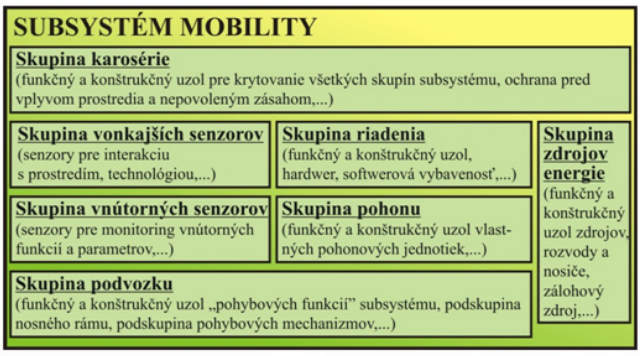
Vo všeobecnosti všetky MSR majú rovnakú funkčnú štruktúru. Jedná sa o subsystémy, ktoré tvoria akýsi základ každého MSR. Ide o subsystém mobility, subsystém riadenia, subsystém senzoriky, subsystém výkonnej nadstavby, ďalej energetický subsystém, subsystém operátora, subsystém pracovného efektoru.



Obr.3 Štruktúra subsystémov v MSR

Fig.3 The structure of the subsystems in MSR

Keďže tento článok pojednáva o mobilných sústavách, ďalej sa budeme zaoberať len subsystémom mobility.



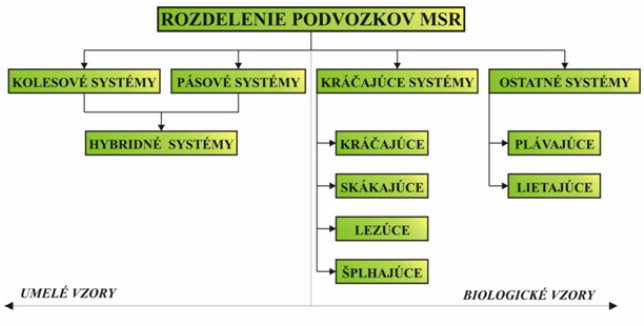
Obr.4 Subsystém mobility[6]

Fig.4 Mobility Subsystém

Subsystém mobility – realizuje voľne programovateľné premiestňovanie MSR v jeho operačnom priestore – technika a technológia premiestňovania je priamo viazaná na charakter priestoru a koncepcie riešenia subsystému – subsystém realizuje stabilizáciu MSR počas pohybu v určenej pracovnej polohe a pri výkone pracovnej funkcie.[6]

Základné rozdelenie podvozkov MSR - súčasný stav

Podľa súčasného stavu vo svete môžeme podvozky kvalifikovať do niekoľkých skupín. Tieto skupiny sa kvalifikujú podľa svojho typu a usporiadania lokomočnej štruktúry. Lokomočné štruktúry rozdeľujeme do dvoch základných skupín. A to, ak svoje usporiadanie lokomočnej štruktúry prevzali z prírodného sveta – biologické vzory alebo svoj vzor majú v umelých systémoch, vytvorených človekom – umelé vzory.



Obr.5 Rozdelenie podvozkov MSR

Fig.5 Distribution undercarriages MSR

Kolesový systém - princíp a konštrukcia vychádza z aplikácie riadeného kolesového podvozku, pre realizáciu požadovaných funkcií v priestore, systém umožňuje obecný pohyb po vodorovnej a šikmej ploche v ľubovoľnom smere a pohybu.[6]

Šplhajúci systém – princíp a konštrukcia vychádza z kombinácie biokinetických modelov hornej a dolnej končatiny biologických tvorov (rameno/noha), systém umožňuje pohyb po zvislej ploche v ľubovoľnom smere pohybu.[6]

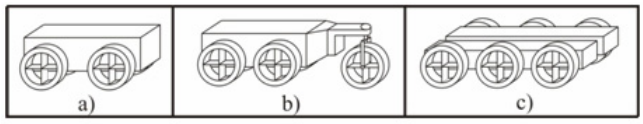
Lezúci systém – princíp a konštrukcia vychádza z biologického modelu plazov, systém umožňuje pohyb po vodorovnej, šikmej a zvislej ploche v ľubovoľnom smere pohybu.[6]

Plávajúci systém – princíp a konštrukcia vychádza z biologického modelu vodných resp. obojživelných živočíchov, systém umožňuje pohyb v kvapalnom prostredí v ľubovoľnom smere pohybu.[6]

Lietajúci systém – princíp a konštrukcia vychádza z biologického modelu vtákov, systém umožňuje pohyb vo vzdušnom prostredí v ľubovoľnom smere pohybu.[6]

MSR s kolesovým podvozkom

Práve tento subsystém mobility je najviac používaným riešením typu podvozku pre MSR. U týchto typov podvozkov je prvoradým problémom správne navrhnutie koncepcie podvozku a to počet a usporiadanie hnacích, hnaných a riadiacich kolies.

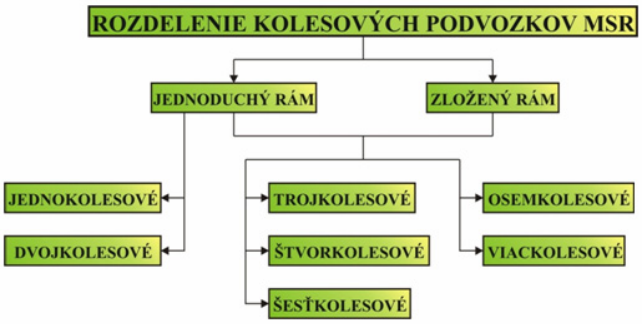


Obr.6 a) štvorkolesový podvozok s riadením všetkých kolies, b) päťkolesový podvozok s jedným riadením kolesom, c) šesťkolesový podvozok s deleným rámom a riadením všetkých kolies

Fig.6 a) four-wheel undercarriage with the regulated of all wheel, b) five-wheel undercarriage with one regulated wheel, c) six-wheel undercarriage with the divide casing and regulated if all wheel

Ďalším faktorom pri návrhu podvozku je jeho využitie v pracovnom prostredí, pretože použitie takéhoto typu podvozku nie je možné v členitom teréne pracovného prostredia. Prechod cez prekážky je daný veľkosťou kolies, avšak treba brať do úvahy, že so zväčšovaním priemerov kolies sa zvyšuje vzdialenosť ťažiska podvozku vzhľadom ku povrchu resp. celého MSR, čo sa prejavuje na zhoršených jazdných

vlastnosti a riadenia. Kolesové podvozky môžeme jednoducho rozdeliť podľa počtu kolies a typu podvozku.

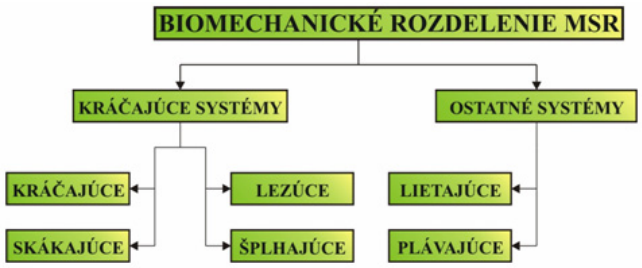


Obr.7 Rozdelenie kolesových podvozkov

Fig.7 Distribution of flywheel undercarriages

MSR s biomechanickým princípom podvozku

Do tejto skupiny MSR patria všetky kráčajúce a skákajúce roboty, roboty s plazivým pohybom a šplhajúce roboty. Vzorom pre tieto riešenia podvozkov je samotné príroda, ktorá svojou evolúciou doviedla takpovediac takéto vzory k dokonalosti. Všetky tieto biomechanické konštrukcie MSR nachádzajú svoje uplatnenie väčšinou v prostredí, kde uplatnenie MSR s kolesovým podvozkom nie je možné. Je to prostredie charakteristické členitým povrchom s prekážkami rôzneho typu a veľkosti.

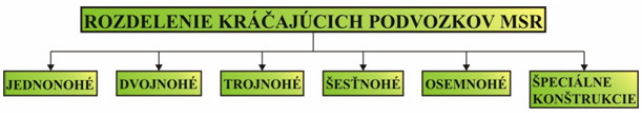


Obr.8 Biomechanické rozdelenie MSR

Fig.8 Biomechanics distribution of MSR

MSR s kráčajúcim podvozkom

Tento subsystém mobility je druhým najpoužívanejším podvozkom. Základné rozdelenie je nasledovné:



Obr.9 Rozdelenie kráčajúcich podvozkov

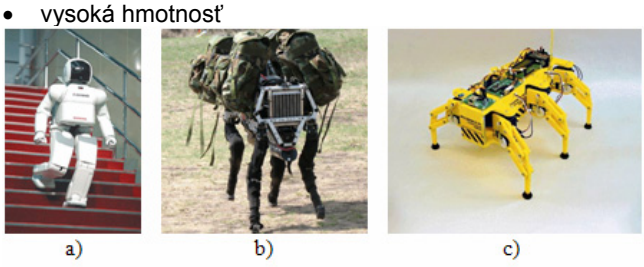
Fig.9 Distribution of walking undercarriages

Výhody kráčajúcich robotov[7]:

- môžu prekonať relatívne vysoké prekážky
- môžu sa pohybovať po schodoch hore a dole
- môžu prekonať priehlbiny a priekopy, pohybovať sa po extrémne členitom povrchu a zdolávať príkre svahy
- môžu sa plynulo pohybovať po značne nerovnom teréne pomocou nastaviteľnej výške tela nad povrchom terénu a to zmenou natiahnutia (vysunutia, zdvihnutia) nohy a vyrovnať tak vlnitosť povrchu
- nohy sa menej zabárajú do povrchu a spôsobujú menšie poškodenia podlažia ako kolesové a pásové podvozky

Nevýhody kráčajúcich robotov[7]:

- vyšší počet nezávisle riadených stupňov voľnosti
- vyšší počet akčných členov (pohony, prevody, senzory)
- zložitejší riadiaci systém
- nutnosť dodávať energiu po dávkach do rôznych miest
- malá energetická účinnosť
- výrobná náročnosť
- nízka nosnosť
- vysoká hmotnosť



Obr.10 a) robot s dvojnohým podvozkom – humanoid[8], b) robot so štvornohým podvozkom[9], c) robot so šesťnohým podvozkom[10]

Fig.10 a) robot with the bipedal undercarriage - humanoid, b) robot with the quadrupedal undercarriage, c) robot with the hexapodous undercarriage

Záver

Článok poukazuje na mobilné servisne systémy. Ich základné rozdelenie, s popisom jednotlivých robotov. Ďalej sa zaoberá subsystémom mobility, kde tento subsystém je rozdelený do dvoch základných kategórií podľa ich podstaty vzniku - umelé alebo biologické vzory, ďalej sú tieto komplexnejšie kategórie systémov rozdelené podrobnejšie, spolu s stručným výkladom.

Poďakovanie

Autori týmto ďakujú Slovenskej grantovej agentúre pre vedu GU VEGA 1/0201/08 „Výskum štruktúr a správania sa modulov mechatronickéj mobilné technické sústavy na úrovni orgánov a stavebných prvkov za účelom zlepšenia vlastnosti mobilnej technickej sústavy“ a GU VEGA 1/0464/09 „Výskum mechatronických sústav imitujúcich lokomóciu hada v obmedzenom a premenlivom priestore.“

Literatúra

[1] WIKIMEDIA COMMONS [online] 2009 [cit. 2008-10-12] Dostupné a internete: <http://commons.wikimedia.org/wiki/File:NASA_Mars_Rover.jpg>

[2] ALTIVIS [online] 2005 [cit. 2008-10-12] Dostupné na internete: < http://www.altivis.fr/-Les-robots-japonais-.html>

[3] KABBODLE [online] 2009 [cit. 2008-10-18] Dostupné na internete: <http://www.kaboodle.com/reviews/irobot-roomba-416-vacuum-cleaning-robot>

[4] THE COMPETITION [online] 2008 [cit. 2008-10-18] Dostupné na internete: <http://www.mobilerobot.org/MRSC/Competition.htm>

[5] 800HIGHTECH [online] 2008 [cit. 2008-10-18] Dostupné na internete: < http://blog.800hightech.com/microsoft-xbox-360-joins-the-military/420/>

[6] SMRČEK, J.: prednášky KAMaM SjF Košice, Košice 2008

[7] LOKOMOČNE MECHANIZMY HUMANOIDNÝCH ROBOTOV, SCIRANKA, M.: NOVUS SCIENTIA 2007 [online]

2007 [cit. 2008-10-25] Dostupné na internete: <http://www.sjf.tuke.sk/novus/papers/533-536.pdf>

[8] TMS [online] 2003 [cit. 2008-10-19] Dostupné na internete: <http://www.tms.org/pubs/journals/JOM/0311/Byko-0311.html>

[9] DARKGOVERNMENT [online] 2008 [cit. 2008-10-19] Dostupné na internete: <http://www.darkgovernment.com/news/the-darpa-military-robot/>

[10] MINDCREATORS [online] 2002 [cit. 2008-10-19] Dostupné na internete: <http://www.mindcreators.com/ResearchOutline.htm >

Abstract

The article deals with a brief dividing mobile service robots and distribution of their mobile subsystem. They are characterized particular types and kinds of undercarriages.

Rastislav Baláž, Ing.

Technická univerzita v Košiciach
Strojnícka fakulta, Ústav špeciálnych technických vied
Katedra aplikovanej mechaniky a mechatroniky
Letná 9
042 00 Košice
00421 55 602 2719
E-mail: rastislav.balaz@tuke.sk

Mária Ádiová, Ing.

Technická univerzita v Košiciach
Strojnícka fakulta, Ústav špeciálnych technických vied
Katedra aplikovanej mechaniky a mechatroniky
Letná 9
042 00 Košice
00421 55 602 2719
E-mail: maria.adiova@tuke.sk

Ľubica Miková, Ing.

Technická univerzita v Košiciach
Strojnícka fakulta, Ústav špeciálnych technických vied Katedra aplikovanej mechaniky a mechatroniky Letná 9
042 00 Košice
00421 55 602 2719
E-mail: lubica.mikova@tuke.sk

Detekcia značiek prostredia pomocou laserového skenera

František Duchoň, Ladislav Jurišica

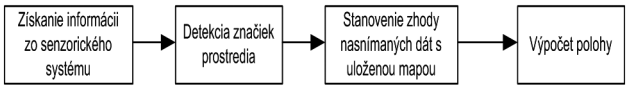
Abstrakt
V mobilnej robotike sa používajú snímače vzdialeností na extrahovanie opisu prostredia, teda vytvorenie istej mapy prostredia. Pre inteligentnú navigáciu mobilného robota v prostredí je však vhodné poznať aj isté významné vlastnosti prostredia, ktoré prispievajú k zvýšeniu kvality informácií o prostredí. Takéto významné vlastnosti prostredia nazývame značky prostredia. Článok sa zaoberá detekciou značiek prostredia s využitím laserového skenera.

Kľúčové slová: značka prostredia, umelé a prirodzené značky prostredia, filtrácia dát, segmentácia dát, detekcia značiek prostredia

Úvod

V robotike sú značky prostredia vlastnosti prostredia, ktoré robot dokáže rozlíšiť z dát získaných zo senzorického systému. Týmito vlastnosťami môžu byť napríklad geometrické tvary (obdĺžniky, čiary, kružnice), ktoré môžu navyše obsahovať prídavné informácie (napr. v podobe čiarových kódov). Aby bol robot schopný určiť svoju polohu relatívne k týmto značkám, poloha značiek býva pevná a jednoznačne určená v prostredí.

Značky musia byť ľahko detekovateľné a identifikovateľné. Charakteristické vlastnosti použitých značiek musia byť známe predtým, ako budú použité pre lokalizáciu, navigáciu, prípadne korekciu chyby odometrie. Hlavnou úlohou pri lokalizácii robota v prostredí je spoľahlivá identifikácia detekovanej značky. Všeobecný proces určenia polohy s použitím značiek prostredia je zobrazený na Obr. 1.



Obr.1 Proces určenia polohy s použitím značiek prostredia

Fig.1 Assignment of position with environment mark process

Z hľadiska pôvodu môžeme definovať 2 typy značiek prostredia:

- prirodzené značky prostredia
- umelé značky prostredia

Prirodzené značky prostredia sa najviac vyskytujú v štruktúrovanom prostredí, ako napr. chodby, priemyselné haly alebo nemocnice. V skutočnosti prirodzené značky fungujú najlepšie, ak sú skutočne vyrobené človekom (ako napr. v prípade štruktúrovaného prostredia). Z tohto dôvodu môžeme definovať pojmy prirodzené a umelé nasledovne:

Prirodzené značky sú objekty alebo ich vlastnosti, ktoré sú už súčasťou prostredia a neslúžia primárne na navigáciu robota.

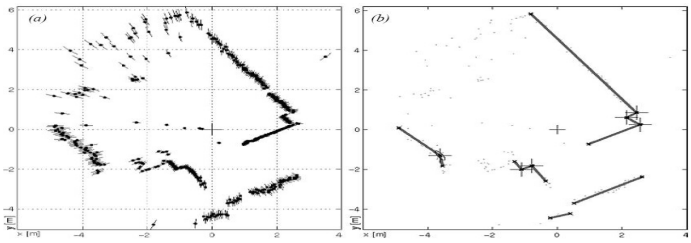
Umelé značky sú špeciálne navrhnuté objekty alebo návestia, ktoré musia byť umiestnené v prostredí za výhradným účelom navigácie robota.

Prirodzené značky prostredia

Najväčším problémom použitia prirodzených značiek býva detekcia a určenie zhody charakteristických vlastností zo senzorického systému. Pre rozpoznanie, detekciu, identifikáciu a zachytenie čo možno najviac vlastností prostredia je najvhodnejší vizuálny senzorový systém. V takomto prípade sú najčastejšie využívanými prirodzenými značkami dlhé kolmé hrany, ako napr. dvere alebo rohy stien, alfanumerické znaky, prípadne stropné svetlá. Jediným kritériom je, aby boli značky rozoznateľné od okolitej scény (pozadia) farebne, alebo kontrastne.

Ak sú na detekciu prirodzených značiek použité diaľkomery (laserový skener, IR skener, ultrazvukový skener – sonar) bývajú dobrými kandidátmi na detekciu najmä jednoznačné rysy objektov ako napríklad rohy alebo hrany dlhých rovných stien (Obr 2.). Vhodný výber vlastností je veľmi dôležitý, pretože je ním určená celá komplexnosť opisu značiek, detekcie a určenia zhody. Správna voľba vlastností značiek takisto znižuje možnosť nejednoznačnosti a zvyšuje presnosť určenia polohy. Systém určenia polohy použitím prirodzených značiek má tieto základné zložky:

- Senzor (najčastejšie vizuálny systém alebo laserový skener) pre detekciu značky prostredia a jej odlíšenia od ostatných objektov.
- Algoritmus určenia zhody nasnímaných značiek s mapou značiek robota.
- Algoritmus výpočtu polohy a chyby určenia polohy.



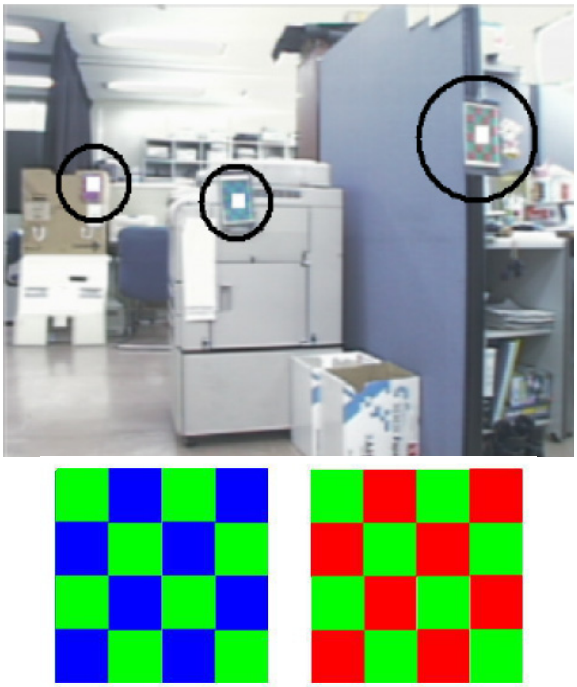
Obr.2 Dáta nasnímané laserovým skenerom a následná extrakcia značiek [1]

Fig.2 Data obtained with laser scanner and consistent extraction of the marks

Umelé značky prostredia

Detekcia a identifikácia umelých značiek prostredia je oveľa jednoduchšia, ak sú optimálne navrhnuté pre čo najväčší možný kontrast s prostredím, s tým, že presná veľkosť a tvar umelých značiek je známa vopred. V tejto oblasti boli testované rôzne druhy značiek, pričom metódy a techniky určenia polohy sa líšili v závislosti od použitých značiek [2]. Väčšina systémov polohovania je založených na použití vizuálnych systémov.

Fukui [3] použil značku tvaru kosoštvorca. Borenstein [4] použil čierny obdĺžnik so 4 bielymi bodkami na rohoch. Kabuka a Arenas v [5] použili napol čierny, napol biely kruh s jedinečným čiarovým kódom pre každú značku. Magee and Aggarwal [6] použili guľu s horizontálnymi a vertikálnymi kalibračnými kruhmi aby docielili trojrozmernú lokalizáciu z jednoduchého obrazu.



Obr.3 Ukážky umelých značiek použitých pre sledovanie značiek pohybujúceho sa robota [7]

Fig.3 Examples of manmade marks used for marks following robot

Presnosť dosiahnutá detekciou umelých značiek prostredia závisí od presnosti parametrov, s akou sú značky extrahované z obrazovej informácie vizuálneho systému, a takisto od relatívnej polohy a uhlu medzi robotom a značkami. Všeobecne platí že presnosť klesá s narastajúcou relatívnou vzdialenosťou robota od značky. V praxi sa definuje rozsah relatívnych uhlov medzi robotom a značkami, kde je zaručená dobrá presnosť. Presnosť určenia polohy robota potom výrazne klesá, ak je relatívny uhol mimo tohto rozsahu.

Množstvo umelých značiek existuje aj pre systémy s využitím laserových skenerov. Najčastejšie sú nasadzované reflektory s čiarovým kódom. Tvar značky je v tomto prípade zvyčajne nedôležitý. Zaujímavou je určite aj metóda od Fenga v práci [8], kde použil kruhovú značku a aplikáciou Houghovej transformácie extrahoval parametre elipsy z obrazu v reálnom čase.

Chyby merania použitím značiek prostredia

Pri využití metódy určenia polohy na základe využitia značiek prostredia sa môžeme stretnúť s dvoma najčastejšími chybami:

1. Pri triangulácii je chyba merania funkciou relatívnej polohy robota R a značkami prostredia L_i :

$$\varepsilon = f(R, L_i) \tag{1}$$

2. Pri určení polohy na základe geometrickej zhody objektu je chyba merania funkciou vzdialenosti d a uhla θ medzi robotom R a značkami prostredia L_i .

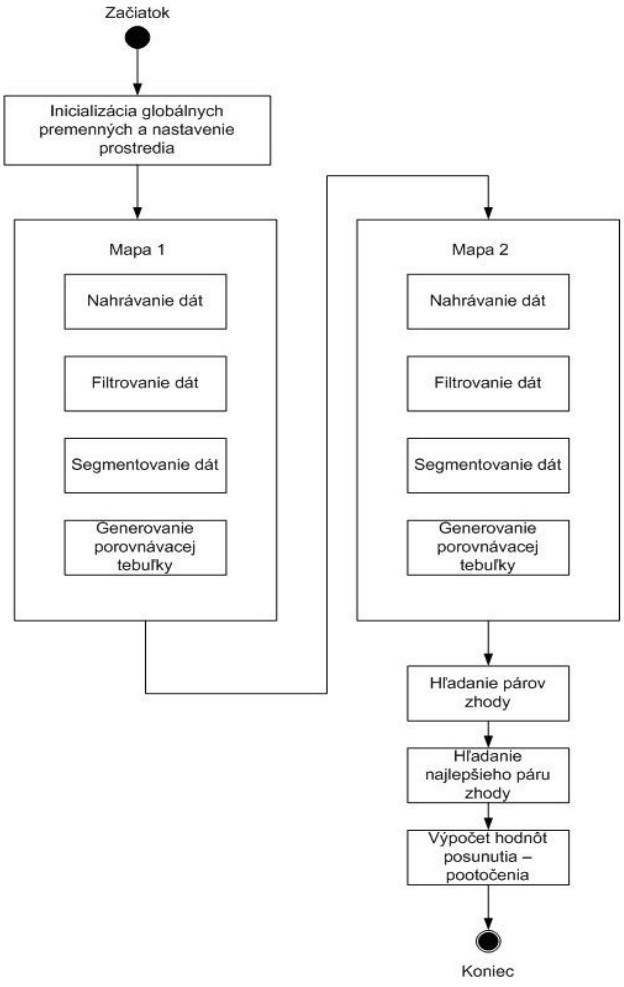
$$\varepsilon = f(d, \theta) \tag{2}$$

Z uvedených informácii môžeme zosumarizovať vlastnosti lokalizácie použitím značiek nasledovne:

- *Prirodzené značky* prostredia poskytujú väčšiu flexibilitu pričom nevyžadujú modifikáciu okolia.
- *Umelé značky* prostredia sú pomerne lacné a môžu v sebe obsahovať prídavnú zakódovanú informáciu.
- Presnosť polohovania závisí od vzdialenosti a uhla medzi robotom a značkou - čím bližšie sa robot nachádza pri značke, tým presnejšie je určenie polohy.
- Okolité podmienky ako napr. osvetlenie môžu byť zdrojom chyby, Značky vôbec nemusia byť detekované, alebo objekty s podobnými vlastnosťami môžu byť omylom detekované ako značky.
- Značky musia byť prítomné v pracovnom okolí robota.
- Navigácia pomocou značiek prostredia vyžaduje informáciu o približnej polohe robota. Ak nie je pozícia známa, robot musí vykonať globálnu lokalizáciu, čo môže predstavovať časovo náročnú operáciu.
- Databáza značiek prostredia a ich umiestnenie v prostredí musí byť korektne spravovaná.

Navrhnutá metóda detekcie značiek

Na otestovanie detekcie prirodzených značiek prostredia pomocou laserového skeneru boli navrhnuté vlastné metódy pozostávajúce z filtrovania dát, segmentácie dát a detekovania značiek prostredia. Schému celkového algoritmu na určenie relatívnej polohy mobilného robota s využitím detekcie značiek možno vidieť na Obr. 4. V úvode aplikácie sa inicializujú všetky potrebné globálne premenné a vlastnosti prostredia. Pomocou týchto hodnôt sa dajú modifikovať vlastnosti použitých filtrov a algoritmov zhody značiek prostredia.



Obr.4 Schéma algoritmu

Fig.4 Algorithm scheme

Filtrácia dát

Dáta z laserového skenera sú vzhľadom k chybám merania zaťažené chybou. Preto je nutné tieto merania upraviť - filtrovať. Pri návrhu filtrov bolo potrebné zachovať charakteristiku prostredia a teda neskresliť dáta, napríklad pri ostrých lomoch apod.

Na základné (hrubé) filtrovanie bola použitá funkcia `RawFilter`. Jej úlohou je odstránenie chýbajúcich a neplatných dát merania. Medzi tieto dáta patria napr. chybové hlásenia skenera. Funkcia nuluje hodnoty nenachádzajúce sa v intervale <*min_res;max_res*>, kde *min_res* predstavuje minimálnu vzdialenosť, pri ktorej považujeme meranie za platné a *max_res* predstavuje maximálnu platnú vzdialenosť. V prípade výskytu chýbajúcich dát v sekvencii platných meraní, ktorých počet je menší ako hodnota *max_var*, funkcia vykoná aproximáciu chýbajúcich dát za účelom zvýšenia konzistencie dát. Premenná *max_var* predstavuje počet neplatných meraní, ktoré budú nahradené aproximovanými hodnotami. Princíp možno vysvetliť na jednoduchom príklade. Máme pole dát s hodnotami vzdialeností napr.:

data = [800, 803, 805, 0, 0, 0, 815, 819, 823…]

Je zrejmé že tri nulové hodnoty v sekvencii platných dát predstavujú chybu merania. Funkcia `RawFilter` túto situáciu detekuje a chybné hodnoty aproximuje. Z poľa dát vezme 5 hodnôt (ak sú k dispozícii, ak nie, vezme sa toľko hodnôt koľko sa dá) pred detekciou chybných údajov. Z týchto hodnôt sa vypočíta priemerný koeficient rastu (prípadne klesania) a tento koeficient sa pripočíta k poslednému platnému údaju. Ďalšie hodnoty sú vypočítané násobkami koeficientov v závislosti od počtu neplatných údajov. V tomto prípade by aproximované údaje boli vypočítane nasledovne:

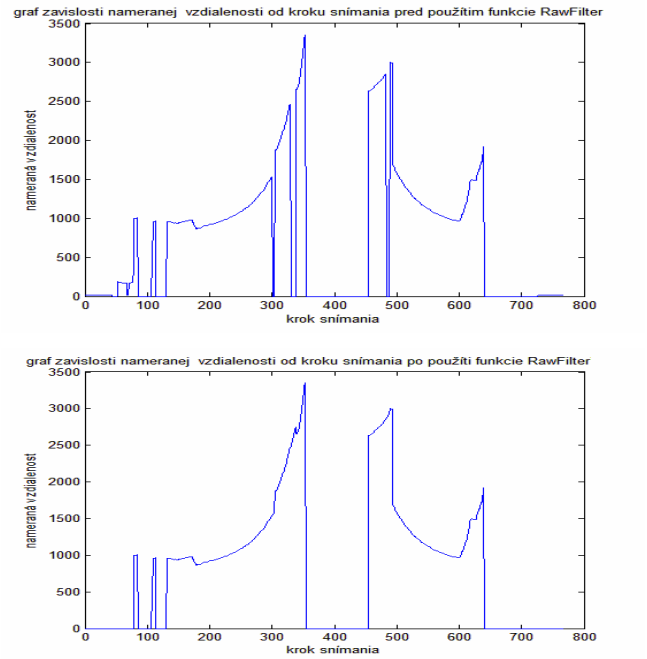
$$koeficient = \left(\frac{(805-803)+(803-800)}{2} \right) = \left(\frac{5}{2} \right) \cong 3$$

(po zaokrúhlení)

Pole data po aproximácii bude vyzerať takto:

data = [800, 803, 805, 808, 811, 814, 815, 819, 823…]

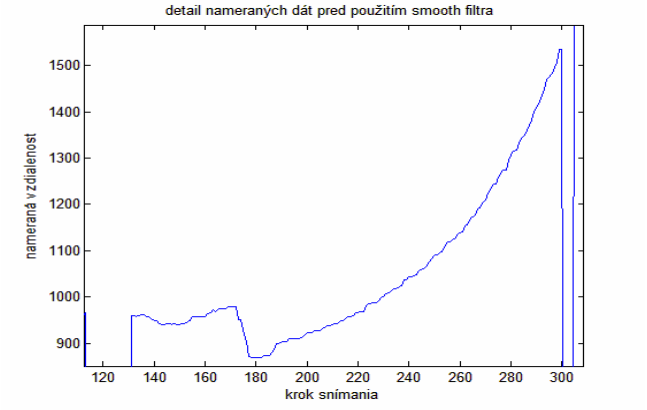
Výstupom funkcie je modifikované pole vzdialeností zbavené základných chýb merania. Ukážky reálnych grafov pred a po použití filtra sú uvedené na Obr. 5.

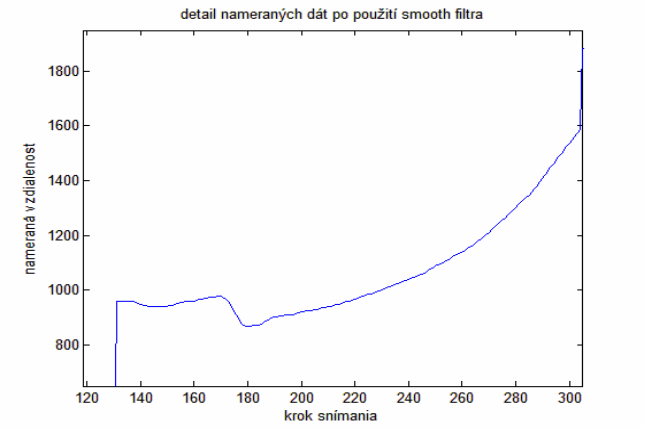


Obr.5 Ukážka nameraných dát pred a po použití filtra RawFilter

Fig.5 Example of measured data before and after using RawFilter

Filter `DataSmoother` vykonáva drobnú korekciu priebehu nameraných dát (Obr. 6). V podstate vyhladzuje krivku priebehu nameraných dát. Jeho základom je použitie funkcie smooth integrovanej v Matlabe. Výstupom funkcie je opäť modifikované pole vzdialeností.



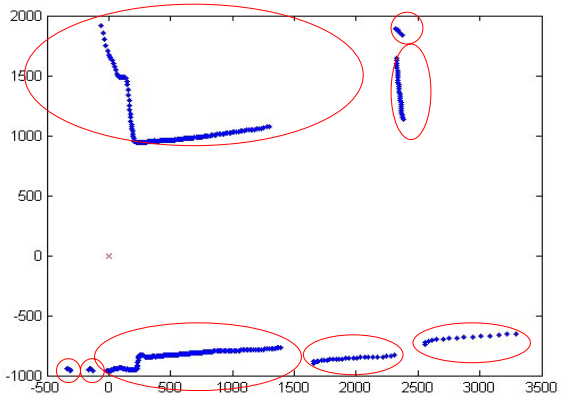


Obr.6 Vyhľadenie časti dát použitím DataSmoother filtra

Fig.6 Smoothing of data part using DataSmoother filter

Segmentácia dát

Vyfiltrované údaje nameraných vzdialeností sa v ďalšom kroku rozdelia na tzv. segmenty. Segmentácia slúži na zjednodušenie práce, keďže algoritmus potom nepracuje s celým poľom údajov, ale len s jednotlivými segmentmi. Segment predstavuje akúkoľvek spojitú sekvenciu dát rôznych od nuly, ktoré spolu logicky súvisia (predstavujú 1 objekt). Ak však v tejto sekvencii dát dôjde k skoku v hodnote vzdialenosti väčšej ako je premenná *segmentDistance*, už hovoríme o novom dátovom segmente. Na obrázku Obr. 7 je naznačené vytvorenie jednotlivých segmentov.



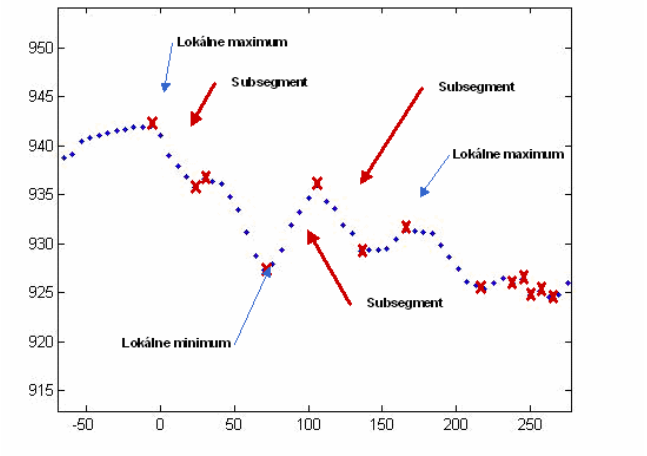
Obr.7 Segmentácia dát v mape prostredia

Fig.7 Segementation of data in environment map

Každý segment je jasne definovaný svojím začiatočným a konečným indexom. Segmenty s malou dĺžkou boli zanedbané, pretože nepredstavujú relevantnú informáciu.

Detekcia značiek prostredia

Pri analýze nameraných dát bolo zistené, že väčšinu významných značiek prostredia (reálnych extrémov) možno detekovať určením lokálnych maxím a miním jednotlivých segmentov. Na určenie týchto extrémov sme použili funkciu `Extr` [9], ktorá vracia indexy lokálnych extrémov v sekvencii dát, v našom prípade teda v danom segmente. S pomocou týchto extrémov boli segmenty rozdelené na ešte menšie segmenty (subsegmenty) - obsahujúce všetky lokálne extrémy daného segmentu. Názorné zobrazenie subsegmentov, lokálnych maxím a miním je uvedené na Obr. 8 (priblížená ukážka nameraných dát v mape prostredia).

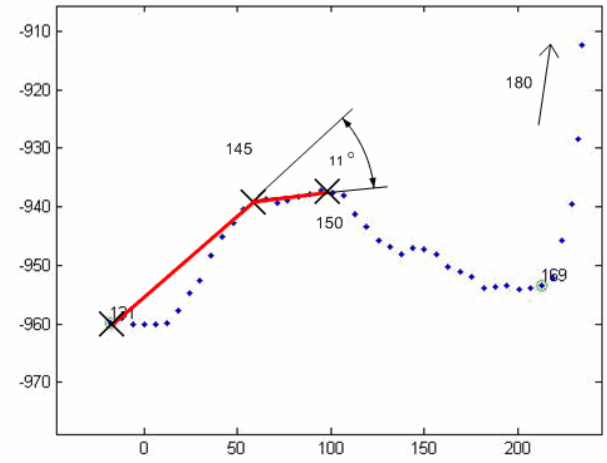


Obr.8 Lokálne minimá a maximá v segmente, zachytené funkciou Extr (priblížené dáta v mape prostredia)

Fig.8 Local minimums and maximums in segment catched by function Extr (zoomed data in environment map)

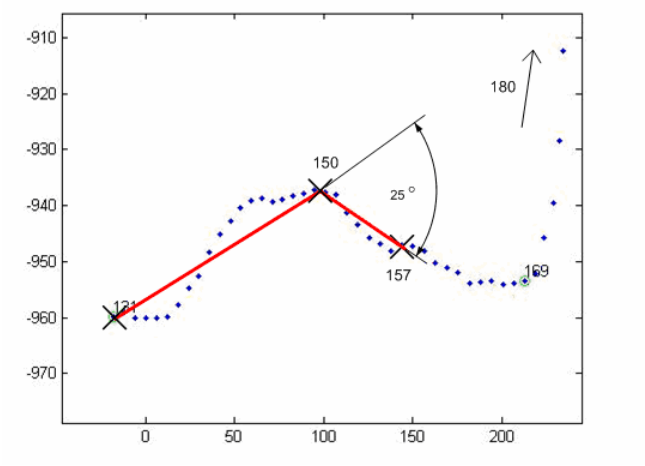
Určením lokálnych miním a maxím segmentov sme získali značné množstvo extrémov. Mnoho z nich je ale nepotrebných, ich informačná hodnota je nízka. Zameriame sa teda na tie, ktoré sa nachádzajú iba na miestach významnejšej zmeny, napr. rohy stien. Na tento účel slúži funkcia `ExtremeProcessing`, ktorá odstraňuje nevýznamné extrémy a významné ponecháva. Odstraňovanie menej významných extrémov vykonávame nasledovne:

Z počiatočných a koncových bodov dvoch po sebe nasledujúcich subsegmentov vytvoríme vektory. Zmeriame ich vzájomný uhol a v prípade, že je väčší ako *minAngle* (40°) a menší ako *maxAngle* (140°) (empiricky zvolené hodnoty) daný bod považujeme za reálny extrém. Ak nie, daný extrém je nevýznamný a ďalej sa s ním už nepočíta. Počiatočný bod prvého vektora prvého subsegmentu začína buď začiatočným bodom segmentu (v prípade vektorizácie prvých dvoch subsegmentov), alebo posledným platným reálnym extrémom. Odstraňovanie nevýznamných extrémov popisujú obrázky 9, 10 a ponechanie významného obrázok 11.



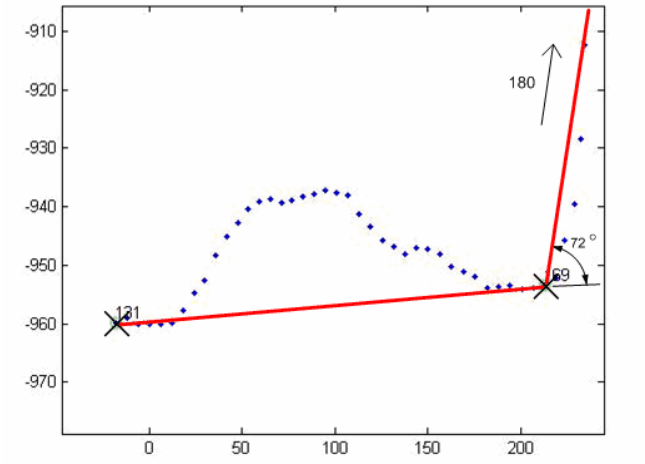
Obr.9 Nevýznamný extrém - bod 145 (priblížené dáta v mape prostredia)

Fig.9 Nonsignificant extreme – point 145 (zoomed data in environment map)



Obr.10 Nevýznamný extrém - bod 150 (priblížené dáta v mape prostredia)

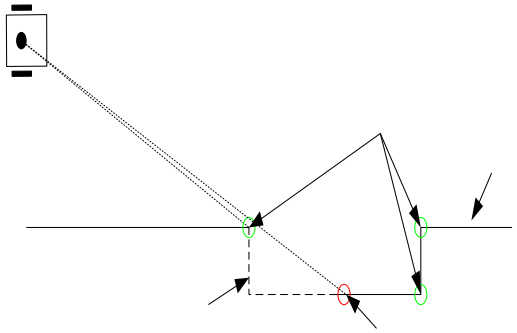
Fig.10 Nonsignificant extreme – point 150 (zoomed data in environment map)



Obr.11 Významný extrém - bod 169 (priblížené dáta v mape prostredia)

Fig.11 Significant extreme – point 169 (zoomed data in environment map)

Kategorizácia extrémov spočíva v rozdelení extrémov do skupiny reálnych, a nami definovaných, tzv. pomocných extrémov. Za reálne extrémy považujeme význačné vlastnosti prostredia. Týmito vlastnosťami môžu byť rohy stien ktoré sú detegované dvoma spôsobmi. Jedným z nich je detekcia na základe výpočtu uhlu medzi dvoma susednými segmentmi, druhým je využitie výskytu nami definovaných, tzv. pomocných extrémov. Pomocné extrémy definujeme nasledovne: ak medzi dvoma po sebe nasledujúcimi krokmi snímania vznikne skok v hodnote vzdialeností poľa *distance* väčší ako premenná *segmentDistance*, potom bod s nižšou hodnotou nameranej vzdialenosti (t.j. ktorý je k robotu bližšie) označíme ako reálny extrém a druhý bod ako pomocný extrém. Pomocné extrémy naznačujú, že prekážka na danom mieste pravdepodobne pokračuje, a preto ho nemôžeme považovať za reálny extrém. S istotou však vieme povedať, že údaj s nižšou hodnotou predstavuje reálny extrém, v našom prípade roh steny. Obr. 12 ilustruje rozdiel medzi reálnymi, a pomocnými extrémami.



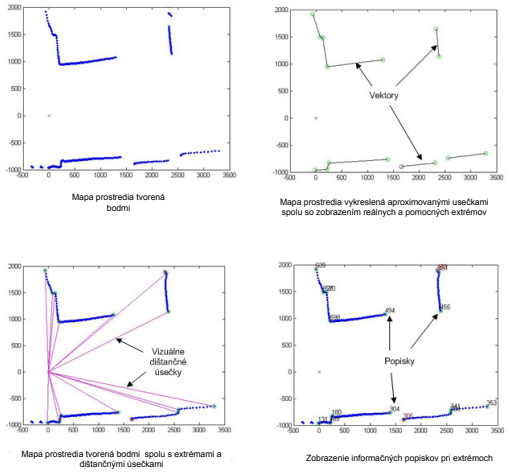
Obr.12 Reálne a pomocné extrémy

Fig.12 Real and helping extremes

Zobrazovanie prostredia (mapy) a jeho značiek

Grafický výstup prostredia a jeho značiek (Obr. 13) zabezpečuje funkcia PlotMap. Umožňuje nám zobraziť resp. skryť:

- mapu tvorenú jednotlivými bodmi poľa *distance*
- reálne extrémy
- pomocné extrémy
- pomocnú vizuálnu dištančnú úsečku (siahajúcu od robota až po extrém)
- mapu aproximovanú úsečkami (vektormi)
- popisy k jednotlivým extrémom (identifikátory bodov)



Obr.13 Ukážka voliteľných možností zobrazenia informácií

Fig.13 Examples of optional information display

Záver

Pri predpoklade pohybu mobilného robota v štruktúrovanom prostredí, dokážeme detekovať značky prostredia z nameraných údajov laserového skenera. Pri určovaní vzťahov mobilný robot – značka, sme predpokladali využitie prirodzených značiek prostredia. Namerané dáta boli spracované filtráciou a segmentáciou, čo zjednodušilo detekciu samotných extrémov a teda významných značiek prostredia. Informáciu o detekovaných prirodzených značkách prostredia potom možno použiť pre iné úlohy mobilnej robotiky, ako je lokalizácia, navigácia alebo korekcia chýb senzorických systémov.

Poďakovanie

Článok vznikol pri riešení projektu VEGA 1/0690/09.

Literatúra

[1] TOMATIS N.: Hybrid, Metric – Topological, Mobile robot navigation, Lausanne, 2001

[2] TALLURI R., AGGARWAL J. K.: Position estimation techniques for an autonomous mobile, Handbook of pattern recognition & computer vision, 1993, Pages: 769 – 801

[3] FUKUI I.: TV Image Processing to Determine the Position of a Robot Vehicle, Pattern Recognition, 1981, Vol. 14, pp. 101-109

[4] BORENSTEIN J.: The Nursing Robot System, Ph. D. Thesis, Technion, Haifa, Israel, June 1987, pp. 146-158

[5] KABUKA M., ARENAS A.: Position Verification of a Mobile Robot Using Standard Pattern, IEEE Journal of Robotics and Automation, 1987, Vol. RA-3, No. 6, pp. 505-516

[6] MAGEE M., AGGARWAL J.: Determining the Position of a Robot Using a Single Calibrated Object, Proceedings of IEEE International Conference on Robotics and Automation, Atlanta, GA, March 13-15 1984, pp. 140-149

[7] KUK-JIN Y., IN-SO K.: Artificial Landmark Tracking Based on the Color Histogram

[8] FENG L., FAINMAN Y., KOREN, Y.: Estimate of Absolute Position of Mobile Systems by Opto-electronic Processor," IEEE Transactions on Man, Machine and Cybernetics, 1992, Vol. 22, No. 5, pp. 954-963

[9] BALDA M.: extr.m, Positions of local extremes in a sequence, 2006

Abstract

In mobile robotics are used distance sensors for extracting environment description, then for creating environment map. For inteligent navigation of mobile robot in environment is advisable to know some notable attributes, which contribute in increasing of information quality about environment. These attributes are entitled environment marks. This paper deals with detection of environment marks by laser scanner.

prof. Ing. Ladislav Jurišica, PhD.,
Ing. František Duchoň

Fakulta elektrotechniky a informatiky
Ústav riadenia a priemyselnej informatiky
Ilkovičova 3
812 19 Bratislava
ladislav.jurisica@stuba.sk
frantisek.duchon@stuba.sk,

Triangulácia polohy mobilného robota s použitím detegovaných prirodzených značiek prostredia

František Duchoň, Ladislav Jurišica

Abstrakt
Cieľom článku je návrh metódy určenia vzťahu posunutie – otočenie medzi dvoma polohami robota (relatívnej polohy robota v prostredí) na základe informácií o okolí robota v týchto polohách získaných pomocou laserového skenera. Predpokladáme, že mapu prostredia ani vzájomné vzťahy medzi vlastnosťami (značkami) prostredia nepoznáme a priori. Na výpočet hodnoty relatívneho posunutia/otočenia medzi dvoma polohami robota je použitá známa metóda triangulácie. Navrhnutá metóda bola overená na reálnom systéme štvorkolesového robota s diferenčným spôsobom riadenia. Vypočítaná hodnota posunutia/otočenia môže byť použitá na korekciu chyby v systéme odometrie mobilného robota.

Kľúčové slová: poloha a pozícia mobilného robota, triangulácia, relatívne posunutie

Úvod

Pri určovaní polohy robota pomocou triangulačných metód môžeme predpokladať dve situácie. V prvej budeme predpokladať, že máme k dispozícii mapu prostredia, t.j. sú známe pozície značiek a vzťahy medzi nimi v prostredí. V tomto prípade ide o výpočet absolútnej polohy metódou 3-bodovej triangulácie [1] , t.j. odhad pozície aj orientácie. V druhej situácii predpokladáme, že mapu prostredia nepoznáme a budeme určovať relatívny posun ε robota medzi dvoma rôznymi polohami v prostredí vzhľadom k značkám prostredia. Táto informácie môže byť následne použitá na korekciu chyby systému odometrie.

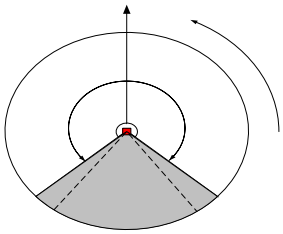
1 Určenie relatívneho posunutia na reálnom zariadení

Na overenie metódy určenia relatívneho posunutia/otočenia mobilného robota medzi dvoma polohami boli namerané dáta z laserového skenera pre tieto dve situácie:

a) posun vpred o 17 cm

b) posun vpred o 28 cm a následné otočenie o 10°

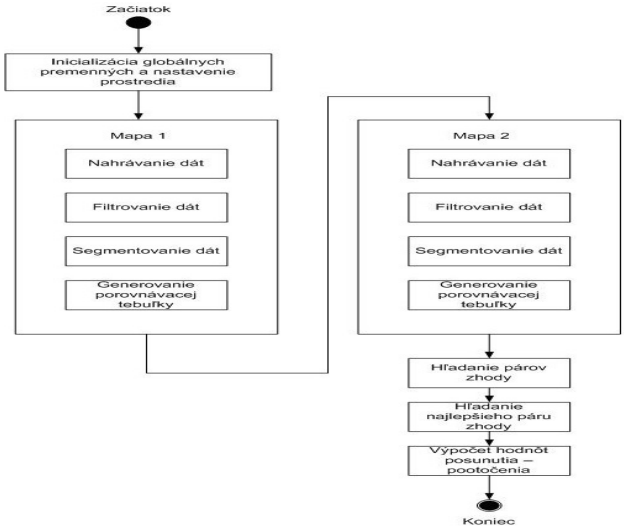
Krok snímania predstavuje pri nami použitom skeneri hodnotu približne 0.36º. Výsledný súbor nameraných dát má teda podobu 768x2 matice, kde prvý stĺpec predstavuje krok snímania a druhý nameranú vzdialenosť. Riadky predstavujú jednotlivé namerané hodnoty vzdialeností pre konkrétne kroky snímania. Skener je schopný merania v rozsahu 240º. Vzájomný vzťah medzi krokom snímania a uhlom vidieť na Obr. 1. Krok 384 predstavuje uhol 0º. Kroky snímania 44 a 726 predstavujú uhly -120 ° a 120º. Skener meria vzdialenosti od 20mm po 4094mm. Vzdialenosti menšie ako 20mm reprezentujú interný chybový kód skenera.



Obr.1 Spôsob uhlovej reprezentácie dát skutočného laserového skenera

Fig.1 Angle representation of data from real laser scanner

Schému celkového algoritmu na určenie relatívnej polohy mobilného robota s využitím detekcie značiek možno vidieť na Obr. 2.



Obr.2 Schéma algoritmu

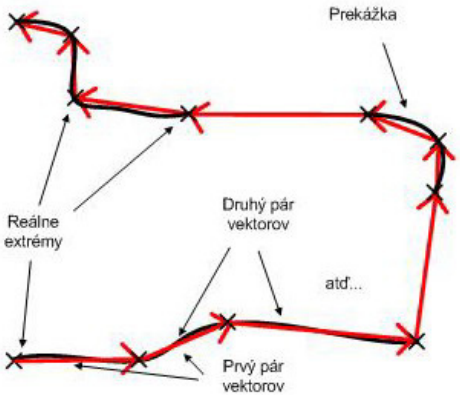
Fig.2 Algorithm scheme

Pri predpokladanom správnom detegovaní značiek prostredia treba medzi dvoma po sebe nasledujúcimi mapami určiť aj zhodu medzi týmito dvoma mapami. Pre určenie zhody medzi dvoma mapami sme využili vzťahy medzi značkami prostredia, ktoré sú nezávislé od polohy a orientácie robota. Tieto vzájomné vzťahy medzi detegovanými značkami predstavujú ich spoločné vlastnosti ako sú vzdialenosť a uhol pod akým sa značky navzájom "vidia".

Najprv sme skúsili metódu porovnania vzájomných vzťahov značiek princípom vytvorenia všetkých možných kombinácii dvojíc vektorov pozostávajúcich z bodov reálnych extrémov. Metóda síce vracala korektné zhody značiek, ale iba v prípade malého počtu reálnych extrémov. Pri mapách s veľkým počtom detegovaných reálnych extrémov už dochádzalo k veľkému počtu kombinácií, čo malo za následok veľkú výpočtovú náročnosť, a tým aj dlhý čas potrebný na výpočet. Preto sme museli nájsť nejaký iný, vhodný spôsob nájdenia zhody medzi dvoma mapami.

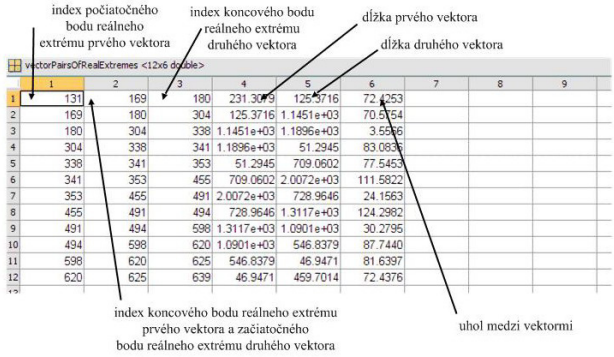
Nový spôsob získania zhody spočíva vo vytvorení poľa dvojíc po sebe nasledujúcich vektorov pozostávajúcich z bodov reálnych extrémov (Obr. 3). Takto sme zachovali princíp metódy porovnania na základe využitia vzájomných vzťahov medzi značkami a zároveň sme sa vyhli generovaniu kombinácii typu "každý s každým".

Účelom porovnávacej tabuľky je zhromažďovanie už získaných a generovanie všetkých nových informácií potrebných k získaniu zhody reálnych extrémov v jednotlivých mapách. Porovnávacia tabuľka sa generuje pre každú mapu zvlášť. Porovnávacia tabuľka (Obr. 4) pozostáva z riadkov – dvojíc vektorov, kde každý stĺpec predstavuje atribút neskoršieho porovnávania (popr. indexovania). Prvé tri stĺpce tabuľky predstavujú indexy bodov tvoriacich vektory (1 - počiatočný bod reálneho extrému prvého vektora, 2 - koncový bod reálneho extrému prvého vektora a začiatočný bod reálneho extrému druhého vektora, 3 - koncový bod reálneho extrému druhého vektora), ďalšie dva stĺpce predstavujú dĺžky vektorov (4 - dĺžka prvého vektora, 5 - dĺžka druhého vektora). Posledný stĺpec predstavuje uhol medzi vektormi.



Obr.3 Princíp generovania porovnávacej tabuľky

Fig.3 Principle of generating comparative table

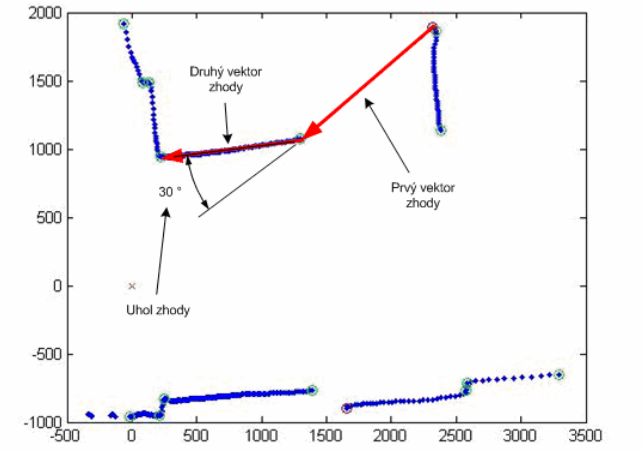


	1	2	3	4	5	6	7	8	9
1	131	169	180	231.3079	125.3716	72.4253			
2	169	180	304	125.3716	1.1451e+03	70.5764			
3	180	304	338	1.1451e+03	1.1896e+03	3.5566			
4	304	338	341	1.1896e+03	51.2945	83.0838			
5	338	341	353	51.2945	709.0602	77.5453			
6	341	353	455	709.0602	2.0072e+03	111.5822			
7	353	455	491	2.0072e+03	728.9646	24.1563			
8	455	491	494	728.9646	1.3117e+03	124.2982			
9	491	494	598	1.3117e+03	1.0901e+03	30.2795			
10	494	598	620	1.0901e+03	546.8379	87.7440			
11	598	620	625	546.8379	46.9471	81.6397			
12	620	625	639	46.9471	459.7014	72.4376			

Obr.4 Ukážka porovnávacej tabuľky

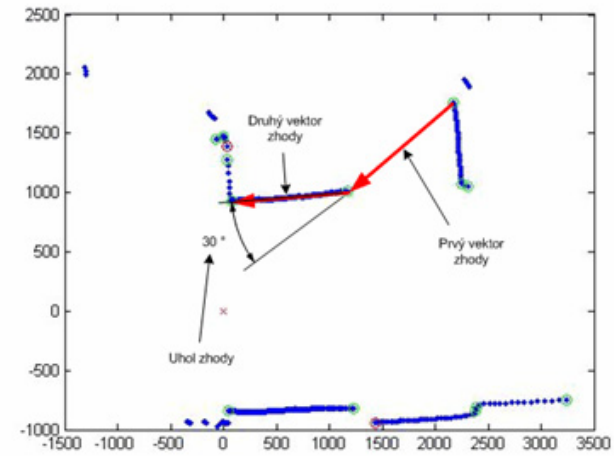
Fig.4 Example of comparative table

Ak sú vygenerované porovnávacie tabuľky pre obe mapy, môžeme začať s porovnávaním ich obsahu a hľadaním následnej zhody. Kombinujú sa záznamy z prvej porovnávacej tabuľky so záznamami z druhej porovnávacej tabuľky. Ak dôjde k zhode v dĺžke prvého vektoru, dĺžke druhého vektoru a v uhle medzi týmito vektormi, jedná sa o zhodu troch reálnych extrémov (Obr. 5 a 6). Výsledný zápis sa nachádza v tabuľke zhody (Obr. 7).



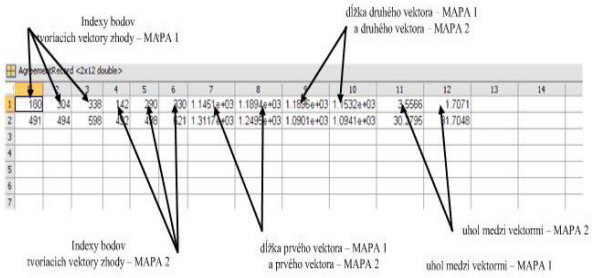
Obr.5 Pár vektorov zhody v mape č.1 (poloha1.dat)

Fig.5 Pair of agreement vectors in map number 1



Obr.6 Pár vektorov zhody v mape č.2 (poloha2.dat)

Fig.6 Pair of agreement vectors in map number 2

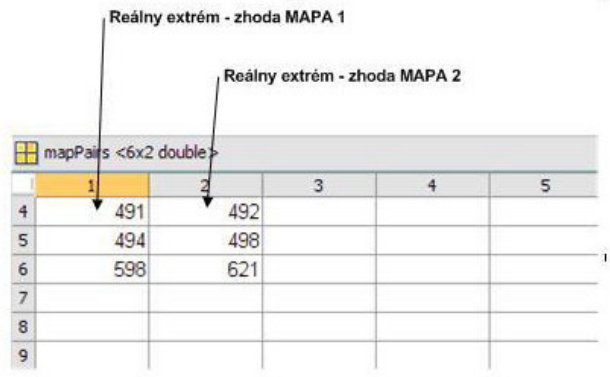


Obr.7 Výsledná tabuľka zhody

Fig.7 Final table of agreement

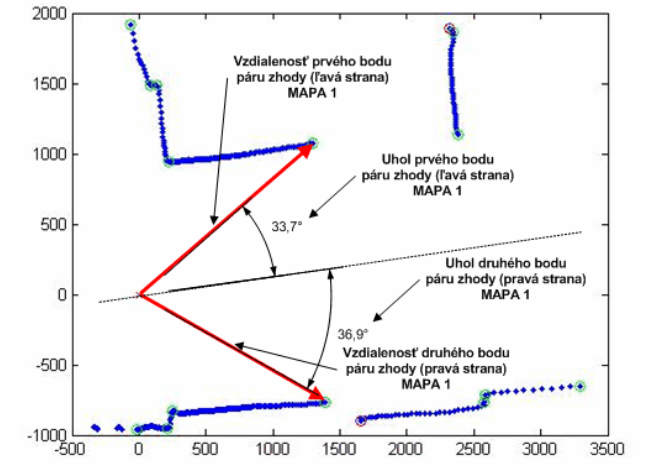
Pri hľadaní párov zhody je možné využiť aj vlastnosť "pravdepodobných" bodov zhody, t.j. bodov, ktoré sa nachádzajú bezprostredne pred alebo za pomocným extrémom, pravdepodobne totiž detegujú hranu objektu. Funkcia hľadania zhody *GetMapPairs* využíva túto vlastnosť s tým rozdielom, že už používa len známe páry zhody (obsiahnuté v poli *MapPairs*). Táto metóda predstavuje modifikáciu vyššie spomenutého princípu kombinácii vektorov "každý s každým" (funkcia *GetOppositeRealExtremeInMap2*).

Ak máme k dispozícií všetky zhody (Obr. 8), našou úlohou je vybrať tú najlepšiu. Najlepšia zhoda poskytuje najpresnejší výsledok. Podmienky sú nasledujúce: Prvý bod sa musí nachádzať po ľavej strane zámerného lúča robota, druhý bod po pravej strane. Najlepšia zhoda je určená najlepším pomerom vzdialeností a uhlov týchto bodov (Obr. 9)



Obr.8 Zhody reálnych extrémov v jednotlivých mapách

Fig.8 Agreement of real extremes in severally maps



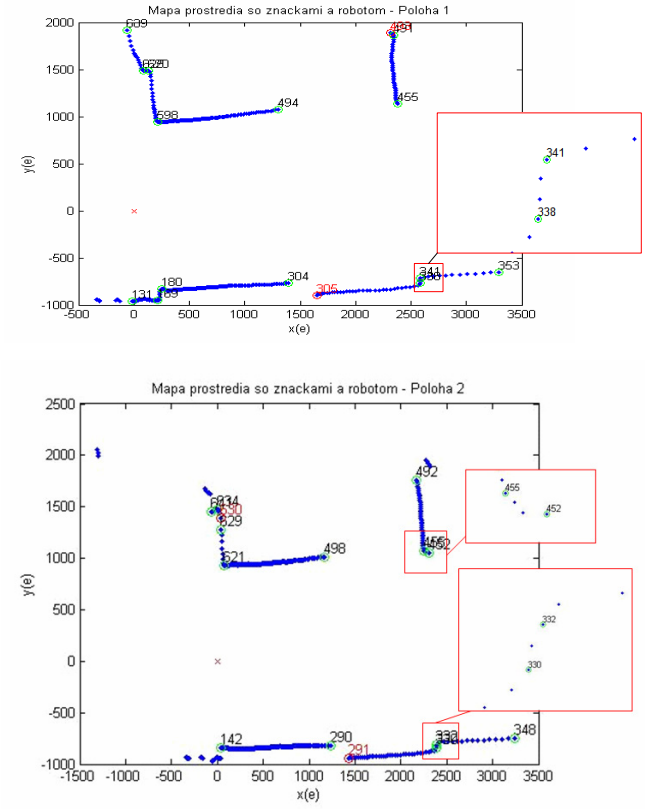
Obr.9 Príklad najlepšej zhody

Fig.9 Example of best agreement

Keď máme k dispozícií najlepšiu zhodu, t.j. indexy značiek zhody, môžeme vypočítať relatívne posunutie, prípadne otočenie robota medzi dvoma polohami robota. Na výpočet sa použije funkcia *GetRobotMovementRelativeChange*, ktorá vezme ako vstupné parametre práve pole najlepšej zhody a hodnoty posunutia/otočenia sú vypočítané metódou opísanou v časti o určení relatívneho posunutia. Predpokladáme že robot sa nedokáže súbežne posúvať aj otáčať, preto ak je vypočítané relatívne posunutie medzi dvoma mapami menšie ako parametrizovateľná hodnota *robotInStopModeDistance* (v našom prípade empiricky nastavená na hodnotu 40mm) je možné vypočítať aj uhol otočenia. Ten sa vypočíta jednoducho odčítaním uhla značky prostredia z polohy 1 o uhol spárovanej značky z polohy 2. Funkcia vráti ako výsledok dĺžku relatívneho posunutia ε medzi dvoma polohami robota, prípadne ak dôjde len k otočeniu robota, funkcia vráti hodnotu natočenia v stupňoch.

2 Výsledky z reálneho zariadenia

Posun vpred o 17cm



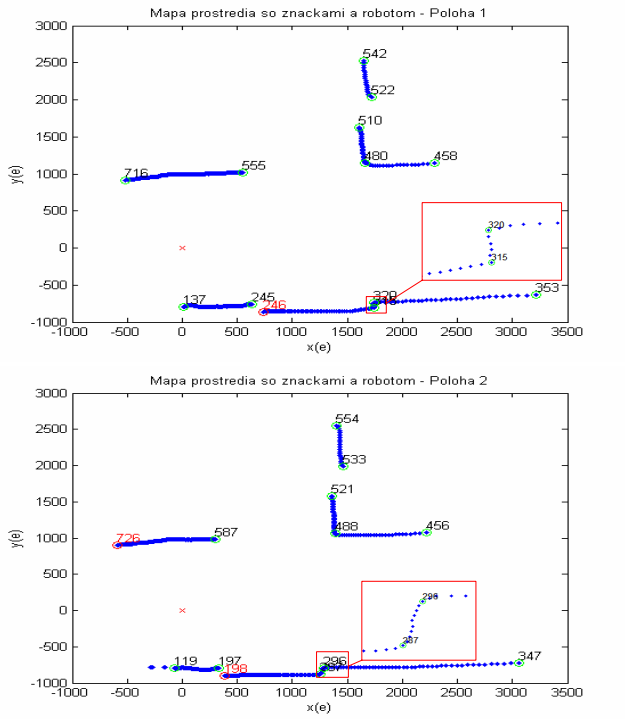
Obr.10 Mapy pre posun vpred o 17cm

Fig.10 Maps for direct movement of 17 cm

Výsledok aplikácie (Obr. 10):

$\varepsilon = 16.31cm$ a $\theta_r = 0°$

Posun vpred o 28cm



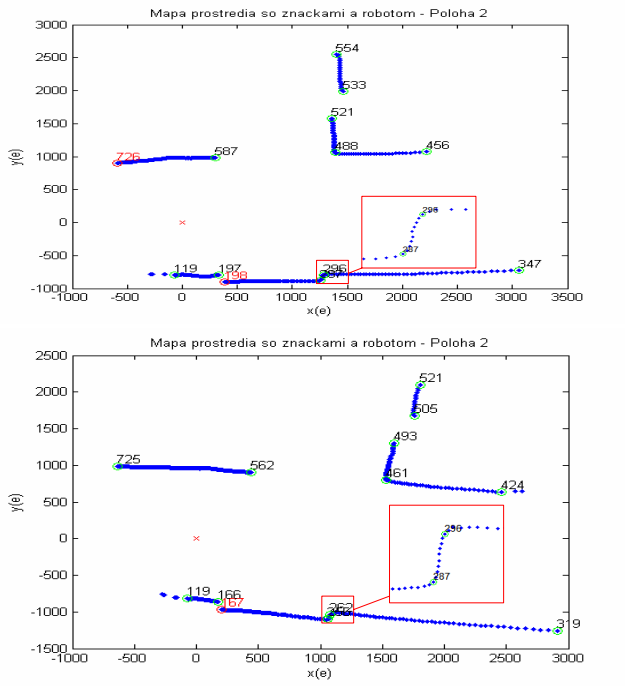
Obr.11 Mapy pre posun vpred o 28cm

Fig.11 Maps for direct movement of 28 cm

Výsledok aplikácie (Obr. 11):

$\varepsilon = 28.3cm$ a $\theta_r = 0°$

Otočenie robota o +10°



Obr.12 Mapy pre otočenie o +10°

Fig.12 Maps for rotation of +10°

Výsledok aplikácie (Obr. 12):

$\varepsilon = 0.185cm$ a $\theta_r = 10.98°$

Záver

Navrhnuté metódy dávajú možnosť určiť relatívnu polohu mobilného robota v prostredí na základe detegovaných umelých značiek alebo prirodzených vlastností objektov. Metódy umožňujú potlačiť chyby, ktoré vznikajú pri odometrii a tým teda zvyšujú kvalitu informácie o polohe robota v prostredí.

Poďakovanie

Článok vznikol pri riešení projektu VEGA 1/0690/09.

Literatúra

[1] BETKE M., GURVITS L.: Mobile Robot Localization Using Landmarks. In: IEEE Transactions on Robotics and Automation, roč. 13, 1997, č. 2, s. 251-26

[2] Matlab Documentation,

http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.shtml

[3] KILPELÄ A.: Pulsed time-of-flight laser range finder techniques for fast, high precision measurement applications, Department of Electrical and Information Engineering, University of Oulu, Finland 2004

[4] KOSKINEN M., KOSTAMOVAARA J., MYLLYLÄ R.: Comparison of the continuous wave and pulsed time-of-flight laser rangefinding techniques, Optics, Illumination and Image Sensing for Machine Vision VI, 1992, Proc. SPIE Vol. 1614, pp. 296-305

[5] TYPIAK A.: Use of laser rangefinder to detecting in surroundings of mobile robot the obsacles, The 25th International Symposium on Automation and Robotics in Construction, June 26-29,2008

Abstract

The aim of this paper is design methods for relation assignation between direct movement and rotation among two positions of robot (relative position in environment) following information about environment in these positions obtained by laser scanner. We assume, that environment map neither relations between attributes (marks) are not known antecedent. For calculation of relative movement (rotation) between two positions of robot is used method of triangulation. Supposed method was verified on real system. Calculated value of movement (rotation) can be used for odometry correction.

prof. Ing. Ladislav Jurišica, PhD.,
Ing. František Duchoň
Fakulta elektrotechniky a informatiky
Ústav riadenia a priemyselnej informatiky
Ilkovičova 3
812 19 Bratislava
ladislav.jurisica@stuba.sk
frantisek.duchon@stuba.sk

Využitie rôznych typov kolesových podvozkov pri lokomócii robotických systémov

Ľubica Miková, Rastislav Baláž, Mária Ádiová

Abstrakt

Široká oblasť priemyslu neustále vyžaduje nové špecifické požiadavky na mobilné robotické systémy. Ruka v ruke s týmito potrebami sa zároveň otvárajú nové otázky pri riešení týchto problémov. Vhodný výber podvozku, teda základnej platformy navrhovaného systému, je jednou z hlavných úloh pri počiatočnom návrhu. Pri počiatočnej analýze návrhu je nutná úplná znalosť všetkých vstupných parametrov a požiadaviek, jednak funkčnosti a taktiež prostredia, v ktorom systém bude pracovať.

Kľúčové slová: mobilný robot, robotické systémy, typy podvozkov

Úvod

Široká oblasť priemyslu neustále vyžaduje nové špecifické požiadavky na mobilné robotické systémy. S týmito potrebami sa zároveň otvárajú nové otázky pri riešení týchto problémov. Vhodný výber podvozku, teda základnej platformy navrhovaného systému, je jednou z hlavných úloh pri počiatočnom návrhu. Pri počiatočnej analýze návrhu je nutná úplná znalosť všetkých vstupných parametrov a požiadaviek, jednak funkčnosti a taktiež prostredia v ktorom systém bude pracovať.

Vývoj servisných robotov prebiehal a prebieha ruka v ruke s rozvojom robotov priemyselných. Oblasť ich aplikácie je široká – jadrový priemysel, zdravotníctvo, stavebníctvo, vojenské a policajné úlohy, likvidácia požiarov, práce v nebezpečnom prostredí – diagnostika, monitoring, údržba, deštrukcia, čistenie, hygiena, transport, manipulácia, identifikácia, vyhľadávanie, zber informácii, pomocné úkony...

V súčasnosti nie sú servisné roboty príliš rozšírené. Najväčšou prekážkou ich širokého rozvoja je neustále značne vysoká cena.

Priekopníkmi rozvoja servisných robotov sú univerzity a rôzne vývojové pracoviská, v ktorých sú hľadané nové cesty a nové prístupy v konštrukcii servisných robotov, nové aplikácie, uplatnenia a optimalizácie.

1. Rozdelenie robotov

Roboty a robotické systémy môžeme rozdeliť podľa viacerých kritérií, ktoré sú predovšetkým závislé na spôsobe a druhu ich použitia.

Jedným z kritérií môžu byt ich schopnosti, t.j. kvalita riadiaceho systému, nosnosť, dosah, presnosť, kinematika a pod.

Na základe týchto vlastností môžeme pri robotických systémoch použiť nasledujúce základné rozdelenie:

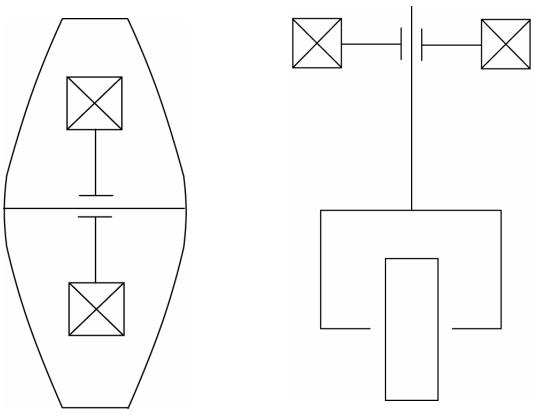
- **Priemyselný robot** – autonómne fungujúci stroj, ktorý je určený k reprodukcii niektorých pohybových a duševných funkcií človeka pri vykonávaní pomocných a základných výrobných operácií bez bezprostrednej účasti človeka a ktorý je k tomuto účelu vybavený niektorými jeho schopnosťami (sluchom, zrakom, hmatom, pamäťou a podobne), schopnosťou samovýuky, samoorganizácie a adaptácie, t.j. prispôsobeniu k danému prostrediu.[1]
- **Servisný robot** - voľne programovateľné kinematické zariadenie, ktoré vykonáva služby čiastočne alebo úplne automaticky. V súčasnosti nie sú servisné roboty príliš rozšírené. Najväčšou prekážkou ich širokého rozvoja je neustále značne vysoká cena. Do budúcnosti sa predpokladá mohutný rozvoj servisných robotov, ktorý dosiahne úrovne rovnajúcej sa súčasnému automobilovému priemyslu.[1]

2. Kinematické štruktúry kolesových robotických systémov

2.1. Jednokolesové a dvojkolesové mobilné robotické systémy

Jedno a dvojkolesové roboty sa v súčasnej dobe vyskytujú iba ako prototypy, na ktorých sú skúmané rovnovážne schopnosti robotov.

Existujú dve základné koncepcie jednokolesových robotov líšiacich sa umiesteným gyroskopu (zotrvačníka).



a) b)

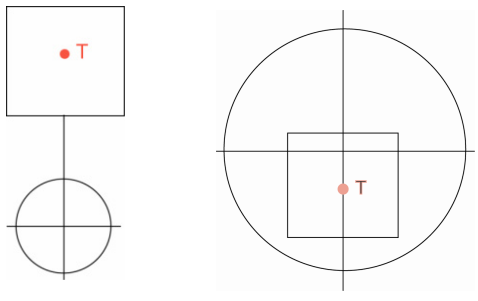
Obr.1 a) zvislý gyroskop [1]
b) Horizontálny gyroskop[1]
Fig.1 a) vertical gyroscope
b) horizontal gyroscope

Robot so zvislým gyroskopom (Obr.1) bol už vyrobený – ide o robot GYROVER a na svete je už niekoľká jeho verzia.

Robot tohto typu je znevýhodnený celkovou obvodovou rotáciou, čo sťažuje integráciu senzorického subsystému do systému robota. Jeho možnou aplikáciou je vojenská aplikácia – nosič zbraňových systémov, prieskum a podobne.

Robot druhého typu s vodorovným gyroskopom (Obr.2) existuje zatiaľ iba ako počítačová simulácia. Je osadený dvoma protichodnými zotrvačníkmi, ktoré zaisťujú jeho vzpriamenú polohu. Výhodou tohto typu umiestnenia zotrvačníka je existencia relatívne statických častí robota, ktoré je možné osadiť vhodným senzorickým subsystémom, manipulátorom či úložnou plochou.

Pri týchto robotoch existujú dve základné koncepcie, ovplyvnené polohou ťažiska robota vzhľadom k osi rotácii kolies.



a) b)

Obr.2 a) ťažisko nad osou rotácie [1]
b) ťažisko pod osou rotácie [1]
Fig.2 a) centre of gravity over a rotation axis
b) centre of gravity under a rotation axis

Prvá varianta sa vyznačuje rovnakými problémami stability ako predchádzajúci jednokolesový robot. Oproti tomu varianta robota s ťažiskom pod osou rotácie kolies je vždy stabilná a nikdy sa nemôže prevrátiť.[1]

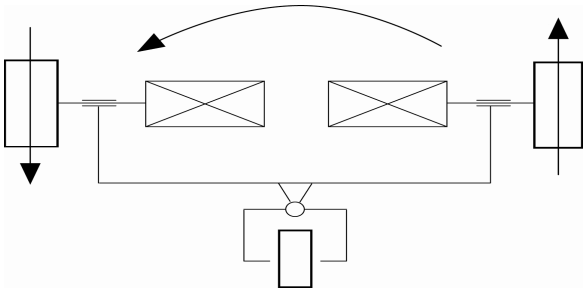
2.2. Tri a štvorkolesové mobilné robotické systémy

Troj- a štvor-kolesové roboty sú najrozšírenejšie roboty vôbec.

Trojkolesový podvozok je oproti štvorkolesovému ľahší, konštrukčne jednoduchší a má aj jednoduchšie riadenie. Nevýhodou trojkolesového podvozku spočíva v jeho stabilite.

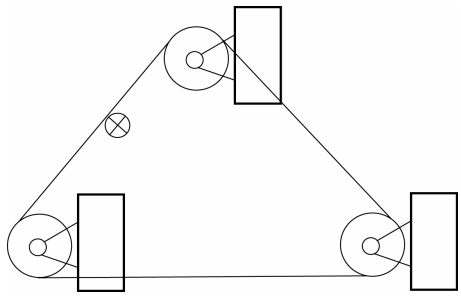
Riadenie 3-4 kolesových podvozkov je možné vykonať v praxi niekoľkými spôsobmi:

Diferenčné riadenie - podvozok robota je osadený dvoma nezávisle poháňanými kolesami a zmena orientácie robota (smeru jazdy) sa vykonáva pomocou rozdielnych otáčok ľavého a pravého kolesa. Zadné koleso je a samovoľne sa natáča do smeru pohybu robota. [1]



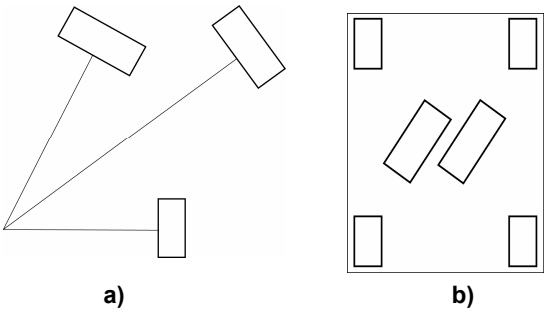
Obr.3 Diferenčné riadenie robota [1]
Fig.3 Robot with differencial control

Synchrónne riadenie - všetky kolesá robota sú poháňané. Ich natočenie do smeru pohybu robota je vykonané pomocou jedného motora, ktorý pomocou istého prevodu (ozubený remeň, reťaz) natočí rovnakým uhlom všetky kolesá. [1]



Obr.4 Synchrónne riadenie robota [1]
Fig.4 Robot with synchronous control [1]

Ackermanov spôsob riadenia – tento spôsob riadenia používajú napr. automobily. Robot má dve závislé otočné kolesá (závislosť je spôsobená lichobežníkovým záprahom medzi nimi). Krútiaci moment je na kolesá privádzaný z jedného pohonu a rozdeľovaný pomocou diferenciálu (súčet obvodových rýchlostí kolies je konštantný). Natočenie kolies je vykonané tak, aby sa ich osi rotácie pretínali v strede otáčania robota (stred zakrivenia trajektórie). [1]



Obr.5 a) Ackermanov spôsob riadenie robota [1]
b) podvozok s viacerými stupňami voľnosti [1]
Fig.5 a) robot with Ackerman control
b) undercarriage with various degree of freedom

Podvozok s viacerými stupňami voľnosti - roboty tohto typu majú v strede dve riadené a poháňané kolesá. V prednej a zadnej časti sú dve a dve oporné kolesá. Robot môže hnané kolesá točiť ktorýmkoľvek smerom a týmto smerom sa potom pohybuje robot. Výhodou tohto robota je vysoká manévrovateľnosť a veľká únosnosť, stabilita je zabezpečená opornými kolesami. [1]

Najzákladnejšie štvorkolesové vozidlo zvyčajne nepoužíva diferenciál. Na každej strane má dve kolesá, ktoré sú spolu spojené a je riadené presne ako trojkolky riadené diferenciálom. Keďže kolesá sú v rade na každej strane a neotáčajú sa pri riadení v zákrute, pri otočení vozidla sa šmýkajú. Takýto efekt šmýkania dáva uvedenej metóde riadenia názov —Skid Steer (šmykové riadenie). Toto usporiadanie nepoužíva diferenciály, aj keď sa nazýva aj diferenciálové riadenie. Vozidlá so šmykovým riadením majú robustný, jednoduchý dizajn s dobrou pohyblivosťou, napriek neefektívnosti šmýkajúcich sa kolies. Keďže kolesá sa nekrútia, je ľahké ich pripojiť k podvozku, pričom nezaberajú priestor potrebný na krútenie [2].

2.3. Šesť a viackolesové mobilné robotické systémy

Šesťkolesové roboty sú väčšinou nasadzované do vonkajšieho prostredia, kde pri pohone všetkých kolies, vhodnej voľbe kolies (priemer, pneumatiky) môžu prekonávať značné nerovnosti povrchu. Môžu sa pohybovať po štrkovitom povrchu s kameňmi, prekonávať obrubníky a menšie priekopy, potoky a rozbahnený terén. Šesťkolesové outdoor roboty sa taktiež používajú k prieskumu cudzích planét.

Obr.6 Robot pracujúci vo vonkajšom prostredí [3]

Fig.6 Robot working in outdoor environment

Vo vnútornom (indoor) prostredí sa šesťkolesové roboty používajú v aplikáciach vyžadujúcich dobrú stabilitu – prevoz nebezpečných látok a zariadení, transport pacientov, preprava vzoriek a podobne. Tieto roboty sú schopné prekonávať prahy a schody. V praxi sa najčastejšie používajú v chemickom a jadrovom priemysle ako diagnostické a servisné zariadenia.

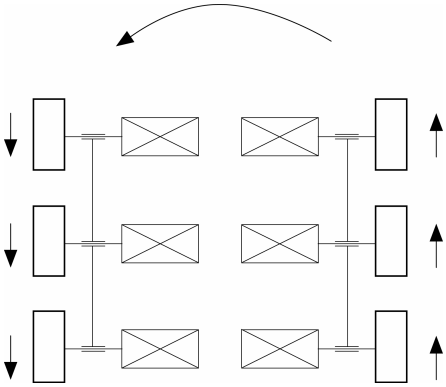
Obr.7 Robot pracujúci vo vnútornom prostredí [4]

Fig.7 Robot working in indoor environment

Pre riadenie šesťkolesového podvozku sa používajú metódy odvodené z riadenia robotov 3 a 4 kolesových – riadenie šmykom (obdoba diferenčného riadenia) a ackermanov spôsob riadenia pre viackolesové roboty.

Riadenie šmykom

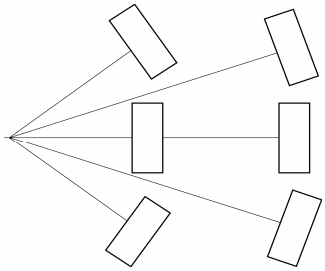
Obdoba diferenčného riadenia, rovnaký princíp, ktorý sa používa pri riadení pásových podvozkov. Nevýhodou tohto spôsobu je vysoké opotrebenie kolies pri zmene smeru jazdy. Výhodou je schopnosť otočiť sa na mieste o 360°.



Obr.8 Riadenie šmykom [1]

Fig.8 Ackerman control for multi-wheels robots

Veľmi sofistikovaný spôsob riadenia šesťkolesového podvozku. Dobrá manevrovateľnosť, nedochádza k opotrebeniu kolies. Nevýhodou je nárast ceny o pohony natáčania kolies.



Obr.9 Ackermanov spôsob riadenie pre viackolesové roboty [1]

Fig.9 Ackerman's control for multi-wheels robots

Záver

Zďaleka najbežnejšou formou usporiadania vozidla je štvorkolesové vozidlo s riadením predných kolies. Je nasledovníkom voza ťahaného koňmi ale počas mnohých desaťročí, od pridania motora, ktorý nahradil kone, prešlo niektorými drobnými a niektorými veľkými zmenami. Najvýznamnejšie zmeny (okrem spaľovacieho motora) boli v systémoch odpruženia a riadenia.
Väčšina odpružení je určená pre vysokorýchlostné riadenie prevažne na hladkých povrchoch, ale významnejšie sú určené na ľudské ovládanie. Napriek svojej popularite a niekedy naozaj fantastickému výkonu v pretekárskych autách a terénnych automobiloch, existuje veľmi málo pružinových systémov párovania. Výnimkou sú odpružené podvozky v niektorých z usporiadaní pásových vozidiel a odpružené štvrté koleso v niekoľkých štvorkolesových dizajnoch.

Poďakovanie

Autori týmto ďakujú Slovenskej grantovej agentúre pre vedu GU VEGA 1/0201/08 „Výskum štruktúr a správania sa modulov mechatronickej mobilnej technickej sústavy na úrovni orgánov a stavebných prvkov za účelom zlepšenia vlastnosti mobilnej technickej sústavy" a GU VEGA 1/0464/09 „Výskum mechatronických sústav imitujúcich lokomóciu hada v obmedzenom a premenlivom priestore.".

Literatúra

[1] VYSOKÁ ŠKOLA BÁŇSKÁ [online] 2004-2009 [cit. 2008-10-7] Dostupné na internete:< http://robot.vsb.cz/cojerobot/servisni.htm >

[2] Sandin, E. Paul: Robot mechanisms and mechanical devices, 2003, ISBN 0-07-142928-X

[3] NASA [online] 2009 [cit. 2009-10-3] Dostupné na internete:<http://photojournal.jpl.nasa.gov/gallery/snt?order=Xdim*Ydim*Zdim&sort=DESC&start=20 >

[3] NASA [online] 2008 [cit. 2009-10-3] Dostupné na internete:<http://photojournal.jpl.nasa.gov/gallery/snt?order=Xdim*Ydim*Zdim&sort=DESC&start=20 >

[4] PC REVUE [online] 2009 [cit. 2009-10-3] Dostupné na internete:< http://www.pcrevue.sk/buxus_dev/generate_page.php?page_id=40505 >

Abstract
The large area of Industry requeres new specific demads for mobil robotics systems. New questions with this demands are opening at this problem. Right choices undercarriage, therefore basic platform of designed system is one of general tasks at early design. At early analysis of design is necessary a complete knowladge all input parameters and deamands of funktion along with enviroment in which the system will work.

Ľubica Miková, Ing.

Technická univerzita v Košiciach
Strojnícka fakulta, Ústav špeciálnych technických vied
Katedra aplikovanej mechaniky a mechatroniky
Letná 9
042 00 Košice
00421 55 602 584
E-mail: lubica.mikova@tuke.sk

Rastislav Baláž, Ing.

Technická univerzita v Košiciach
Strojnícka fakulta, Ústav špeciálnych technických vied
Katedra aplikovanej mechaniky a mechatroniky
Letná 9
042 00 Košice
00421 55 602 2719
E-mail: rastislav.balaz@tuke.sk

Mária Ádiová, Ing.

Technická univerzita v Košiciach
Strojnícka fakulta, Ústav špeciálnych technických vied
Katedra aplikovanej mechaniky a mechatroniky
Letná 9
042 00 Košice
00421 55 602 2719
E-mail: maria.adiova@tuke.sk

Lokomócia robotov s plazivým pohybom

Mária Ádiová, Ľubica Miková, Rastislav Baláž

Abstrakt
Článok sa zaoberá robotmi s lokomóciou napodobňujúcou pohyb hada. Had sa pohybuje rôznymi spôsobmi, ktoré sú závislé od prostredia, v ktorom sa nachádza. Tieto pohyby sú charakterizované v tomto článku.

Kľúčové slová: robot, pohyb, had

Úvod

Človek sa od prvopočiatku snažil vyhnúť nebezpečnej prací a tak hľadal vhodné prostriedky aby nebezpečnú prácu nemusel vykonávať. Jediným takýmto riešením je nahradenie činnosti človeka strojom – robotom. Táto myšlienka je v dnešnej dobe čoraz aktuálnejšia a zasahuje už do všetkých častí každého odvetvia.
Jedným takýmto odvetvím v robotike sú servisné roboty. Oblasť využitia týchto robotov nachádzame v nebezpečnom prostredí, kde je nebezpečné riskovať ľudský život. Za nebezpečné prostredie môžeme považovať – jadrový priemysel, vojenské, policajné a záchranárske úlohy, manipulácia s nebezpečným odpadom a mnoho ďalších ... Veľa krát pri návrhu týchto mobilných robotov sa človek inšpiruje prírodou. Jednou takouto inšpiráciou je využitie lokomócie hada. Robot s takouto " schopnosťou " pohybu nachádza využitie všade tam, kde sa robot s klasickou koncepciou, či samotný človek nemôže vôbec dostať. Je to prostredie stiesňujúce, často krát s obmedzeným priestorom pre pohyb. Tieto roboty – hadovité roboty predovšetkým nachádzajú svoje uplatnenie pri záchranárskych činnostiach, ale veľa krát nachádzajú svoje uplatnenie aj pri inšpekcií potrubí.

Subsystém mobility hadovitých robotov

Mobilné hadovité roboty majú uplatnenie v celých radách servisných činnosti. Pre roboty tohto typu je charakteristicky plazivý pohyb a modely pre ich konštrukciu je telo hada. Môžu sa pohybovať dopredu alebo do strán v závislosti na ich konštrukcií a účelu použitia. Väčšinou sa využívajú k inšpekcií potrubí, vzduchotechnických zariadeniach, v chodbách s veľmi nízkou svetlosťou, vyhľadávanie obetí v troskách budov a pod. [1]

Výhody hadovitých robotov [2]

- **stabilita** - ak sa kolesový robot začne šmýkať z útesu, môže sa prevrhnúť. U kráčajúcich robotov, ak dôjde k porušeniu ťažiska, tiež sa prevrhnú. Na rozdiel od nich, je stabilita hadovitých robotov vyššia. Ak predsa dôjde k pádu hadovitého robota je väčšia šanca, že robot nestratí možnosť pohybu a stabilitu a bude sa hýbať ďalej v obmedzenej miere.
- **odolnosť v teréne** - odolnosť v teréne je schopnosť prechádzať hrboľatý terén. Hadovitý robot dokáže prejsť aj cez prekážky, ktoré siahajú do výšky, ktorá je rovná polovici jeho dĺžky.
- **trakcia** - trakcia je sila, ktorá môže byť vyvinutá pri hadovitom pohybe. Pohon je bežne obmedzený tiažou a koeficientom trenia. Ťažná sila môže byť vysoká u prirodzených hadov. Had sa môže namáhať silou rovnej až tretine svojej vlastnej váhy. Rozdelenie hmotnosti na veľkú plochu, v porovnaní s plochou kráčajúcich robotov alebo kolesových robotov, má za následok menšiu deformáciu pôdy. Prirodzené hady môžu udržiavať ťah do štvor- alebo päť- násobku svojej váhy
- **účinnosť**
- **veľkosť -** veľkosť je závislá na dizajne. Malá čelná plocha hadích robotov dovolí preniknutie do menších prierezových plôch než ekvivalentné kráčajúce alebo kolesové prostriedky.
- Pri valcovitej forme hada, na udržanie rovnakého objemu, ak je priemer znížený o polovicu, dĺžka je štyrikrát väčšia.
- Mechanizmy s malým čelným prierezom s podobným objemom a hmotnosťou môžu mať za následok veľmi dlhé mechanizmy.
- **Redundancia -** Pri hadovitých robotoch aj s poruchou niekoľkých aktivátorov sa udrží čiastočná mobilita. Samozrejme, za cenu zníženej účinnosti a mobility.
- **Tesnosť-** Kontinuálny nedierovaný povrch hadovitého robota nemá žiadny problém byť vystavený rôznym okoliam.[4]

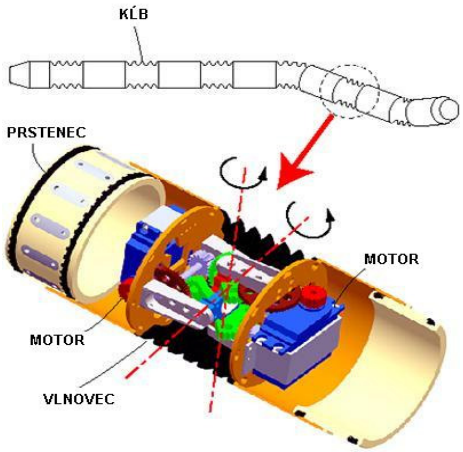
Nevýhody hadovitých robotov [2]

- Dizajn
- Konštrukcia
- Riadenie
- Stupeň voľnosti
- Rýchlosť

Konštrukcia hadovitých robotov

Konštrukcia tela robota sa skladá z množstva článkov vzájomne spojených s kĺbmi s dvoma alebo s troma stupňami voľnosti. Dĺžka robota (počet jeho článkov) závisí na type vykonávanej servisnej úlohy a prostredia, v ktorom sa bude pohybovať. Prostredie bude mať ďalej vplyv i na dĺžku jednotlivých článkov tela robota, napr. pre aplikácie v potrubí musia byť články robota navrhnuté tak, aby robot mohol

v potrubí prekonať všetky jeho zakrivenia. V tomto prípade môžu mať jednotlivé články po obvode kolesá umožňujúci ľahší pohyb v potrubí. Pre aplikáciu v značne členitom alebo skalnatom teréne budú dĺžky článku dlhšie, aby mohol robot zdolávať väčšie prekážky. Pre pohon jednotlivých článkov sa využívajú servomechanizmy alebo jednosmerné motory. Každý článok má svoj kĺb. Vo vnútri článku sú vždy umiestnené pohony pre daný kĺb. Prvý článok predstavuje hlavu robota, na ktorom je prevažné umiestnená kamera. Okrem kamery to môže byť osvetľovacia technika a ďalšie senzory. Na konci tela robotu býva chvost. [1]



Obr.4 Subsystém mobility [3]

Fig.4 Mobility Subsystém

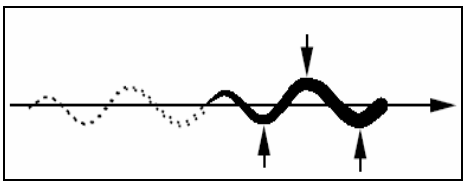
Klasifikácia hadovitých robotov podľa spôsobu pohybu

Hady vykonávajú mnoho druhov pohybov, s ktorými sa prispôsobujú k životnému prostrediu, a môžeme ich rozdeliť do 4 typov:

1. bočný pohyb – je spôsob plazenia používaná hadom, ako je štrkáč, ktorý žije na púšti. Pri svojom pohybe dvíha časť svojho tela počas plazenia. V tomto spôsobe pohybu nie je posuvný pohyb medzi telom a plazením po povrchu. Jeho dynamická charakteristika je, že telo má zvyčajne kontakt zo zemou zhora. Kvôli tejto charakteristike odpor šmykového trenia je malý, aj v pohyboch prostredí, ktoré nie sú pevné, tak ako piesočnatá zem, pohybová efektívnosť je vysoká. [4]

Bočný pohyb vyžaduje minimálne tri kontaktné body pre kontinuálny pohyb. Dve vytvárajú silu a tretí vyrovnáva pohyb v jednom smere.

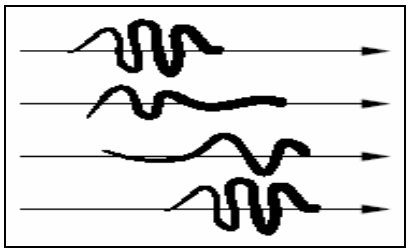
Bočný pohyb sa nehodí pre hladké povrchy (pre nízke trenie) a úzke chodby. Nie je vhodný ani pre kratších a hrubších hadov, lebo u nich sa zmenšuje účinnosť. [2]



Obr.2 Bočný pohyb [2]

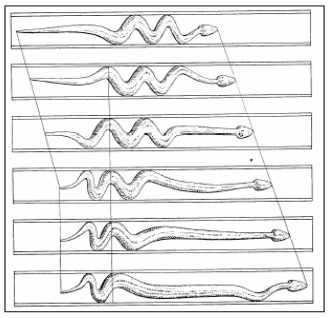
Fig.2 Lateral movement

2. harmonikový pohyb – je spôsob plazenia používaný hadmi obmedzenými v nejakej priamej ceste cez úzku priamku ako aj pri hadoch umiestnených na poschodovom povrchu, ktoré sú mimoriadne šmykľavé. Predovšetkým pri konfigurácií plazenia na poschodovom povrchu sa používa vlastnosť, že index statického trenia je väčší než index dynamického trenia.



Obr.3 Harmonikový pohyb [2]

Fig.3 Concertina movement



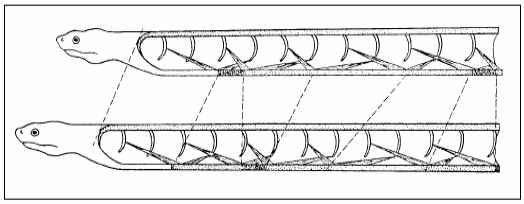
Obr.4 Harmonikový pohyb v obmedzenom priestore [2]

Fig.4 Concertina movement in the constraints space

3. priamočiary pohyb – je spôsob plazenia vykonávaná špeciálnym zoskupením, ako u veľkých hadoch (štrkáč a vretenica), keď sa blíži ich korisť alebo, keď sa plazia nad hladkým povrchom. [4]

Pôvodne sa myslelo, že je spôsobený pohybom rebier, ale v skutočnosti sa na ňom podieľa iba svalstvo pripojené k elastickej koži.

Pri priamočiarom pohybe niekoľko častí tela sa dotýka pôdy a na krok sa využíva symetrické sťahovanie. Tento pohybový mechanizmus je jednoduchý. [2]

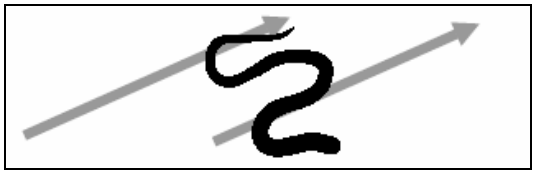


Obr.5 Priamočiary pohyb [2]

Fig.5 Rectilinear movement

4. stranový pohyb – stranový ohyb je použitie kontinuálneho a striedavého kývnutia bočného zakrivenia. Pohyb je využívaný na piesku alebo kyprej pôde. Sú tam len dve dotykové plochy zatiaľ čo sa had pohybuje. Technika minimalizuje sklz a je dokonca viac výkonný než bočný pohyb

Stranový ohyb je používaný hlavne hadmi v neobývaných regiónoch, kde je kyprá pôda. Vývoj stranového ohybu súvisí s vysokými teplotami povrchu a pravdepodobne minimalizuje kontaktné body s povrchom. [2]



Obr.6 Stranový pohyb [2]

Fig.6 Sideways movement

Záver

Článok poukazuje na mobilné servisne roboty s lokomóciou napodobňujúcou pohyb hada. Lokomócia hada je rozdelená do niekoľkých skupín pomocou ktorých sa had môže pohybovať. Každá z týchto skupín je jednotlivo charakterizovaná s ich opisom plazenia sa po určitých povrchoch, v určitých priestoroch a v závislosti na druhu hada.

Poďakovanie

Autori týmto ďakujú Slovenskej grantovej agentúre pre vedu GU VEGA 1/0201/08 „Výskum štruktúr a správania sa modulov mechatronickéj mobilné technické sústavy na úrovni orgánov a stavebných prvkov za účelom zlepšenia vlastnosti mobilnej technickej sústavy“ a GU VEGA 1/0464/09 „Výskum mechatronických sústav imitujúcich lokomóciu hada v obmedzenom a premenlivom priestore.“

Literatúra

[1] KÁRNÍK, Ladislav: Servisní roboti. Ostrava: VŠB – Ostrava, 2004. 144 s. il. ISBN 80-248-0626-6

[2] TÓTH Gabriel: Evolúcia neurokontrolera pre riadenie pohybu hadovitého robota. Diplomová práca. TU FEI KKUI, Košice 2008, vedúci DP (Ing. Marek Bunzel, PhD)

[3] Trossen robotics [online] 2009 [cit. 2009-3-20] Dostupné na internete: <http://www.trossenrobotics.com/images/blogposts/2007/RobotSnakeJointDiagram.jpg>

[4] HIROSE Shigeo: Biologically Inspired Robots – Snake-like locomotors and manipulators: Oxford University Press, 1993. 210 s.

Abstract

The article deals with motion robots to mimic the movement of a snake. Snake it varies in different ways, which are dependent on the environment in which it is located. These movements are characterized in this article.

Mária Ádiová, Ing.

Technická univerzita v Košiciach
Strojnícka fakulta, Ústav špeciálnych technických vied
Katedra aplikovanej mechaniky a mechatroniky
Letná 9
042 00 Košice
00421 55 602 2719
E-mail: maria.adiova@tuke.sk

Ľubica Miková, Ing.

Technická univerzita v Košiciach
Strojnícka fakulta, Ústav špeciálnych technických vied
Katedra aplikovanej mechaniky a mechatroniky
Letná 9
042 00 Košice
00421 55 602 2719
E-mail: lubica.mikova@tuke.sk

Rastislav Baláž, Ing.

Technická univerzita v Košiciach
Strojnícka fakulta, Ústav špeciálnych technických vied
Katedra aplikovanej mechaniky a mechatroniky
Letná 9
042 00 Košice
00421 55 602 2719

E-mail:rastislaz.balaz@tuke.sk

Modelovanie dynamiky robotov v prostredí SimMechanics

Viola Vavrinčíková, Darina Hroncová

Abstrakt
Práca sa zaoberá riešením priamej a inverznej úlohy dynamiky robotov v prostredí Matlab a SimMechanics. Pri riešení priamej úlohy dynamiky ide o vyšetrenie pohybu mechanického systému robota pri pôsobení daných síl a momentov. Riešenie inverznej úlohy dynamiky je náročnejšie, lebo hľadá zodpovedajúce sily, pri ktorých predpísaný pohyb efektora robota nastane. Priama i inverzná úloha sú riešené numericky na základe modelov pomocou blokových schém. Výsledky sú prezentované časovým priebehom kinematických a dynamických funkcií riadenia robotov.

Kľúčové slová: priemyselný robot, počítačová simulácia, Matlab, SimMechanics.

Úvod

Priemyselné roboty predstavujú zložité mechatronické zariadenia, ktoré pozostávajú z viacerých funkčných podsystémov úlohou ktorých je zabezpečenie rôznych typov činnosti robota.

Väčšina priemyselných robotov, ktoré sa v súčasnosti v praxi využívajú sú priemyselné roboty stacionárneho typu. Predstavujú také typy robotov, ktoré sú pevne ukotvené k základu a zmena manipulačného priestoru je u nich možná iba na základe prekonfigurácie kinematickej štruktúry. Splnenie požiadavky pre technicky a ekonomicky efektívne nasadenie robotov je možné hlavne na základe modulárneho prístupu k architektúre robotických zariadení.

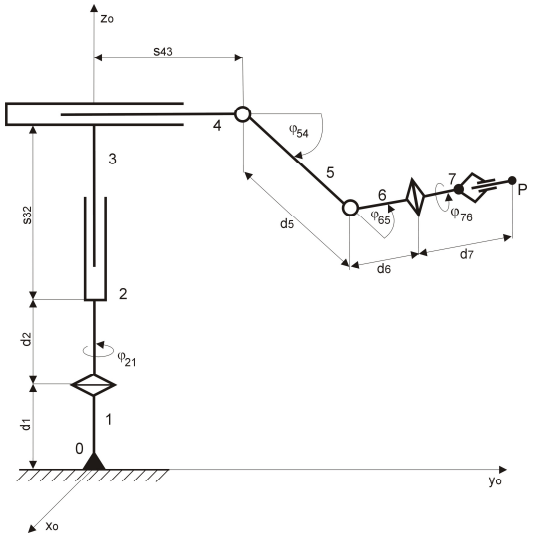
Základom konštrukcie priemyselných robotov sú mechanické systémy s viacerými stupňami voľnosti pohybu, tvorené prevažne otvorenými kinematickými reťazcami. Pri mechanickej koncepcii robotov je potrebné okrem stupňov voľnosti pohybu zohľadniť aj kinematický princíp, ktorý ich zabezpečuje. Pri kinematickom riešení mechanických systémov robotov je implicitne formulovaný aj problém modelovania pracovných pohybov robota pri definovanom pohybe výkonného člena. Pri predpísanej polohe efektora v závislosti na čase je možné určiť kinematické funkcie dráhového riadenia, čo nazývame inverznou úlohou kinematiky robotov. Po určení kinematických funkcií riadenia je možné odvodiť aj dynamické funkcie riadenia pohonov.

Matlab a SimMechanics sú vhodným programovým prostriedkom pri vytváraní počítačových modelov mechanických systémov robotov. Počítačové modelovanie je účinný nástroj, ktorý urýchli a skvalitní i riešenie priamej a inverznej úlohy dynamiky priemyselných robotov.

1. Popis mechanického systému robota

Kinematické reťazce priemyselných robotov sú prevažne otvorené a skladajú sa z dvoch častí. Prvú časť tvorí polohovacie zariadenie, tvorené jednotlivými článkami a posuvnými a rotačnými kinematickými dvojicami.

Druhú časť kinematického reťazca priemyselného robota tvorí zariadenie pre orientáciu, tvorené prevažne rotačnými kinematickými dvojicami s jedným, dvomi alebo tromi stupňami voľnosti pohybu. Mechanický systém otvoreného kinematického reťazca priemyselného robota je ukončený efektorom, teda výkonným členom robota.



Obr. 1 Kinematická schéma priemyselného robota

Fig. 1 Kinematical scheme of industrial robot

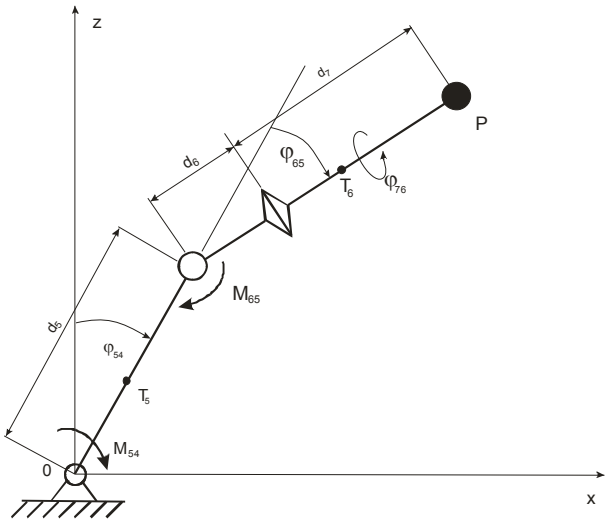
Skúmaný priemyselný robot podľa obr. 1 tvorí otvorený kinematický reťazec so 6° voľnosti pohybu. Prvé tri stupne voľnosti zabezpečujú pohyb polohovacieho zariadenia daného robota. Mechanizmus pre orientáciu má taktiež tri stupne voľnosti pohybu.

V nasledujúcej časti sa zameriame na riešenie priamej a inverznej úlohy dynamiky mechanizmu pre orientáciu v prostredí SimMechanics. Polohu pracovného bodu *P* efektora robota vzhľadom na súradnicový systém mechanizmu pre orientáciu podľa obr. 2 môžeme vyjadriť nasledovne:

$$x_P = d_5 \sin\varphi_{54} + (d_6 + d_7)\sin(\varphi_{54} + \varphi_{65}) \quad (1)$$

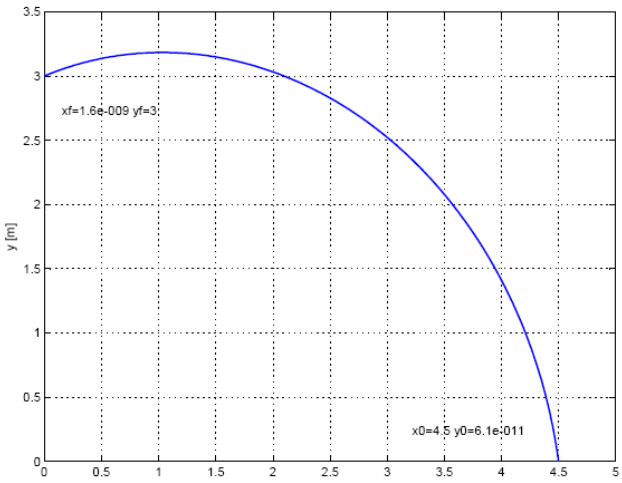
$$y_P = d_5 \cos\varphi_{54} + (d_6 + d_7)\cos(\varphi_{54} + \varphi_{65}) \quad (2)$$

Úlohou je vyšetriť potrebný pohyb v kinematických dvojiciach, ktorý by zabezpečil premiestnenie pracovného bodu *P* efektora robota z polohy P_0 (0;3) do polohy P_1 (4,5;0). Na začiatku a na konci pohybu majú byť rýchlosť a zrýchlenie bodu *P* nulové.



Obr.2 Kinematická schéma mechanizmu pre orientáciu

Fig. 2 Kinematical scheme of mechanism for orientation



Obr. 3 Trajektória koncového bodu efektora

Fig. 3 Trajektory of the endpoint of the effector

Časový priebeh uhlov pootočenia φ_{54} a φ_{65} hľadáme v tvare polynómu piateho stupňa

$$\varphi_{54}(t) = a_1 t^5 + a_2 t^4 + a_3 t^3 + a_4 t^2 + a_5 t + a_6 \quad (3)$$

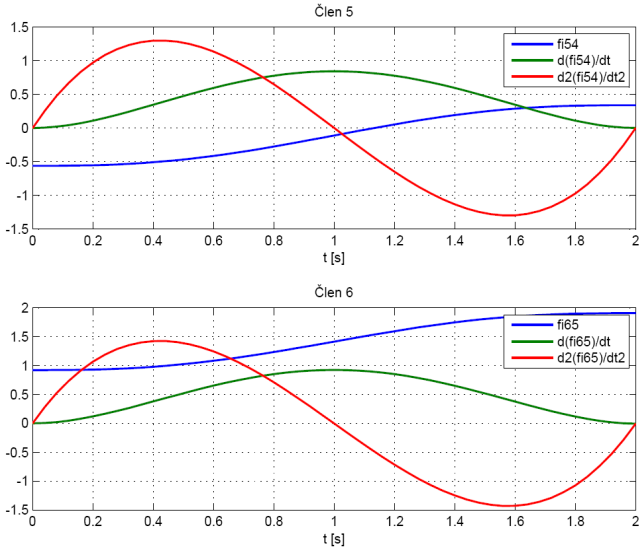
$$\varphi_{65}(t) = b_1 t^5 + b_2 t^4 + b_3 t^3 + b_4 t^2 + b_5 t + b_6 \quad (4)$$

Konštanty v týchto polynómoch sa určia zo začiatočných a koncových podmienok pre pohyb pracovného bodu *P*, pre d_5=2, d_6=1 a d_7=2.

Nakoľko uhol pootočenia φ_{76} nemá vplyv na polohu resp. pohyb pracovného bodu môžeme ho považovať za konštantný a preto nevystupuje ako kinematická funkcia riadenia pohonov.

Trajektória bodu *P* znázornená na obr. 3 odpovedá premiestneniu zo začiatočnej do koncovej polohy.

Priebehy uhlov pootočenia, uhlových rýchlostí a uhlových zrýchlení jednotlivých členov mechanizmu pre orientáciu v závislosti na čase sú uvedené na obr. 4.



Obr. 4 Priebehy kinematických veličín v závislosti na čase

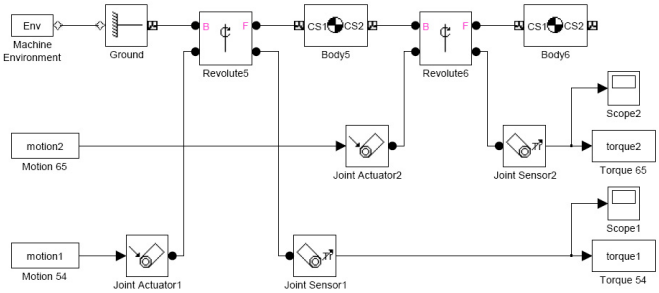
Fig. 4 Time histories of kinematical variable

Uhly pootočenia jednotlivých článkov mechanizmu pre orientáciu vyjadrené ako funkcie času reprezentujú kinematické funkcie riadenia mechanického subsystému robota.

2. Riešenie inverznej úlohy

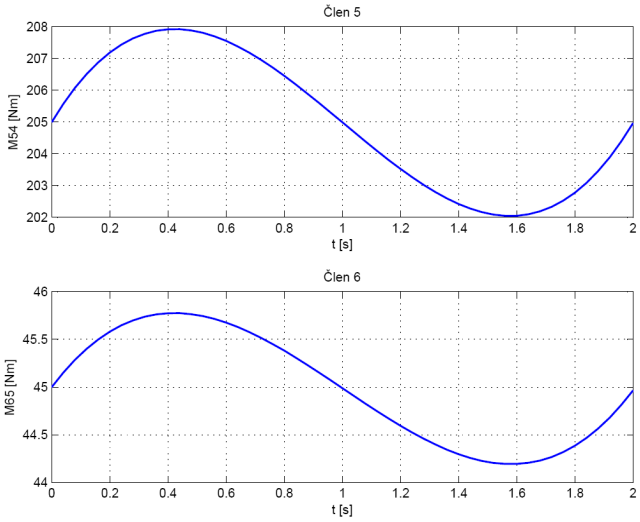
Pri riešení inverzného problému sa určia veľkosti momentov M_{54}, M_{65} potrebných na to, aby sa mechanický systém dostal z jednej polohy do druhej. Vstupmi pre riešenie sú predpísané pohyby jeho jednotlivých členov. Efektor robota sa má z polohy zadanej P_0 premiestniť do požadovanej polohy P_1 po trajektórii určenej v predchádzajúcej kapitole.

Schéma modelu pre riešenie inverzného problému je na obr. 5



Obr. 5 Bloková schéma modelu mechanického systému v SimMechanics (inverzná úloha)

Fig. 5 Block diagram of the mechanical system in SimMechanics (inverse dynamics)



Obr. 6 Priebeh hnacích momentov mechanického systému v závislosti na čase

Fig. 6 Time histories of driving moments of the mechanical system

Priebehy momentov M_{54} a M_{65} v kinematických dvojiciach členov robota v závislosti na čase získané riešením modelu na vyššie uvedenej blokovej schéme sú na obr. 6.

Uvedené momenty predstavujú dynamické funkcie riadenia pohonov mechanizmu pre orientáciu. Hnacie momenty M_{54} , M_{65} sú vyvodené spravidla servopohonmi v kinematických dvojiciach. Pohybové rovnice odvodené pomocou Lagrangeových rovníc 2. druhu môžeme stručne zapísať v maticovom tvare

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} \ddot{\varphi}_{54} \\ \ddot{\varphi}_{65} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \begin{bmatrix} \dot{\varphi}_{54}^2 \\ \dot{\varphi}_{65}^2 \end{bmatrix} + \\ + \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} \dot{\varphi}_{54}\dot{\varphi}_{65} + \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} g = \begin{bmatrix} M_{64} \\ M_{65} \end{bmatrix} \qquad (5)$$

kde

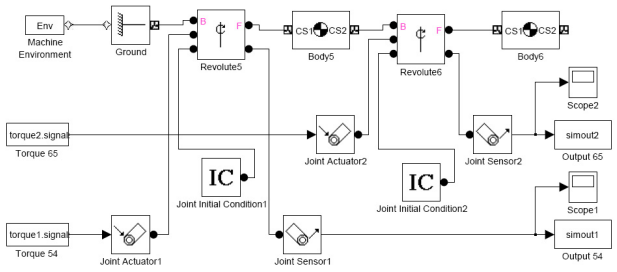
$$a_{11} = m_5 l_5^2 + I_5 + (m_6 + m_7)(d_5^2 + l_6^2 + 2 l_5 l_6 \cos\varphi_{65}) + I_6$$
$$a_{12} = (m_6 + m_7) l_6^2 + (m_6 + m_7) d_5 l_6 \cos\varphi_{65} + I_6$$
$$a_{21} = a_{12} \qquad a_{22} = (m_6 + m_7) l_6^2 + I_6$$
$$b_{11} = 0 \qquad b_{12} = -(m_6 + m_7) d_5 l_6 \sin\varphi_{65}$$
$$b_{21} = (m_6 + m_7) d_5 l_6 \sin\varphi_{65} \qquad b_{22} = 0$$
$$c_1 = -2(m_6 + m_7) d_5 l_6 \sin\varphi_{65} \qquad c_2 = 0$$
$$e_1 = -(m_5 + m_6 + m_7) l_5 \sin\varphi_{54} - (m_6 + m_7) l_6 \sin(\varphi_{54} + \varphi_{65})$$
$$e_2 = -(m_6 + m_7) l_6 \sin(\varphi_{54} + \varphi_{65})$$

pričom d_5, d_6, d_7 sú dĺžky členov robota podľa obr. 2, hmotné momenty zotrvačnosti ku ťažiskám sú I_5, I_6 a vzdialenosť ťažiska člena 5 od osi rotácie je I_5 resp. vzdialenosť spoločného ťažiska členov 6, 7 je I_6. Pohyb mechanizmu pre orientáciu sa uskutočňuje vo vertikálnej rovine, preto uvažujeme i vplyv tiaže.

V Lagrangeových pohybových rovniciach sa neuvažujú vôle a trecie účinky v jednotlivých kinematických dvojiciach ani elastické deformácie jednotlivých členov. Rovnica (5) predstavuje sústavu dvoch diferenciálnych rovníc 2. rádu. Pre numerické riešenie popri zadaných hodnotách hmotnostných, geometrických a kinematických veličín robota je treba uvažovať i ohraničenia maximálnych hodnôt hnacích momentov servopohonov.

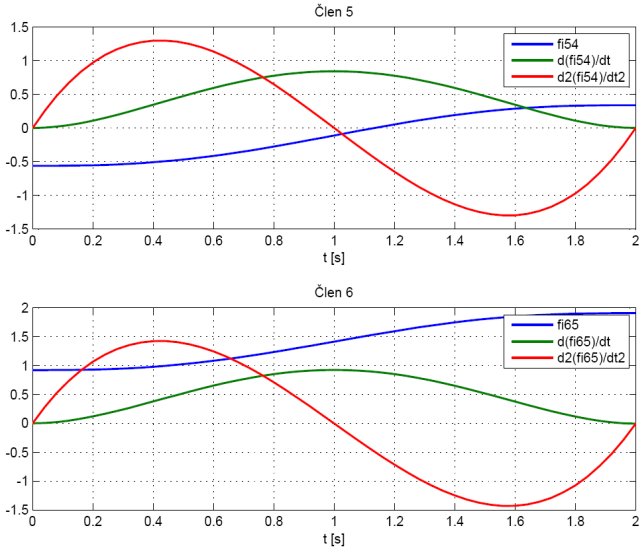
3. Riešenie priamej úlohy

Pri riešení priamej úlohy dynamiky mechanického systému sa vyšetruje priebeh výchyliek, uhlových rýchlostí a uhlových zrýchlení pri známych zovšeobecnených silách resp. momentoch vyvodzujúcich pohyb. Po aplikovaní vypočítaných momentov z inverznej úlohy na model pre riešenie priamej úlohy dynamiky (obr. 7) by sme mali dostať časový priebeh kinematických veličín zhodný s priebehom, ktorý sme určili na začiatku riešenia



Obr. 7 Bloková schéma modelu mechanického systému v SimMechanics (priama úloha)

Fig. 7 Block diagram of the mechanical system in SimMechanics (forward dynamics)



Obr. 8 Priebehy kinematických veličín v závislosti na čase

Fig. 8 Time histories of kinematical variable

Priebehy kinematických veličín získané zo simulácie viď obr. 8 v súlade s očakávaním korešpondujú s priebehmi, ktoré sme určili z pohybového stavu mechanizmu a ktoré sú uvedené na obr. 4. Ide o časový priebeh uhlov pootočenia jednotlivých členov mechanizmu pre orientáciu, ďalej o priebehy ich uhlových rýchlostí a uhlových zrýchlení. Zovšeobecnené sily mechanického systému predstavujúce momentové účinky v kinematických dvojiciach sú zároveň považované za dynamické funkcie riadenia pohonov priemyselného robota.

Záver

V práci sme sa zaoberali určovaním kinematických a dynamických funkcií riadenia mechanického systému priemyselného robota. Prezentovaný postup riešenia priamej a inverznej úlohy dynamiky vyšetrovaného mechanizmu pre orientáciu využíva modelovanie v prostredí SimMechanics.

SimMechanics umožňuje simulovať viachmotové mechanické systémy, ktoré sú bežným problémom v inžinierskej praxi. Úlohy sú riešené numericky na základe modelov pomocou blokových schém. Po zvládnutí metodiky riešenia sa ukazuje ako vhodný nástroj na riešenie praktických problémov technickej praxe. Významnú úlohu môže zohrať i pri dynamickom riešení priemyselných robotov a manipulátorov.

Poďakovanie

Práca vznikla za podpory vedeckej grantovej agentúry VEGA MŠ SR pri riešení grantovej úlohy č. 1/0201/08 Výskum štruktúr a správania sa modulov mechatronickej mobilnej technickej sústavy na úrovni orgánov a stavebných prvkov za účelom zlepšenia vlastnosti mobilnej technickej sústavy a grantovej úlohy č.1/0464/09 Výskum mechatronických sústav imitujúcich lokomóciu hada v obmedzenom a premenlivom priestore.

Literatúra

[1] CRAIG, J.: Introduction to Robotics, Mechanics and Control, Reading: Addison-Wesley Company, 1995

[2] HRONCOVÁ, D.: Príspevok k riešeniu priamej a inverznej úlohy dynamiky v prostredí SimMechanics. In: Modelovanie mechanických a mechatronických sústav MMaMS´2007, Herľany, 2007

[3] KARBAN, P.: Výpočty a simulace v programech Matlab a Simulink, Brno, Computer Press, 2006

[4] KOZÁK, S., KAJAN, S.: Matlab – Simulink I, Bratislava, STU Bratislava, 2006

[5] SEGĽA, Š.: Effect of small elastic deformations on the dynamics of multibody systems, In: Proc. Of the 7th International Workshop on Robotics in Aple-Adria-DanubeRegion RAAD 1998, Smolenice, 1998

[6] SHABANA, A., A.: Dynamics of Multibody Systems (2nd edition), Cambridge University Press, 1998

[7] SCHLOTTER, M.: Multibody System Simulation with SimMechanics, Unviersity of Canterbury, 2003

[8] VAVRINČÍKOVÁ, V., BARANOVÁ, E.: Dynamic Funktions of Cotnrol of Robot Arm, In: IX. International conference on the theory of machines and mechanisms, Liberec, 2004

[9] WALDRON, K., J., KINZEL, G., L.: Kinematics, Dynamics and Design of Machinery, John Wiley & Sons, Inc., New York, 1994

Abstract

In the paper are solved direct an inverse problems of robot dynamic in the Matlab and SimMechanics environment. The solution of direct problem deals with analysis of robot mechanical system movement under action of given forces and moments. The solution of inverse task is more complicated. Here are search the forces under which we receive prescribed movement of effector. Direct and inverse problems are solved numerically on the base of models and schemas of blocks. The results are verified by time dependent charts of kinematical and dynamical functions of robot control.

Key words: industrial robot, computer simulation, Matlab, SimMechanics.

Doc. RNDr. Viola Vavrinčíková, CSc.

TU Košice
Strojnícka fakulta
Katedra technickej mechaniky a mechatroniky
Letná 9, 040 01 Košice

Ing. Darina Hroncová

TU Košice
Strojnícka fakulta
Katedra technickej mechaniky a mechatroniky
Letná 9, 040 01 Košice
darina.hroncova@tuke.sk

Vytvorenie vizuálnej podoby modelu pre fyzikálny model v Microsoft Robotics Developer Studio

František Duchoň, Martin Štrenger

Abstrakt
Na vytvorenie komplexného modelu v Microsoft Robotics Developer Studio sa nepredpokladá len vytvorenie fyzikálneho modelu robota, ale aj prenesenie alebo vytvorenie jeho vizuálnej podoby, teda istého vizuálneho modelu. Použitím jednoduchých grafických programov ako je Google SketchUp v kombinácii so zložitejšími nástrojmi ako je Blender, dokáže užívateľ pomerne jednoducho a rýchlo vytvoriť vizuálny model, či už reálneho alebo práve vytváraného robota. Tento článok sa zaoberá vytvorením vizuálneho modelu pomocou týchto dvoch nástrojov. Článok vysvetľuje na jednoduchých príkladoch, ako postupovať v návrhu a vyhnúť sa nedostatkom jednotlivých nástrojov.

Kľúčové slová: vizuálny model robota, Microsoft Robotics Developer Studio, Google SketchUp, Blender

Úvod

Microsoft Robotics Developer Studio (MRDS, pred tým Microsoft Robotics Studio) umožňuje vytvárať a simulovať vlastné navrhnuté roboty v rôznych prostrediach a rôznych situáciách. Pri potrebe vytvorenia modelu vlastného robota v MRDS, vzniká požiadavka vytvorenia aj vlastnej vizuálnej podoby tohto modelu, tzv. vizuálneho modelu. Vizuálne modely pri ich bežnom použití nemajú na priebeh simulácie v MRDS nijaký vplyv, ak za vplyv nepočítame použitie časti výpočtového výkonu. Ich účelom je teda istá grafická reprezentácia skutočného robotického systému. Tento článok sa zaoberá jednoduchým vytvorením vizuálneho modelu robota, ktorý je elementárnym predpokladom na vytvorenie fyzikálneho modelu robota v MRDS a teda celkovej modelovej reprezentácie vlastného robotického systému.

1 Podpora modelov v Microsoft Robotics Developer Studio

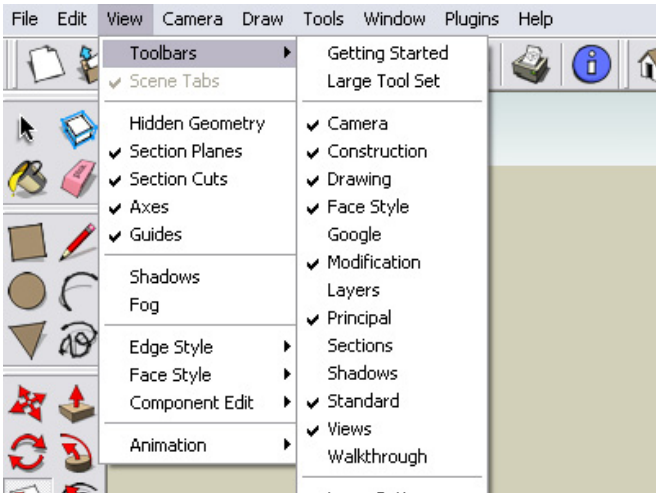
Microsoft Robotics Developer Studio umožňuje používať len 3D modely vo formáte *.bos* alebo *.obj*. Pri modeloch vo formáte *.obj* sa často vyskytuje aj súbor s rovnakým názvom a príponou *.mat*, ktorý obsahuje informácie o farbách pre model. V prípade ak je vytvorený vizuálny model robota vo formáte *.obj*, MRDS ho prekonvertuje na formát *.bos* a pôvodný súbor ponechá. Pri vytváraní vizuálneho modelu vo všeobecnosti nezáleží na použitom grafickom editore. Jedinou všeobecnou požiadavkou je, aby výsledný model po importovaní do MRDS bol správne otočený a so správnymi rozmermi. Veľmi ľahko sa môže stať, že súradnicový systém editora nekorešponduje so súradnicovým systémom v MRDS, keďže MRDS používa pravotočivý súradnicový systém *ZXY*. Správnosť rozmerov sa dá zistiť buď porovnaním s vytvoreným fyzikálnym modelom v MRDS, vtedy sa oba modely prekrývajú, alebo porovnaním s objektom referenčnej veľkosti. Ak sa rozmery nezhodujú, najčastejšie stačí v grafickom editore upraviť mierku celého modelu.

2 Vytvorenie vizuálneho modelu v Google SketchUp

Program Google SketchUp je grafický editor určený pre architektov. Hodí sa však aj na vytvorenie modelu robota. Dôvodom je jeho jednoduchosť používania a voľná dostupnosť na internete. Voľne dostupná verzia programu SketchUp však neobsahuje nástroj na ukladanie 3D modelov vo formáte *.obj* a preto je nutné do adresára *Google SketchUp\Plugins* skopírovať plugin, ktorý umožní export modelov aj v tomto formáte. Najbežnejší a taktiež voľne dostupný plugin má názov *obj_export1.1.rb*. Treba pri ňom počítať s istými obmedzeniami. Jedná sa najmä o neschopnosť exportovať farby a textúry. Dokáže exportovať iba geometriu. Ďalším obmedzením je schopnosť exportovať skupiny objektov len najvyššej úrovne. To znamená, že ak je viacero skupín spojených do jedného objektu a následne je tento objekt exportovaný, všetky podskupiny nižšej úrovne zaniknú. Ďalším obmedzením je fakt, že daný plugin exportuje celú geometriu v palcoch nezávisle od nastavenia jednotiek editora.

2.1 Úvodné nastavenia Google SketchUp

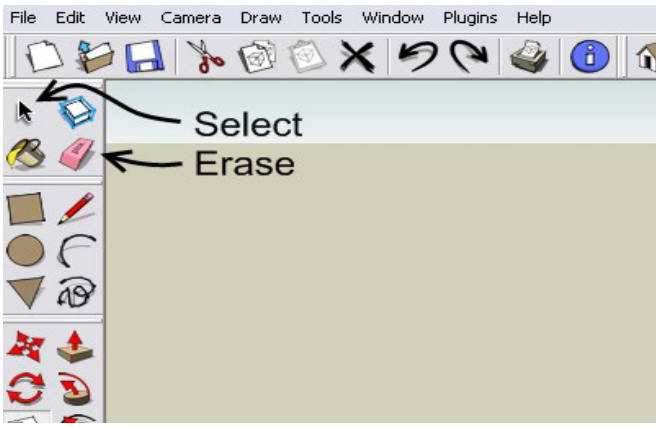
Po spustení editora je vhodné najskôr nastaviť rozmerové jednotky, aj keď na zvolených jednotkách v prípade importovania modelu do MRDS v skutočnosti nezáleží. Výber je vhodný preto, aby zvolené jednotky zodpovedali užívateľskej zvyklosti. Ďalšou možnosťou je výber 3D pohľadu alebo 2D plochy a taktiež na tomto výbere nezáleží, keďže je to možné zmeniť aj v samotnom programe. Dôležitým voľbou je ale výber nástrojov, ktoré užívateľ chce používať. To je možné zvoliť v hornom menu v položke *View* a v nej položkou *Toolbars*. Takto je možné zvoliť zobrazenie používaných nástrojov. Odporúčané nastavenia sú zobrazené na Obr. 1.



Obr.1 Odporúčané nastavenia používaných nástrojov

Fig.1 Recommended settings for used tools

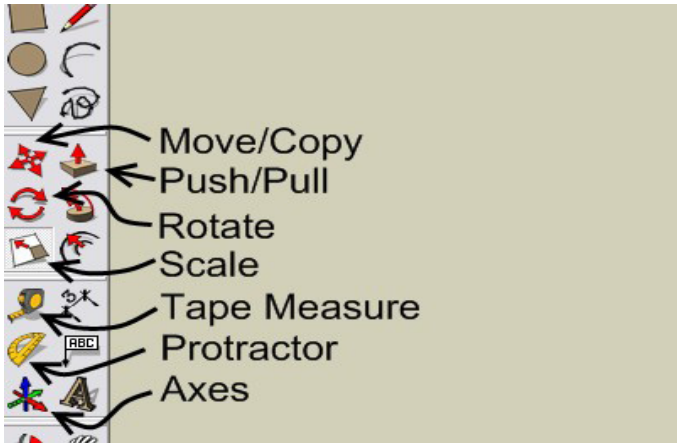
Pre vytvorenie vizuálneho modelu robota stačí poznať len niekoľko základných nástrojov. Prvým z nich je nástroj *Select*. *Select* slúži na označenie jednotlivých objektov nakreslených v editore. Ďalším nástrojom je nástroj *Erase*, ktorý slúži na mazanie objektov.



Obr.2 Nástroje Select a Erase

Fig.2 Select and Erase tools

Za týmito nástrojmi nasleduje skupina nástrojov, pomocou ktorých je možné nakresliť jednotlivé geometrické útvary ako je štvoruholník, čiara, kruh, polygón a ďalšie. V ďalšej skupine nástrojov sú dôležité nástroje *Move/Copy*, *Push/Pull*, *Rotate* a *Scale*. Nástroj *Move/Copy* slúži na premiestnenie alebo vytvorenie kópie aktuálne označených objektov. Nástroj *Push/Pull* slúži vytvorenie trojrozmerného objektu z dvojrozmernej predlohy alebo upravenie rozmerov. Pomocou *Rotate* môžeme jednotlivé objekty otáčať okolo jednotlivých osí. Posledným nástrojom zo skupiny je nástroj *Scale*, ktorý slúži na zmenu mierky. Pri vytváraní komplikovanejšieho modelu dobre poslúžia nástroje *Tape Measure* a *Protractor*, ktorým je možné vytvoriť pomocné vodiace čiary a nástroj *Axes*, ktorým je možné zmeniť stred súradnicovej sústavy a orientáciu osí. Aby nevznikli problémy so zlou orientáciou modelu, je vhodné kresliť model tak, aby smer pohybu dopredu bol v kladnom smere červenej osi a výška modelu bola v kladnom smere zelenej osi (obr.4).



Obr.3 Ďalšie nástroje

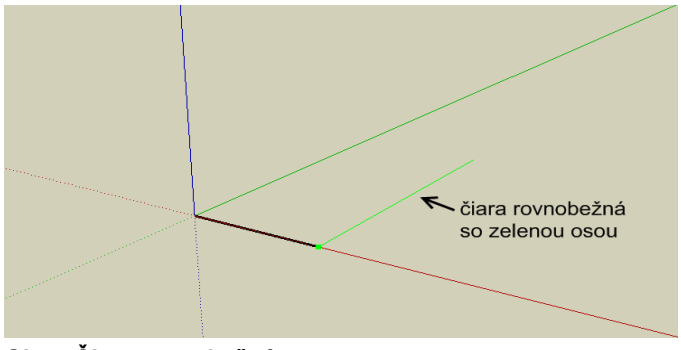
Fig.3 Other tools

2.2 Práca s programom Google SketchUp

SketchUp patrí do skupiny programov s veľmi jednoduchým ovládaním. Zmenu uhla pohľadu možno vykonať prostredným tlačidlom na myši a pohybom myši nastaviť požadovaný uhol. Horizontálny a vertikálny posun pohľadu sa robí obdobným spôsobom pri podržaní klávesy SHIFT. Priblíženie alebo oddialenie sa vykonáva pomocou kolieska na myši.

Kreslenie útvarov je taktiež jednoduché. Prvým krokom je zvolenie požadovaného nástroja (geometrického útvaru). Kreslenie následne pozostáva najčastejšie z dvoch krokov, a to zvolenie počiatočného a koncového bodu, pričom funkcia týchto bodov je daná zvoleným nástrojom. Napríklad ak je zvolený nástroj čiary, potom tieto dva body predstavujú začiatok a koniec čiary.

Kresliť je možné dvomi spôsobmi. Jeden spôsob je ten, že prvý bod zadefinuje užívateľ kliknutím myšou na miesto, kde ho chce mať umiestnený a koncový bod zadefinuje opäť kliknutím. Druhý spôsob spočíva v tom, že prvý bod zadefinuje kliknutím, pričom tlačidlo nepustí a druhý bod zadefinuje až keď tlačidlo na myši uvoľní. Kreslenie užívateľovi uľahčuje samotný editor. Príkladom je zmena farby čiary podľa toho, s ktorou osou je rovnobežná.



Obr.4 Čiara rovnobežná s osou

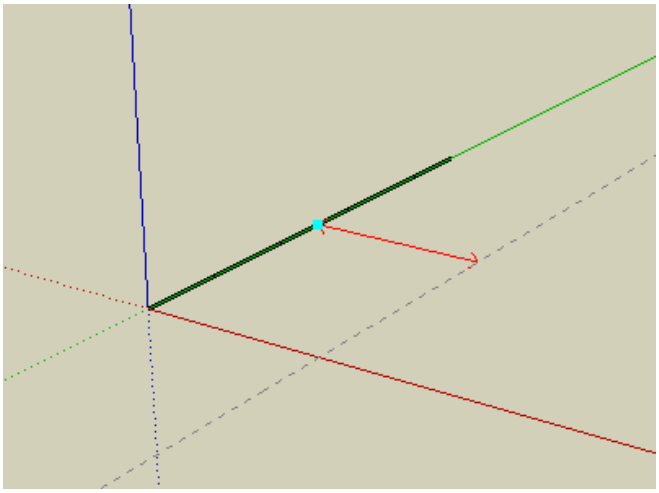
Fig.4 Parallel line with axis

Editor ponúka aj ďalšiu pomôcku. Ak už užívateľ dosiahol požadovaný smer čiary, ale čiara nemá požadovanú dĺžku, stlačením a držaním klávesy *SHIFT* sa zablokujú všetky pohyby okrem pohybov v danom smere. Ak má užívateľ napríklad zvolenú čiaru rovnobežnú s osou, po pridržaní tlačidla *SHIFT* sa bude pri pohybe myšou daná čiara iba predlžovať alebo skracovať, pričom jej aktuálnu dĺžku možno vidieť v stavovom riadku vpravo dole. Najjednoduchším spôsobom je však nakresliť útvar pomocou zadania parametrov, v prípade čiary teda iba jej dĺžky. Po zadefinovaní

začiatočného bodu užívateľ nastaví požadovaný smer a napíše požadovanú dĺžku. Potvrdí ju stlačením klávesy *ENTER*. Rovnakým spôsobom sa dajú kresliť aj ostatné útvary.

2.3 Kreslenie v programe Google SketchUp

Nástroj *Push/Pull* slúži na predĺženie, skrátenie alebo posúvanie už nakreslených objektov. Takto je možné jednoduchým spôsobom vytvoriť kváder z už nakresleného obdĺžnika (odpadá teda nutnosť kresliť každú čiaru kvádra zvlášť). Pri tvorbe modelu môžu prácu uľahčiť pomocné čiary. Pomocné čiary sa v editore zobrazujú ako prerušované nekonečne dlhé čiary. Pomocné čiary sa dajú vytvoriť pomocou nástrojov *Tape Measure* a *Protractor*. Prvý nástroj vytvára paralelné pomocné čiary a druhý nástroj vytvára čiaru, ktorá zviera s inou čiarou želaný uhol. Pri vytváraní pomocných čiar musí byť vedľa kurzora myši malý symbol plus. Ak sa tam symbol nenachádza, treba stlačiť klávesu *CTRL* a symbol by sa mal zobraziť. Nástroj *Tape Measure* okrem iného slúži aj na zmenu rozmerov celého modelu. Stačí ak užívateľ kliknutím určí začiatočný a koncový bod a následne napíše želanú hodnotu.



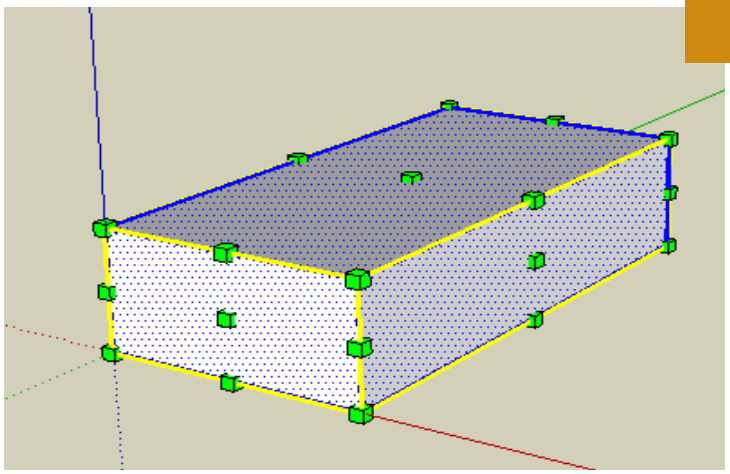
Obr.5 Pomocné čiary

Fig.5 Helping lines

Na posunutie alebo otočenie hotových objektov sa využívajú nástroje *Move/Copy* a *Rotate*. Ak chce užívateľ objekt posunúť, najprv ho označí nástrojom *Select* a následne si zvolí nástroj *Move/Copy*. Týmto nástrojom môže objekt presunúť na požadované miesto, prípadne ak stlačí klávesu *CTRL* môže vytvoriť kópiu označeného objektu. Rovnakým spôsobom funguje aj nástroj *Rotate*.

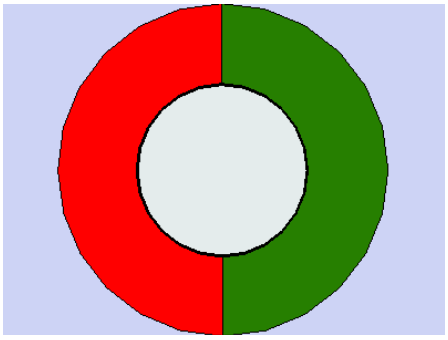
2.4 Export modelu do formátu .obj

Na exportovanie modelu sa používa napríklad už spomínaný plugin obj_export1.1.rb. Plugin je možné nájsť v hornom menu v záložke Plugins. Pre exportovanie stačí zadať cestu kam sa má výsledný súbor uložiť. Predtým je však vhodné spojiť niektoré objekty do skupín, hlavne v tom prípade, ak ich bude potrebné ešte zafarbiť. Plugin totižto nedokáže exportovať použité farby a materiály. Pre vytvorenie skupín stačí dané objekty označiť a pravým tlačidlom na myši vyvolať kontextové menu. Z menu potom treba vybrať položku *Make Group*. Používaný formát podporuje len jednoduchú geometriu, preto je potrebné niektoré objekty rozdeliť pomocou čiar. Týka sa to hlavne rôznych otvorov v stenách.



Obr.6 Nástroj Scale

Fig.6 Scale tool

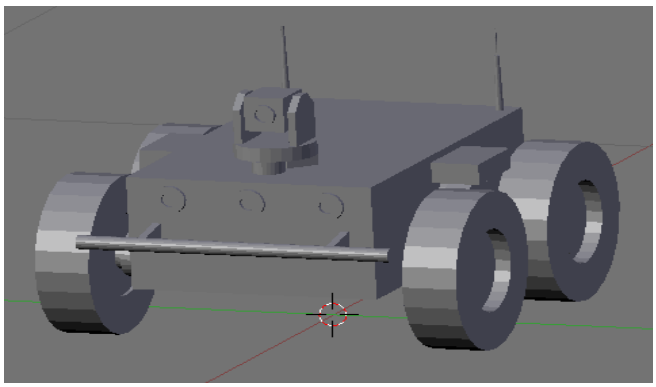


Obr.7 Rozdelenie objektu pomocou čiar

Fig.7 Splitting-up of object with lines

3 Práca s programom Blender

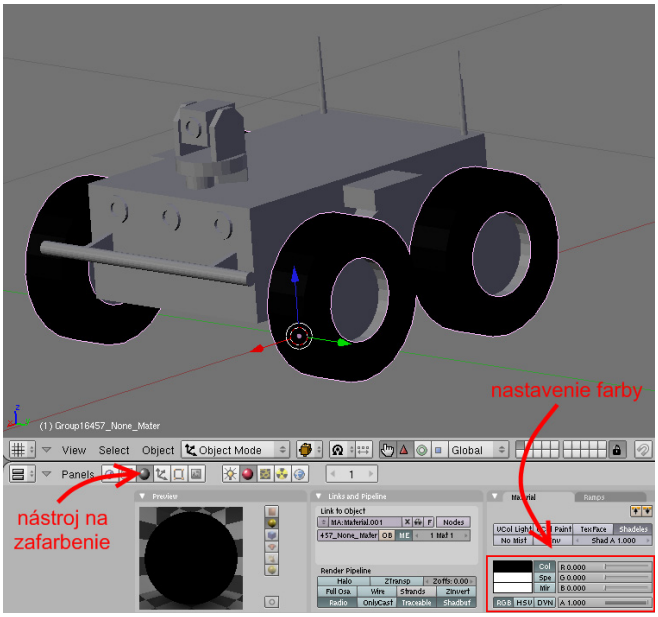
Program Blender je voľne dostupný grafický editor, ktorý slúži na tvorbu 3D modelov a animácií. Umožňuje vytvorenie kompletného modelu (vrátane farieb apod.) a jeho exportovanie do formátu *.obj*. Pre vyššiu náročnosť ovládania je ho vhodné použiť len na posledné úpravy a pridanie farieb do vizuálneho modelu robota. Ovládanie kamery je rovnaké ako v programe Google Sketchup. Výrazný rozdiel je v označovaní objektov. Objekty sa označujú pravým tlačidlom na myši. Po spustení Blender sa v editore objaví nakreslená kocka, ktorú je možné zmazať stlačením klávesy *DELETE* a potvrdením. Vizuálny model robota treba najskôr importovať. To je možné spraviť cez hlavné menu *File*, položku *Import* a voľbu *Wavefront(.obj)*.



Obr.8 Model importovaný z Google SketchUp do Blender

Fig.8 Model imported from Google SketchUp to Blender

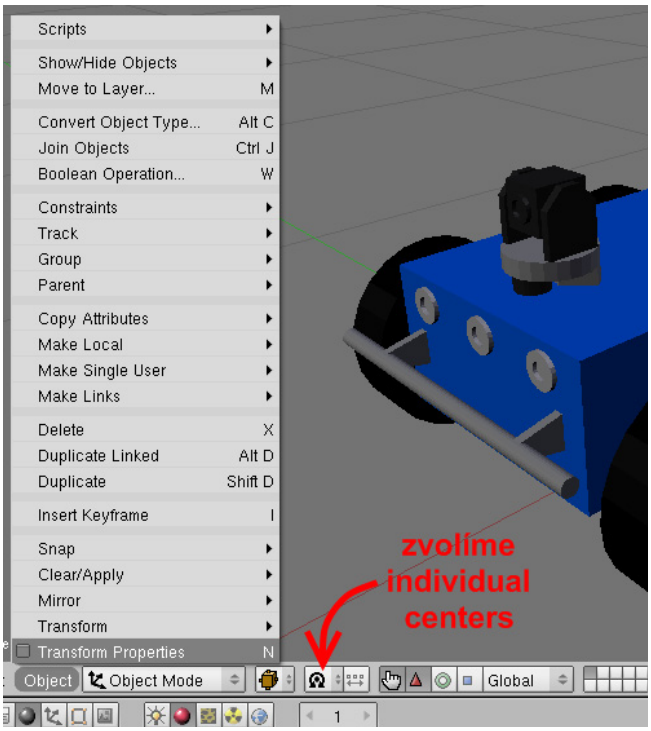
Model robota je síce vo formáte *.obj*, neobsahuje však žiadne informácie o farbách (Obr. 8). Nástroj pre editovanie farieb je možné zvoliť stlačením *F5* alebo kliknutím na odpovedajúcu ikonku (Obr. 9). Pri farbení sa zafarbí danou farbou všetko, čo je aktuálne označené. Preto je dobré v Google SketchUp na konci práce vhodne pospájať objekty do jednotlivých skupín.



Obr.9 Farbenie modelu

Fig.9 Model colouring

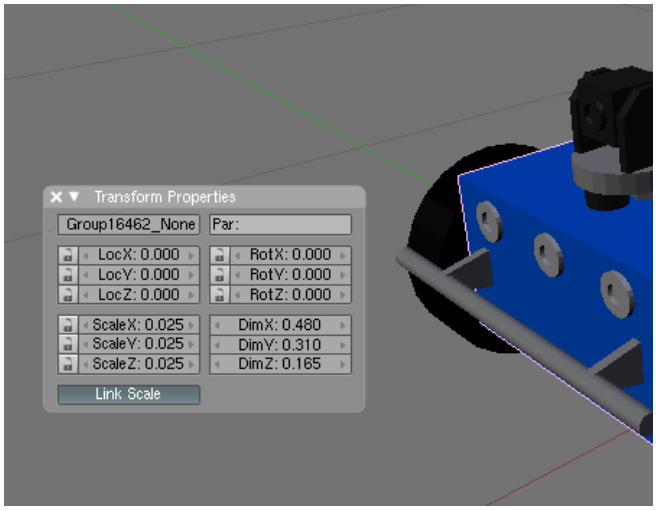
Ak je potrebné zmeniť rozmery modelu, treba nastaviť *Pivot* na *Individual Centers* (Obr. 10). Z menu treba vybrať položku *Object*, následne sa objavia dve možnosti.



Obr.10 Zmena mierky

Fig.10 Scale changing

Prvá možnosť je zmeniť mierku pomocou nástroja *Scale*. Pri takomto spôsobe zmeny mierky je vhodné mať označený celý model. Mierku je možné nastaviť pohybom myši, vhodnejšie je však napísať mierku priamo. Odpovedajúce rozmery modelu v metroch (pre potreby MRDS) užívateľ dosiahne zadaním mierky 0.0254 (prepočet z palcov). Druhou možnosťou je nastaviť rozmery pomocou *Transform Properties*. Nástroj umožňuje nastaviť priamo rozmery modelu alebo mierku. Nevýhodou však je, že umožňuje meniť rozmery len jednej skupiny súčasne, a preto užívateľ musí meniť rozmery každej skupiny zvlášť. Pri menení rozmerov je vhodné mať zapnutý nástroj *Link Scale,* aby sa menili rozmery vo všetkých osiach súčasne.



Obr.11 Zmena mierky pomocou Transform Properties

Fig.11 Scale changing with Transform Properties

Ak užívateľ dokončil všetky úpravy, môže model exportovať. Postup pri exportovaní modelu je podobný ako pri jeho importovaní. Najprv užívateľ označí všetky časti modelu, najjednoduchšie stlačením klávesy *A*. Z vrchného menu zvolí položku *File* a z otvoreného menu zvolí *Export* a vyberie *Wavefront(.obj)*. V otvorenom okne zadá názov výsledného súboru a klikne na *Export Wavefront OBJ*. Po exportovaní vzniknú dva súbory, jeden s príponou **.obj* a druhý s príponou **.mat*. Tieto súbory pre importovanie do MRDS treba skopírovať do adresára *Microsof Robotics Developer Studio\store\media.*

Záver

Pomocou dvoch voľne dostupných grafických nástrojov je možné navrhnúť vizuálnu podobu modelu robotického systému, ktorý je možne ďalej používať v Microsoft Robotics Developer Studio. Zvolený postup návrhu umožňuje vytvoriť vizuálne modely robotických systémov od jednoduchých až po komplexné rozmermi alebo štruktúrou. Takto zvoleným postupom môže užívateľ importovať modely vlastných reálnych robotov do prostredia MRDS, čo mu umožní jednoduchým spôsobom testovať navrhnuté inteligentné správanie robotov, bez nutnosti riskovať poškodenie reálneho robotického systému.

Poďakovanie

Článok vznikol pri riešení projektu VEGA 1/0690/09.

Literatúra

[1] Google SketchUp Reference Manual, **http://sketchup.google.com/support/bin/topic.py?topic=13702**

[2] Blender 3D, Tutorial Links List, **http://en.wikibooks.org/wiki/Blender_3D:_Tutorial_Links_List#Official_Blender_Documentation**

Abstract

For creating of a complex model of mobile robot in Microsoft Robotics Developer Studio, it is necessary not only to create physical model of robot, byt also creating of visual form of that model, certain visual model. Using simple grapgic tools such as Google SketchUp in combination with complicated tools such as Blender, can lead to simple and fast creation of visual model, whether real robot or actually designed robot. This paper deals with creating such a visual model using these two tools. Paper explains on simple examples, how to proceed in design and how to avoid of shortages of each tool.

Ing. František Duchoň,
Bc. Martin Štrenger

Fakulta elektrotechniky a informatiky
Ústav riadenia a priemyselnej informatiky
Ilkovičova 3
812 19 Bratislava
frantisek.duchon@stuba.sk
mastr@ynet.sk

Tvorba fyzikálneho modelu v Microsoft Robotics Developer Studio

Ladislav Jurišica, František Duchoň, Martin Štrenger

Abstrakt
Elementárnym predpokladom na vytvorenie inteligentného systému v simulačnom prostredí je zostavenie takého modelu systému, ktorý sa maximálne približuje správaniu sa reálneho systému v reálnom prostredí. Microsoft Robotics Developer Studio je vhodným simulačným prostredím. Okrem programovania inteligencie robotických systémov umožňuje zapracovať do simulačného prostredia aj vlastné modely robotických systémov. Keďže je o tomto štúdiu málo odbornej literatúry a práca s týmto štúdiom vyžaduje isté vedomosti a skúsenosti, tento článok sa zaoberá vytvorením fyzikálneho modelu diferenciálne riadeného mobilného robotického systému v tomto prostredí.

Kľúčové slová: Microsoft Robotics Developer Studio, fyzikálny model robota, simulácia mobilného robota

Úvod

Microsoft Robotics Developer Studio (MRDS, pred tým Microsoft Robotics Studio) umožňuje vytvárať a simulovať vlastné navrhnuté roboty v rôznych prostrediach a rôznych situáciách. Môže sa jednať o kolesové mobilné roboty s rôznym počtom a usporiadaním kolies, kráčajúce roboty alebo klasické 6-osé roboty využívané v priemysle. Taktiež je možné jednotlivé typy kombinovať a vytvoriť tak napríklad štvorkolesový robot, ktorý bude mať na sebe robotickú ruku. Základom všetkého je vytvorenie čo najlepšieho fyzikálneho modelu daného robota pre MRDS. Zahŕňa to vytvorenie modelu daných rozmerov a hmotnosti, vytvorenie modelu motorov, senzorov a nakoniec samotného prostredia, v ktorom sa bude daný robot pohybovať. Tento článok sa zaoberá vytvorením fyzikálneho modelu diferenciálne riadeného robota z už vopred pripraveného vizuálneho modelu robota.

1 Stručný opis Microsoft Robotics Developer Studia

Microsoft Robotics Developer Studio je vývojové prostredie určené pre operačné systémy Windows [5] a umožňuje vývojárom jednoduché navrhnutie robotických aplikácií pre rôzne hardvérové platformy. Obsahuje vizuálne vývojové prostredie, ktoré umožňuje jednoduchý návrh a odladenie rôznych robotických aplikácií. MRDS využíva realistické 3D modely s licencovaným fyzikálnym modelom AGEIA™ PhysX™ [1], ktorý umožňuje simulácie robotických modelov s fyzikou reálneho sveta. Okrem samotného prostredia, umožňuje používateľovi generovať štandardné služby pre hardvér a softvér a teda aj komunikáciu s robotmi pomocou webového alebo Windows rozhrania. Programovací model MRDS môže byť aplikovaný na rôzny hardvér, pričom umožňuje prechod z jednej platformy na inú. Vývojár si môže zvoliť programovací jazyk, v ktorom chce programovať. MRDS dokáže pracovať s jazykmi C#, VB.NET, MS Iron Python a MS Visual Programming Language. Základné pojmy pre Microsoft Robotics Developer Studio:

- **Concurrency and Coordination Runtime (CCR)** – proces pre súbežnosť a koordináciu, zabezpečuje koordináciu správ bez nutnosti manuálne tvoriť vlákna, semafory a zámky. CCR je založené na asynchrónnom prechode správ.
- **Decentralized System Services (DSS)** – služby, ktoré poskytujú hostiteľské prostredie pre ostatné služby, umožňujúce úlohy ako ladenie, záznam, monitorovanie, bezpečnosť, sprístupnenie a dátovú stálosť.
- **Application** – aplikácia je spojenie služieb, ktoré pomocou riadenia môžu byť spriahnuté pre vykonanie želanej úlohy
- **Contract** – kontrakt je skrátený opis služby, ktorý opisuje jej správanie tak, aby ju mohli použiť aj iné služby. Kontrakt je zvyčajne identifikovaný pomocou HTTP URI.
- **Manifest** – manifest je XML dokument, ktorý opisuje súbor služieb, ktoré majú byť použité pri behu DSS. Manifest môže byť použitý na explicitné prepísanie štandardných vlastností služieb ako napríklad aktuálne meno služby alebo s ktorým partnerom alebo službou má spolupracovať. V spojení s vytvorením služby pomocou manifestu, služby môžu byť vytvorené programovo pomocou iných služieb alebo pomocou použitia Control Panelu v internetovom prehliadači. Služba Control Panel je spojenie služieb a je prítomná vždy, keď je spustený DSS proces (DSS node).
- **Orchestration** – v spojení s robotickými aplikáciami predstavuje úlohu zosúladenia vstupov zo senzorického systému a ovládanie súboru akčných členov na základe týchto vstupov. Cieľom je poskytnúť funkčnosť a vystupovať ako nezávislá aplikácia alebo ako časť (komponent) nadradenej úrovne riadenia.
- **Partner** – partner je služba spojená s inými službami so zámerom partnerstva. Partnerstvo je spôsob opisu vzťahu medzi dvomi službami v takom smere, v ktorom je umožnené použitie podriadených členov pri použití neskoršieho zviazania.
- **Proxy** – zástupca, je to vygenerovaná zostava, ktorá odkrýva kontrakt služby tak, aby túto službu mohli vyu-

žiť aj ostatné služby. Zostava (proxy assembly) obsahuje verejný operačný port a typy stavov vyjadrené službou. Keď služby vytvárajú medzi sebou partnerstvo spoja sa medzi sebou pomocou proxy assembly. Proxy je možné pomocou použitia príkazového riadku príkazom DssProxy. Tento príkaz je automaticky zapojený do procesu kompilácie počas generovania novej služby použitím príkazu DssNewService. Výsledkom spustenia DssProxy je projekt Visual Studia, ktorý je vytvorený v adresári Proxy v priečinku aktuálnej služby.

Služby sú základný stavebný blok pri tvorbe aplikácií MRDS. Aplikácia môže byť napríklad nahradená tromi službami, teda namiesto každého bloku jedna služba. Keďže služby môžu medzi sebou komunikovať rovnako dobre v rámci jedného uzlu (node) ako aj po sieti, tieto tri služby môžu bežať všetky naraz na jednom počítači alebo každá zvlášť na inom počítači spojenými pomocou internetu. Kvôli rôznym ohraničeniam však služby o sebe „nevedia“ až kým nie sú spustené. Táto forma spájania sa nazýva partnerstvo. Partnerstvo je spôsob opisu vzťahu medzi dvomi službami v takom smere, v ktorom je umožnené použitie podriadených členov pri použití neskoršieho zviazania. Kompozícia pomocou partnerstva je cestou k poskytnutiu vyššieho stupňa abstrakcie. MRDS Runtime obsahuje súbor služieb, ktoré poskytujú veľa často používaných a potrebných funkcií ako sú monitorovanie, ladenie, riadenie životného cyklu služieb a manažér podpisov (subscription manager).

2 Vytvorenie vlastných služieb

Fyzikálny model robota pre Microsoft Robotics Developer Studio je možné vytvoriť v prostredí Microsoft Visual C# Express Edition 2008 v jazyku C#. Tento jazyk sa využíva na vytváranie väčšiny programov a služieb pre MRDS. Vytváranie služieb pre Microsoft Robotics Developer Studio vyžaduje veľa vedomostí a skúseností. Okrem toho zatiaľ neexistuje žiadna programová dokumentácia opisujúca fungovanie jednotlivých služieb. Službu si užívateľ môže vytvoriť pomocou príkazového riadku *DSS Command Prompt* z Windows menu *Štart* > *Programy* > *Microsoft Robotics Developer Studio*. Otvorí sa okno s príkazovým riadkom. Ďalším krokom je nastavenie sa do pracovného adresára, v ktorom sa vytvorí nová služba. Nová služba sa vytvára príkazom *DssNewService* spolu s parametrami ako sú *namespace* a názov. Takto vytvorená služba predstavuje projekt pre Microsoft Visual C#, ktorý je možné ďalej editovať. Príklad:

>Dss*NewService*
/namespace:Microsoft.Robotics.Services.Simulation.Drive
/service:SimulatedCustomDifferentialDrive

V príklade sa teda vytvorila služba s názvom *SimulatedCustomDifferentialDrive*. Ako vidno podľa parametra *namespace*, projekt patrí do skupiny simulovaných podvozkov a tento parameter slúži na zaradenie do požadovanej skupiny pre lepší prehľad. Názov a zaradenie tejto novej služby nebolo vybrané náhodne. Vytváranie vlastných služieb nie je jednoduché, a preto je vhodné využiť už existujúce služby. V tomto prípade sa jedná o existujúcu službu *SimulatedDifferentialDrive*, ktorú je možné nájsť v hlavnom adresári MRDS v podadresári *Samples\Simulation\DifferentialDrive*. Vytvorený projekt predstavujúci službu sa nachádza v adresári s rovnakým menom ako je meno služby. Navyše v adresári */bin/services/* budú po skompilovaní vytvorené nasledujúce súbory, ktoré obsahujú okrem názvu služby aj čas, kedy boli vytvorené, teda rok a mesiac:

SimulatedCustomDifferentialDrive.Y2008.M10.dll

SimulatedCustomDifferentialDrive.Y2008.M10.pdb

SimulatedCustomDifferentialDrive Y2008.M10.proxy.dll

SimulatedCustomDifferentialDrive Y2008.M10.proxy.pdb

SimulatedCustomDifferentialDrive.Y2008.M10.transform.dll

SimulatedCustomDifferentialDrive.Y2008.M10.transform.pdb

V prípade úspešného vytvorenia služby možno daný projekt otvoriť v prostredí Microsoft Visual C#. Po otvorení projektu sa objavia súbory, ktoré obsahujú základnú štruktúru programu. Najskôr je nutné nastaviť referencie. Po kliknutí pravým tlačidlom myši na položku *References* treba vybrať *Add Reference*. Zobrazí sa okno s viacerými záložkami. Po otvorení záložky.*NET* treba vybrať referencie, ktoré bude užívateľ neskôr v programe potrebovať. Keďže ako vzor bola použitá služba *SimulatedDifferentialDrive*, je možné si pozrieť referencie, inak povedané knižnice, ktoré daná služba potrebuje a následne ich je nutné pridať do projektu. Význam niektorých referencií je nasledovný:

PhysicsEngine – knižnica fyzikálneho jadra

RoboticsCommon – knižnica rôznych robotických služieb

SimulationCommon, SimulationEngine – knižnice simulačného prostredia

Okrem týchto knižníc si projekt môže niekedy vyžiadať pridať aj referencie s rovnakým názvom, ale zakončeným príponou *.Proxy*. Niekedy si program vyžiada pridať aj mnoho iných referencií.

3 Vytvorenie štruktúry programu

Microsoft Robotics Developer Studio už po nainštalovaní obsahuje vytvorené služby zodpovedajúce reprezentácii diferenciálneho podvozku. Tieto služby však obsahujú mnoho nepresností, a preto je vhodné vytvoriť si služby vlastné, pričom pôvodné služby možno použiť ako vzor. Služba *SimulatedDifferentialDrive* síce predstavuje službu pre diferenciálny podvozok, ale neobsahuje žiadne informácie o fyzikálnom modeli robota. Jedná sa teda iba o obslužnú službu, ktorá sprostredkováva akúsi komunikáciu s modelom. Túto službu možno prakticky použiť bez zmeny, je však viazaná len na jeden fyzikálny model.

Nie je povinnosťou vytvoriť všetky funkcie, ktoré materská trieda poskytuje. Materskou triedou pre *SimulatedDifferentialDrive* je trieda *Drive*, ktorej zdrojový kód je dostupný v adresári *\Samples\Common*. Tento kód už netreba kompilovať, pretože je obsiahnutý hneď po inštalácii MRDS. Navyše pri kompilácii často vyhodí množstvo chýb a môže sa stať, že prepíše pôvodnú funkčnú verziu. To môže vyústiť až do nutnosti preinštalovať celé prostredie MRDS. Zdrojové kódy zo súborov *Drive.cs*, *DriveState.cs* a *DriveTypes.cs* však môžu poslúžiť na nahliadnutie do štruktúry programu a pochopenie niektorých súvislostí. Pre potreby užívateľa-začiatočníka postačí upraviť vlastný program podľa vzoru *SimulatedDifferentialDrive*, pričom sa dá použiť takmer celá štruktúra programu. Program sa však musí prispôsobiť vytvorenej triede *SimulatedCustomDifferentialDrive*. Prvým krokom je kontrola referencií. *SimulatedCustomDifferentialDrive* by mal v zásade používať rovnaké referencie ako *SimulatedDifferentialDrive*. Druhým krokom je použitie referencií pomocou termínu using. Symboly nasledujúce medzi príkazom *using* a znamienkom = predstavujú určitú formu substitúcie. Namiesto vypisovania celého názvu referencie stačí napísať v programe už len skrátený názov. Dôležité je hlavne použitie nasledovných referencií:

```
using submgr = Microsoft.Dss.Services.SubscriptionManager;
using diffdrive = Microsoft.Robotics.Services.Drive.Proxy;
```

```
using simtypes = Microsoft.Robotics.Simulation;
using simengine = Microsoft.Robotics.Simulation.Engine;
using physics = Microsoft.Robotics.Simulation.Physics;
```

Ako vidno z uvedených referencií, v programe sa používajú objekty patriace do triedy *Drive*. Rovnako sa používajú aj objekty patriace pod simulačné prostredie a fyzikálne jadro. Následne je potrebné zabezpečiť komunikáciu tejto služby s ostatnými službami, napríklad aj so simulačným prostredím a fyzikálnym jadrom. Za komunikáciu je zodpovedný *Subscription Manager*, ktorý umožňuje ostatným službám vymieňať si údaje medzi sebou. Do programu je teda potrebné doplniť nasledujúce riadky:

```
#region Simulation Variables
simengine.SimulationEnginePort _simEngine;
simengine.SimulatedCustomDifferentialDriveEntity _entity;
simengine.SimulationEnginePort _notificationTarget;
#endregion
[Partner("SubMgr", Contract = submgr.Contract.Identifier,
CreationPolicy = PartnerCreationPolicy.CreateAlways)]
submgr.SubscriptionManagerPort _subMgrPort = new
submgr.SubscriptionManagerPort();
string _subMgrUri = string.Empty;
```

V regióne Simulation Variables sú zadefinované potrebné porty, cez ktoré *SimulatedCustomDifferentialDrive* komunikuje so simulačným prostredím pomocou správ. Syntax *#region* sa používa na sprehľadnenie programu a umožňuje dočasne skryť kód umiestnený medzi hranicami regiónu. Za touto oblasťou nasleduje spustenie partnerskej služby, v tomto prípade je partnerskou službou už spomínaný *SubscriptionManager* a funkcia *Start()*, ktorá je obdobou funkcie *Main()* používanej v jazyku C. Obsahuje vo veľkej miere iba príkazy vytvárajúce inštancie potrebných objektov, vytvorenie komunikačných portov a spustenie obslužných funkcií zachytávajúcich prichádzajúce správy. Je možné sem doplniť obslužné funkcie zachytávajúce správy prichádzajúce na porty, ktoré boli vytvorené samostatne. V druhom riadku vyššie uvedenej časti programu chce program používať niečo, čo ešte nie je vytvorené. Je to preto, lebo daná služba a ani vzorová služba neobsahujú informácie o fyzikálnej interpretácii daného modelu. Samotný fyzikálny model pre triedu *SimulatedDifferentialDrive* sa nachádza v súbore *Entities.cs* umiestnenom v adresári *\Samples\Simulation\Entities*. V tomto súbore sú definované aj všetky ostatné objekty, ktoré sa využívajú v simulačnom prostredí. Je v ňom možné nájsť definované rôzne typy snímačov, fyzikálne modely rôznych robotických platforiem, ale aj objekty, ktoré slúžia na modelovanie simulovaného prostredia. Tento súbor je však opäť len ukážkou a sám o sebe sa nedá skompilovať. Predstavuje však vynikajúcu pomoc pri vytváraní vlastných objektov pre simulačné prostredie. Model diferenciálneho podvozku možno nájsť v regióne *Robot Platforms* vrátane konkrétnych modelov robotov postavených na tomto podvozku.

4 Vytvorenie fyzikálneho modelu robota

Pre vytvorenie fyzikálneho modelu robota je potrebné do projektu *SimulatedCustomDifferentialDrive* pridať nový súbor predstavujúci novú triedu. Táto trieda bude reprezentovať fyzikálny model robota. Prvým krokom je teda vytvorenie triedy s názvom *SimulatedCustomDifferentialDriveEntitiy*. Triedu je možné pridať kliknutím pravého tlačidla myši na názov projektu v okne *Solution Explorer*. Otvorí sa menu a z neho treba vybrať položku *Add* a ďalej New Item. Z daných možností typ *Class* a dole v riadku treba uviesť názov novej triedy, čo je v tomto prípade *SimulatedCustomDifferentialDriveEntities.cs*. Ďalším krokom je úprava názvu *namespace* na *Microsoft.Robotics.Simulation.Engine*. Z originálneho *Entities.cs* je možné skopírovať všetky *using* príkazy. Ak budú pri kompilácii chýbať niektoré referencie, je ich možné pridať neskôr. V ďalšom kroku je potrebné upraviť deklaráciu triedy *SimulatedCustom-DifferentialDrive*. Ako si možno všimnúť v programe, z ktorého sa vychádza, trieda musí byť verejná a derivovaná z triedy *VisualEntity*. Trieda *VisualEntity* je základná trieda pre všetky objekty, ktoré sa zobrazujú v simulačnom prostredí. Deklarácia triedy je nasledovná:

```
public class SimulatedCustomDifferentialDriveEntity: VisualEntity
```

4.1 Región State

V tele triedy *SimulatedCustomDifferentialDriveEntity* sa na začiatku programu nachádza región *State*. Definujú sa tu rôzne parametre modelu. Hlavnými parametrami sú váha (MASS), rozmery (DIMENSIONS), pozícia pravého a ľavého kolesa súčasne s pozíciou podporného kolieska (CASTER_WHEEL) a vzdialenosť podvozku od zeme (CHASSIS_CLEARANCE). Ďalej nasledujú údaje o polomere, šírke a váhe predných kolies a rovnako aj zadného podporného kolieska. Posledným takýmto údajom je ešte vzdialenosť osi kolies od stredu podvozku (FRONT_AXLE_DEPTH_OFFSET). V regióne *State* sa okrem spomenutých premenných nachádzajú ešte aj definície niektorých špecifických premenných a objektov, ktoré sú označované ako *DataMember*. Charakteristickými črtami pre takýto typ objektov sú funkcie *Set* a *Get* pre nastavenie a získanie hodnoty danej premennej. Vďaka týmto vlastnostiam je ich možné nastaviť a prezerať aj z iných tried. Do tejto skupiny patrí napríklad premenná *IsEnabled* označujúca stav kedy je motor zapnutý alebo vypnutý. Rovnako sem patria aj objekty reprezentujúce kolesá a objekty predstavujúce tvar robota. Klasické kolesá sú reprezentované objektom *WheelEntity*, ktorého interakcia s prostredím je zakomponovaná vo fyzikálnom jadre. Podporné koliesko je najčastejšie v podobe guľôčky (SphereShape) a samotný tvar robota je dosiahnutý poskladaním kvádrov (BoxShape) rôznych veľkostí do jedného celku.

4.2 Metóda Inictialize

Model robota sa skladá z kvádrov rôznych veľkostí spojených do jedného celku. Kvádre sa používajú z jednoduchého dôvodu. Pre fyzikálne jadro je veľmi ľahké pracovať s kvádrami hlavne kvôli jednoduchosti výpočtu ťažiska a ostatných fyzikálnych veličín, ako aj na testovanie možnosti kolízie a interakcie s ostatnými objektmi v prostredí. Fyzikálne jadro viaže všetky fyzikálne veličiny na ťažisko daného objektu. Pri použití objektu s jediným komplexným tvarom fyzikálne jadro nedokáže správne určiť jeho ťažisko a preto s ním nedokáže ďalej pracovať. Okrem ťažiska objektu je dôležité určiť aj hmotnosť objektu a rovnako aj vlastnosti materiálu, z ktorého je objekt zhotovený. Medzi tieto vlastnosti patrí napríklad trenie. Za regiónom *State* nasleduje konštruktor pre novú triedu a za ním metódy, ktoré sú upravenými verziami metód z triedy *VisualEntity*. Prvá takáto metóda je metóda *Initialize*. Parameter override značí, že daná metóda je už definovaná v triede *VisualEntity*, z ktorej je odvodená a zároveň umožňuje zmeniť kód programu vo vnútri metódy. *Initialize* obsahuje príkazy, ktoré sa spustia iba raz pri prvom vytváraní daného vizuálneho objektu. V metóde *Initialize* je volaná aj funkcia *Programmatically-BuildModel*.

4.3 Funkcia ProgrammaticallyBuildModel

Funkcia *ProgrammaticallyBuildModel* slúži na vytvorenie vlastného fyzikálneho modelu robota. V tejto funkcii sa spájajú všetky objekty, z ktorých sa model robota skladá. Do tejto časti sa nepridávajú žiadne senzory ani podobné zariadenia, ale len podvozok. Najskôr môže užívateľ začať s definíciou podporného kolieska. Hlavné parametre, ktoré

sa pre podporné koliesko dajú nastaviť sú polomer, poloha a materiál:

```
if (_casterWheelShape == null)
{
// add caster wheel as a basic sphere shape
CasterWheelShape = new SphereShape(new SphereShapeProperties("rear wheel", 0.001f, new Pose(CASTER_WHEEL_POSITION), CASTER_WHEEL_RADIUS));
CasterWheelShape.State.Name = "Caster wheel";
// a fixed caster wheel has high friction when moving laterely, but low friction when
it moves along the
// body axis its aligned with. We use anisotropic friction to model this
CasterWheelShape.State.Material = new MaterialProperties("small friction with anisotropy", 0.5f, 0.5f, 1);
CasterWheelShape.State.Material.Advanced = new MaterialAdvancedProperties();
CasterWheelShape.State.Material.Advanced.AnisotropicDynamicFriction = 0.3f;
CasterWheelShape.State.Material.Advanced.AnisotropicStaticFriction = 0.4f;
CasterWheelShape.State.Material.Advanced.AnisotropyDirection = new Vector3(0, 0, 1);
}
```

Zadávaním názvov premenných namiesto čísiel užívateľ zabezpečí určitú modularitu modelu. Okrem toho treba ošetriť aj možnosť, keby sa mal daný objekt, v tomto prípade podporné koliesko, vytvoriť druhý krát, čo nie je možné. Materiál, z ktorého je podporné koliesko má malé trenie a pôsobí len v smere jednej osi, v tomto prípade v smere pohybu robota. Príkazom:

```
base.State.PhysicsPrimitives.Add(_casterWheelShape)
```

sa pridá podporné koliesko k fyzikálnemu modelu. Ďalšími podobnými príkazmi sa dá poskladať telo robota. Následne tieto časti robota je nutné vložiť do fyzikálneho jadra pomocou príkazu:

```
base.CreateAndInsertPhysicsEntity(physicsEngine);
```

Okrem samotného tela robota a podporného kolieska je potrebné nastaviť parametre dvoch hlavných kolies. Tieto kolesá sú objekty, ktoré sú priamo zakomponované vo fyzikálnom jadre a majú trochu odlišné správanie ako ostatné objekty. Pri ich vytváraní užívateľ musí ošetriť prípad, ak by už kolesá boli náhodou vytvorené. Môže sa tak stať napríklad v prípade načítavania modelu robota z XML súboru. Nasledujúca časť programu je len pre ľavé koleso:

```
// if we were created from xml the wheel entities would already be instantiated
if (_leftWheel != null && _rightWheel != null)
return;
// front left wheel
WheelShapeProperties w = new WheelShapeProperties("front left wheel", FRONT_WHEEL_MASS, FRONT_WHEEL_RADIUS);
// Set this flag on both wheels if you want to use axle speed instead of torque
w.Flags |= WheelShapeBehavior.OverrideAxleSpeed;
w.InnerRadius = 0.7f * w.Radius;
w.LocalPose = new Pose(LEFT_FRONT_WHEEL_POSITION);
_leftWheel = new WheelEntity(w);
_leftWheel.State.Name = State.Name + ":" + "Left wheel";
_leftWheel.State.Assets.Mesh = _wheelMesh;
_leftWheel.Parent = this;
//_leftWheel.WheelShape.WheelState.Material =
new MaterialProperties("wheel", 0.5f, 0f, 1f);
// wheels must have zero friction material.The wheel model will do friction
differently
```

Kolesá netreba vkladať do fyzikálneho jadra. Pre každé koleso však užívateľ musí nastaviť rodiča (Parent).

4.4 Metóda Dispose

Metóda *Dispose* slúži na uvoľnenie prostriedkov po ukončení simulácie. V prípade ukážkového modelu sa jedná len o hlavné kolesá.

4.5 Metóda Update

Metóda *Update* je najdôležitejšou súčasťou modelu. Táto metóda je pravidelne volaná pri každom novom prekreslení simulácie. Z toho vyplýva, že presnosť celej simulácie závisí od počtu snímkov za sekundu, ktoré je schopný počítač zobraziť. Pre každý snímok fyzikálne jadro prepočítava všetky veličiny vzhľadom na ubehnutý čas od posledného snímku. Z toho vyplýva, že celé správanie sa modelu je prepočítavané pre časové intervaly, ktoré vznikajú medzi jednotlivými snímkami. Fyzikálne jadro má určité obmedzenia a nedokáže správne pracovať pri veľkých časových intervaloch. To znamená, že ak počítač dokáže spracovať len jeden snímok za sekundu, potom celá simulácia bude nepoužiteľná. Za veľký časový interval možno považovať už aj časový úsek v trvaní 0,1s. Aby však nevznikali takéto problémy, je možné v simulačnom editore nastaviť fixný časový krok. Tým sa zabezpečí to, že všetky snímky budú generované v rovnakých časových úsekoch, ale celá simulácia nebude prebiehať v reálnom čase a môže byť spomalená. Neznamená to však, že robot bude tiež reagovať spomalene. Ak je napríklad fixný časový krok 0,001s a počítač je schopný vykresliť 100 snímkov za sekundu, potom simuláciu, ktorá by trvala 1s bude vykresľovať 10s.

V metóde *Update* užívateľ zadefinuje správanie sa podvozku. Fyzikálne jadro sa síce postará o interakciu kolies a tela robota s prostredím, ale to, ako rýchlosťou sa budú kolesá točiť pre každý snímok musí zadefinovať užívateľ. Parameter, ktorým užívateľ určuje rýchlosť kolies má názov *Wheel.AxleSpeed*. Tento parameter predstavuje priamo uhlovú rýchlosť, ktorou sa koleso v danom okamihu otáča. Ukážka vzorového programu *Entities.cs*:

```
float left = _leftWheel.Wheel.AxleSpeed + _leftTargetVelocity;
float right = _rightWheel.Wheel.AxleSpeed + _rightTargetVelocity;

if (Math.Abs(left) > 0.1)
{
if (left > 0) _leftWheel.Wheel.AxleSpeed -= SPEED_DELTA;
else _leftWheel.Wheel.AxleSpeed += SPEED_DELTA;
}
if (Math.Abs(right) > 0.1)
{
if (right > 0) _rightWheel.Wheel.AxleSpeed -= SPEED_DELTA;
else _rightWheel.Wheel.AxleSpeed += SPEED_DELTA;
}

// update entities in fields
_leftWheel.Update(update);
_rightWheel.Update(update);
```

Okrem toho sú v danej triede zadefinované ešte tieto premenné:

```
const float SPEED_DELTA = 0.5f;
float _leftTargetVelocity;
float _rightTargetVelocity;
```

Sú to zadané požadované rýchlosti pre ľavé koleso (*_leftTargetVelocity*) a pre pravé koleso (*_rightTargetVelocity*). Tieto požadované rýchlosti sa zadá-

vajú pomocou funkcie *SetAxleVelocity*. Pri pohybe robota vpred sú rýchlosti kolies kladné, pri pohybe vzad záporné. Ďalej sú tu pomocné premenné *left* a *right*, v ktorých sú uložené rozdiely aktuálnej uhlovej rýchlosti daného kolesa a požadovanej rýchlosti. Znamienko "+" v tomto prípade vzniklo z chybnej reprezentácie modelu robota, kde pri pohybe robota vpred sú uhlové rýchlosti obidvoch kolies záporné. Podľa hodnoty týchto pomocných premenných sa následne určuje, či má koleso zrýchliť alebo spomaliť. Treba si však všimnúť, že uhlová rýchlosť kolies vzrastá a klesá vždy lineárne a navyše s pevne definovaným krokom SPEED_DELTA. Tento spôsob reprezentácie pohonov nemožno považovať za dostatočný. Je možné nahradiť lineárne vzťahy odozvou systému prvého rádu.

```
const float T = 0.5f;
double Ts = update.ElapsedTime;
float left = _leftWheel.Wheel.AxleSpeed;
if (Math.Abs(_leftTargetVelocity) >= 0.01)
{
_leftWheel.Wheel.AxleSpeed = (left + (_leftTargetVelocity - left) * ((float)Ts / T));
}
else
{
_leftTargetVelocity = 0;
if (Math.Abs(_leftWheel.Wheel.AxleSpeed) < 0.01)
_leftWheel.Wheel.AxleSpeed = 0;
}
```

Daná časť programu je len pre pohon ľavého kolesa. Premenná *T* predstavuje časovú konštantu systému prvého rádu. Premenná *Ts* predstavuje čas ubehnutý od vytvorenia posledného snímku. Ak nie je v simulácii nastavený presne definovaný krok, potom by jej nahradenie za konštantu mohlo spôsobiť rôzne chyby, pretože čas medzi jednotlivými snímkami je premenlivý a záleží od aktuálneho stavu prostredia. Pomocná premenná *left* tentoraz uchováva hodnotu uhlovej rýchlosti kolesa z posledného snímku. Ak požadovaná hodnota uhlovej rýchlosti *_leftTargetVelocity* nie je dostatočne veľká, potom ju môžeme považovať za nulovú, robot nebude vykonávať žiaden pohyb. Rovnako ak by uhlová rýchlosť kolesa mala byť menšia ako definovaná hodnota, môžeme ju považovať za nulovú. Nasledujúcim príkazom užívateľ odošle hodnotu novej uhlovej rýchlosti fyzikálnemu jadru na spracovanie.

```
// update entities in fields

_leftWheel.Update(update);
```

Metóda *Update* môže okrem iného obsahovať rôzne iné funkcie. Ide hlavne o príkazy pre funkcie *DriveDistance* a *RotateDegrees*, ktoré slúžia na definovanie pohybu robota o požadovanú vzdialenosť alebo otočenie robota o požadovaný uhol. Obe tieto funkcie však pre svoju činnosť využívajú presnú pozíciu a orientáciu robota priamo zo simulačného prostredia a súčasne sa spoliehajú na spomínaný model pohonov s konštantným prírastkom rýchlosti.

4.6 Región Motor Base Control

Región *Motor Base Control* obsahuje funkcie, ktoré spracovávajú požiadavky na činnosť pohonov. Zahrnuté sú tu funkcie *SetVelocity* a *SetMotorTorque*. Okrem týchto funkcií obsahuje konštanty a funkcie, ktoré obmedzujú maximálne možné rýchlosti, aké je robot schopný vyvinúť:

```
const float MAX_VELOCITY = 20.0f;
const float MIN_VELOCITY = -MAX_VELOCITY;

float ValidateWheelVelocity(float value)
{
if (value > MAX_VELOCITY)
return MAX_VELOCITY;
if (value < MIN_VELOCITY)
return MIN_VELOCITY;
return value;
}
```

Niektoré funkcie z pôvodného vzorového programu však treba upraviť, aby nebolo možné obísť zadané obmedzenia:

```
public void SetMotorTorque(float leftWheel, float rightWheel)
{
if (leftWheel * _motorTorqueScaling > MAX_VELOCITY)
   leftWheel = MAX_VELOCITY / _motorTorqueScaling;
if (leftWheel * _motorTorqueScaling < MIN_VELOCITY)
    leftWheel = MIN_VELOCITY / _motorTorqueScaling;
if (rightWheel * _motorTorqueScaling > MAX_VELOCITY)
    rightWheel = MAX_VELOCITY / _motorTorqueScaling;
if (rightWheel * _motorTorqueScaling < MIN_VELOCITY)
    rightWheel = MIN_VELOCITY / _motorTorqueScaling;
SetAxleVelocity(leftWheel * _motorTorqueScaling, rightWheel *_motorTorqueScaling);
}
```

Funkcia *SetMotorTorque* nastavuje rýchlosť na základe požadovaného výkonu. Vstupné parametre by mali byť z intervalu <0;1> pričom motor ide na maximum pri hodnote 1. Ďalšou podobnou funkciou je funkcia *SetVelocity*, ktorá nastavuje požadovanú uhlovú rýchlosť na základe požadovanej obvodovej rýchlosti kolesa:

```
public void SetVelocity(float left, float right)
{
left = ValidateWheelVelocity(left);
right = ValidateWheelVelocity(right);
// v is in m/sec - convert to an axle speed
// 2Pi(V/2PiR) = V/R
SetAxleVelocity(left / _leftWheel.Wheel.State.Radius,
right / _rightWheel.Wheel.State.Radius);
}
```

Obe uvedené funkcie volajú funkciu *SetAxleVelocity*. Až táto funkcia nastavuje požadované uhlové rýchlosti *leftTargetVelocity* a *rightTargetVelocity* pre jednotlivé kolesá:

```
private void SetAxleVelocity(float left, float right)
{
if (_leftWheel == null || _rightWheel == null)
   return;
if (left > 0)
{
   _leftTargetVelocity = (float)(2 * Math.PI);//left;
   _rightTargetVelocity = (float)(2 * Math.PI); //right;
}
else _leftTargetVelocity = _rightTargetVelocity = 0;
}
```

5 Vytvorenie celkového modelu robota

Vzorový príklad sa dá opäť nájsť v programe *Entities.cs* v regióne *Platforms*. Tento program je však len pomôcka a užívateľský model treba upraviť podľa triedy *SimulatedCustomDifferentialDriveEntity*. Model robota bude predstavovať trieda odvodená od triedy *SimulatedCustomDifferentialDriveEntitiy*. Na začiatku tejto triedy sa nachádza štandardný konštruktor. Za ním konštruktor, ktorý obsahuje parameter pre počiatočnú polohu robota. Tento konštruktor obsahuje konkrétne parametre pre robot vytvorený užívateľom. Najskôr užívateľ definuje konkrétne číselné hodnoty pre premenné, ktoré boli definované už v triede *SimulatedCustomDifferentialDriveEntity*. Ide hlavne o rozmery a hmotnosti jednotlivých častí robota. Tieto hodnoty treba zadať podľa skutočného stavu. Rozmery by mali byť rovnaké ako rozmery už vytvoreného grafického (vizuálneho) modelu robota. Po zadaní rozmerov nasleduje časť, kde sa vytvárajú konkrétne časti robota. Najprv však užívateľ nastaví názov, hmotnosť a počiatočnú polohu modelu:

```
base.State.Name = "MotorBaseWithThreeWheels";

base.State.MassDensity.Mass = MASS;

base.State.Pose.Position = initialPos;
```

Za týmito jednoduchými príkazmi nasledujú ďalšie, veľmi dôležité príkazy, ktoré vytvárajú geometriu fyzikálneho modelu. Treba si uvedomiť, že pri vytváraní nasledujúcich objektov je veľmi dôležitý referenčný bod, na ktorý bude vztiahnutá celá geometria. Pozícia a orientácia modelu robota sa vzťahuje na priesečník osí súradnicového systému robota. Preto je vhodné, aby stred úsečky spájajúcej stredy hlavných kolies robota ležal nad týmto bodom. Netreba zabudnúť, že súradnicový systém v simulačnom prostredí je pravotočivý súradnicový systém *ZXY*. Vlastnosti objektu predstavujúceho telo robota možno nastaviť nasledujúcim spôsobom:

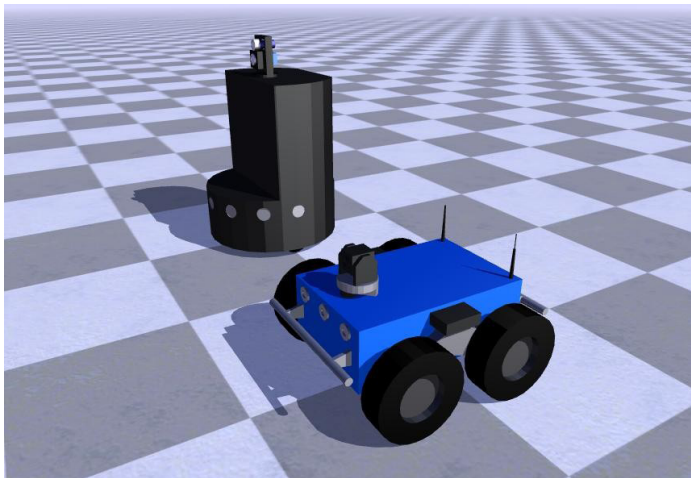
```
// pozícia tela robota

BoxShapeProperties motorBaseDesc = new BoxShapeProperties("telo", MASS, new Pose(new Vector3(0, CHASSIS_CLEARANCE + CHASSIS_DIMENSIONS.Y / 2, 0)), CHASSIS_DIMENSIONS);
```

V tomto prípade je nastavený názov, hmotnosť a pozícia tela robota. Uložené sú v premennej *motorBaseDesc.* Telo robota je v tomto prípade predstavované kvádrom. Všeobecne pre celé simulačné prostredie platí, že poloha kvádra nie je definovaná ako pozícia jedného z jeho vrcholov, ale ako pozícia stredu daného kvádra. Pre robot teda platí, že stred jeho tela sa nachádza nad priesečníkom súradnicových osí systému robota – súradnice X a Z sú nulové. Keďže telo v podobe kvádra má mať výšku, ktorú definuje premenná *CHASSIS_DIMENSIONS* a zároveň má byť v určitej výške nad zemou, potom ypsilonová súradnica stredu kvádra musí byť rovná súčtu premenných *CHASSIS_CLEARANCE* a polovičnej výške kvádra (*CHASSIS_DIMENSIONS.Y / 2*). Ak užívateľ chce používať robot s komplexnejšou geometriou a telo robota modeluje viacerými kvádrami naskladanými na seba, potom pre každý tento kváder musí zadať tieto rozmery. Okrem týchto vlastností musí užívateľ ešte zadať vlastnosti materiálové:

```
motorBaseDesc.Material = new MaterialProperties("high friction", 0.0f, 1.0f, 20.0f);

MeshRotation = new Vector3(0, -90, 0);

MeshTranslation = new Vector3(0, 0.005f, 0);

ChassisShape = new BoxShape(motorBaseDesc);
```

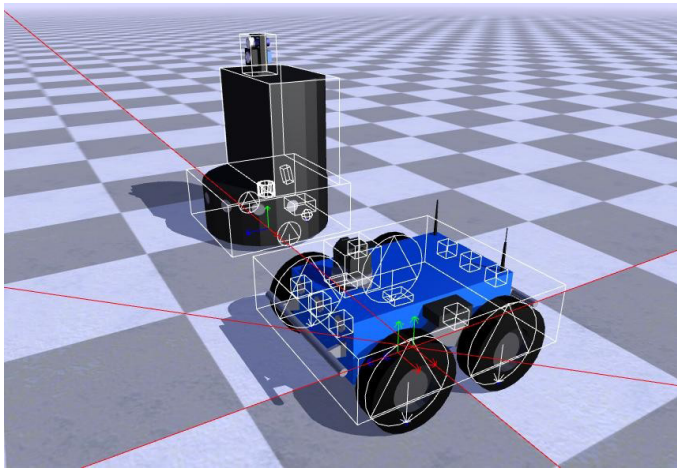
Pri materiálových vlastnostiach sa zadávajú koeficienty dynamického a statického trenia. Okrem týchto materiálových vlastností ešte možno nastaviť pozíciu a orientáciu vizuálneho modelu v prípade, že by sa jeho pozícia alebo orientácia nezhodovala s fyzikálnym modelom. Všetky doposiaľ zadané vlastnosti a parametre užívateľ vloží do premennej *ChassisShape*, ktorá je definovaná už v triede *SimulatedCustomDifferentialDriveEntitiesEntity* a predstavuje telo robota. Poslednou vecou, ktorú je nutné vyriešiť, je zadefinovanie presnej pozície a rozmerov kolies robota:

```
// rear wheel is also called the caster
CASTER_WHEEL_POSITION = new Vector3(0, // center of chassis CASTER_WHEEL_RADIUS, // distance from ground
CHASSIS_DIMENSIONS.Z / 2 - CASTER_WHEEL_RADIUS); // all the way at the back of the robot
// NOTE: right/left is from the perspective of the robot, looking forward
FRONT_WHEEL_MASS = 0.10f;
RIGHT_FRONT_WHEEL_POSITION = new Vector3(
CHASSIS_DIMENSIONS.X / 2 + 0.01f,// left of center
FRONT_WHEEL_RADIUS,// distance from ground of axle
FRONT_AXLE_DEPTH_OFFSET); // distance from center, on the z-axis
LEFT_FRONT_WHEEL_POSITION = new Vector3(
-CHASSIS_DIMENSIONS.X / 2 - 0.01f,// right of center
FRONT_WHEEL_RADIUS,// distance from ground of axle
FRONT_AXLE_DEPTH_OFFSET); // distance from center, on the z-axis
```



Obr.1 Importovanie vizuálnych (grafických) modelov do prostredia Microsoft Robotics Developer Studio

Fig.1 Visual models importing into Microsoft Robotics Developer Studio environmnet



Obr.2 Modely podvozkov v Microsoft Robotics Developer Studio

Fig.2 Chassis models in Microsoft Robotics Developer Studio

Pri súradnicovom systéme ZXY je pri nulovom otočení okolo osi Y robot natočený v smere kladnej osi Z. Ak sa pritom stred robota nachádza nad stredom súradnicového systému, potom ľavé koleso bude mať zložku X kladnú a pravé koleso bude mať zložku X zápornú. Ako si možno všimnúť z uvedeného programu, kolesá sú medzi sebou vymenené. Okrem zápornej uhlovej rýchlosti kolies, vymenenie kolies vlastne spôsobilo otočenie celého robota o 180 stupňov a to znamená, že orientácia robota nie je v súlade so skutočnou orientáciou, s ktorou pracuje fyzikálne jadro. Preto, ak by užívateľ chcel riešiť otázku autonómnej navigácie napríklad podľa nejakej mapy a pri výpočte odchýlok orientácie robota a uhla, pod ktorým vidí cieľový bod, vznikne chyba ak tento uhol počíta pomocou trigonometrických funkcií.

Zadefinovaním rozmerov a pozície kolies je ukončená tvorba fyzikálneho modelu robota. Tento model je už pripravený na použite v simulácii, pričom jednotlivé snímače sa k modelu pridajú až pri vytváraní simulovaného prostredia.

Záver

Microsoft Robotics Developer Studio je nástroj, ktorý slúži na simulovanie mobilných robotov. Základným predpokladom na vytvorenie takejto simulácie je vytvorenie vlastného modelu mobilného robota. Vytvorenie modelu robota bez niektorých skúseností nemusí byť jednoduché. Ak však takýto model dokážeme vytvoriť, v konečnom dôsledku môžeme vytvárať a simulovať inteligentné správanie robotov (napr. bezkolízny prechod prostredím), čo v neposlednom rade vedie na energetické šetrenie, paralelný vývoj alebo zamedzeniu mechanických poškodení reálnych robotických systémov.

Poďakovanie

Článok vznikol v rámci riešenia projektu VEGA 1/0690/09.

Literatúra

[1] AGEIA Technologies: Advanced Gaming Physics Defining The New Reality In PC Hardware, March 2006, http://www.ageia.com/pdf/wp_advanced_gaming_physics.pdf

[2] MRS User Guide, http://msdn2.microsoft.com/en-us/library/bb881626.aspx

[3] Microsoft Robotics Studio Now Available to Provide Common Development Platform, http://www.robocup-us.org/Media/2006-MSRS-release.pdf

[4] Microsoft Robotics Studio Community, http://forums.microsoft.com/MSDN/default.aspx?ForumGroupID=383&SiteID=1

[5] DUCHOŇ F., ŠTRENGER M., Vývojové prostredie Microsoft Robotics Studio, AT&P Journal Plus 1, 2008

[6] JOHNS K., TAYLOR T., Professional Microsoft Robotics Studio (Wrox Programmer To Programmer), Wiley Publishing, Inc., ISBN: 9780470141076

Abstract

For creating an inteligent system in simulation environemnt , it is necessary to form such a model of that system, which maximally corensponds with real system in real environment. Microsoft Robotics Developer Studio is exactly like this simulation environment. Besides programming of inteligence of robotics system, it provides to create own models of robotics systems. In that there is only few specialized literature about this studio and to work with this studio is required some experience and knowledge, this paper is about creating of physical model of own differentially driven mobile robotics system in Microsoft Robotics Developer Studio.

prof. Ing. Ladislav Jurišica, PhD.,
Ing. František Duchoň,
Bc. Martin Štrenger

Fakulta elektrotechniky a informatiky
Ústav riadenia a priemyselnej informatiky
Ilkovičova 3
812 19 Bratislava
ladislav.jurisica@stuba.sk
frantisek.duchon@stuba.sk
mastr@ynet.sk

Vytvorenie vlastných snímačov, prostredia, robota a simulácia v Microsoft Robotics Developer Studio

František Duchoň, Ladislav Jurišica, Martin Štrenger

Abstrakt
Pre komplexné vytvorenie simulačného prostredia na implementáciu inteligentých algoritmov riadenia mobilných robotických systémov v Microsoft Robotics Developer Studio je potrebné vytvoriť vlastný model robota. Samotný model robota by bol nepoužiteľný, preto je nutné vytvoriť aj prostredie, v ktorom sa algoritmy budú testovať, ale aj snímače, ktoré bude robot používať ako spätnú väzbu z tohto prostredia. Tento článok sa zaoberá vytvorením vlastného snímača v prostredí Microsoft Robotics Developer Studio a vytvorením vlastného prostredia, v ktorom má mobilný robotický systém inteligentne vykonávať svoju činnosť.

Kľúčové slová: Microsoft Robotics Developer Studio, snímač, vytvorenie vlastného snímača, simulácia, vytvorenie prostredia, Floorplan

Úvod

Pre implementovanie inteligentných spôsobov správania sa mobilných robotov v prostredí Microsoft Robotics Developer Studio (MRDS) je nutné vyriešiť aj otázku vytvorenia tohto prostredia ako aj otázku spätnej väzby z tohto prostredia, teda vytvorenie snímačov pre daný model mobilného robota. MRDS už síce obsahuje niekoľko typov snímačov, ale jedná sa väčšinou o snímače s pevne zadefinovanými parametrami čo môže užívateľa vo veľkej miere obmedzovať.

1 Dostupné snímače

Microsoft Robotics Developer Studio podporuje veľké množstvo snímačov. Z týchto snímačov však drvivá väčšina pre svoje fungovanie potrebuje aj daný hardvér, preto napríklad nie je možné v simulácii použiť snímače GPS alebo sonary. Snímače používané v simulácii musia byť schopné interakcie so simulovaným prostredím a preto musia byť riešené softvérovo.

1.1 Snímač kolízie

Medzi najjednoduchšie využívané snímače patria snímače kontaktu. Údaje z týchto snímačov nadobúdajú len dve hodnoty, a preto je ich vyhodnotenie nenáročné. V simulačnom prostredí MRDS sú takéto snímače predstavované jednoduchými útvarmi, najčastejšie kvádrami, ktoré majú aktívny parameter kontrolujúci kontakt kvádra s ostatnými objektmi v simulácii:

```
EnableContactNotifications = true;
```

Takýto simulovaný snímač sa v MRDS nazýva *SimulatedBumper* a jeho zdrojový kód je možné nájsť v adresári *Samples\Simulation\Sensors\Bumper* a v súbore *Entities.cs* v regióne *Sensors*. Takéto snímače však hrajú pri navigácii robota len okrajovú úlohu, pretože pri inteligentnej navigácii treba využívať oveľa sofistikovanejšie senzory.

1.2 Kamera

V dnešnej dobe sa čoraz viac v mobilnej robotike využívajú kamery ako snímače. Používajú sa hlavne vďaka ich dostupnosti a neustále klesajúcej cene. Navyše sa jedná o zariadenie so širokým využitím. Jedným z najlepšie rozvinutých algoritmov videnia je algoritmus na sledovanie farebnej škvrny. V reálnom svete, aby mohol byť takýto systém použitý pre navádzanie, musí programátor zvoliť príslušnú farbu a naučiť algoritmus rozpoznávať túto farbu. Kamera potom môže dávať na výstupe pozíciu danej farby v jej zornom poli. Kamera ako snímač má však isté obmedzenia. Má obmedzený počet snímkov za sekundu, a preto nemusí byť schopná zaznamenať a následne nasledovať rýchlo sa pohybujúci objekt danej farby. Citlivosť kamery na farbu je ovplyvnená svetelnými podmienkami. Pri prechode zo svetlého prostredia do tmavého alebo naopak môže stratiť sledovaný objekt. V neposlednom rade je nutné zvoliť jednoznačnú farbu. Podobné farby môžu predstavovať pre robot jednu a tú istú. Microsoft Robotics Developer Studio umožňuje využívať rovnaké algoritmy ako sa používajú pri reálnych kamerách. Obraz z reálnej kamery sa spracováva snímok po snímku, pričom každý snímok predstavuje jeden obrázok. Rovnako MRDS umožňuje získať a spracovávať obrázky zo simulovanej kamery, ktorá je dostupná v MRDS. Simulovaná kamera predstavuje len akýsi ďalší pohľad na simulované prostredie. Okrem toho simulovaná kamera umožňuje nastaviť užívateľovi rôzne parametre ako sú napríklad rozlíšenie a obnovovacia frekvencia. Nevýhodou používania simulovaných kamier je však vyššia náročnosť na výpočtové prostriedky pri použití viacerých takýchto kamier, pretože grafické jadro musí prepočítavať pohľad pre každú kameru osobitne. Simulovaná kamera môže plniť rôzne úlohy. Okrem spomenutého algoritmu pre sledovanie farebnej škvrny sa môžu aplikovať algoritmy na rozoznávanie tvarov, sledovanie iných objektov v simulácii a podobne. Okrem toho zo simulovanej kamery možno vytvoriť úplne nový typ snímača. Ako príklad uvedieme jednoduchý snímač farby. Pri tomto snímači nastavíme nízke rozlíšenie

kamery a nižšiu frekvenciu obnovovania. Následne pomocou prahovania a aplikácie rôznych filtrov dostaneme ako výstup prevládajúcu farbu v zosnímanom obraze.

1.3 Laserový snímač vzdialenosti

Laserové snímače vzdialenosti patria medzi optické snímače, ktoré merajú vlastnosti odrazeného svetla a na základe získaných informácií určia vzdialenosť alebo iné vlastnosti objektu, od ktorého sa svetlo odráža. Väčšina laserových snímačov pracuje len v jednej rovine, pričom laserové lúče sa usmerňujú pomocou rotujúceho zrkadla a dokážu pokryť niekedy aj viac ako 180° uhol. Najväčšou výhodou takýchto systémov je, že môžu zaznamenať pasívne objekty v širokom zornom poli a zároveň v dostatočnej vzdialenosti, najčastejšie okolo 10 metrov. Ak sú navyše objekty z reflexného materiálu, môžu ich zaznamenať aj do vzdialenosti 100 metrov. Rýchlosť snímania je v rozmedzí 1 až 100 Hz. Namerané údaje sú však závislé od rýchlosti pohybu mobilného robota s takýmto snímačom. Vysoké rozlíšenie sa dosahuje pri frekvenciách okolo 3 až 10 Hz. Pri takýchto frekvenciách je však nutné kompenzovať pohyb systému a preto tento typ snímača nie vhodný pre rýchle roboty. Medzi nevýhody laserových snímačov patrí najmä ich cena, rozmery, hmotnosť, vysoká spotreba energie a zložitosť mechanického prevedenia. Za nevýhodu možno považovať aj znižovanie rozlíšenia pri zvyšovaní vzdialenosti v dôsledku rozbiehania vysielaných lúčov.

MRDS dokáže pracovať aj s reálnym snímačom, Okrem toho v MRDS existuje rovnaký snímač určený pre simulačné prostredie. Preto môže každý užívateľ vytvárať a testovať svoje algoritmy navrhnuté pre navigáciu pomocou takéhoto typu snímačov. Laserový snímač určený pre simuláciu je opäť vytvorený programovo. To znamená, že tento snímač je tiež iba simuláciou reálneho snímača. Pracuje však na rovnakom princípe ako reálny snímač. Do simulovaného prostredia sú virtuálne vysielané „laserové“ lúče a pre každý lúč sa testuje jeho dopad na prekážku. Pre každý takýto laserový lúč musí prebehnúť výpočet jeho dráhy počínajúc v nulovej vzdialenosti od snímača. Postupne sa táto vzdialenosť zväčšuje až kým lúč nenarazí na prekážku alebo sa nedostane na obmedzenie dosahu lúča. Ako vidno, pre každý lúč sa musí vykonať veľké množstvo testov. Okrem toho výpočet musí prebehnúť pre všetky lúče a navyše niekoľko krát za sekundu. Použitie takéhoto snímača opäť zvýši výpočtovú náročnosť simulácie. Simulovaný laserový snímač, ktorý MRDS štandardne obsahuje, má napevno zadefinované parametre. Tento simulovaný snímač je orientovaný na robote vždy dopredu. Nie je teda možnosť použiť viacero takýchto snímačov otočných o určitý uhol. Okrem toho je fixne zadefinované uhlové rozlíšenie 0,5°, zorné pole snímača 180°, obnovovacia frekvencia 10Hz a maximálny dosah 8m. Z toho vyplýva, že výpočet musí prebehnúť pre 361 lúčov 10 krát za sekundu na veľkú vzdialenosť.

2 Upravenie laserového snímača vzdialenosti

Microsoft Robotics Developer Studio už obsahuje zdrojový kód pre simulovaný laserový snímač vzdialenosti. Tento kód je možné použiť ako podklad na vytvorenie vlastného laserového snímača vzdialenosti, ktorému bude možné meniť parametre a zároveň to umožní pridať na robot niekoľko nezávislých laserových snímačov naraz. Vzorový kód sa nachádza v adresári *Samples\Simulation\Sensors\ LaserRangeFinder* a v súbore *Entities.cs*. Tento kód však neobsahuje časť pre výpočet kontaktu lúča s prekážkou. Obsahuje len časť pre zadefinovanie parametrov snímača a jeho vytvorenie. Dôležitý je potom hlavne súbor *SimulatedLRF.cs* a súbor *Entities.cs*. V súbore *SimulatedLRF.cs* sa nachádza časť kde sa definujú parametre simulovaného snímača:

```
const float LASER_RANGE = 8f;

void CreateDefaultState()

{
        _state.Units = sicklrf.Units.Millimeters;
        _state.AngularRange = 180;
        _state.AngularResolution = 0.5f;
}
```

V súbore *Entities.cs* v regióne *Sensors* sa nachádza trieda *LaserRangeFinderEntity*. V tejto triede sa nachádza konštruktor a metóda *Update*.

Parametre snímača sú fixne zadefinované. Tieto parametre sa nedajú pre tento snímač zmeniť a preto je nutné si vytvoriť vlastný snímač, ktorý bude obsahovať konštruktor zabezpečujúci vytvorenie snímača so zadanými parametrami. Najprv je potrebné vytvoriť vlastnú vizuálnu entitu laserového snímača, ktorá bude obsahovať funkcie pre zadanie parametrov. Následne sa do programu doplnia príkazy, ktoré zabezpečia, že pri vytváraní snímača budú použité zadané parametre.

V projekte vlastného snímača je potrebné vytvoriť novú triedu, v tomto prípade nazvanú ***SimulatedLaserRangeFinderEntities***. Do nej treba skopírovať telo triedy *LaserRangeFinderEntity* zo súboru *Entities.cs.* Nasleduje krok vytvorenia si vlastných premenných, do ktorých sa uloží hodnoty zadaných parametrov:

```
float _angularResolution;
/// <summary>
/// Angular resolution of laser sensor
/// </summary>
[DataMember]
[Description("The angular resolution of the laser rangefinder.")]
public float AngularResolution
{
        get { return _angularResolution; }
        set { _angularResolution = value; }
}
int _angularRange;
/// <summary>
/// Angular range of laser sensor
/// </summary>
[DataMember]
[Description("The angular range of the laser rangefinder.")]
public int AngularRange
{
        get { return _angularRange; }
        set { _angularRange = value; }
}
float _Range;
/// <summary>
```

```
/// Maximum range of laser sensor
/// </summary>
[DataMember]
[Description("Maximum range of the laser rangefinder.")]
public float Range
{
        get { return _Range; }
        set { _Range = value; }
}
```

Následne treba vytvoriť konštruktor, ktorý uloží hodnoty parametrov do vlastných premenných:

```
/// <summary>
/// Default constructor
/// </summary>
public CustomLaserRangeFinderEntity() { }
/// <summary>
/// Initialization constructor
/// </summary>
/// <param name="localPose"></param>tný
public CustomLaserRangeFinderEntity(Pose localPose, float refreshRate,
float angularResolution, int angularRange, float range)
{
        // create a new instance of the laser pose so we dont re-use the raycast reference
        // That reference will be updated regularly
        BoxShapeProperties box = new BoxShapeProperties("SickLRF", 0.5f, localPose, new Vector3(0.05f, 0.05f, 0.05f));
        _laserBox = new BoxShape(box);
        State.Assets.Effect = "LaserRangeFinder.fx";
        SCAN_INTERVAL = refreshRate;
        AngularResolution = angularResolution;
        AngularRange = angularRange;
        Range = range;
}
```

V metóde *Update* treba upraviť určitú časť kódu, aby nevznikla chyba z dôvodu, že orientácia pôvodného laserového snímača je prispôsobená pôvodnému modelu robotov:

```
// the LRF looks towards the negative Z axis (towards the user), not the positive Z axis
// which is the default orientation. So we have to rotate its orientation by 180 degrees
_raycastProperties.OriginPose.Orientation = TypeConversion.FromXNA( TypeConversion. ToXNA(State.Pose. Orientation) * xna.Quaternion.CreateFromAxisAngle(new xna.Vector3(0, 1, 0), (float)Math.PI));
```

Llaserový snímač bol teda otočený o 180°. Tento problém vznikol kvôli zlej orientácii modelu robota. Navyše z danej časti programu vidno, že nie je možnosť ovplyvniť orientáciu laserového snímača, pretože orientácia daného laserového snímača sa skladá len z orientácie modelu robota a otočenia o spomínaných 180°. Ak chce užívateľ otočiť snímač o nejaký uhol vzhľadom na robot a keďže snímač je reprezentovaný aj fyzicky vo forme kvádra, natočenie laserového snímača bude zodpovedať natočeniu tohto kvádra voči modelu robota. Upravená časť programu bude vyzerať preto nasledovne:

```
_raycastProperties.OriginPose.Orientation = Parent.State.Pose.Orientation *_laserBox.BoxState.LocalPose.Orientation;
```

Takto si teda užívateľ dokáže vytvoriť vlastný laserový snímač vzdialenosti s názvom *CustomLaserRangeFinder*. Tento snímač má rovnaké vlastnosti ako pôvodný simulovaný laserový snímač, líši sa však možnosťou nastavenia parametrov. Túto vlastnosť je možné využiť pre vytvorenie iných snímačov. Môže ísť napríklad o infračervené snímače alebo ultrazvukové snímače. Tieto typy snímačov nie sú pre prostredie MRDS implicitne vytvorené.

2.1 Vytvorenie iných snímačov zmenou parametrov laserového snímača vzdialenosti

Infračervený snímač možno vytvoriť z laserového snímača nastavením nasledujúcich parametrov. Parameter *Range* nastavuje dosah snímača, parameter *AngularRange* nastavuje uhol snímania a v prípade nastavenia malej hodnoty je zabezpečené snímanie objektov len oproti snímaču. Parameter *AngularResolution* je možné nastaviť podľa potreby. Tento parameter predstavuje uhol medzi susednými lúčmi, vo väčšine prípadov vyhovuje nastavenie 0,5° alebo 1°. Na malé vzdialenosti stačí menší počet lúčov (väčší uhol medzi lúčmi), na väčšie vzdialenosti väčší počet lúčov. Posledným parametrom je *RefreshRate*, ktorým sa nastavuje počet vzoriek za sekundu.

Ultrazvukový snímač sa dá nastaviť podobne. Opäť je teda potrebné len nastaviť parametre pre maximálny dosah pre snímač, uhol v ktorom snímač vysiela lúče ako aj uhol medzi jednotlivými lúčmi a počet vzoriek za sekundu. Tieto typy snímačov však dávajú na výstupe iba jednu hodnotu, pretože snímajú vzdialenosť od najbližšej prekážky. Túto skutočnosť treba zohľadniť v algoritme, kde sa vyhodnocujú získané údaje zo snímača. Takisto v tomto algoritme je potrebné softvérovo nastaviť minimálnu možnú vzdialenosť od ktorej je schopný snímač merať.

Pri používaní viacerých snímačov v simulácii je vhodné nastaviť im príslušné mená. Najmä ak má laserový snímač reprezentovať infračervený alebo ultrazvukový snímač vzdialenosti, pričom na spracovanie údajov chce užívateľ využívať iné algoritmy. Ak má model robota kruhovú podstavu a snímače vzdialenosti umiestnené po obvode robota, všetky tieto snímače je možné umiestniť do stredu robota, nastaviť ich príslušnú orientáciu a v algoritme spracovania údajov odpočítať od nameranej vzdialenosti polomer kruhovej podstavy robota. V simulácii sa kontakt laserového lúča s objektom vyznačuje červenou bodkou.

3 Vytvorenie vlastného inkrementálneho snímača

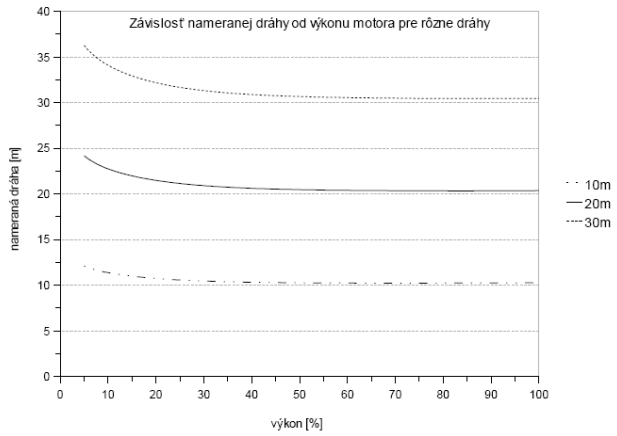
Microsoft Robotics Developer Studio dokáže pracovať s rôznymi robotickými platformami a využívať informácie z rôznych snímačov avšak akákoľvek implementácia inkrementálneho snímača do simulácie tu nie je. Najjednoduchším riešením je vytvoriť takýto snímač priamo v modeli robota, kde má užívateľ prístup k jednotlivým premenným patriacim ku kolesám. Kolesá sú však samostatné objekty a aj keď im je možné nastaviť mnoho parametrov, informácia o otočení kolesa chýba. Prvou možnosťou je teda vytvoriť vlastné koleso, ktoré bude túto možnosť podporovať a

umožní tak vytvoriť akúsi náhradu inkrementálneho snímača. Druhou možnosťou je využitie skutočnosti, že uhlová rýchlosť kolesa sa počíta v metóde *Update* v modeli robota. Keďže poznáme uhlovú rýchlosť a čas, ktorý uplynul od posledného snímku, vieme vypočítať uhol, o ktorý sa koleso pootočilo. Tento istý spôsob sa dá aplikovať aj na vlastné koleso, pretože tak isto obsahuje metódu *Update*. Záleží na používateľovi, ktorú možnosť si vyberie. Model vlastného kolesa má nasledujúci kód:

```
[DataContract]
public class CustomWheelEntity : WheelEntity
{
        float Increment = 0;
        public CustomWheelEntity()
        {
        }
public CustomWheelEntity(WheelShapeProperties wheelShape)
: base(wheelShape)
        { }
public override void
Initialze(Microsoft.Xna.Framework.Graphics.GraphicsDevice device, PhysicsEngine physicsEngine)
{
        base.Initialize(device, physicsEngine);
}
public override void Update(FrameUpdate update)
{
        base.Update(update);
        // update the increment for the next frame
        Increment = (float)(Wheel.AxleSpeed * update.ElapsedTime);
}
public float GetIncrement ()
{
        return Increment;
}
}
```

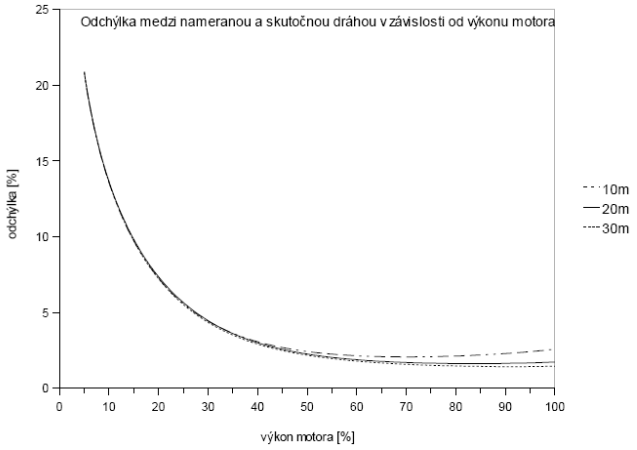
Ak užívateľ používa model vlastného kolesa, musí v programe pre model robota doplniť tento kód na koniec programu (teda do súboru SimulatedCustomDifferentialDriveEntity.cs). Okrem toho treba v modeli robota zmeniť typ kolies na vlastné kolesá.

Vlastný inkrementálny snímač má podobné nepresnosti ako aj reálne snímače (Obr. 1). Veľkosť chyby je závislá od rýchlosti a rovnako aj od dráhy. Čím pomalšie sa robot pohybuje, tým väčšia chyba vzniká. Teoreticky pri veľmi malej rýchlosti by chyba rástla donekonečna. Chyba vzniká v dôsledku zlého modelu simulovaného kolesa. Tento model je priamo implementovaný vo fyzikálnom jadre a ovplyvniť ho je možné len do určitej miery. Závislosť odchýlky nameranej dráhy od skutočne prejdenej dráhy je na Obr.2. Podľa zistenej závislosti by bolo teoreticky možné odchýlku aspoň čiastočne korigovať, lenže odchýlka bola meraná len pre priamočiary pohyb s konštantnou rýchlosťou pohybu.



Obr.1 Závislosť nameranej dráhy od výkonu motora

Fig.1 Relation of measured path from engine performence



Obr.2 Odchýlka medzi nameranou a prejdenou dráhou

Fig.2 Deviation between measured and distance moved

4 Simulácia s vlastným modelom robota

Snímače sa na robot pridávajú až pri simulácii. Ako príklad využijeme existujúci robot z Ústavu riadenia a priemyselnej informatiky. Reálny model obsahuje 9 ultrazvukových snímačov vzdialenosti a rovnaký počet i umiestnenie infračervených snímačov. Keďže infračervené snímače slúžia na zaznamenanie blízkych objektov a ultrazvukové snímače na zaznamenanie vzdialenejších objektov, každý takýto pár snímačov bol spojený do jedného snímača. Okrem toho je robot vybavený kamerou umiestnenou na vrchu robota. Skladanie modelu pozostáva z postupnosti funkcií s parametrami, ktoré pridávajú jednotlivé objekty k modelu robota a následne sa celý model robota vloží do simulácie:

```
void AddURPI_Robot_1(Vector3 position)
{
URPI_Robot_1 robotBaseEntity = CreateMotorBase(ref position);
CustomLaserRangeFinderEntity laser;
for (int i = 0; i < 7; i++)
{
        // Create Laser entity and start simulated laser service
        laser = CreateURPI_Robot_1_Sonar(90 - i * 30, i + 1);
```

```
            // insert laser as child to motor base
            robotBaseEntity.InsertEntity(laser);
}
// Create Laser entity and start simulated laser service
laser = CreateURPI_Robot_1_Sonar(187.5f, 8);
// insert laser as child to motor base
robotBaseEntity.InsertEntity(laser);
// Create Laser entity and start simulated laser service
laser = CreateURPI_Robot_1_Sonar(180 - 7.5f, 9);
// insert laser as child to motor base
robotBaseEntity.InsertEntity(laser);
// create Camera Entity ans start SimulatedWebcam service
CameraEntity camera = CreateURPI_Robot_1_Camera();
// insert as child of motor base
robotBaseEntity.InsertEntity(camera);
// Finally insert the motor base and its two children
// to the simulation
SimulationEng ne.GlobalInstancePort.Insert (robotBaseEnti-
ty);
}
```

Ako vidno z uvedenej časti programu, najskôr bol vytvorený podvozok uložený v premennej *robotBaseEntity* pomocou funkcie *CeateMotorBase*. Pomocou funkcie *CreateURPI_Robot_1_Sonar* sa pridali ultrazvukové snímače. Ako posledná bola pridaná kamera pomocou funkcie *CreateURPI_Robot_1_Camera*.

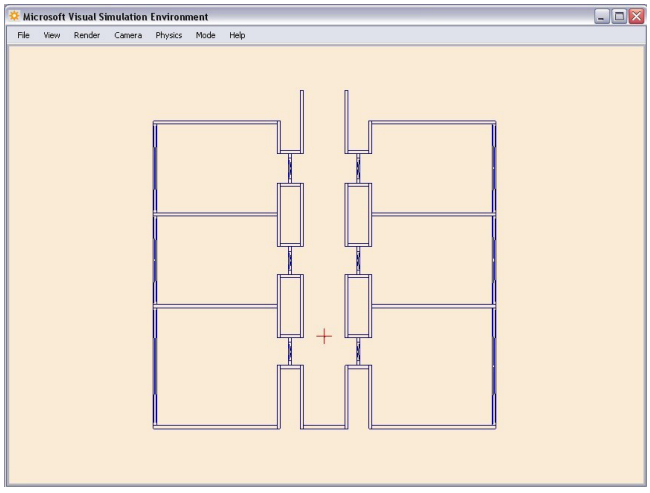
Treba si všimnúť, že všetky laserové snímače sú umiestnené v rovnakom bode a líšia sa len svojou orientáciou. Znamená to, že pri spracovávaní informácií z týchto snímačov užívateľ nesmie zabudnúť odpočítať vzdialenosť snímača od povrchu robota, teda polomer podstavy robota.

```
private CustomLaserRangeFinderEntity CreateUR-
PI_Robot_1_Sonar(float angle, int i)
{
            // Create a Custom Laser Range Finder Entity .
            // Place it above base, rotate and set the parame-
            ters
            CustomLaserRangeFinderEntity laser = new Cus-
            tomLaserRangeFinderEntity(
            new Pose(new Vector3(0, 0.175f, 0),
            // Rotation in Euler angles
            UIMath.EulerToQuaternion(TypeConversion. ToX-
            NA(new Vector3(0, angle, 0)))), 0.5f, // refresh rate
            1f, // angular resolution      25, // angular range
            3f); // range
            laser.State.Name = "URPI_Robot_1_Sonar_" +
            i.ToString();
            laser.LaserBox.State.DiffuseColor = new Vec-
            tor4(0.25f, 0.25f, 0.8f, 1.0f);
            laser.LaserBox.State.Dimensions = new Vec-
            tor3(0.05f, 0.05f, 0.05f);
            // Create LaserRangeFinder simulation service and
            specify
            // which entity it talks to
            lrf.Contract.CreateService( ConstructorPort, Micro-
            soft.Robotics.Simulation.Partners.CreateEntityPart
            ner( "http://localhost/" + laser.State.Name));
            return laser;
}
```

Laserový snímač je upravený tak, aby sa správal ako ultrazvukový snímač vzdialenosti. Jeho perióda vzorkovania je nastavená na pol sekundy, zorný uhol je nastavený 25° a uhlové rozlíšenie je 1°. Okrem toho bol obmedzený maximálny dosah snímača na 3m a každý laserový snímač má vlastný názov. Spôsob vyhodnocovania ultrazvukového snímača je samozrejme odlišný od laserového. Túto skutočnosť však treba riešiť softvérovo.

5 Vytvorenie simulačného prostredia s nástrojom Floorplan

Pre testovanie algoritmov simulovaných robotov treba vytvoriť vhodné prostredie. Prostredie sa dá vytvoriť klasickým spôsobom pridávaním rôznych vizuálnych objektov alebo pomocou nástroja *Floorplan*. Tento nástroj umožňuje vytvoriť prostredie podobné vnútorným priestorom budov. Aby užívateľ mohol tento nástroj používať musí mať spustené simulačné prostredie Visual Simulation Enviroment a zapnutý editovací mód. Editovací mód je možné spustiť pomocou menu *Mode* alebo stlačením klávesy *F5*. Následne z menu *Entity* treba vybrať položku *New*. Otvorí sa okno, v ktorom je vidieť zoznam rôznych objektov použiteľných v simulácii. Zo zoznamu sa dá vybrať položka *FloorplanEntity*. Otvorí sa nám ďalšie okno, kde sa zadáva pozícia vzťažného bodu. V zozname v ľavom stĺpci v simulačnom prostredí pribudol nový objekt. Po označení tohto objekt a kliknutia na tlačidlo *Edit Entiy* sa otvorí okno nástroja *Floorplan* (Obr. 3).



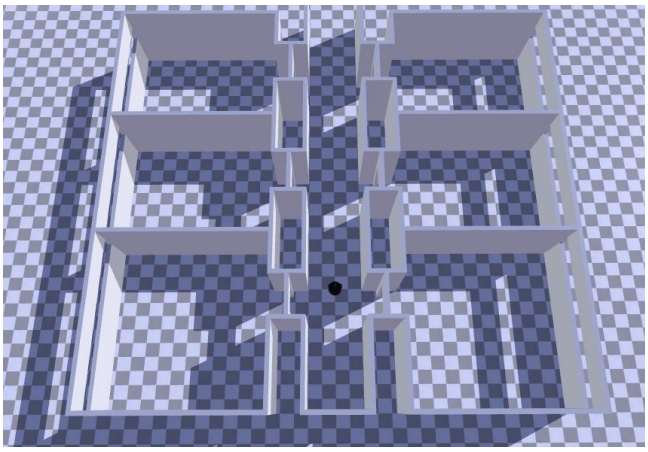
Obr.3 Nástroj Floorplan

Fig.3 Floorplan tool

Navigácia na pracovnej ploche nástroja *Floorplan* je jednoduchá. Pohyb po pracovnej ploche sa vykonáva pomocou stlačenia ľavého tlačidla myši a pohybu myši zároveň. Priblíženie alebo oddialenie sa vykonáva stlačením a držaním tlačidla *CTRL* a pohybom myšou doľava alebo doprava.

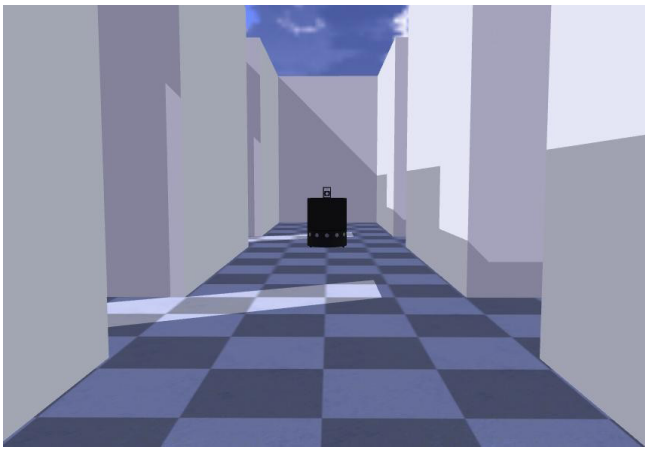
V tomto nástroji môžeme pridať tieto objekty: múr (Wall), dvere (Door) alebo okno (Window). Pre všetky tri typy platia rovnaké pravidlá. Po výbere typu sa na pracovnej ploche nástroja *Floorplan* zobrazí obdĺžnik. Obdĺžnik užívateľ najskôr označí, čím sa zobrazia sa dve farebné značky. Týmito značkami sa dajú nastaviť požadované rozmery a orientácie objektu. Ak sú značky štvorce, potom možno nastaviť roz-

mery objektu naťahovaním. Premiestňovanie objektov je možné kliknutím na daný objekt, podržanie tlačidla myši a následné umiestnenie tohto objektu na želané miesto. Ak je už užívateľ s vytvorením simulovaného prostredia hotový, klikne na *OK* a vráti sa do simulačného prostredia kde už možno vidieť vytvorené objekty. Toto vytvorené prostredie si je možné uložiť ako XML súbor pomocou položky *Save Scene As* z menu File. Tento súbor je potom pripravený pre použitie pri testovaní robota.



Obr.4 Vytvorené prostredia

Fig.4 Created environment



Obr.4 Robot vo vytvorenom prostredí

Fig.4 Robot in created environment

Záver

Výsledkom vytvorenia modelu mobilného robota spolu so snímačmi a prostredím v Microsoft Robotics Developer Studio je možnosť využiť tieto modely pri návrhu a optimalizácii rôznych algoritmov správania sa mobilných robotov. Výsledky z týchto optimalizácií je možné testovať na reálnych systémoch.

Poďakovanie

Článok vznikol pri riešení projektu VEGA 1/0690/09.

Literatúra

[1] AGEIA Technologies: Advanced Gaming Physics Defining The New Reality In PC Hardware, March 2006, http://www.ageia.com/pdf/wp_advanced_gaming_physics.pdf

[2] MRS User Guide, http://msdn2.microsoft.com/en-us/library/bb881626.aspx

[3] Microsoft Robotics Studio Now Available to Provide Common Development Platform, http://www.robocup-us.org/Media/2006-MSRS-release.pdf

[4] Microsoft Robotics Studio Community, http://forums.microsoft.com/MSDN/default.aspx?ForumGroupID=383&SiteID=1

[5] DUCHOŇ F., ŠTRENGER M., Vývojové prostredie Microsoft Robotics Studio, AT&P Journal Plus 1, 2008

[6] JOHNS K., TAYLOR T., Professional Microsoft Robotics Studio (Wrox Programmer To Programmer), Wiley Publishing, Inc., ISBN: 9780470141076

[7] BORENSTEIN J., EVERETT H.R., FENG L.: „Where am I?“ Sensors and Methods for Mobile Robot Positioning [pdf súbor]. University of Michigan 1996, ftp://ftp.eecs.umich.edu/people/johannb/pos96rep.pdf

[8] JURIŠICA L., MURÁR R.: Reprezentácia prostredia mobilného robotického systému (1). AT&P Journal, 2005, č.8, str. 60-61,

http://www.atpjournal.sk/casopisy/atp_05/pdf/atp-2005-08-60.pdf

Abstract

For implementation of inteligent mobile robot control algorithms in Microsoft Robotics Developer Studio, it is necessary to create own model of robot and complex simulation environment. Alone model of robot will be useless, so it needs to create environment, in which will robot work, but also sensors, which will robote use for feedback information from this environment. This paper contents creating of own sensor in Microsoft Robotics Developer Studio and creating of own environment, in which will mobile robotic system intelligently perform its activities.

prof. Ing. Ladislav Jurišica, PhD.,
Ing. František Duchoň,
Bc. Martin Štrenger

Fakulta elektrotechniky a informatiky
Ústav riadenia a priemyselnej informatiky
Ilkovičova 3
812 19 Bratislava
ladislav.jurisica@stuba.sk
frantisek.duchon@stuba.sk
mastr@ynet.sk

Archív Archive

AT&P journal PLUS1: Adaptívne a nelineárne riadenie systémov (tlačená verzia, vydané 2001)
Adaptive and nonlinear control systems (printed version, published 2001)

AT&P journal PLUS2: Robotika, mechatronika, diskrétne výrobné systémy (tlačená verzia, vydané 2001)
Robotics, mechatronics, discrete manufacturing systems (printed version, published 2001)

AT&P journal PLUS3: Robustné systémy riadenia (tlačená verzia, vydané 2002)
Robust control systems (printed version, published 2002)

AT&P journal PLUS4: Samonastavujúce sa systémy v riadení procesov (tlačená verzia, vydané 2003)
Selftuning systems in process control (printed version, published 2003)

AT&P journal PLUS5: Robotické systémy (elektronická – CD verzia, vydané 2004)
Robotics systems (electronic – CD version, published 2004)

AT&P journal PLUS6: Mechatronika (elektronická – CD verzia, vydané 2005)
Mechatronics (electronic – CD version, published 2005)

AT&P journal PLUS7: Umelá inteligencia v praxi (elektronická – CD verzia, vydané 2005)
Artifical intelligence in Practise (electronic – CD version, published 2005)

AT&P journal PLUS 1 2006: Mechatronické systémy (elektronická – CD verzia, vydané 2006)
Mechatronic systems (electronic – CD version, published 2006)

AT&P journal PLUS 2 2006: Inteligentné meracie systémy (elektronická – CD verzia, vydané 2006)
Intelligent measurement systems (electronic – CD version, published 2006)

AT&P journal PLUS 1 2007: MMaMS'2007 (elektronická – CD verzia, vydané 2007)
MMaMS'2007 (electronic – CD version, published 2007)

AT&P journal PLUS 2 2007: Riadenie procesov (elektronická – CD verzia, vydané 2007)
Process Control (electronic – CD version, published 2007)

AT&P journal PLUS 1 2008: Mobilné robotické systémy (elektronická – CD verzia, vydané 2008)
Mobile robotic systems (electronic – CD version, published 2008)

AT&P journal PLUS 2 2008: Riadenie v energetike (elektronická – CD verzia, vydané 2008)
Control of Power Systems (electronic – CD version, published 2008)