KATEDRA APLIKOVANEJ INFORMATIKY
FAKULTA MATEMATIKY FYZIKY A INFORMATIKY
UNIVERZITA KOMENSKÉHO, BRATISLAVA

SLEDOVANIE POHYBUJÚCEJ SA LOPTIČKY MOBILNÝM ROBOTOM

(Diplomová práca)

BC. TOMÁŠ RAPČAN

Študijný program : Kognitívna veda
Medziodborové štúdium: 9.2.8 Umelá inteligencia
3.1.9 Psychológia

Školiteľ: doc. Ing. Igor Farkaš, PhD.

Bratislava 2010

Čestné vyhlásenie

Čestne vyhlasujem, že som diplomovú prácu vypracoval samostatne s použitím uvedenej literatúry.

Bratislava 12. 5. 2010 .....

Poďakovanie

Chcem sa poďakovať vedúcemu práce doc. Ing. Igorovi Farkašovi, PhD. za jeho odbornú pomoc, nápady pred i počas písania práce.

Abstrakt

Rapčan, Tomáš. *Sledovanie pohybujúcej sa loptičky mobilným robotom.* [Diplomová práca]. Univerzita Komenského v Bratislave. Fakulta Matematiky, Fyziky a Informatiky; Katedra Aplikovanej Informatiky.

Školiteľ: Ing. Igor Farkaš, PhD: FMFI UK, 2010, 77 s.

Diplomová práca sa zaoberá vytváraním simulácie mobilného robota sledujúceho pohybujúcu sa loptičku pomocou RBF siete (Radial Basis Function).

Opisuje základné teoretické poznatky týkajúce sa umelých neurónových sietí potrebných k implementácii simulácie. Poukazuje na klasický (symbolový) prístup riešenia podobného problému, ktorý je komplementárny k umelým neurónovým sietam. Práca prezentuje výsledky testovania efektívnosti modelu vzhľadom na rýchlosť a vzhľadom na úroveň šumu vo vstupných (obrazových) dátach.

Kľúčové slová: umelá neurónová sieť, RBF sieť, simulácia, mobilný robot.

Abstract

Rapčan, Tomáš. *Simulation of mobile robot tracking a free rolling ball.* [Master's thesis]. Comenius University in Bratislava Faculty of Mathematics, Physics and Informatics; Departement of Applied Informatics.

Supervisor: Ing. Igor Farkaš, PhD: FMFI UK, 2010, 77p.

This Master's thesis describes an implementation of an experiment in which a mobile robot tracks (follow) a free rolling ball using Radial Basis Function (RBF) network.

It describes the basic theoretical knowledge of artificial neural networks necessary to implement the simulation. It refers to the classical (symbolic) approach for a similar problem, which is complementary to the artificial neural networks. This work presents the results of testing the effectivity of the model with respect to speed and the level of noise in input data.

Keywords: artificial neural network, RBF network, simulation, mobile robot, tracking.

Predhovor

Diplomová práca sa zaoberá implementáciou a následným testovaním simulácie mobilného robota sledujúceho pohybujúcu sa loptičku pomocou RBF siete (Radial Basis Function). Venuje sa implementácii a následnému otestovaniu efektívnosti natrénovanej RBF siete v prípade viacerých spôsobov zašumenia vstupných dát počas interakcie s virtuálnym prostredím a efektívnosti v prípade zmeny rýchlosti pohybu robota a loptičky po scéne. Poukazuje na charakteristické správanie sa modelu, slabiny ale i devízy zvoleného prístupu preukázané v testovaní.

Práca sa venuje i opisu teoretickým poznatkom vzťahujúcich sa k umelým neurónovým sieťam a taktiež poukazuje na možnosť riešenia podobného zadania práce pomocou klasického (symbolového) prístupu.

Pri nadobúdaní vedomostí potrebných k vyhotoveniu zadania sme využili literatúru, publikované materiály a osobné konzultácie týkajúce sa spracovávaného problému.

Prácu je možné použiť ako podklad v prípade tvorby podobnej simulácie s cieľom porovnania si vedomostí v danej problematike.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam skratiek a značiek

SOM	samoorganizujúca mapa
RBG	farebný model s tromi farebnými zložkami (červená, modrá, biela)
UNS	umelá neurónová sieť.
c	centrum bázovej funkcie
d	učiaci signál
σ	citlivosť bázového neurónu
I	výpočet hodnoty obrazového bodu v škále šedí
net	suma cez všetky vstupy vynásobené príslušnými váhami
r	polomer centra bázového neurónu
w	váha neurónu
x	vstup neurónu
$\|x\|$	euklidovská vzdialenosť vektoru x
y	výstup neurónu
Θ	aktivačný prah
α	rýchlosť učenia
δ_i	chyba výstupu
φ	nelineárna aktivácia bázového neurónu

Úvod

Človek si ani neuvedomuje, že systémy, ktoré pracujú na princípoch spadajúcich do odvetvia umelej inteligencie a kognitívnej vedy, nie sú v dnešných dňoch v našom okolí ničím výnimočným. Úspešne sa aplikujú nielen v rozhodovaní, riadení ale aj v netechnických smeroch.

Môžeme povedať, že umelá inteligencia predstavuje relatívne široké spektrum prístupov, ktoré v mnohých prípadoch vychádzajú z úplne iných predpokladov a ich spoločným cieľom je riešenie rovnakých problémov. Ide o problémy, ktoré sa pokúšajú simulovať riešenia problémov biologických systémov, ako je napr. analýza a vyhodnocovanie najrôznejších situácií za účelom nájdenia adekvátneho riešenia. Keďže správanie biologických systémov (najmä ľudského mozgu) je extrémne komplexné, zatiaľ nie je možné ani jedným zo známych prístupov umelej inteligencie správanie takéhoto komplexného systému modelovať. Konekcionistický model (Connectionist model) je jedným z prístupov, ktorý sa o to pokúša. Už názov prístupu poukazuje na hlavný princíp funkčnosti. Ide o riešenie problémov pomocou vysokej miery prepojenia medzi jednotlivými výpočtovými jednotkami na elementárnej úrovni. Celkový model sa nielenže pokúša modelovať správanie biologických systémov jeho vonkajším prejavom (teda generovanými výstupmi), ale aj jeho vnútornou štruktúrou. Princíp fungovania modelu je postavený na znalostiach spadajúcich najmä do odvetvia neurovedy, v užšom zmysle ide o matematickú aproximáciu biologických neurónov (umelé neurónové siete).

Rôzne prístupy v rámci umelej inteligencie majú za následok vznik systémov, ktoré sa venujú modelovaniu rovnakých problémov na základe odlišných princípoch funkčnosti. Na jednej strane to môže byť model umelej neónovej siete, ktorá pracuje pomocou malých výpočtových jednotiek. Na druhej strane to môže byť model klasického prístupu, ktorý bude riešiť ten istý problém na prostredníctvom procesov, ktoré narábajú so symbolmi. Symboly predstavujú zjednodušené vnútorné reprezentácie vonkajšieho sveta.

Pri odlišnostiach, aké dané prístupy ukazujú sa vynára otázka, či sú schopné oba prístupy dosiahnuť porovnateľné výsledky v rovnakých úlohách, čo bolo motiváciou k uskutočneniu diplomovej práce.

V práci navrhneme simuláciu mobilného robota, ktorý má sledovať pohybujúcu sa loptičku pomocou špeciálneho druhu umelej neurónovej siete. Očakávame, že neurónová sieť si dokáže bezproblémovo osvojiť schopnosť sledovania daného objektu. Cieľom práce je testovanie správania sa implementovanej simulácie v rôznych podmienkach. Ide o testovanie efektivity umelej neurónovej siete so zašumenými vstupnými dátami a vplyv zmeny rýchlosti pohybu objektov po scéne za účelom zistenia jej efektivity i v takto stanovených podmienkach.

Práca bude pozostávať z piatich častí, a to z teoretickej časti týkajúcej sa umelých neurónových sietí, opisu konkrétnej implementácie klasického prístupu, návrhu implementácie simulácie, realizácie simulácie a z časti zaoberajúcou sa dosiahnutými výsledkami.

V prvej kapitole sa zameriame na problematiku týkajúcu sa umelých neurónových sietí výlučne z teoretického pohľadu. Pozornosť upriamime najmä všeobecné poznatky a informácie potrebné k realizovaniu práce, teda na umelú neurónovú sieť typu RBF (Radial Basis Function).

Druhá kapitola bude venovaná špeciálnemu prípadu použitia klasického prístupu na riešenie rovnakého problému v reálnych podmienkach. Priblížime v nej všetky vykonané kroky, ktoré budú potrebné k jej úspešnej realizácii.

V tretej kapitole sa budeme zaoberať návrhom realizácie za účelom správneho naplánovania postupu práce.

Štvrtá kapitola bude popisovať najdôležitejšie algoritmy naprogramované v simulácii.

Posledná kapitola bude venovaná dosiahnutým výsledkom z vykonaných testov na simulácii s ohľadom na zadanie práce.

Text diplomovej práce doplníme o tabuľky a obrázky, ktoré uľahčia orientáciu a pomôžu sprehľadniť spracovávanú problematiku.

Ciele práce

Hlavným cieľom práce je vytvoriť simuláciu mobilného agenta sledujúceho pohybujúcu sa loptičku riadeného umelou neurónovou sieťou, pričom daný robot musí udržiavať adekvátnu vzdialenosť od sledovaného objektu. Vstup umelej neurónovej siete bude reprezentovaný bázovými neurónmi (pod neurónmi máme v práci na mysli, pokiaľ nie je uvedené inak, matematickú aproximáciu biologického neurónu). Pre lepšiu organizáciu a vyššiu efektivitu práce je vhodné zadanie práce rozdeliť na postupnosť niekoľkých elementárnych krokov.

- Vyhľadanie vhodných zdrojov na zlepšenie orientácie v problematike týkajúcej sa umelých neurónových sietí a senzomotorickej koordinácie v simulovaných robotoch.
- Spresnenie kritérií kladených na výslednú simuláciu.
- Výber vhodného vývojového prostredia na vytvorenie simulácie so zadanými kritériami.
- Adekvátne oboznámenie sa so zvoleným vývojovým prostredím.
- Prerozdelenie postupu implementácie za účelom čiastkového testovania funkčnosti simulácie. Implementácia:
    - jednoduchého trojdimenzionálneho prostredia s testovaním jeho vhodnosti,
    - RBF vrstvy s testovaním výstupu na špecifický podnet,
    - výstupnej (riadiacej) vrstvy neurónov.
- Trénovanie umelej neurónovej siete v “modelových“ situáciách a neskôr v trojdimenzionálnej (ďalej len 3D) simulácii.
- Testovanie efektívnosti implementácie modelu vzhľadom na rýchlosť a vzhľadom na úroveň šumu vo vstupných dátach.
- Vytvorenie dokumentačnej časti práce, kde je nevyhnutné vyjadriť sa k poznatkom a záverom, ktoré boli dosiahnuté v priebehu tvorby simulácie.

Hlbšia analýza zadania nám umožní lepšiu organizáciu práce a zvýši to hodnotu výslednej simulácie.

1 ÚVOD DO NEURÓNOVÝCH SIETÍ

V prvej kapitole sa budeme venovať všeobecným teoretickým poznatkom z umelých neurónových sietí, ich aplikácií ako aj konštrukcii. V závere kapitoly sa zameriame na problematiku týkajúcu sa bázovej umelej neurónovej siete, ktorú plánujeme implementovať v simulácii.

1.1 Všeobecný úvod

Teória týkajúca sa modelov umelých neurónových sietí (ďalej len UNS) je inšpirovaná najmä biologickou funkciou ľudského mozgu, no pochádza z viacerých vedných disciplín ako napr.: neuroveda, matematika, štatistika a iné.

Z histórie sa dozvedáme, že počiatky UNS, taktiež známych ako konekcionistické modely alebo paralelné distribuované systémy, začínajú s predstavením jednoduchých modelov biologických neurónov (McCulloch a Pitts 1943). Ich modely boli prezentované ako jednotky, ktoré vo vzájomnom zapojení dokážu vykonávať výpočtové úlohy. V roku 1969 poukázal Minsky a Papert na nedostatky vyššie uvedených modelov. Kritika modelu mala za následok pokles záujmu o výskum v tejto oblasti.

Opätovný záujem o UNS viditeľne vzrástol po niekoľkých významných teoretických objavoch na začiatku deväťdesiatych rokov. K zvýšenému záujmu prispela v najväčšej miere možnosť trénovania UNS pomocou algoritmu spätného šírenia chyby (error back propagation) (Rumelhart, Hinton a Williams, 1986) a v neposlednom rade zvýšenie výpočtovej kapacity počítačového hardvéru.

V dnešných dňoch sú UNS aplikované na mnohé problémy, no najčastejšie na aproximáciu funkcií, klasifikáciu alebo na mapovanie funkcií za predpokladu, že je použité dostatočné množstvo trénovacích dát.

Vo všeobecnosti je tendencia pripisovať výpočtovú silu UNS jej dvom výrazným vlastnostiam. Prvou vlastnosťou, ktorá vedie zainteresovaných k takémuto tvrdeniu je ich paralelnosť. Dovoľujeme si pripomenúť, že paralelnosť, inak povedané aj distribuovanosť, je v softvérovej implementácii len simulovaná. Reálnu distribuovanosť vieme dosiahnuť hardvérovou implementáciou. Druhou zo spomenutých vlastností je schopnosť učiť sa a generalizovať. Učenie v tomto prípade reprezentuje schopnosť siete aproximovať nejakú, pre nás významnú funkciu. Generalizácia zabezpečuje

generovanie adekvátnych výstupov k vstupom, s ktorými sa UNS počas trénovania nestretla.

UNS v širšom zmysle charakterizujeme ako výpočtový model s vlastnosťami ako adaptácia (učenie), schopnosť generalizovať, alebo organizovať. Haykin (1999) uvádza definíciu UNS ako adaptovateľného stroja nasledovne: „Neurónová sieť je masívne paralelný distribuovaný systém zostavený z jednoduchých výpočtových jednotiek, ktoré majú prirodzenú schopnosť ukladať experimentálne nadobudnuté vedomosti, pričom je k nim možný stály prístup. Podobajú sa mozgu v dvoch aspektoch:

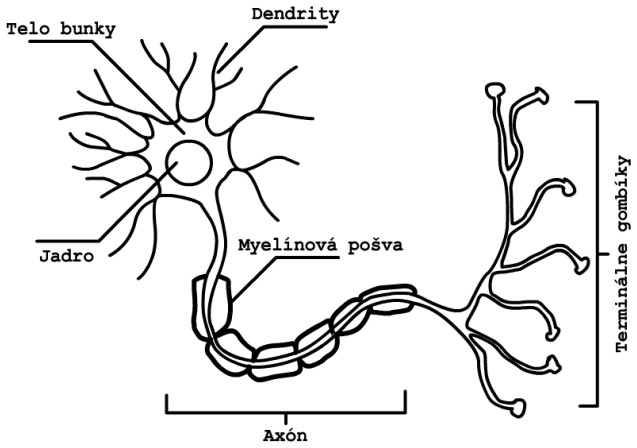
1. sieť nadobúda vedomosti z jej okolia pomocou učiaceho procesu,
2. sila spojení medzi neurónmi, ktoré sú známe ako synaptické váhy, slúžia na ukladanie získaných znalostí.“

Modelovanie správania systémov pomocou UNS je nekonkrétne, to znamená, že sieť s tou istou topológiou môže byť použitá na modelovanie rôznych procesov za predpokladu, že použitý model bude opäť trénovaný na nových dátach. Na druhej strane musíme upozorniť na to, že táto abstrakcia nie je len výhodou. „Vedomosti“ nadobudnuté počas trénovania nie je možné získať explicitne zo siete, pretože sú uchované vo váhach výpočtových jednotiek.

1.2 Konštrukcia UNS

Podľa Haykina (1999) môžeme povedať, že mozog je vysoko komplexný, nelineárny a paralelný počítač, ktorý má schopnosť organizovať vnútornú štruktúru za účelom zdokonaľovania sa pri vykonávanej akcii.

Presne číslo hovoriace o počte neurónov v ľudskom mozgu nie je známe, ale vo všeobecnosti sa uvádza číslo 10^{11}. Mozog obsahuje niekoľko typov neurónov. Nasledujúci obrázok zjednodušene ilustruje jeden z nich.



Obr. 1.1: : Biologický neurón

Na obr. 1.1 máme možnosť vidieť vstupy do neurónu reprezentované dendritmi. Ďalej je znázornené telo bunky, ktoré obsahuje jadro neurónu. Výstup neurónu reprezentuje axón, ktorý je obalený myelinovou pošvou. Myelin slúži na rýchlejší prenos akčných potenciálov (informácie) od daného biologického neurónu k ostatným. Axón ukončujú terminálové gombíky, kde sa končí cesta akčného potenciálu v rámci jedného neurónu. Informácie sa medzi neurónmi šíria prostredníctvom neurotransmiterov cez synaptické štrbiny. Počet synaptických spojení sa počas života človeka mení, no v dospelosti je to zhruba 5-7 tisíc synapsií na jeden neurón.

Ako sme už naznačili, biologické neurónové siete sú mimoriadne komplexný systém, ktorého presnú analógiu nie je v dnešných dňoch možné vytvoriť a ani nie je isté, že niekedy bude. UNS môžeme charakterizovať niekoľkými hlavnými črtami, ktoré sú do určitej miery analógiou biologických neurónových sietí, a to nasledovne:

- Je to súbor výpočtových jednotiek (neurónov).
- Každá jednotka obsahuje aktivačnú funkciu. Výstup tejto funkcie je aj výstupom výpočtovej jednotky (frekvencia pálenia biologického neurónu).
- Prepojenia medzi jednotkami sú orientované a majú pridelené váhy (synaptické spojenia). Veľkosť váhy kvantifikuje účinnosť vstupného signálu, ktorý prechádza daným váhovaným spojením na jednotku, do ktorej vstupuje. Všetky váhy vynásobené príslušnými vstupmi, pričom sa na jednotke počíta ich suma a porovnáva sa s prahom (pri biologických neurónoch môže byť sumácia aj časová). Spojenia môžu byť excitačné alebo inhibičné.

- Prah (z angličtiny threshold) určuje úroveň, po prekročení ktorej bude daný neurón aktivovaný (prah excitácie).
- Učiace pravidlo je metóda určená na osvojovanie nových informácií.

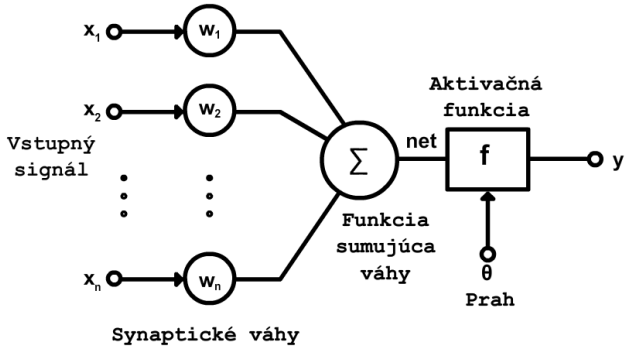
V nasledujúcej časti sa budeme podrobnejšie venovať základným stavebným prvkom UNS a zákonitostiam, ktoré platia medzi nimi.

1.2.1 Výpočtová jednotka (perceptrón)

Výpočtová jednotka je základný stavebný prvok UNS, ktorú tiež označujeme ako perceptrón (ďalej bude výpočtová jednotka označovaná ako neurón, pokiaľ sa nebude myslieť inak). Má za úlohu vykonávať relatívne jednoduchú prácu. Vstupné signály z vonkajšieho prostredia, alebo zo susedných neurónov sú cez synaptické váhy prepočítavané pomocou aktivačnej funkcie na výstup, ktorý sa distribuuje ďalej, či už k ďalším neurónom, alebo len k výstupu. V prípade distribuovaných systémov v porovnaní so symbolovo založenými systémami, sú cez ich komunikačné kanály (synaptické spojenia) prenášané a neskôr spracovávané numerické informácie.

Funkčnosť biologických neurónových sietí je do značnej miery známa, ale stále nie úplne jasná. Matematický model neurónu reprezentuje zjednodušenú verziu biologického neurónu.

Nasledujúci obrázok zobrazuje matematickú aproximáciou biologického neurónu.



Obr. 1.2: Perceptrón

Už len pri pohľade na obr. 1.2 sa dá pozorovať podobnosť v stavbe, ďalej si povieme o ich funkčnej podobnosti.

Prepojenia medzi neurónmi

Každý neurón poskytuje príspevok k vstupu neurónu, na ktorú je napojený.

Výstup neurónu je daný vzťahom:

$$y = f\left[\sum_{k=1}^{n} x_k w_k - \theta\right] \qquad 1.1$$

kde n – je počet vstupov, Θ je aktivačný prah, $\boldsymbol{x}$ je vstupný vektor a $\boldsymbol{w}$ je váhový vektor na k-ty neurón. Zo vzťahu vyplýva, že perceptrón vykonáva globálne mapovanie vstupov na výstup.

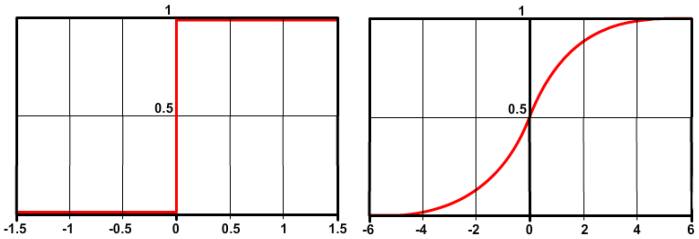
Kladný príspevok váhy w_k je braný ako excitačný a záporný príspevok naopak, ako inhibičný. Váhy reprezentujú distribuovanú formu uchovanej informácie v UNS.

V topológii UNS vo všeobecnosti rozlišujeme medzi tromi druhmi neurónov:

- vstupné neuróny (dostávajú vstupy z vonkajšieho prostredia),
- skryté neuróny (ich vstupy a výstupy ostávajú len vo vnútri UNS) a
- výstupné neuróny (posielajú dáta von z UNS).

Aktivácia neurónu

Aktiváciu neurónu reprezentuje pravidlo, ktoré dá efekt celkového vstupu na aktivačnú funkciu a súčasne udržiava jeho numerickú stabilitu. Spomenieme si aspoň dve bežne používané aktivačné funkcie: krokovú a sigmoidálnu. V poradí druhá funkcia bude neskôr predmetom aplikácie v simulácii.



Obr. 1.3: Aktivačné funkcie – skoková (vľavo), sigmoidálna (vpravo)

Binárna funkcia zobrazená na obr. 1.3 (ľavá strana) je funkcia, ktorá má vstup aj výstup binárny a používa sa na binárne klasifikovanie.

Sigmoidálna funkcia znázornená na obr. 1.3 (pravá strana) je najčastejšie používaný druh aktivačnej funkcie v UNS. Z neurológie vieme, že i vstupno-výstupná charakteristika biologického neurónu má taktiež tvar sigmoidy (Bois-Reymond, 1948). Sigmoidálna funkcia je daná nasledovným vzťahom:

$$f(net) = 1/(1+e^{-net}) \qquad 1.2$$

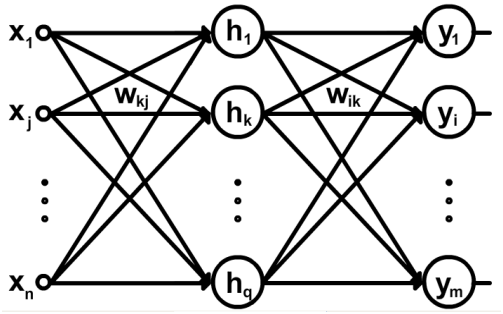
pričom *net* predstavuje sumu cez všetky vstupy vynásobené s príslušnými váhami. Vzťah na výpočet *net*-u je:

$$net = \left[\sum_{k=1}^{n} x_k w_k \right] \qquad 1.3$$

V tejto časti sme si predstavili parametre základného stavebného prvku UNS. Ďalej si opíšeme niektoré z možností vzájomných prepojení medzi nimi.

1.3 Topológie UNS

Ľudský mozog je vynikajúca výpočtová jednotka, ktorá je reprezentovaná aj neurónmi zapojenými vo vrstvách. Viacvrstvový model sa osvedčil i pri aplikáciách v UNS. Obr. 1.4 znázorňuje viacvrstvový perceptrón.



Obr. 1.4: Viacvrstvová UNS

Takáto topológia je typický príklad UNS s použitím jednej skrytej vrstvy. Výstup siete sa počíta na základe vzťahu č. 1.1 uvedeného pri perceptrón s postupom výpočtu od vstupov x_1-x_n cez skryté neuróny h_1-h_q, až k výstupným neurónom y_1-y_m. Pre objasnenie si uvedieme príklad. Výstup neurónu h_k dosiahneme prostredníctvom nasledovného vzťahu č. 1.4:

$$h_k = f\left[\sum_{j=1}^{n+1} w_{kj} x_j \right] \qquad 1.4$$

a výstup neurónu y_i dosiahneme pomocou vzťahu č. 1.5:

$$y_i = f\left[\sum_{k=1}^{q+1} w_{ik} h_k \right] \qquad 1.5$$

kde $n+1$ a $q+1$reprezentujú prah daného neurónu.

Je zrejmé, že spôsob, akým sú neuróny medzi sebou usporiadané, hovorí o vzore prenosu dáta v sieti. Hlavný podiel na tom majú orientované váhy. Podľa smeru šírenia informácie v UNS hovoríme o dvoch typoch sietí, ktoré popíšeme v ďalšej časti.

1.3.1 Dopredné UNS

Dopredné (feed-forward) UNS sú špecifické jednosmerným tokom dát od vstupných neurónov k výstupným. Z toho vyplýva, že spracovávanie dát prebieha cez všetky použité vrstvy UNS, ale bez spätných prepojení medzi neurónmi, podobne ako na obr. 1.4. Spätné prepojenia sú prepojenia medzi akýmkoľvek výstupom neurónu a iným vstupom neurónu umiestneného v rovnakej vrstve alebo ktorejkoľvek vrstve pred ňou. Dopredné UNS sa vyznačujú jednoduchou implementáciou a všestrannou použiteľnosťou napr.: v klasifikácii údajov, v aproximácii funkcií, v modelovaní systémov, ktoré nám nie sú známe a v mnohých iných problémoch.

1.3.2 Rekurentné UNS

Rekurentné UNS (z angličtiny recurrent) obsahujú minimálne jedno spätné spojenie. Spätné spojenie má za následok cyklické ovplyvňovanie aktivity daného neurónu. Rekurentné UNS sa často označujú ako UNS s časovým posunom, alebo kontextom.

Z biologického hľadiska sú UNS tohto typu o niečo viac relevantné ako dopredné, pretože v mozgu nie sú spojenia medzi neurónmi iba jednosmerné. Aplikácia rekurentných sietí nie je taká jednoduchá ako v prípade dopredných sietí. Je to ovplyvnené ich trénovaním, ktoré je zložitejšie. Napriek tejto skutočnosti nájdeme rôzne aplikácie v problematike predikcie, modelovaní asociatívnych pamätí, analýze reči, riadení a iné.

Ďalej spomenieme niekoľko druhov rekurentných UNS.

Elmanova – obsahuje „pamäťové“ neuróny určené na uchovanie stavu skrytých neurónov z predchádzajúceho kroku.

Jordanova – pamäťové neuróny počítajú svoj stav z výstupu siete a svojho stavu v predchádzajúcom kroku.

Hopfieldova – každá vstupný neurón je aj výstupný a zároveň je prepojený aj so všetkými ostatnými neurónmi v sieti.

Z ohľadom na zadanie práce rekurentné UNS nie je potrebné bližšie vysvetľovať. Z tohto dôvodu sa posunieme na ďalšiu časť teórie, a to trénovanie UNS.

1.4 Trénovanie

UNS sú v zásade schopné naučiť sa teoreticky s ľubovoľnou presnosťou akúkoľvek spojitú funkciu. To znamená, že sa dokážu naučiť takmer čokoľvek, čo dokáže číslicový počítač. UNS je nevyhnutné podrobiť trénovaniu, aby sme dosiahli s danými vstupmi požadované výstupy. Ako sme si už spomenuli v úvode, učenie je reprezentované úpravou synaptických váh a prahových potenciálov neurónov.

Existujú dva spôsoby ako nastaviť tieto váhované spojenia na správnu hodnotu:

1. nastavenie váh pomocu vopred naučených a uložených hodnôt,
2. trénovanie pomocou učiaceho pravidla, pričom rozoznávame dva druhy trénovania: učenie s učiteľom, učenie bez učiteľa.

1.4.1 Učenie s učiteľom

Učenie s učiteľom aplikujeme za predpokladu, že poznáme k vstupom, ktoré sú predkladané UNS príslušné (očakávané) výstupy. V prípade správne zvolenej topológie sa dokáže UNS priblížiť k požadovanému výsledku pomocou učiaceho pravidla. Učiace pravidlo má za úlohu upravovať váhy na vstupoch do neurónov tak, aby sa minimalizovala vstupná chyba s ohľadom na očakávaný výsledok . Nižšie uvedené pravidlo (1.6) platí pre perceptrón s binárnou aktivačnou funkciou (obr. 1.3) :

$$w_j(t+1) = w(t) + \alpha(d - y)x_j \qquad 1.6$$

kde $w_j(t+1)$ predstavuje novú váhu, $w(t)$ starú váhu, α rýchlosť učenia (konštanta ovplyvňujúca celkovú rýchlosť učenia), d učiaci signál (očakávaný výstup), ktorý je vopred známy, y príslušný výstup neurónu a x_j predstavuje vstup k príslušnej váhe.

Spätné šírenie chyby

V prípade použitia viacvrstvových sietí sa nám situácia významne komplikuje. Úprava váh UNS prebieha v opačnom smere spracovania vstupných informácií, teda od poslednej vrstvy dopredu. Algoritmus, ktorý upravuje váhy týmto spôsobom sa nazýva algoritmus spätného šírenia chyby (error backpropagation learning). Váhy výstupnej vrstvy sa vypočítajú relatívne ľahko i keď je situácia do určitej miery skomplikovaná.

Používa sa na to rovnaká rovnica ako v prípade jednovrstvového perceptrónu, ktorý je potrebné len trochu upraviť. Uvedené vzorce sa dajú aplikovať na obr. 1.4:

$$w_{ij}(t+1) = w_{ij}(t) + \alpha\delta_i h_k \qquad 1.7$$

kde w_{ij} reprezentujú váhy na skrytej vrstve a δ_i reprezentuje nasledovné:

$$\delta_i = (d_i - y_i) f_i' \qquad 1.8$$

Po vykonaní úprav váh vo výstupnej vrstve môžeme upraviť i skrytú. Keďže na skrytej vrstve nevieme priamo vypočítať chybu (nepoznáme očakávaný výstup), tak musíme chybu posúvať od výstupnej vrstvy dopredu . V tejto fáze je nevyhnutné použiť δ_i z predchádzajúceho kroku. Vzhľadom na túto skutočnosť sa metóda nazýva učenie so spätným šírením chyby.

$$v_{kj}(t+1) = v_{kj}(t) + \alpha\delta_k x_j \qquad 1.9$$

kde δ_k reprezentuje nasledovné:

$$\delta_k = (\sum_i w_{ik}\delta_i) f_k' \qquad 1.10$$

Chybová funkcia

Ak chceme, aby UNS dosahovala, čo najlepšie výsledky, pri trénovaní, alebo pri zmene architektúry (zmena počtu neurónov, vrstiev) je nutné použiť tzv. chybovú funkciu.

$$E_k = \frac{(d_k - y_k)^2}{2} \qquad 1.11$$

Funkcia sa počíta pre každý neurón zvlášť. Po získaní všetkých hodnôt sa vypočíta ich celková suma. Výstup funkcie je možné sledovať. Po ustálení celkovej chyby siete pri zvolenom minime môžeme učenie zastaviť. Iný spôsob prerušenia trénovania je po určitom počte zbehnutých cyklov.

1.4.2 Učenie bez učiteľa

Učenie bez učiteľa je nazývané aj samoorganizujúce sa učenie. Používa sa napr. za účelom zhlukovania, a triedenia vstupných dát do skupín. Výstupy takéhoto druhu UNS nie sú vopred známe.

Algoritmus určený na trénovania UNS tohto druhu je založený na súťaživom princípe. Neuróny sa pod vplyvom učiaceho pravidla stávajú citlivé na určitý druh vstupu, pričom

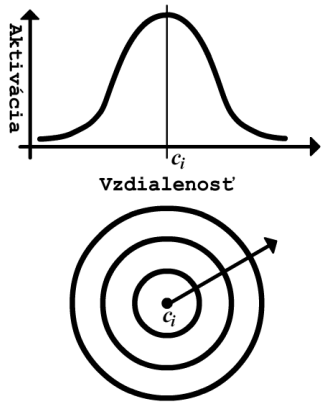
súťažia medzi sebou o to, ktorý z nich bude mať najvyššiu aktivitu. K jednému stimulu je víťazný vždy iba jeden neurón, pričom podobné vstupné vzory vyvolávajú odozvu na neurónoch, ktoré sú v topológii blízko seba.

Za účelom vhodného zobrazenia výstupných dát sú zvyčajne neuróny pri implementácii organizované do jednorozmernej alebo do dvojrozmernej štruktúry.

1.5 RBF siete

Táto kapitola bude venovaná pre nás veľmi dôležitej bázovej (Radial Basis Function) UNS. Bázická UNS (ďalej len RBF sieť) v princípe prestavuje alternatívny prístup k viacvrstvovému perceptrónu. RBF sieť je v podstate hybridná, pretože nie všetky jej neuróny pracujú na rovnakom princípe. Vrstva, ktorá je v kontakte so vstupnými dátami musí reprezentovať radiálna bazová funkcia (viď. nižšie).

Dopredné UNS vykonávajú globálne mapovanie vstupov, kde každý z nich ovplyvňuje celkový výstup UNS. Na druhej strane RBF siete pracujú na lokálnom mapovaní Klasický perceptrón dostáva na spracovanie kompletný vstup určený pre UNS, pričom bázické neuróny sú špecifikované na určitú oblasť vstupu, teda každý z nich má možnosť vidieť len časť vstupných dát. Iba tie vstupy ovplyvnia výstup, ktoré sú blízko špecifického receptívneho poľa, centra RBF neurónu. Obr. 1.5 znázorňuje náčrt jedného receptívneho poľa RBF neurónu s centrom c_i.

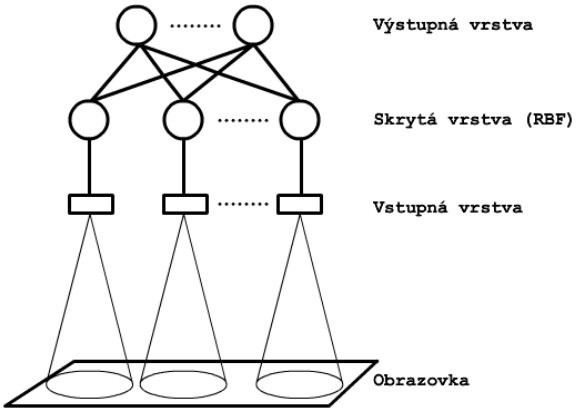


Obr. 1.5: Receptívne pole bázového neurónu

Schopnosť RBF siete rozoznať, či sú vstupné dáta blízko trénovaných dát alebo mimo nich poskytuje významnú výhodu oproti viacvrstvovému perceptrónu a prispela k ich vhodnosti použitia aj v klasifikácií dát.

Matematické odôvodnenie RBF sietí siaha až do roku 1965, kedy Cover špecifikoval problém klasifikácie vzorov v priestore s vyšším počtom dimenzií ako jednoduchší, než v priestore nižším počtom dimenzií. Odtiaľto pochádza dôvod častého aplikovania priestoru s vysokým počtom dimenzií v RBF sieťach. Úloha klasifikácie je riešená pomocou transformovania vysoko dimenzionálneho priestoru nelineárnym spôsobom.

Pri pohľade na topológiu RBF siete na obr. 1.6 môžeme konštatovať, že sa do určitej miery podobná vyššie spomenutej doprednej UNS pozostávajúcej z perceptrónov, ktoré sú zapojenej v architektúre s jednou skrytou vrstvou.



Obr. 1.6: Architektúra jednoduchej RBF siete

Podľa Haykin (1999) by sa konštrukcia RBF sietí dala charakterizovať v jej najjednoduchšom prevedení, kde sú zahrnuté tri vrstvy nasledovne: „Vstupná vrstva je zostavená zo vstupných jednotiek, ktoré sú prepojené s prostredím. Druhá, jediná skrytá vrstva v architektúre aplikuje nelineárnu transformáciu zo vstupného priestoru do skrytého. Vo väčšine aplikácií je skrytý priestor vysoko dimenzionálny. Výstupná vrstva je lineárna a nahrádza odozvu UNS na aktivačný signál predložený vstupu“.

1.5.1 RBF vrstva

Ako sme už naznačili, bázová vrstva RBF siete používa iný vzorec na výpočet svojho výstupu ako výstupná vrstva pri UNS. Výstup bázovej vrstvy je definovaný nasledovným výrazom:

$$\varphi\left(\left\| x - c_i \right\|\right) \qquad 1.12$$

ktorý zvyčajne reprezentuje nelineárna funkcia, tiež označovaná ako bázová funkcia, kde $\|\cdot\|$ označuje Euklidovskú vzdialenosť dvoch vektorov(**x**, **c**). Známe centrá bázových

funkcií reprezentuje c_i. Rozmiestnenie centier sa dá realizovať niekoľkými možnými spôsobmi. Keďže našej situácii bude vyhovovať pevné pridelenie nami zvolených súradníc centier bázových neurónov, tak sa ďalej nebudeme venovať tejto problematike.

Bázové funkcie sú špecifické ich monotónnym stúpaním, alebo klesaním v závislosti od centra. Funkciu φ je možné nahradiť nasledovnými verziami bázických funkcií:

1. Multikvadratická:

$$\varphi(r) = (r^2 + c^2)^{\frac{-1}{2}} \qquad 1.13$$

2. Inverzná multikvadratická:

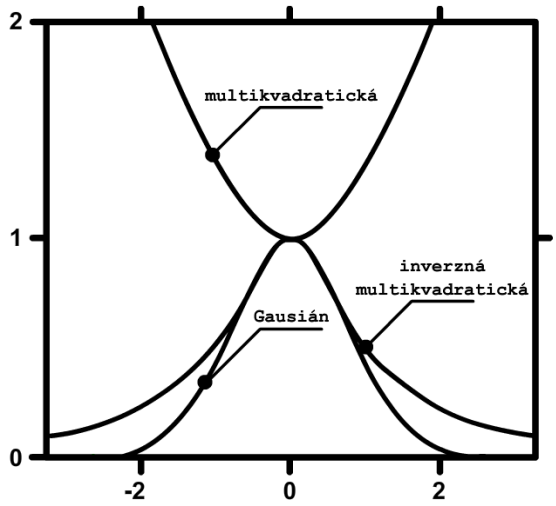
$$\varphi(r) = \frac{1}{(r^2 + c^2)^{\frac{1}{2}}} \qquad 1.14$$

3. Gaussián:

$$\varphi(r) = \left(\frac{-r^2}{2\sigma^2} \right) \qquad 1.15$$

Kde r predstavuje polomer centra bázového neurónu a σ reprezentuje rádius funkcií, inak povedané citlivosť bázového neurónu. Obe premenné je nevyhnutné nájsť empiricky.

Zobrazenie uvedených bázových funkcií je na nasledovnom obrázku.



Obr. 1.7: Graf jednorozmerných bázových funkcií

1.5.2 Výstupná vrstva

V tejto časti upriamime našu pozornosť na teoretické poznatky týkajúce sa celkového výstupu RBF siete. Formálne ho môžeme vyjadriť nasledovne:

$$F(x) = f\left[\sum_{i=1}^{N} w_i \varphi(\|x - c_i\|)\right] \qquad 1.16$$

Daný vzťah je v princípe rovnaký z vyššie uvedeným vzorcom na výpočet výstupu perceptrónu (rovnica č. 1.1). Jediný rozdiel je v ich vstupnej informácii, ktorú v tomto prípade reprezentuje výstup bázovej vrstvy. Aktivačná funkcia môže byť napr. sigmoidálna (rovnica č. 1.2).

Učenie tejto siete prebieha s učiteľom a je realizované len na výstupnej vrstve Použité je rovnaké učiace pravidlo ako v prípade perceptrónu so sigmoidou.

2 PROBLÉM SLEDOVANIA LOPTIČKY MOBILNÝM ROBOTOM

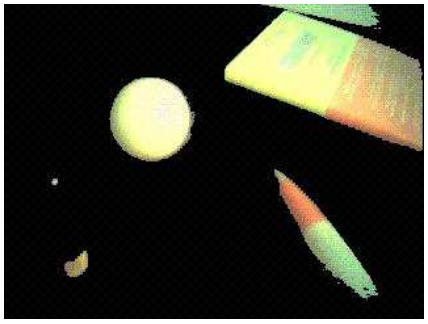
Kapitola je venovaná z pohľadu UNS komplementárnemu prístupu riešenia daného problému.

Metód na spracovanie obrazu za účelom rozpoznania, alebo vyhľadávania objektov je viacero. V prípade ich implementácie si musíme byť vedomí, že algoritmy určené pre aplikácie v mobilných robotoch by nemali byť veľmi výpočtovo náročné.

Konkrétna špecifikácia kapitoly bude smerovaná k problematike spojenej s princípmi algoritmov použitých v reálnom robotovi. Ide o implementáciu tradičného prístupu sledovania bielej loptičky realizovanú Andrejom Lúčnym (2002).

2.1 Predspracovanie obrazu

Vstupný obraz z prostredia pre robota je reprezentovaný kamerou snímajúcou obraz v RBG móde. RGB mód predstavuje farebný model, kde je možné takmer všetky farby vytvoriť prostredníctvom zmiešania troch farebných zložiek. V tomto prípade musíme uvažovať o predspracovaní relatívne zložitého obrazu. Nasledujúci obr. 2.1 predstavuje príklad snímaného obrazu.



Obr. 2.1: Farebný obraz získaný z kamery reálneho robota (Lúčny, 2002)

Pre ujasnenie si uvedieme špecifikáciu predspracovania. Predspracovanie obrazu predstavuje operácie s obrazovými dátami na nízkej úrovni abstrakcie a vykonáva sa po zosnímaní a digitalizácii obrazu. Sonka, Hlavac, Boyle (2007) ho charakterizovali ako proces, ktorý zvyčajne zníži informačnú nosnosť obrazových dát, ktoré nie sú pre ďalšiu analýzu dát relevantné. Niektoré techniky môžu naopak zlepšiť určité povahové črty obrazových dát ako potlačenie deformácie a skreslenia obrazu.

Prvá technika použitá na predspracovanie obrazu, ktorú si priblížime v nasledovnej podkapitole je konverzia obrazu do škály šedí.

2.1.1 Konverzia obrazu do škály šedí

O obraze generovanom v RGB móde vieme povedať, že každý bod (pixel) v tomto obraze je kódovaný 24 bitmi. Je to zapríčinené tromi farebnými zložkami R – červená, G – zelená, B – modrá a im (pre každú zvlášť) prislúchajúcou 8 bitovou reprezentáciou farebnej intenzity (3*8=24 bitov). Osem bitov každej farebnej zložky vypovedá o kódovaní farebnej intenzity v intervale od 0 po 255. Plne farebný obraz v RGB móde vzniká vzájomným zmiešaním všetkých troch intenzít jej prvkov. Z toho vyplýva skutočnosť, že máme 2563 kombinácií pre každý bod obrazu. Takýto obraz by bol bez predspracovania zdrojom mnohých komplikácií pri rozoznávaní hľadaného objektu. Pri snahe zjednodušiť proces rozoznávania loptičky je vhodné obraz transformovať do škály šedí. Škála šedí reprezentuje, ako sa dá predpokladať i z jej názvu, v porovnaní s RGB už len jedná farebná zložka s rovnakým intervalom farebnej intenzity ako v RGB v intervale <0, 255>.

Nižšie uvedený výraz reprezentuje výpočet hodnoty obrazového bodu v škále šedí:

$$I = 0.255R + 0.587G + 0.114B \qquad 2.1$$

Ako máme možnosť pozorovať na obr. 2.2, transformácia výstupu kamery z RGB módu do škály šedí nám ponúka značné zjednodušenie vstupných dát, pričom nestrácame informáciu o tvaroch objektov na scéne, čo je pokladané za prínos.

Obr. 2.2: Obraz získaný z kamery reálneho robota predspracovaný do škály šedí (Lúčny, 2002)

2.1.2 Detekcia hrán

Ďalší z možných krokov k zjednodušeniu obrazu, ktoré konštruktéra robota doviedli k rozoznaniu loptičky vo vstupnom obraze, je detekcia hrán objektov. Detekcia hrán sa

podľa Sonka, Hlavac, Boyle (2007) radí medzi gradientné operácie, ktoré signifikantne redukujú dáta obrazu, pričom neznižuje zrozumiteľnosť jeho obsahu v mnohých smeroch. Gradientné operácie zjednodušene vyhľadávajú zmeny jasovej funkcie s najväčším gradientom. V podstate ide o hľadanie nespojitostí v obrazových dátach, pričom sa do úvahy berie konkrétny pixel a jeho blízke okolie.

Veľkosť gradientu dvojrozmernej skalárnej funkcie v bode so súradnicami *(x, y)* vieme vypočítať nasledovne:

$$|g(x,y)| = \sqrt{\left(\frac{\partial g}{\partial x}\right)^2 + \left(\frac{\partial g}{\partial y}\right)^2} \qquad 2.2$$

kde vektor *g(x, y)* reprezentuje uhol v radiánoch, teda je orientovaný v smere maximálnej strmosti funkcie *g(x, y)* v danom bode.

Keďže sa jedná o diskrétnu realizáciu výpočtu gradientu, je nevyhnutné príslušné parciálne derivácie aproximovať v určitom okolí bodu, napr. 8-susednosť, teda pre masku s rozmerom 3x3. V prípade spomínaného mobilného robota bol na detekciu hrán použitý jeden z možných prístupov zvaný Sobelov operátor.

Sobelov operátor používa prvú parciálnu deriváciu, čo hovorí o jeho smerovej závislosti. Z toho vyplýva, že pre každý bod sa zisťuje skutočnosť, či ním hrana prechádza. Ak áno, tak vieme určiť aj jej smer.

Existuje viac druhov masiek Sobelovho operátora. Lúčny (2002) zvolil nasledovnú:

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad 2.3$$

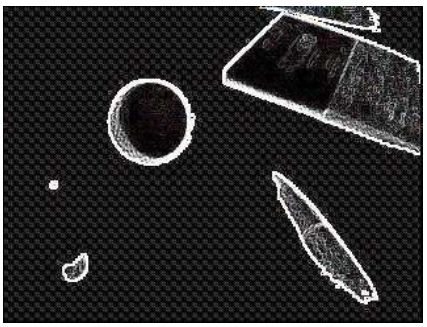
Hodnoty jasu daných obrazových bodov vstupu:

$$\begin{bmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \end{bmatrix} \qquad 2.4$$

Nasledujúci výraz reprezentuje výpočet novej jasovej hodnoty obrazového bodu s polohou *(i, j)*:

$$z(i,j) = (-1)x_1 + 0x_2 + 1x_3 + (-2)x_4 + \ldots + 1x_9 \qquad 2.5$$

Po aplikácií Sobelovho operátora autor uvádza zvýraznenie vertikálnych hrán, pričom boli horizontálne hrany zanedbané. Problém horizontálnych hrán bol vyriešený otočením Sobelovej masky o 90° a následným prekrytím výsledných obrazových dát. Výsledok je zobrazený na obr. 2.3.



Obr. 2.3: Detekcia hrán z obrazu získaná aplikáciou Sobelovho operátora (Lúčny, 2002)

Obrazové dáta sú v tejto časti reprezentované v intervale <0, 255>.

Nevýhodou takéhoto prístupu je nedostatok vhodných dôvodov na výber správnej (konkrétnej) veľkosti okolia pre operátora.

2.1.3 Segmentácia prahovaním

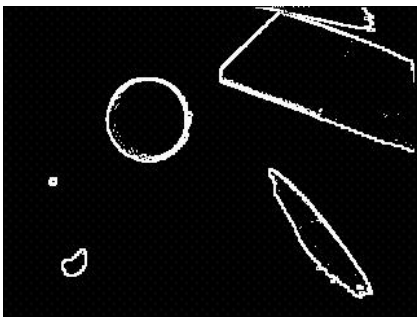
Pre potrebu rozoznania loptičky na scéne je vhodnejšia reprezentácia obrazových dát v intervale <0, 1> než v intervale <0, 255>. Za účelom takéhoto druhu optimalizácie Lúčny (2002) zvolil metódu prahovania.

Prahovanie je jedna z najstarších a výpočtovo najmenej náročných segmentačných techník. Hlavný predpoklad prahovania je, že väčšina objektov a im patriace časti obrazu sú reprezentované zhruba konštantnou schopnosťou odraziť, či pohltiť svetlo.

Za predpokladu, že v spracovávanom obraze je dostatočne veľký kontrast medzi rozpoznávaným objektom a zvyškom scény, ako je to v prípade bielej guľôčky a jej okolitého prostredia, použiť prah, ktorý reprezentuje úroveň jasová za účelom oddelenia objektu s daným prahom od zvyšku obrazu. Princíp konverzie je jednoduchý. Jasová intenzita každého obrazového bodu vstupných dát sa porovnáva s vopred empiricky určeným prahom. Ak intenzita daného bodu presahuje hodnotu prahu, algoritmus pridelí daný bod objektu, teda nahradí jeho jasovú hodnotu 1 inak 0.

Do algoritmu sa môže rovnako pridať okolie spracovávaného bodu. Táto podmienka má za úlohu redukovať osamostatnené body ako napr. šum.

Autor uvádza doplňujúcu informáciu, že nájdenie vhodného prahu nie je jednoduché. Závisí to od snímaného prostredia, osvetlenia a tak podobne. V danej implementácii použil prahovú hodnotu na úrovni 200. Na obr. 2.4 je možné pozorovať značnú redukciu šumu.



Obr. 2.4: Segmentácia prahovaním (Lúčny, 2002)

2.2 Rozpoznanie loptičky

Posledná fáza implementácie je zameraná na rozoznanie loptičky v predspracovanom obraze. Pre ujasnenie uvedieme, že zatiaľ bol obraz upravený do stavu, kde ho reprezentujú dáta v intervale *<0, 1>* pomocou jeho konverzie do škály šedí, následnej detekcie hrán a prahovania. Všetky tieto kroky boli nevyhnutné pre aplikáciu nasledovného, autorom zvoleného algoritmu, ktorý má za úlohu rozpoznanie loptičky vo vstupných dátach.

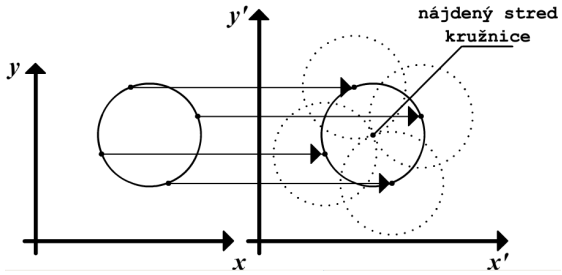
2.2.1 Houghova transformácia

Houghova transformácia sa využíva najmä pri detekcii objektov s vopred známym tvarom. Pôvodne bola navrhnutá na detegovanie rovných čiar, no neskôr bola upravená i na detegovanie kriviek. Jej hlavnou výhodou je robustnosť. Poradí si s rozoznávaním objektov, ktoré sa prekrývajú, i v prípade použitia zašumených vstupných dát. Houghova transformácia pracuje na nasledovnom princípe.

- Pre každý obrazový bod nájdenej hrany sa generujú kružnice do parametrického priestoru s polomerom *r* z intervalu *<a, b>*, pričom *a, b* reprezentujú hranice vhodne zvoleného intervalu polomeru kružníc, ktoré budú generované. Autor použil *r* z intervalu <10, 200>.
- Pre každú vygenerovanú kružnicu si algoritmus ukladá polohu jej bodov.
- Pre každý bod prienikov rovnakého *r* počítame ich sumu.

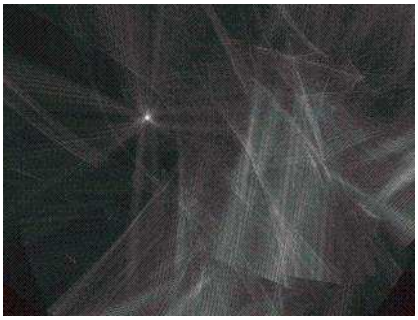
- Bod s najvyšším počtom prienikov reprezentuje stred nájdenej kružnice s daným r.

Obr. 2.5 ilustruje aplikovanie Houghovej transformácie z priestoru *x, y* do parametrického priestoru.



Obr. 2.5: Ilustrácia Houghovej transformácie

Nasledovný obr. 2.6 reprezentuje predspracovaný obraz po Houghovej transformácii. Jasný biely bod poukazuje na stred hľadaného objektu.



Obr. 2.6: Predspracovaný obraz po Houghovej transformácii (Lúčny, 2002)

Nevýhodou tejto transformácie je veľká výpočtová náročnosť.

2.3 Navádzanie robota

Po dokončení implementácie vyššie spomenutých algoritmov autor spomína poslednú fázu implementácie. Zo získaného obrazu bolo potrebné vygenerovať adekvátne príkazy pre podvozok robota s ohľadom na jeho polohu voči loptičke. Za týmto účelom v prezentácii uvádza nasledovnú stratégiu:

- ak je loptička veľmi vpravo, vydávame príkaz na otáčanie sa doprava,
- ak je veľmi vľavo, doľava,
- ak je loptička veľmi veľká, cúvame,
- ak veľmi malá, ideme dopredu.

2.4 Dosiahnuté výsledky

Lúčny (2002) v prezentácii neuvádza správanie robota, ale po návšteve jeho prezentácie venovanej tomuto robotovi sa dozvedáme, že robot je schopný vykonávať od neho očakávané reakcie na podnety relatívne dobre.

3 NÁVRH SIMULÁCIE

K dosiahnutiu uspokojivého správania sa simulácie v krátkom časovom horizonte by mal napomôcť správny návrh postupnosti plánovaných krokov. V tejto kapitole sa budeme zaoberať návrhom celej simulácie od 3D prostredia až po návrh UNS.

3.1 Návrh 3D prostredia

3D prostredie, v ktorom sa má odohrávať simulácia, by malo byť hlavne v počiatočnej fáze trénovania UNS jednoduché. Objekty na scéne určenej na vykonávanie akcie robotického agenta (ďalej len robota) neuvažujeme iné než guľu, ktorú by mal sledovať. Farebné prevedenie budeme škálovať do vysokého kontrastu za účelom jednoduchého rozlíšenia sledovaného objektu (lopty) od zvyšného prostredia.

Na začiatok bude postačovať scéna, inak povedané hracia plocha, kde sa budú môcť pohybovať objekty. Celej hracej ploche bude pridelená čierna farba. Priestor určený na vykonávanie akcie robota bude nevyhnutné pozorovať prostredníctvom kamery, ktorú umiestnime v primeranej výške nad hracou plochou.

Na hraciu plochu umiestnime dva objekty. Jedným z nich je robot a druhým lopta. Loptu bude reprezentovať jednofarebná guľa. Dizajn robota nie je dôležitý, preto bude mať jednoduché telo v tvare kvádra.

Ovládanie oboch objektov v počiatočnej fáze programovania prepojíme na klávesnicu, aby boli overené vlastnosti naprojektovaného prostredia. Ovládanie robota bude neskôr prepojené na UNS.

Za účelom získania vstupu pre UNS je reprezentantovi robota nutné prideliť kameru. Predbežne budeme uvažovať kameru s rozlíšením 150 x 100 pixlov.

Po prevedení úspešného trénovania budú v 3D prostredí uskutočnené zmeny so snahou urobiť prostredie o niečo bližšie k realite. Zmeny sa budú týkať najmä zníženia kontrastu medzi guľou a hracou plochou. Pod znížením kontrastu máme na mysli zmenu ideálnych podmienok v 3D prostredí predpripravenom na trénovanie v 3D prostredí, ktoré bude viac podobné skutočnosti. Prevedieme to pomocou pridania bodového osvetlenia (vznik tieňa) a rovnako je v pláne pridanie stien a podlahy s možnosťou zmien textúr na všetkých spomenutých objektoch. Táto časť bude uskutočnená s cieľom testovania natrénovanej UNS so zašumenými dátami.

Súčasne s vytváraním 3D prostredia budeme pracovať aj na používateľskom prostredí.

3.2 Návrh používateľského prostredia

Na riešenie tohto problému nebude venovaná zvýšená pozornosť ako pri ostatných častiach návrhu. Je to zapríčinené skutočnosťou, že v tomto prípade sa jedná o simuláciu, teda o aplikáciu, ktorá je zameraná na modelovanie určitej situácie, pričom sa kladie dôraz na prevedenie daného modelu, a nie na dizajn a jednoduchosť používateľského prostredia.

Používateľské prostredie budeme modifikovať podľa aktuálnej potreby bez hlbšieho psychologického významu, ako to býva u výučbových aplikácií. Bude musieť spĺňať jedinú podmienku, a to účelnosť.

3.3 Návrh UNS

Podľa rozdelenia funkcií RBF siete budeme členiť aj návrh postupu implementácie.

3.3.1 Bázová vrstva

V prípade implementácie UNS začneme s bázovou vrstvou. Ide o vyhotovenie UNS pozostávajúcej iba z bázových neurónov bez realizovaného prepojenia s vopred naprogramovaným 3D prostredím a rovnako bez vrstvy pozostávajúcej z lineárnych neurónov. Pri zahájení zvolíme nižší počet bázových neurónov s cieľom jednoduchého odstraňovania prípadných chýb v implementácii. V momente, keď bude bázová vrstva vykazovať adekvátnu aktivitu k prezentovaným vstupom, budeme môcť rozšíriť počet jej neurónov na verziu schopnú rozoznávať obrazový vstup v rozlíšení 150 x 100.

Inicializácia centier

Na začiatok môžeme predpokladať polomer každého centra bázového neurónu s veľkosťou 8 pixlov. Medzery medzi konkrétnymi centrami bázových neurónov inicializujeme s rozostupmi 10 pixlov do matice, aby sme poryli celý vstup. Keďže vstup do RBF siete reprezentuje obraz s rozlíšením 150 x 100 bodov, vznikne nám matica s bázovými neurónmi s rozmerom 15 x 10. Takéto pokrytie vstupu by malo zaručiť dostatočnú citlivosť UNS na rozohnanie zmeny na vstupe.

Výstupná funkcia

Neuróny tejto vrstvy budú reprezentovať funkciu gaussovského typu (rovnica č. 1.15).

Citlivosť (σ) najprv nastavíme na rovnakú hodnotu ako r. Za účelom zlepšovania výsledkov UNS ju neskôr budeme podľa potreby obmieňať dovtedy, pokiaľ sa neuspokojíme s dosahovanými výsledkami.

3.3.2 Lineárna vrstva

Ako už vieme bázová vrstva sa napája bezprostredne na lineárnu, ktorá bude slúžiť na navádzanie podvozku robota.

Aktivačná funkcia

Na realizovanie výstupnej vrstvy budú použité perceptróny (rovnica č. 1.1) so sigmoidálnou aktivačnou funkciou (rovnica č. 1.2).

Úloha výstupných neurónov

Na začiatok budeme uvažovať päť neurónov na výstup za účelom vykonávania piatich základných pohybov potrebných na udržanie robota v adekvátnej vzdialenosti od sledovaného objektu. Pohyby budú neurónom pridelené nasledovne, pri zachovaní jeho akcie s ohľadom na poradie zľava doprava od 0. po 4. neurón.

Obr. 3.1: Akcie výstupnej vrstvy

Víťazná akcia bude určená víťazným neurónom. Pod víťazným neurónom rozumieme tú akciu, ktorá dosiahla na aktuálny podnet najvyššiu aktivitu.

Učiaci algoritmus

Ako sme už spomenuli, RBF sieť je v princípe hybridná, pričom jej bázové neuróny nepodliehajú trénovaniu. V tejto vrstve bude trénovaná iba vrstva s lineárnymi neurónmi. Bude potrebné implementovať algoritmus určený na ohodnocovanie aktuálnej polohy lopty voči robotovi. Výstup algoritmu bude reprezentovať číslo neurónu, ktorý bude dosahovať najvyššiu aktivitu.

Spôsob trénovania

Keďže ide o unipolárne sigmoidálne neuróny, tak ich aktivita bude v intervale <0, 1>. Číslo víťazného neurónu generované príslušným algoritmom (podkapitola 4.1.5) bude zasielané učiacemu algoritmu. Podľa neho bude vedieť rozhodnúť, ako upraviť váhy

neurónov. To znamená, ak mal daný neurón vyhrať, učiacemu signálu priradí 1, inak 0. Celý priebeh trénovania sa bude uskutočňovať na hracej ploche.

V prípade prejavenia ťažkostí pri trénovaní bude architektúra s jednou vrstvou lineárnych perceptrónov doplnená o ďalšiu vrstvu. V takom prípade bude musieť byť prevedená úprava v učiacom algoritme z jednoduchej verzie určenej pre jednovrstvový perceptrón, spomenutý v podkapitole 1.4.1., na učiaci algoritmus so spätným šírením chyby uvedený v tej istej podkapitole.

4 TVORBA SIMULÁCIE

V tejto kapitole si priblížime samotnú tvorbu aplikácie a všetky jej fázy. Výber programovacieho jazyka a vývojového prostredia

Pre realizáciu diplomovej práce použijeme programovací jazyk spoločnosti Sun Microsystem s názvom Java. Hlavnými dôvodmi, ktoré viedli k nášmu rozhodnutiu boli:

- *Relatívna jednoduchosť programovania*.
- *Vývojové prostredie* – Javu je možné programovať vo vynikajúcom, voľne dostupnom vývojovom prostredí Eclipse.
- *Dostupnosť materiálov* – k programovaciemu jazyku existuje množstvo dokumentácie potrebnej k doplneniu absentujúcich vedomostí či už z „bežných" programátorských úkonov alebo aj vedomostí špeciálne týkajúcich sa programovania 3D prostredia určeného pre simuláciu.
- *Multiplatformovosť* – Java má značnú podporu v komunite, ktorá je blízka otvorenému softvéru a ponúka takmer vždy vítanú multiplatformovosť. Táto vlastnosť vyplýva z toho, ako bola navrhnutá. Aplikácie naprogramované v jazyku Java nie je problém preniesť na rôzne počítačové platformy – možnosť skompilovania programu na jednej platforme a spustenia na inej.
- *Bezpečnosť* – pri vývoji zvoleného jazyka sme kládli dôraz na bezpečnosť. Predtým, než sa spustí bytový kód, kontroluje sa jeho syntax. Toto opatrenie zabraňuje pádu programu v prípade poškodeného kódu.
- *Voľná licencia* – v neposlednom rade Sun Microsystem nevyžaduje na používanie programovacieho jazyku Java licenčné zmluvy a ani žiadne povinné aktualizácie ako je to u mnohých iných jazykoch určených na vývoj aplikácií.

Na druhej strane medzi najväčšie nevýhody Javy patrí hardvérová náročnosť jej aplikácií.

4.1 Implementácia 3D prostredia a jeho súčastí

Konkrétny vývoj aplikácie sme začali tvorbou 3D prostredia. Tomuto kroku predchádzalo nadobúdanie skúseností týkajúcich sa problematiky implementovania 3D grafiky v programovacom jazyku Java. Máme na mysli pokusy o vytvorenie

jednoduchých objektov na scéne, ich pohybu, navádzania z klávesnice a podobne. Pri vývoji prostredia by sme si mali byť vedomí podmienok jeho tvorby ako napr. jednoduchosť a funkčnosť.

4.1.1 3D prostredie

Použitý programovací jazyk Java a vývojové prostredie Eclipse zapríčinili hneď v úvode neočakávaný problém. Eclipse pri snahe o spustenie, čo i len jednoduchých aplikácií pracujúcich s príkazmi určených pre 3D grafiku, hlásil výnimku - chýbajúcu knižnicu „j3dcore-ogl“. Tento v podstate nepatrný problém, keďže požadovaná knižnica je na internete dostupná, bol neočakávaný najmä z toho dôvodu, že Java je, ako sme skôr spomenuli, multiplatformová. Požadovanú knižnicu bolo nevyhnutné v prípade operačného systému Ubuntu v 64-bitovej verzii vložiť do nasledovného adresára:

- /usr/lib/jvm/java-6-openjdk/jre/lib/amd64.

Cieľový adresár i verzia knižnice sa líšia v závislosti od operačného systému, pod ktorým chceme program s obsahom 3D grafiky spúšťať. Ďalej budú uvedené cieľové adresáre pre Ubuntu v 32-bitovej verzii a Windows XP taktiež v 32-bitovej verzii:

- /usr/lib/jvm/java-6-openjdk/jre/lib/i386,
- \WINDOWS\system32.

V operačnom systéme Windows sú použité nasledovné štyri knižnice: j3dcore-d3d.dll, j3dcore-ogl.dll, j3dcore-ogl-cg.dll a j3dcore-ogl-chk.dll knižnice, pričom v Linuxe je len jedna.

Postupné vyhľadávanie jednoduchých príkladov z 3D grafiky nás doviedlo k publikácii s názvom Java 3D Programing (Selman, 2002), kde boli pre nás mnohé vhodné príklady implementácie 3D prostredia. V publikácii sa nachádza aj opis zdrojového kódu, v ktorom je naprogramovaná scéna s dvomi na sebe nezávisle pohybujúcimi sa objektmi, ktoré bolo možné ovládať z klávesnice. Objekty majú na hracej ploche s určitými rozmermi voľnosť pohybu v dvoch súradnicových osiach: x-ovej a y-ovej. Oba objekty majú pridelené kamery. Ďalšia kamera je umiestnená nad hracou plochou, ktorá umožňuje sledovať pohyb na nej.

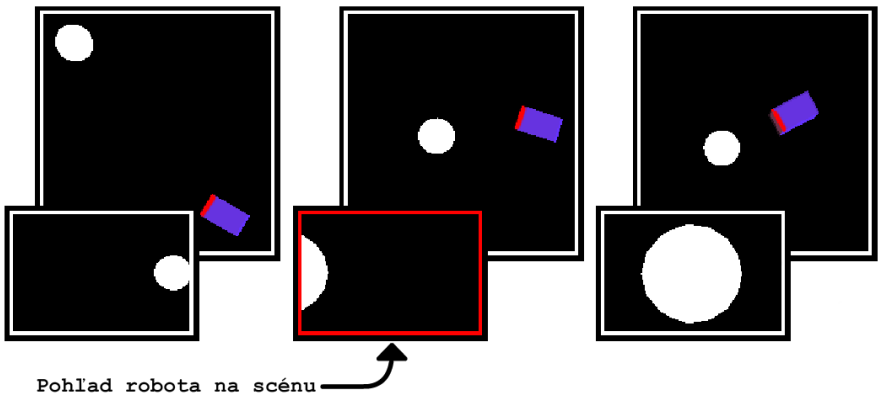
Vyhotovenie opísanej aplikácie do značnej miery vyhovovalo požiadavkám kladeným na prevedenie 3D prostredia určeného pre našu simuláciu. Táto skutočnosť rozhodla

o nasledovnom použití daného kódu ako východiskový bod implementácie potrebného pre našu prácu.

V danom algoritme bolo nevyhnutné previesť nasledovné zmeny.

- Zväčšenie hracej plochy. Pod hracou plochou myslíme priestor, kde sa bude môcť pohybovať robot a lopta bez obmedzení v dvoch súradnicových osiach.
- Zmena tvaru objektov pohybujúcich sa po scéne na guľu a robota, ktorého reprezentuje kváder.
- Doplnenie textúr na steny a podlahy, pričom v tejto fáze boli použité textúry v čiernej farbe. Farba lopty bola nastavená na bielu. Farebné rozlíšenie bolo použité úmyselne s cieľom zvýraznenia kontrastu farieb medzi loptou a zvyškom scény z dôvodu pripravovaného trénovania UNS v tomto prostredí. Dvojfarebná textúra bola pridelená aj robotovi, aby sme jednoduchšie rozpoznávali prednú a zadnú časť robota.
- Nastavenie z-ovej osi súradníc objektov do rovnakej výšky.
- Úprava konštánt pohybu za účelom získania väčšej citlivosti:
  - nastavenie rýchlostí pohybu vpred oboch objektov na hodnotu 0 f, (pričom f predstavuje vzdialenosť na hracej ploche)
  - nastavenie rýchlosti otáčania oboch objektov na hodnotu 0,02 f (v tomto prípade 0,0174 f predstavuje uhol s veľkosťou jedeného stupňa).
- Na záver úprava zorného poľa obidvoch kamier, ktoré boli nastavené na rozmer 100 x 150 pixlov.

Nasledujúci obrázok ilustruje vzhľad 3D prostredia po prevedených úpravách.

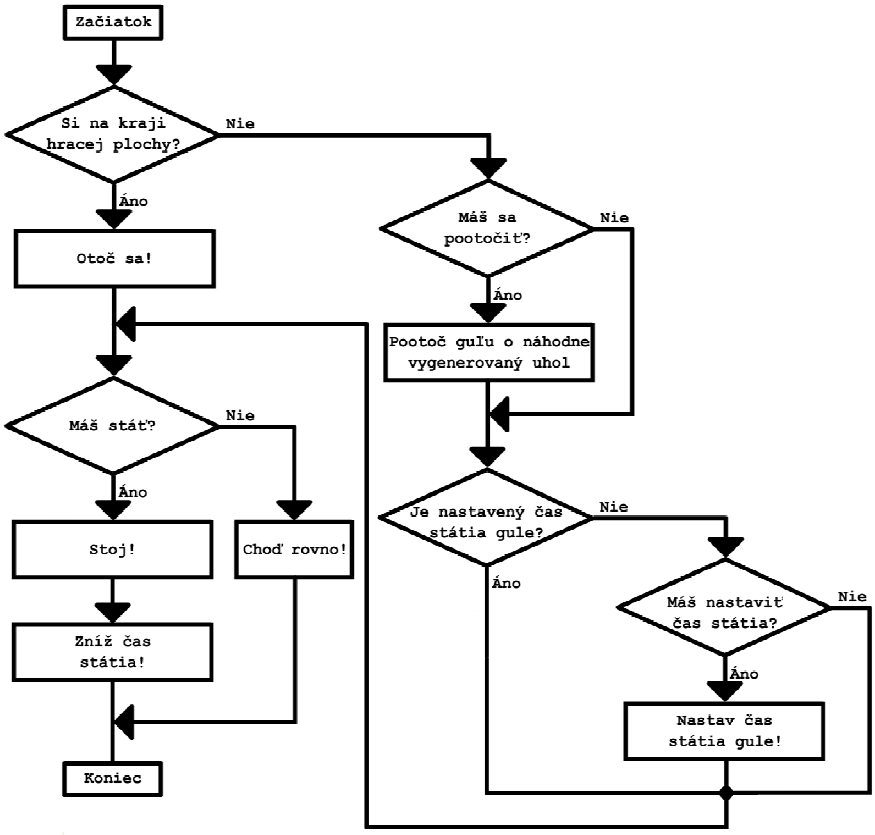


Obr. 4.1: Hracia plocha

4.1.2 Generátor náhodného pohybu lopty

Doposiaľ opísaný stav projektu umožňoval implementáciu algoritmu slúžiaceho na generovanie automatického pohybu lopty po hracej ploche, ktorá bola realizovaná s cieľom automatizácie fázy učenia UNS.

Guľa sa pohybuje nepretržite smerom vpred v rámci hracej plochy. Výnimku tvorí iba situácia, keď je guli nastavený príkaz na zastavenie jej pohybu na určitý čas. Smer a veľkosť uhlu natočenia generujeme náhodne. Algoritmus je bližšie vysvetlený na obr. 4.2 pomocou vývojového diagramu.



Obr. 4.2 : Generátor náhodného pohybu lopty

4.1.3 Transformácia obrazu do škály šedí

Obrazový výstup z kamery robota je reprezentovaný dvojrozmerným poľom, v ktorom sú uložené hodnoty intenzity jasu každého pixlu v 24-bitovom RGB móde. Daný obraz je potrebné predspracovať, aby sme zjednodušili proces trénovania. Zvolili sme konverziu obrazu do 8 bitovej škály šedí. Hodnoty dát v dvojrozmernom poli reprezentujúcom výstup obrazovej informácie robota boli dôsledkom konverzie

transformované do intervalu <0, 255>. Transformovaný obraz je vykresľovaný do používateľského prostredia.

4.1.4 Prahovanie

Po úspešnej aplikácii transformácie obrazu do škály šedí bude vhodné obraz ešte zjednodušiť, keďže zadanie práce obsahuje testovanie modelu so zašumenými vstupnými dátami. V ideálnych podmienkach, t.j. keď bude mať 3D prostredie vysoký kontrast predpokladáme, že UNS by si mala bezproblémovo osvojiť požadované schopnosti. Síce obraz po konverzii do škály šedí je v intervale <0, 255>, ale na scéne sa nám nachádzajú len dve farby, a to čierna = 0 a biela = 255. Komplikácie očakávame až po pridaní bodového osvetlenia alebo textúr. Nami naprogramovaná UNS bude s každým pootočením loptičky, s použitou textúrou alebo tieňom postavená pred zásadne iný vstupný vektor. Vzniknutá situácia by sa trénovala veľmi ťažko (ak by sa vôbec dala natrénovať). Z toho vyplýva dôvod implementácie prahovania. Prahovanie bolo inicializované na hodnotou = 205.

4.1.5 Generátor učiaceho signálu

Po úspešnej implementácii automatického pohybu lopty a prevodu snímaného obrazu z kamery robota prišla na rad implementácia algoritmu určeného na vyhodnocovanie aktuálnej polohy lopty voči robotovi, resp. k jeho zornému poľu kamery. Hlavná úloha algoritmu bude generovanie učiaceho signálu pre UNS vo fáze jej učenia.

Algoritmus pracuje na jednoduchom princípe, ktorý využíva jednoduchosť 3D prostredia vo svoj prospech. Obmedzenie pohybu objektov na scéne len v dvoch osiach, a to x-ovej a y-ovej osi, umožňuje stály predpoklad pohybu objektov pozdĺž horizontálnej osi kamery robota. Umiestnenie objektov v rovnakej výške zaručuje, že každý výskyt lopty v zornom poli robota budeme môcť očakávať v strede jeho zorného poľa.

Algoritmické riešenie problému je nasledovné. V dvojrozmernom poli výstupu kamery transformovaného do škály šedí sa vyhľadá 49. riadok. Tento riadok spolu s 50. riadkom prislúchajú presnému horizontálnemu stredu zorného poľa robota. Každý výskyt lopty je sprevádzaný zmenou aktivity pixlov tohto riadku z čiernej na bielu. Jednoduchou podmienkou pri prehľadávaní daného riadku nájdeme prvý svietiaci pixel, pričom si uložíme jeho súradnice a následne nájdeme aj posledný svietiaci pixel s jeho

súradnicami. Získané súradnice nám dávajú informáciu o aktuálnom priemere lopty v zornom poli. Súčasne si vieme odvodiť aj polohu lopty v priestore, pričom máme na mysli, či je vpravo, vľavo alebo v strede zorného poľa. Podľa veľkosti lopty vieme určiť aj hĺbku priestoru, teda ako ďaleko sa lopta nachádza od robota. Zo získaných informácií je možné vyvodiť päť základných pohybov robota: vľavo, vpravo, dopredu, dozadu a stoj. Tieto pohyby budú slúžiť na učenie UNS.

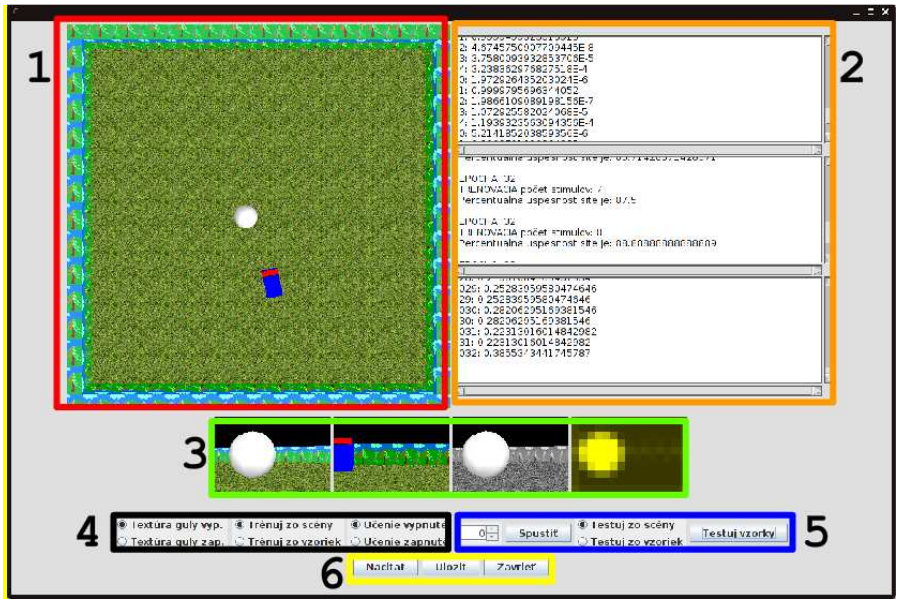
Dôležité je upozorniť na situáciu, keď kamera nebude mať v zornom poli loptu. V takomto prípade sa bude generovať učiaci signál na otáčanie sa robota doprava.

Správnosť učiaceho algoritmu bola testovaná priamym prepojením generovaných signálov určených na trénovanie UNS na pohyb robota.

Nedostatky, ktoré sa v tejto časti prejavili, boli odstránené takmer vždy v momente rozpoznania problému.

4.2 Implementácia používateľského prostredia

Ako sme už skôr spomenuli, dizajn používateľského prostredia aplikácie nie je v tomto prípade prioritou. Z tohto dôvodu bolo upravované väčšinou za „pochodu“. To značí, že dizajn nebol špeciálne pojednávaný pred začatím implementácie, ako to bolo v prípade konkrétneho vzhľadu a správania sa 3D prostredia. Výsledný dizajn používateľského prostredia je možné vidieť na nasledujúcom obrázku.



Obr. 4.3: Používateľské prostredie aplikácie

Na obr. 4.3 máme možnosť vidieť používateľské prostredie aplikácie. Je tam

zvýraznených niekoľko rámikov, ktoré členia aplikáciu do nasledovných celkov.

Rámik č.1 zvýrazňuje hraciu plochu, kde máme možnosť pozorovať robota a loptu.

Rámik č.2 ohraničuje tri polia určené na „servisné“ výpisy, a to nasledovne (zhora nadol):

- aktivácie výstupných neurónov,
- percentuálnu úspešnosť UNS v danej epoche s jej chybou a na záver
- aktivácie neurónov v RBF vrstve.

Rámik č.3 poukazuje na štyri obrazovky, ktoré reprezentujú zľava doprava:

- Obraz z kamery robota.
- Obraz z kamery lopty.
- Obrazovka určená na zobrazovanie obrázkov určených na trénovanie UNS z predpripravených obrazov alebo na vykresľovanie predspracovaného obrazu v škále šedí. Čo daná obrazovka vykresľuje záleží na móde učenia, v ktorom sa nachádzame.
- Obrazovka slúži na zobrazenie aktivácie RBF neurónov.

Rámik č.4 zvýrazňuje (zľava doprava) možnosť zvoliť:

- zapnutie alebo vypnutie textúry lopty,
- trénovanie zo scény alebo zo vzoriek v prípade, ak je učenie zapnuté,
- zapnúť alebo vypnúť trénovanie.

V rámiku č.5 sa nachádza možnosť (zľava doprava):

- Šípky slúžiace k nastaveniu počtu bázových neurónov, ktoré majú byť nulované („zabité“). Interval je obmedzený počtom bázových jednotiek.
- Tlačidlo spustiť, ktoré sa vzťahuje k rámiku č. 4 umožňuje zahájenie testovania alebo trénovania z 3D scény (hracej plochy) a trénovania zo vzoriek.
- Možnosť výberu testovania z pripravených vzoriek alebo z pripravených testov v 3D scéne.

Rámik č.6 poukazuje na tri rôzne tlačidlá, zľava doprava nasledovne:

- Prvé zľava slúži na načítanie váh UNS (za predpokladu, že niekedy pred tým boli už váhy uložené).

- Druhé v poradí slúži na ukladanie váh UNS po trénovaní.
- Posledné slúži na ukončenie programu.

4.3 Implementácia neurónovej siete

V nižšie uvedenom texte si rozoberieme dôležité časti práce týkajúce sa implementácie UNS s príslušnými algoritmami.

4.3.1 RBF vrstva

Bázová vrstva si vyžaduje pre uskutočnenie výpočtov aktivít neurónov vygenerovanie pozície centra zorného poľa, ktorým bude daný neurón získavať jej pridelenú časť vstupného vektoru.

Bázové neuróny vygenerujeme pri inicializácii programu. Každá vygenerovaný neurón je priradený do spájaného zoznamu, s ktorým sa neskôr pracuje ako s reprezentantom vstupnej vrstvy.

Generovanie centier bázových funkcií

Každému bázovému neurónu je nevyhnutné prideliť časť vstupného vektoru, ktorý bude sledovať. V podkapitole 3.3.1 bolo navrhnuté použitie bázových neurónov s r = 8 pixlov. Rozostavené sú v matici s medzerami medzi jednotlivými neurónmi s veľkosťou 10 pixlov. Tabuľka 4.1 zobrazuje implementáciu uvedeného návrhu.

Tabuľka 4.1: Implementácia generátora centier bázových funkcií

```
public void GeneratePositionOfCentroids(){
      // Konštanty vhodné pre rozmer vstupu 150x100.
      int hStep = 10;   // Posun na horizontálnej osi.
      int wStep = 10;   // Posun na vertikálnej osi.
      int hAktual = 4;  // Aktuálna pozícia na horizontálnej osi.
      int wAktual = 4;  // Aktuálna pozícia na vertikálnej osi.
      // Generovanie pozícií centier podľa nastaveného počtu.
      for (int i = 0; i < tmp_centroid.length; i++){
            // Prejdi na nový riadok ak..
            if (hAktual % 154 == 0){ //si prekročil šírku obrazu.
                  hAktual = 4; // Posuň hor. pozíciu na začiatok.
                  wAktual += wStep; // Posun na vertikálnej osi.
            }
            //Inak sa posúvaj na horizontálnej osi.
            tmp_centroid[i][0] = hAktual; // Priraď x-ovú súradnicu.
            tmp_centroid[i][1] = wAktual; // Priraď y-lonovú súrad.
            hAktual += hStep; //Posun na horizontálnej osi.
      }
}
```

Vygenerovaním pozícií centier pomocou *r* umožňuje vypočítanie prislúchajúcej časti výstupu. Zjednodušene povedané, pridelíme im súradnice pixlov vstupu, na ktorých budú sledovať okolie. Vytvorili sme algoritmus, ktorým jednoducho do čierneho plátna s rozmerom výstupu kamery vykreslíme k príslušnému stredu centra bázového neurónu biely kruh. Jeho polomer je vopred zadefinovaný.

Vygenerované centrá reprezentujeme pomocou dvojrozmerného poľa, ktoré je zhodné s rozmerom poľa výstupu kamery. Ako vieme, obrazová informácia bola predspracovaná do intervalu <0, 1>. Aktuálne vygenerované centrá sú v intervale <0, 255>, a preto je potrebné vykonať transformáciu do <0, 1>. Transformácia bola uskutočnená delením hodnoty centra 255. Pole, reprezentujúce centrum bázového neurónu, bude použité na výpočet Euklidovskej vzdialenosti vstupného vektoru od daného centra.

Výpočet výstupu vrstvy

Pre všetky neuróny bázovej vrstvy realizujeme výpočet výstupu. Výpočet každého neurónu sa líši vzhľadom na jeho centrum. Euklidovská vzdialenosť vstupu od centra sa počíta V cykle cez šírku a výšku poľa vstupu. Tabuľka 4.2 zobrazuje implementácia výpočtu bázovej vrstvy.

Tabuľka 4.2: Implementácia výpočtu výstupu bázovej vrstvy

```
public double rbf_GetOutput(int input[][]){
      overlap = 0;
      rbf_output = 0;
      // Prechádza cez celý vstup
      for (int i = 0; i < input.length; i++) { // výška
            for (int j = 0; j < input[i].length; j++) { // šírka
                  // Vyp. Euklidovskú vzdialenosť akt.Centra od vstupu.
                  if ( centroid[i][j] == 1 ) {
                  overlap += Math.pow((input[i][j]–centroid[i][j]), 2);
                  }
            }
      }
      // Vypočíta sa Gaussián (sigma bola inicializovaná ^2).
      rbf_output = Math.exp( -overlap / ( sigma * 2 ) );
      return rbf_output;
}
```

Ukončením implementácie bázových neurónov sme dosiahli nevyhnutný kontakt s UNS s okolitým prostredím. Vzájomné prepojenie medzi RBF sieťou a 3D prostredím je opísané v kapitole 4.4. Teraz nás čaká implementácia výstupnej vrstvy a jej učiacich mechanizmov. Následne si budeme môcť overiť schopnosť UNS osvojiť si požadované vlastnosti.

4.3.2 Výstupná vrstva

Výstupná vrstva bola naprogramovaná ako druhá v poradí. Dôvodom bol fakt, že na výpočet výstupu lineárnej vrstvy sa používa výstup z bázovej vrstvy.

Generovanie lineárnych neurónov prebieha analogicky ako v prípade bázových neurónov pri inicializácii programu. Aj tieto neuróny sú prideľované do spájaného zoznamu, s ktorým sa pracuje ako s celou vrstvou.

Výpočet výstupu vrstvy

Vrstve prichádza na vstup jednorozmerné pole s výstupmi bázových neurónov.

Nasledovná tabuľka zobrazuje konkrétnu implementáciu výpočtu výstupu jedného neurónu tejto vrstvy.

Tabuľka 4.3: Implementácia výpočtu aktivácií výstupnej vrstvy

```
public double GetOutput( double[] input ) {
      double net = 0;
      // Net = suma cez všetky váhy a vstupy.
      for (int i = 0; i < number_of_inputs; i++) {
            net += input[i] * weights[i];
      }
      // Výpočet aktivačnej funkcie.
      output = ActivationFunction(net- threshold);
      return output;
}
```

Kde je sigmoidálna aktivačná funkcia reprezentovaná:

Tabuľka 4.4: Implementácia výpočtu sigmoidálnej aktivačnej funkcie

```
public double ActivationFunction(double net){
      return 1.0 / ( 1.0 + Math.pow( Math.E,( -1.0 * net ) ) );
}
```

Výstup aktivačnej funkcie neurónu (v intervale <0, 1>) reprezentuje jeho výstup.

4.3.3 Učiaci algoritmus

Záverečným krokom v programovaní UNS je učiaci algoritmus. Ako vieme, na úpravu váh je nevyhnutné vedieť, ako mala byť aktívna UNS a aká bola. Číslo víťazného neurónu (v algoritme: `index_of_winer`) sa zvolí v generátore učiaceho signálu (podkapitola 4.1.5). Nasledovná tabuľka zobrazuje implementáciu učiaceho algoritmu pre jeden neurón.

Tabuľka 4.5: Implementácia učiaceho algoritmu

```
public void Train(double[] input, int index_of_winner) {
      double d; // Učiaci signál.
      // Podmienka pridelí učiaci signál podľa toho,
      if ( index_of_winner == neuron_id ) { //
            d = 1; // či neurón vyhral
      } else {
            d = 0; // alebo nie.
      }
      // Vzorec na úpravu váh.
      for ( int i = 0; i < weights.length; i++ ) {
            weights[i] += alpha * (d - output) * input[i];
      }
      // Vzorec na úpravu prahu.
      threshold += alpha * (d - output) * (-1.0); // update threshold
      // Učenie je možné vypnúť po ustálení celkovej chyby siete
      neuron_error = Math.pow((d - output), 2) / 2;
      epocha++; // alebo po určitom počte epoch.
}
```

V algoritme je taktiež možné vidieť naprogramovanú chybovú funkciu a počítadlo epoch. Spomenuté doplnky slúžia na zastavenie trénovania podľa potreby. Buď je to ustálenie chyby, alebo po prekonaní počtu učiacich epoch.

Aktuálny stav implementácie umožňoval prepojenie UNS s 3D prostredím. Nasledovná podkapitola bude venovaná tomuto problému.

4.4 Prepojenie UNS a 3D prostredia

V predchádzajúcich častiach sme pracovali s dvomi samostatnými celkami. Jeden z nich bola UNS a druhým 3D prostredie. Za účelom splnenia zadania musíme spomenuté časti navzájom prepojiť.

Obe časti bežia v samostatných vláknach. Kvôli oživeniu odozvy používateľského prostredia sme navádzanie robota pomocou UNS a automatický pohyb lopty presunuli do novovytvoreného vlákna určeného pre UNS. Vlákno pre UNS rieši nasledovné úlohy:

1. snímanie obrazu z pohľadu robota a jeho predspracovanie (podkapitola 4.1.3 a 4.1.4),
2. ohodnotenie polohy lopty voči robotovi (podkapitola 4.1.5),
3. výpočet výstupu celej UNS na základe získanej obrazovej informácie a jej trénovanie v prípade, ak je zapnuté (podkapitola 4.3),
4. vykonanie automatického pohybu gule (podkapitola 4.1.2).

Vieme, že loptu je možné navádzať z klávesnice alebo pomocou automatického generátora pohybu. V prípade povoleného automatického generátora môžeme stále využiť možnosť navádzania lopty prostredníctvom klávesnice. Kombinácia oboch navádzaní spôsobí, že navádzaním lopty pomocou klávesnice zasahujeme do automatického navádzania lopty (dochádza k súpereniu navádzaní).

Prepojenie oboch celkov nám umožňuje vykonanie testov, ktorými sa budeme zaoberať v nasledovnej kapitole.

5 VÝSLEDKY

Záverečná časť práce bude venovaná trénovaniu UNS a dosiahnutým výsledkom. V úvode sa vrátime do štádia testovania samotnej bázovej vrstvy a následne sa posunieme na fázu celkového trénovania siete spojenú s prezentovaním dosiahnutých výsledkov.

5.1 Testovanie bázovej vrstvy

Bázová vrstva opísaná v podkapitole 4.3.1 bola bezprostredne po implementácii podrobená jednoduchému testovaniu, pričom sa ukázali jej mierne nedostatky. Hľadanie zdroja problému a snaha o zjednodušenie orientácie v aktiváciách bázových neurónov nás doviedla k realizovaniu vizuálneho zobrazenia aktivity bázovej vrstvy.

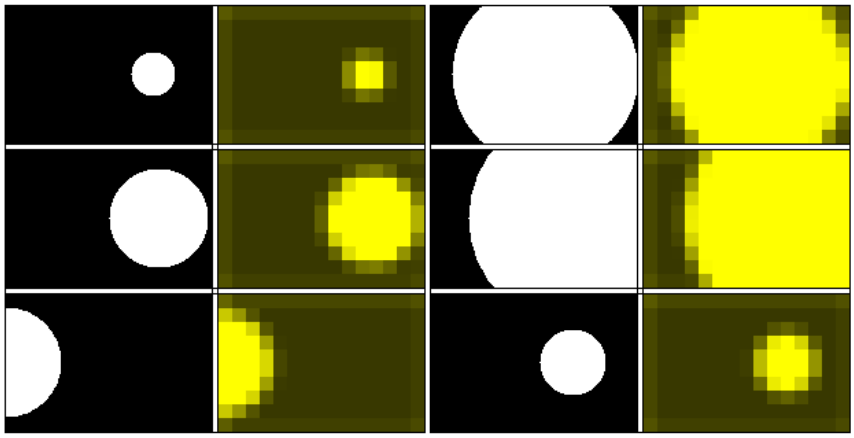
5.1.1 Vizualizácia aktivity RBF neurónov

Algoritmus zobrazenia pracuje na jednoduchom vykresľovaní aktivácií jednotlivých neurónov vynásobených hodnotou 255 v rámci niektorej z farebných zložiek RGB módu. Aktivácie sú vykreslené v matici, podľa ich umiestnenia na snímanom obraze, kamerou robota.

Vizualizácia aktivít bázových neurónov sa preukázala ako dobre zvolený krok v odstraňovaní chýb. Pozorovanie aktivít vo viacerých prípadoch ukázalo príliš malé rozdiely v aktiváciách výstupoch neurónov v prípade prezentácie širokého spektra stimulov.

Opätovná revízia kódu odhalila nepozornosť v použití vzorca na výpočet výstupu bázového neurónu, konkrétne časti týkajúcej sa výpočtu Euklidovskej vzdialenosti medzi vstupným a výstupným vektorom. Upravenie kódu prinieslo definitívne jasne rozpoznateľné rozdiely v aktiváciách bázových neurónov.

Nasledovný obr. 5.1 ilustruje aktivácie bázových neurónov na niekoľko rôznych vstupov.

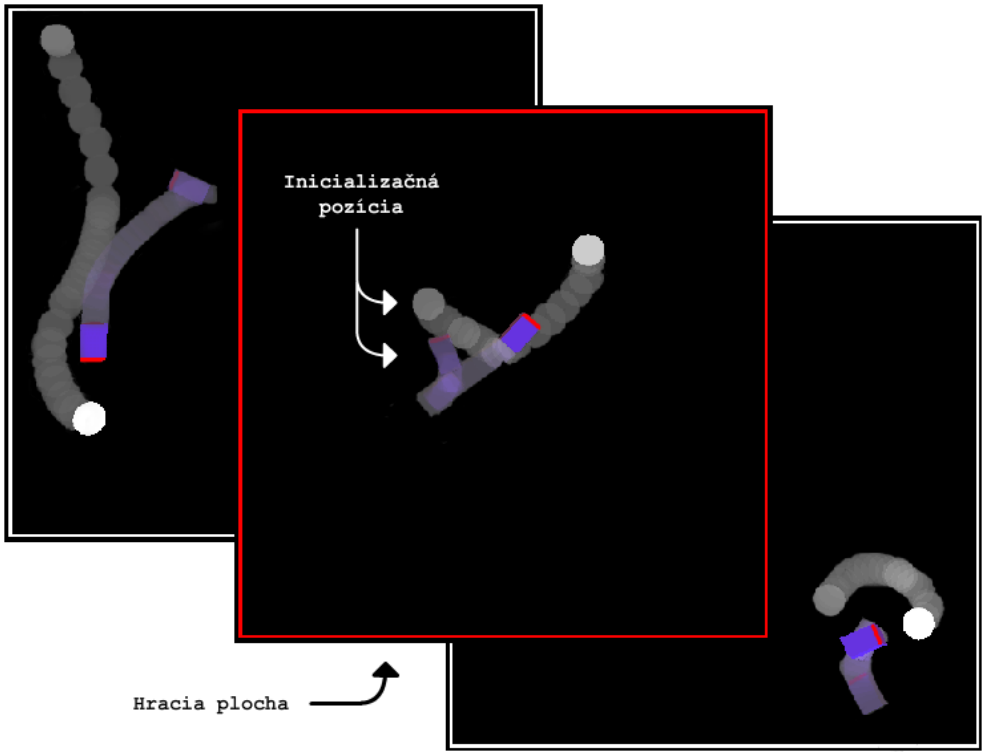


Obr. 5.1: Rôzne aktivácie bázových neurónov

Už na prvý pohľad sú jednoducho a bezproblémovo rozoznateľné správne aktivácie bázovej vrstvy. Plne aktívny neurón dosahuje hodnotu výstupu = 1.

5.2 Úprava algoritmu na trénovanie UNS

Pri návrhu simulácie sme uvažovali o jednom druhu trénovania, a to zo scény pomocou automatického generátora pohybu lopty, alebo pri jej ručnom navádzaní z klávesnice. Nasledujúci obr. 5.2 ilustruje reakciu robota na pohybujúcu sa loptičku po scéne.

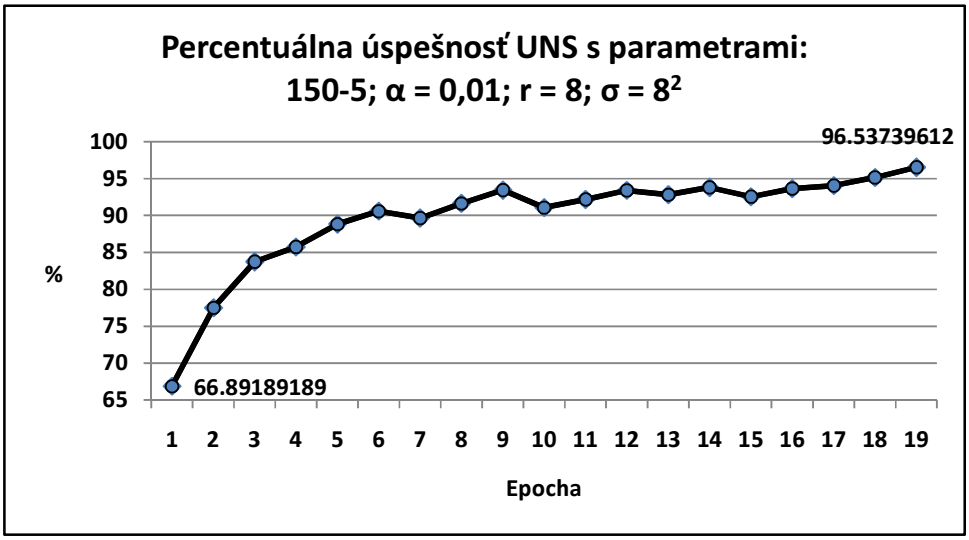


Obr. 5.2: pohyb robota po scéne vo fáze trénovania z 3D prostredia

Pri prvej konfrontácii s trénovaním UNS v 3D prostredí implementovaná UNS preukazovala prijateľné schopnosti osvojovať si požadované vlastnosti. Napriek dobrému výsledku bol zmenený názor na zvolený spôsob trénovania. Trénovanie zo scény sa javilo ako nie príliš objektívne. Za účelom zobjektívnenia fázy trénovania bol dodatočne implementovaný algoritmus na trénovanie z vopred vygenerovaných stimulov, pričom sa ohodnocovala chyba UNS a percentuálna úspešnosť na trénovanej množine v jednotlivých epochách.

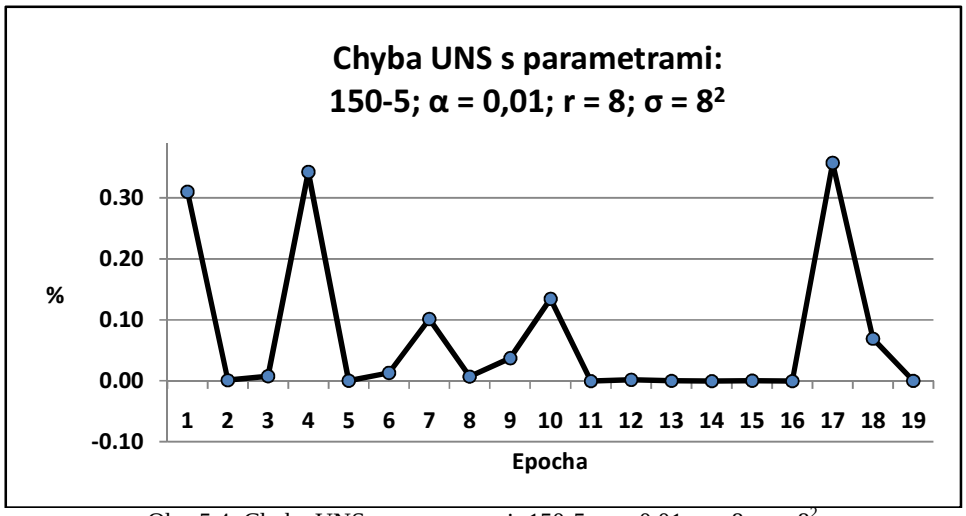
Algoritmus na generovanie stimulov je veľmi jednoduchý. Do čierneho plátna sa na 49. riadku od jeho nultého pixlu generujú kruhy s nasledovnými polomermi: r(t) = 4, r(t+1) = r(t) + 4, r (t+2) = r(t+1) + 4, …, až po r = 80 pixlov, vždy s rozostupmi 4 pixle. Z toho vyplýva, že dostaneme 18 kruhov na jednu súradnicu x, y, teda celkový počet stimulov (vynímajúc čierne) je rovný 740. Počas trénovania sa predkladané stimuly zobrazujú v používateľskom prostredí (podkapitola 4.2).

V grafe na obr. 5.3 je zobrazená percentuálna úspešnosť UNS s architektúrou, ktorej vlastnosti boli už spomenuté kapitole týkajúcej sa návrhu UNS. Počet bázových neurónov: 150 ks; r = 8; $\sigma = 8^2$; s odstupom medzi jednotlivými neurónmi = 10; inicializácia prvého neurónu na súradniciach x = 4, y = 4. Na riadenie bolo použitých päť výstupných neurónov. Topológiu UNS budeme ďalej označovať nasledovne: 150-5, kde prvé číslo bude hovoriť o počte bázových neurónov a druhé o počte výstupných neurónov.



Obr. 5.3: Percentuálna úspešnosť UNS s parametrami: 150-5; α = 0,01; r = 8; $\sigma = 8^2$

Vidíme, že UNS dosiahla v 19. epoche vyššiu úspešnosť ako 96%, čo môžeme pokladať za vynikajúci úspech. Nasledovný graf na obr. 5.4 Obr. 5.4zobrazuje vývoj chyby.



Obr. 5.4: Chyba UNS s parametrami: 150-5; α = 0,01; r = 8; σ = 8^2

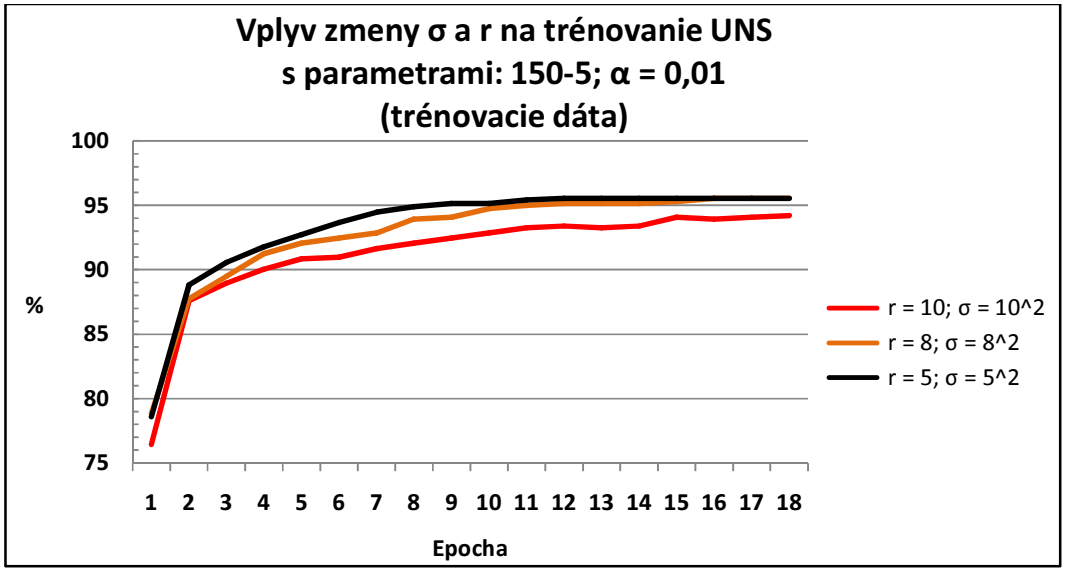
Pri snahe nájsť optimálne nastavenia UNS bola množina stimulov rozdelená na testovaciu a trénovaciu. Pri pokusoch však UNS veľké rozdiely percentuálnej úspešnosti na trénovacej a testovacej množine neukazovala. Rôzne modifikácie pomerov skupín mali vždy počas trénovania tendenciu vykazovať zhruba rovnakú úspešnosť UNS. Táto skutočnosť nás doviedla k vygenerovaniu ďalšej množiny obrázkov, teraz o čosi zložitejšej množiny za účelom ich použitia v testovacej množine.

V tomto prípade sa generovanie začalo na pomyselnom -11 pixli plátna s kruhmi s polomerom väčším ako je vzdialenosť do 0. pixlu plátna a končilo na pomyselnom 160. pixli plátna s kruhmi opäť väčšími ako je rozdiel medzi reálnou veľkosťou plátna a aktuálnou polohou algoritmu na vykresľovanie. Veľkosť polomerov bola nastavená od troch pixlov s krokom 3 až po veľkosť r = 80 pixlov. Takýmto algoritmom sme získali ďalších 1490 stimulov s celými alebo s časťami loptičiek. Po doplnení súboru stimulov o 6 čiernych sme získali 1496 stimulov určených na použitie výpočtu percentuálnej úspešnosti na testovacej množine.

5.3 Hľadanie optimálnej architektúry UNS

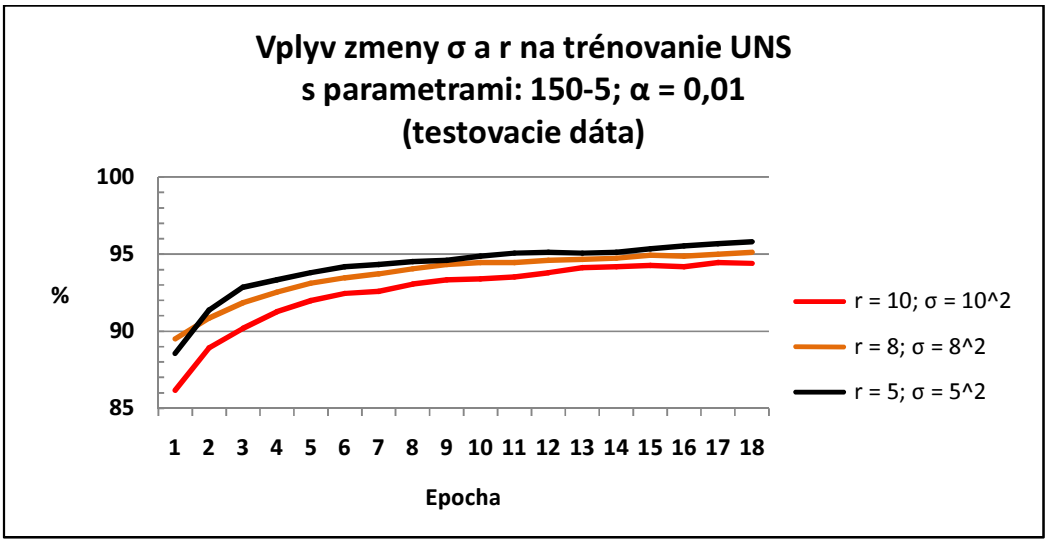
Za účelom dosiahnutia čo najlepších výsledkov v testovaní UNS v 3D prostredí sa teraz budeme venovať hľadaniu parametrov UNS s čo možno najlepšími dosahovanými

výsledkami. Nižšie uvedený graf na obr. 5.5 ukazuje vplyv zmeny σ na percentuálnu úspešnosť UNS pri trénovaní.



Obr. 5.5: Vplyv zmeny σ a r na trénovanie UNS s parametrami: 150-5; α = 0,01(trénovacie dáta)

V grafe sú zobrazené tri krivky znázorňujúce percentuálnu úspešnosť siete s rovnakou topológiou 150-5 iba so zmenenou sigmou. Oranžová čiara reprezentuje $\sigma = 8^2$, čierna reprezentuje $\sigma = 5^2$, a červená $\sigma = 10^2$. Nasledujúci graf na obr. 5.6 znázorňuje vplyv zmeny σ a r na trénovanie UNS.



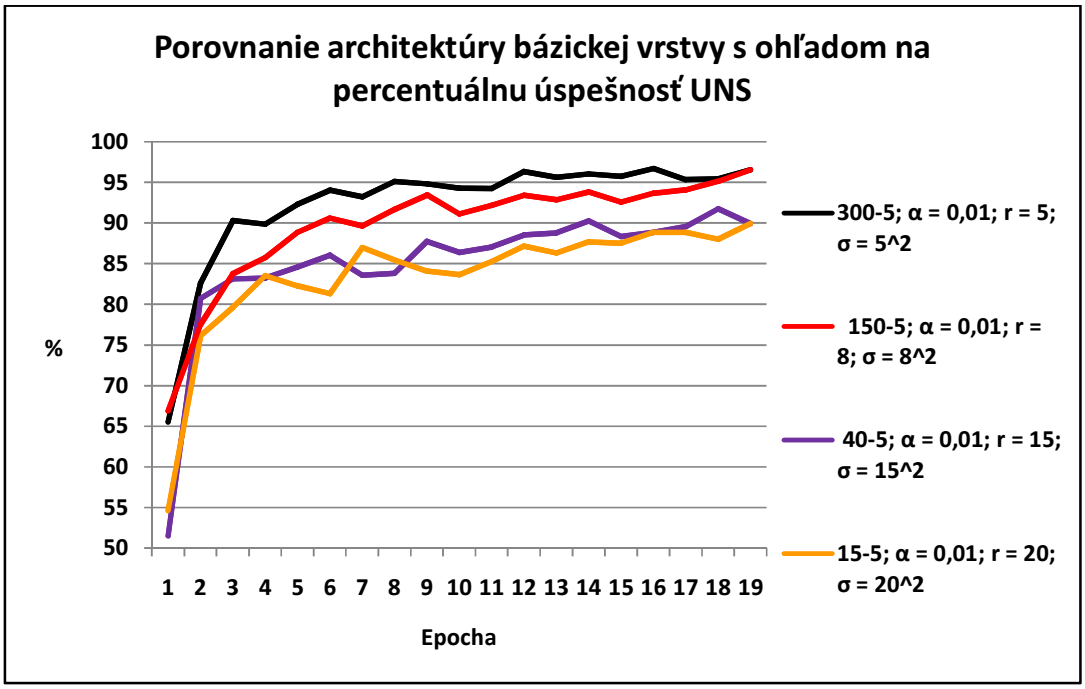
Obr. 5.6: Vplyv zmeny σ a r na trénovanie UNS s parametrami: 150-5; α = 0,01(testovacie dáta)

Pri porovnaní kriviek vidíme, že rozdiel na trénovacích a testovacích dátach nedosiahol veľký rozdiel. Ďalej vidíme, že pri danej architektúre 150-5 boli zhruba rovnako

úspešné nasledovné nastavenia bázickej vrstvy: $\sigma = 5^2$, r = 5 a $\sigma = 8^2$ a r = 8. Vidíme, že prvé spomenuté parametre dopadli o trošku lepšie, čo nebolo očakávané. Očakávanie bolo odvodené od skutočnosti, že v bázovej vrstve boli nastavené medzery medzi jednotlivými neurónmi 10 pixlov. S nastavením na bázickej vrstve r = 5 boli na stimule miesta, ktoré nesnímal ani jeden z bázových neurónov. Výsledok je zrejme spôsobený čistotou stimulov a veľkosťami polomerov kruhov na stimuloch. Ani jeden z prezentovaných stimulov nemal polomer zobrazenej lopty menší ako 4 pixle, čo pri danej čistote dát nespôsobilo žiadne problémy pri rozoznávaní. Pod čistotou myslíme to, že tam nie je žiaden šum. Sú to ideálne stimuly.

5.4 Vplyv zmeny architektúry bázovej vrstvy na celkovú úspešnosť UNS

Nielen zmena *σ* a *r* zorného poľa bázových funkcií má vplyv na celkovú efektivitu UNS, ale samozrejme aj jej topológia. V prípade najjednoduchšieho typu RBF siete, t.j. s tromi vrstvami, ako bolo opísané v podkapitole 1.5, sa dá manipulovať len s počtom skrytých neurónov, teda bázových. Nasledovný obr. 5.7 znázorňuje graf s porovnaním percentuálnej úspešnosti štyroch rôznych architektúr .



Obr. 5.7: Porovnanie architektúry bázickej vrstvy s ohľadom na percentuálnu úspešnosť UNS

Vidíme, že najlepšiu percentuálnu úspešnosť dosiahla UNS s topológiou 300-5. Táto UNS bola v podstate výpočtovo náročnejšia ako doteraz niekoľkokrát testovaná

topológia 150-5, ale v konečnom dôsledku sme po 19 epochách dosiahli zhruba rovnakú úspešnosť (96%). Testovanie v 3D prostredí s vypnutým učením nepreukázalo viditeľné problémy robota pri sledovaní lopty. Chýbajúce 4% predstavovali najmä stimuly, ktoré sa nachádzali na rozhraní dvoch pohybov a stimuly, ktoré reprezentovali loptu s malým polomerom.

Ďalšie dve spomenuté architektúry dosiahli podstatne nižšiu úspešnosť. Architektúry 40-5 a 15-5 neboli po ukončení 19. epochy trénovania schopné vykonávať pohyb v prípade, keď bolo potrebné nasledovať loptu. Je to pravdepodobne spôsobené priveľkým zorným poľom bázového neurónu potrebným na rozoznanie vzdialenej lopty. Robot zvládal natáčanie v smere lopty i cúvanie, no nasledovať loptu bolo v podstate nemožné.

5.5 Testovanie natrénovanej UNS v 3D prostredí so zašumenými dátami

Po porovnaní percentuálnej úspešnosti viacerých architektúr UNS bola za účelom ďalšieho testovania v 3D scéne vybraná architektúra so 150 bázovými neurónmi s r = 8 a $\sigma = 8^2$. Dôvodom bola jej nie príliš veľká výpočtová zložitosť a dobrá percentuálna úspešnosť. Učenie bolo zastavené po 26. epoche pri 96.86% úspešnosti. Takto naučená RBF sieť nemala pri testovaní v jednoduchom prevedení 3D scény žiadne viditeľné problémy so sledovaním lopty. Ako sme už spomenuli, zadanie práce vyžaduje testovanie efektívnosti natrénovanej RBF siete so zašumenými dátami. Pod zašumením dát rozumieme zvýšenie zložitosti vstupných dát pridaním textúr na scénu a loptu alebo bodového osvetlenia lopty. V nižšie uvedených testoch sa nevykonala žiadna zmena vo váhach UNS. Našou snahou je poukázať na všeobecnú schopnosť UNS – generalizáciu. Na uskutočnenie objektívnych testov bol dodatočne implementovaný algoritmus, v ktorom vykoná lopta niekoľko presne zadaných akcií.

5.5.1 Algoritmus testovania úspešnosti UNS

V 3D prostredí sme realizovali testy, pomocou ktorých sme do určitej miery objektívne ohodnotili pohyb robota voči aktuálnej polohe lopty. Test sa skladal z dvoch hlavných častí:

1. test so statickou loptou,
2. test s dynamickou loptou.

Prvý test sa skladal z 5-tich častí. Pri týchto testoch sme loptu postupne rozmiestňovali na 5-tich miestach v scéne. Úlohou UNS bolo reagovať na dané podnety a dostať sa k lopte do požadovanej vzdialenosti. UNS mala predpísaný počet krokov, ktoré mohla vykonať. Vždy, po uplynutí počtu týchto krokov, sa lopta presunula na novú pozíciu a pozícia robota bola reštartovaná na štartovaciu pozíciu, ktorá bola vo všetkých prípadoch nemenná.

Lopta sa postupne objavovala na týchto miestach:

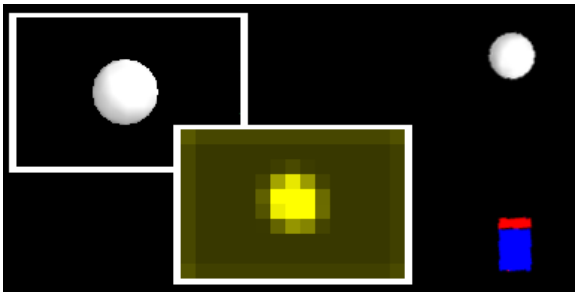
- Vľavo pred robotom (veľká vzdialenosť),
- vpravo pred robotom (veľká vzdialenosť),
- presne oproti robotovi (malá vzdialenosť),
- vľavo pred robotom (malá vzdialenosť),
- vpravo pred robotom (malá vzdialenosť).

Druhý test (guľa už nebola statická) sa skladal zo 4 častí. Aj pri týchto testoch bolo úlohou UNS reagovať na dané podnety. V týchto testoch vykonávala lopta pohyb po scéne v tvare písmena U. V prvých dvoch častiach tohto testu je lopta umiestnená blízko k robotovi (na ľavej strane a na pravej strane od robota) a vykonáva pohyb smerom od neho. Po prejdení určitej vzdialenosti sa lopta v niekoľkých krokoch otočí a začne sa vracať naspať. Pozícia robota a lopty sa vždy po skončení danej časti testu nastaví na štartovaciu pozíciu. Zvyšné dva testy prebiehajú analogicky ako tie prvé. Lopta je vždy umiestnená ďalej od robota na príslušnej strane (na ľavej alebo na pravej) a pohybuje sa smerom k nemu. Po prejdení určitej vzdialenosti sa postupne zmení smer a pohybuje sa smerom preč od robota. Každý pohyb lopty je sprevádzaný dvomi reakciami UNS.

K výpočtu percentuálnej úspešnosti používa algoritmus na ohodnocovanie polohy lopty voči robotovi (podkapitola 4.1.5) Výstup algoritmu sa porovnáva s výstupom siete a delí sa počtom vykonaných krokov robota na scéne.

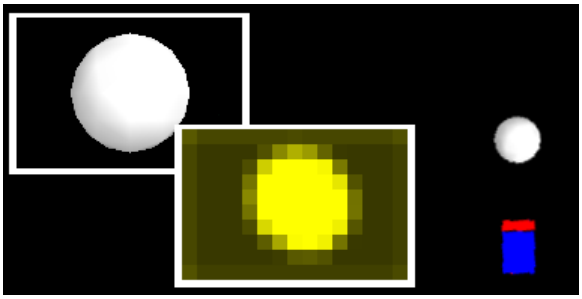
5.5.2 Bodové osvetlenie lopty

Ako prvé sme testovali správanie UNS s pridaním bodového osvetlenia na loptu. Nasledujúce obrázky obr. 5.8 a obr. 5.9 ilustrujú interval adekvátnej vzdialenosti robota od lopty v podmienkach s pridaním jemného bodového osvetlenia. Toto osvetlenie malo za následok vznik tieňu na spodnej strane lopty.



Obr. 5.8: Horná hranica akceptovateľnosti vzdialenosti lopty voči robotovi s obrazom snímaným kamerou robota a k nemu zobrazená vizualizácia prislúchajúcej aktivácie bázovej vrstvy

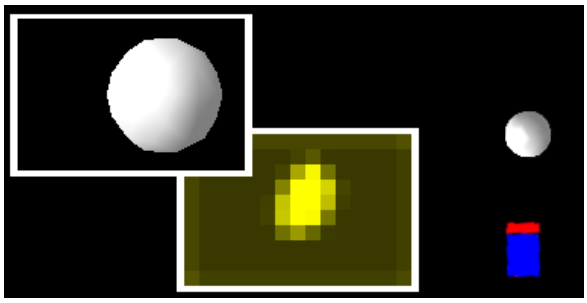
Na obr. 5.8 je znázornený robot a lopta v polohe, keď UNS berie rozmer lopty ako ešte stále postačujúci na to, aby negenerovala pohyb vpred. Na druhej strane generátor učiaceho signálu ohodnotil postavenie ako zlé. Rozdiel v očakávanej veľkosti od aktuálnej veľkosti lopty bol niekoľko pixlov. Takéto zlé ohodnocovanie vznikalo, len málokedy, no najmä v situácii, keď bola lopta statická a robot prichádzal priamo k nej. V ľavom hornom rohu obrázku je zobrazený snímaný obraz kamerou robota bez predspracovania. Na ľavej dolnej strane lopty pozorujeme slabý tieň. V strednej dolnej časti obr. 5.8 vidíme vizualizáciu aktivácie bázovej vrstvy. UNS napriek tieňu reaguje na podnet adekvátnym (možno trochu menším) aktivačným obrazcom. Upozorníme, že prah v predspracovávaní obrazu pre RBF sieť je nastavený na intenzitu jasu = 205 v škále šedí.



Obr. 5.9: Dolná hranica akceptovateľnosti vzdialenosti lopty voči robotovi s obrazom snímaným kamerou robota a k nemu vizualizácia prislúchajúcej aktivácie bázovej vrstvy

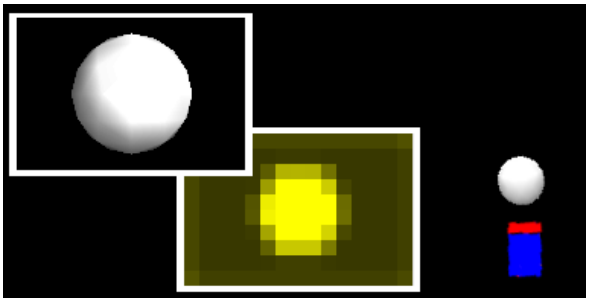
Obr. 5.9 ilustruje dolnú akceptovateľnú hranicu polohy lopty voči robotovi. V tomto prípade UNS ešte negenerovala pohyb vzad. Na vizualizácii aktivácie máme teraz možnosť pozorovať oválny tvar, nie guľu, ako by to bolo za ideálnych podmienok. Tento tvar je spôsobený nižšou aktivitou bázových neurónov umiestnených nad miestami gule, ktoré sú pokryté tieňom.

Pozrime sa na ďalší príklad, no teraz s intenzívnejším zatienením lopty (obr. 5.10).



Obr. 5.10: Intenzívnejšie zatienenie lopty

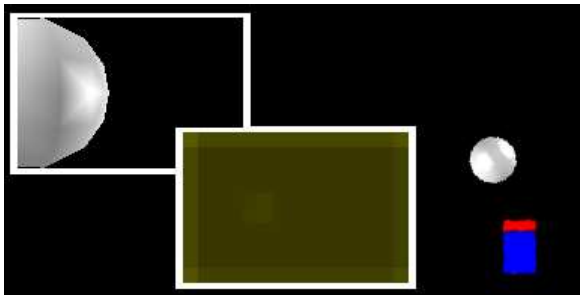
Intenzívnejší tieň je pekne pozorovateľný aj na aktiváciách bázových neurónov. Vidíme, že aktívne neuróny sú umiestnené iba nad časťou lopty bez tieňa. V tomto prípade išlo opäť o svetlo dopadajúce jemne zo strany. Pri obehnutí lopty robotom z druhej (tmavšej) strany sa situácia ešte zhoršila. Hlavnou črtou správania UNS v opísanej situácii bolo chybné vyhodnocovanie robota a jeho aktuálnej vzdialenosti od lopty. Výstupná vrstva vyhodnocovala vzdialenosť zvyčajne väčšiu ako bola v skutočnosti, keďže bolo aktívnych menej bázových neurónov. S cieľom získania adekvátnej vzdialenosti robota od lopty (z pohľadu UNS) musel robot prísť bližšie k lopte, čo bolo ohodnocovacím algoritmom označené ako chybná pozícia. Opísanú situáciu ilustrujeme na obr. 5.11, kde UNS vyhodnotila polohu robota voči lopte ako správnu. V porovnaní s „ideálnymi“ podmienkami zobrazenými na obr. 5.8 a obr. 5.9 sa robot nachádza omnoho bližšie k lopte.



Obr. 5.11 Neadekvátne vyhodnotená situácia pod vplyvom tieňu

V prípade umiestnenia svetelného zdroja s dopadom svetelných lúčov kolmo na hraciu plochu bola sieť schopná počas spusteného trénovania doučiť sa problematické určovanie vzdialenosti. Táto situácia nebola ďalej rozvinutá, pretože hrozila príliš veľká špecifikácia UNS (osvojila by si iba jeden typ podnetu).

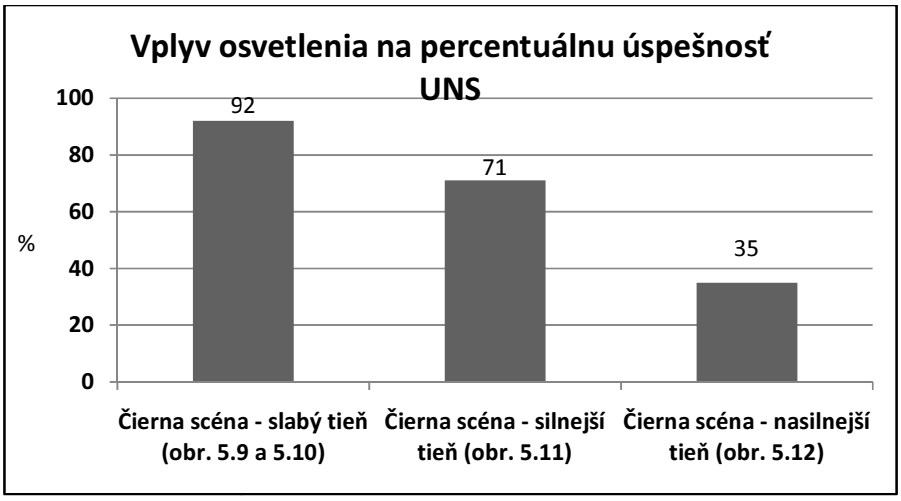
Nasledovný obr. 5.12 ilustruje príliš tmavú loptu. Na aktivácii bázovej siete nie je možné rozoznať takmer žiadne aktivácie okrem malých aktivácií štyroch neurónov na mieste odlesku. UNS nebola vôbec schopná nasledovať loptu.



Obr. 5.12: Nedostatočný stimul pre RBF sieť pod vplyvom silného tieňu

UNS v tomto prípade nebola takmer vôbec schopná nasledovať loptu. Do určitej miery si vedela udržať správny smer natočenia, no i ten bol veľmi problematický.

V nižšie uvedenom grafe na obr. 5.13 máme možnosť vidieť percentuálnu úspešnosť ohodnotenú pomocou algoritmu určeného na testovanie úspešnosti (podkapitola 5.5.1)



Obr. 5.13: Vplyv osvetlenia na percentuálnu úspešnosť UNS

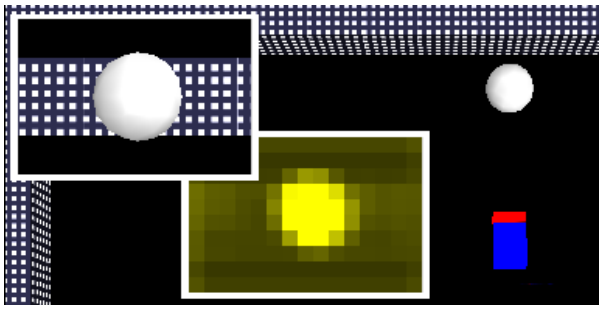
Zobrazenie úspešnosti jasne poukazuje, že najmenšia intenzita tieňu vykazovala najlepšiu percentuálnu úspešnosť v zvolených testoch. Výsledok bol s ohľadom na stimuly, ktorými bola UNS trénovaná, očakávaný.

Ukázali sme si správanie natrénovanej siete s podnetmi zašumenými tieňom. Nasledovná podkapitola bude pojednávať o správaní UNS s tými istými váhami v prostredí rozšírenom o tieň na lopte, o textúry na stenách a na podlahe.

5.5.3 Textúry stien a podlahy

Vzhľadom k tomu, že natrénovaná UNS prejavila schopnosť do určitej miery generalizovať, môžeme pokračovať v testovaní v zložitejších podmienkach. Zložitejšie (reálnejšie) podmienky zabezpečíme pridaním textúr na objekty umiestnené na scéne (podlaha, steny). Bodové osvetlenie sme nechali zapnuté na nižšej intenzite ukázanej na obr. 5.8 a obr. 5.9, kde UNS ešte ukazovala vynikajúcu schopnosť sledovať loptu.

Pridanie tmavšej textúry – farby, ktorá by dosahovala po transformácii do škály šedí nižšie jasové hodnoty ako 205, je bezpredmetné. Je to odôvodnené prahovaním. Nižšie jasové hodnoty sú jednoducho odfiltrované a RBF sieť sa s nimi nekonfrontuje. Zaujímavejšie je to v prípade použitia svetlejšej textúry, napr. textúry čiernobielej mriežky (malé biele štvorčeky) použitej na stene scény zobrazenej na obr. 5.14.



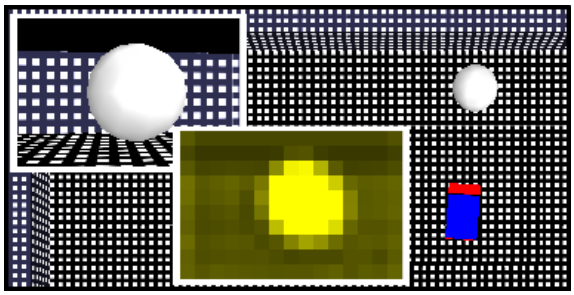
Obr. 5.14: Použitie mriežkovej textúry na stene hracej plochy

Po umiestnení textúr na stenu je vidieť nízku a nejasnú aktivitu po celej dĺžke výskytu textúry, ale loptu je možné stále rozoznať. Aktivita bázovej vrstvy bola v danej situácii stále postačujúca na to, aby robot dokázal sledovať smer pohybu lopty. O niečo horšie to bolo pri upravovaní vzdialenosti medzi loptou a robotom, no i napriek obmedzeniam takmer vždy UNS realizovala správnu akciu. Tento problém sa prejavoval najmä pri väčšej vzdialenosti medzi loptou a robotom. Ďalším prejaveným problémom bola strata schopnosti hľadania loptičky v prípade, ak ju nemá v zornom poli kamery. Aktivácia spôsobená textúrou na stene vyvolala aktivitu na hranici nerozhodnosti pre UNS. Robot nerozhodne kmital zľava doprava.

Použitie rovnakej textúry iba na podlahe hracej plochy nepreukázalo veľmi badateľné zmeny oproti takmer ideálnym podmienkam zobrazených na obr. 5.8 a obr. 5.9. UNS bola schopná generovať správne príkazy na navádzanie robota.

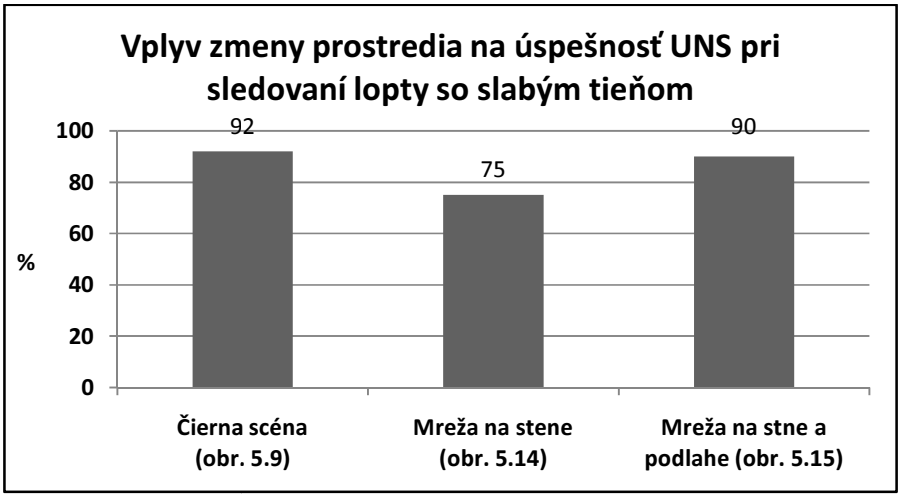
Ďalší test spočíval v použití rovnakej textúry na podlahu a na stenu hracej plochy. Správanie UNS by sa dalo porovnať so správaním v prípade použitia textúry iba na

stene. Sledovanie smeru loptičky sa nezdalo byť problematické. Problém nastal s vyhľadávaním loptičky, rovnako ako v predchádzajúcom prípade, keď nebola v zornom poli. Robot opätovne nerozhodne kmital zľava doprava. Výsledky UNS sa v tejto situácii dali považovať za uspokojivé. Vizualizácia opísanej situácie sa nachádza na nasledovnom obr. 5.15.



Obr. 5.15: Použitie mriežkovej textúry na stene a podlahe hracej plochy

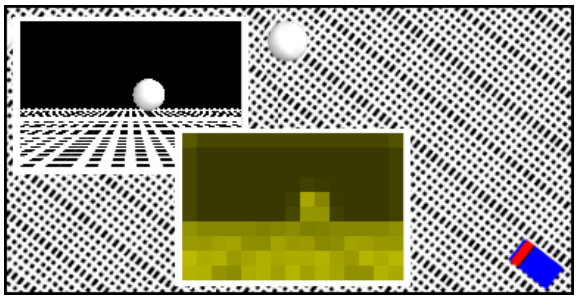
Nasledujúci obr. 5.16 zobrazuje graf percentuálnej úspešnosti troch spomenutých prípadov v tejto podkapitole.



Obr. 5.16: Vplyv zmeny prostredia na úspešnosť UNS pri sledovaní lopty so slabým tieňom

Máme možnosť pozorovať, že najlepšiu aktivitu vykazovala podľa očakávaní jednoduchá scéna (92%). V testoch vyšli rozdielne situácie s použitím textúry na stene, a na stene a podlahe, čo je pre nás trochu prekvapivé. Percentuálny rozdiel vznikol v prípade, keď sa lopta dostala do bezprostrednej blízkosti robota a míňala ho po strane. V tejto situácii nedokázala UNS generovať dostatočne dobre signál na natáčanie sa v smere pohybujúcej sa loptičky, pričom v jednoduchej situácii bola reakcia na podobnú situáciu vykonaná správne.

Invertovaním textúry sme dosiahli čierne štvorčeky na bielom podklade. Správanie UNS bolo vyskúšané najskôr s textúrou na podlahe. Situácia na obr. 5.17 zobrazuje relatívne vysokú aktivitu neurónov umiestnených nad podlahou. V takomto prostredí UNS reagovala len generovaním pohybu vzad. Situáciu zjavne vyhodnotila ako loptu umiestnenú veľmi blízko robota.

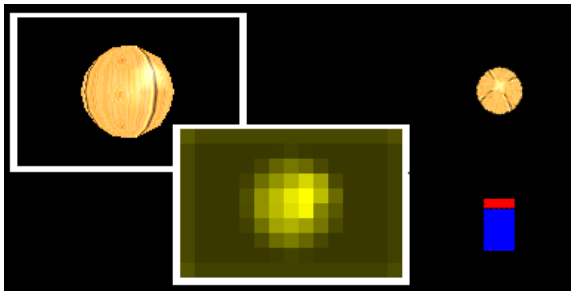


Obr. 5.17: Použitie prevažne bielej textúry na podlahe hracej plochy

Na vyššie uvedenom obrázku vidíme, že aktivácia loptičky sa dá stále rozoznať, ale nami postavená a natrénovaná UNS už nedokázala generovať pohyb vpred. Pokiaľ bola loptička blízko robota, UNS vedela do určitej miery korigovať smer natočenia k loptičke, ale inak len generovala pohyb dozadu. UNS v teste dosiahla menej než 10%. Z dôvodu nízkej úspešnosti UNS už pri prvom teste neboli vykonané žiadne ďalšie testy.

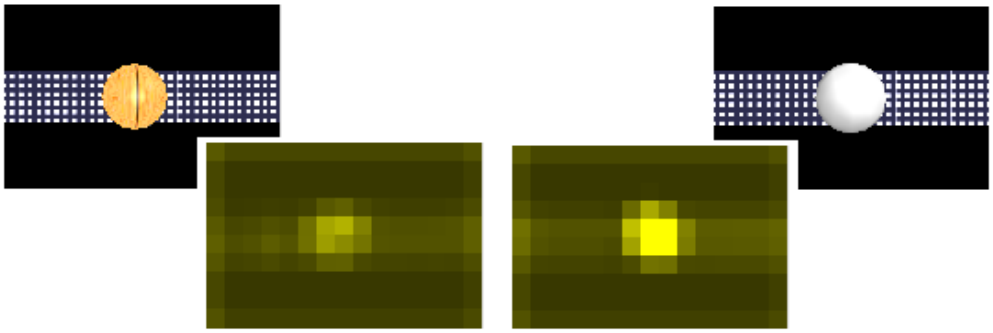
5.5.4 Textúra lopty

Na obr. 5.18 vidíme, že textúra dreva spôsobila, že aktivácie bázových neurónov sú o niečo slabšie v porovnaní so situáciou ilustrovanou na obr. 5.8, kde bol použitý na loptu iba jemný tieň. Napriek tomu textúra dreva na loptičke nespôsobovala veľké problémy. Objavovala sa najmä neistota UNS v centrovaní lopty v zornom poli.



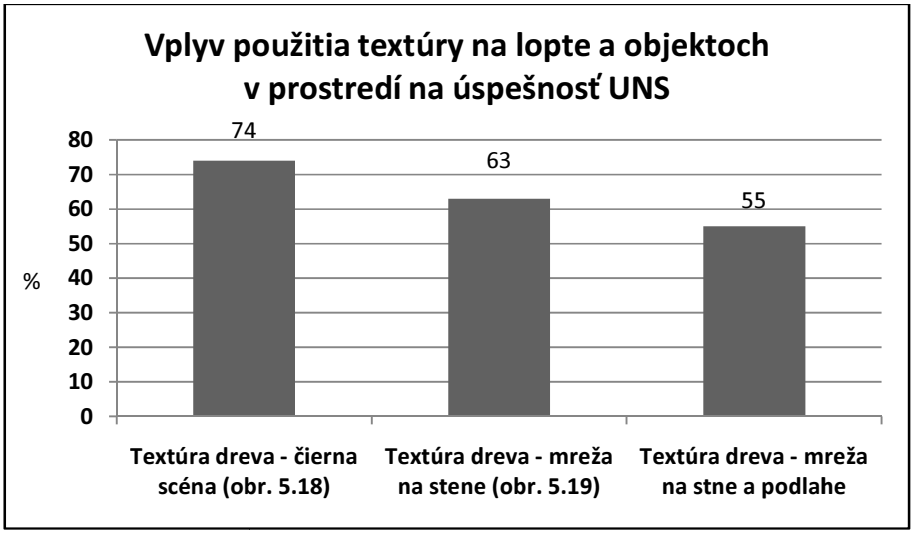
Obr. 5.18: Použitie textúry dreva na loptičke

Po pridaní textúry tmavej mriežky na stenu, použitej v situácii ilustrovanej na obr. 5.14, a drevenej textúry lopty, UNS pri sledovaní lopty napredovala „nesmelo“. Generovala pomedzi pohyby vpred aj natáčanie do strán. Cúvanie robota v prípade potreby bolo bezproblémové. V situácii, keď robot neisto kmital, bola za behu aplikácie zmenená textúra späť na bielu. Robot reagoval vycentrovaním a následným napredovaním k lopte. Obr. 5.19 ilustruje rozdiel aktivácie bázových neurónov s použitím textúry a bez jej použitia na lopte.



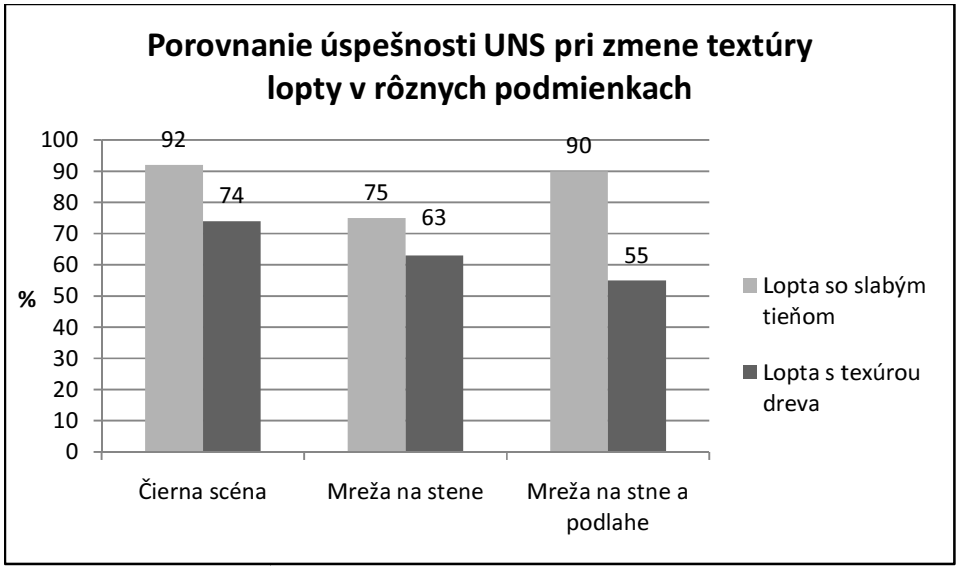
Obr. 5.19: Ilustrácia rozdielu aktivácii v rovnakej situácii s rozdielnou textúrou lopty

Obr. 5.20 zobrazuje graf s percentuálnou úspešnosťou UNS v troch prípadoch použitia textúry na lopte, a to nasledovne: jednoduchá scéna, mreža na stene a mreža na stene a podlahe. Vidíme, že UNS si zhoršila úspešnosť i v jednoduchom prípade, pričom v ostatných dvoch prípadoch sa javil pohyb robota i bežnému pozorovateľovi ako často mylný.



Obr. 5.20: Vplyv použitia textúry na lopte a objektoch v prostredí na úspešnosť UNS

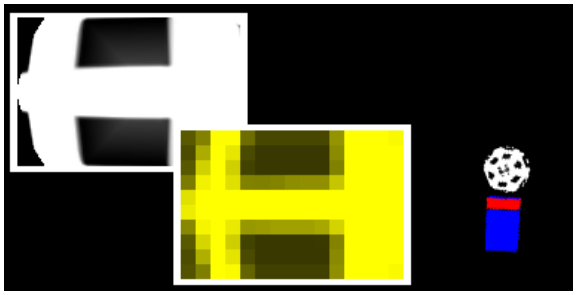
Obr. 5.21 zobrazuje graf s porovnanie úspešnosti UNS pri zmene textúry lopty v rôznych podmienkach.



Obr. 5.21: Porovnanie úspešnosti UNS pri zmene textúry lopty v rôznych podmienkach

V prípade použitia textúry na lopte, bielej mriežky (čierne štvorčeky na bielom podklade) sa problém prejavoval pri pohybe robota vpred a pri malej vzdialenosti medzi robotom a loptou. Pri druhom spomenutom prípade UNS generovala pohyby zľava doprava podľa toho, na ktorej strane prevažovali aktivácie. Situáciu približuje obr. 5.22, kde robot v danom momente vykonával pohyb doprava. V bode, keď na pravej strane dosiahol neaktívne miesta a naopak na ľavej už opäť aktívne, sieť vyhodnotila situáciu ako presne opačnú – generovala pohyb doľava.

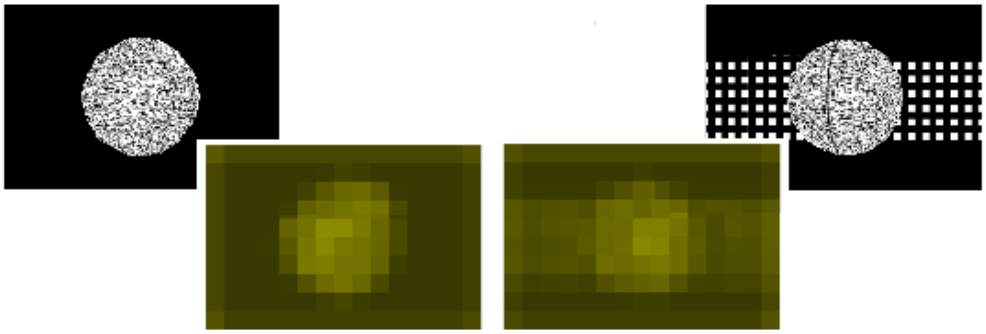
V momente, keď UNS prekonala neaktívnu plochu na ľavej strane (nastalo zvýšenie aktivácií bázových neurónov na uvedenej strane), UNS generovala pohyb vpravo. Naopak, keď UNS dosiahla neaktívnu plochu na pravej strane, generovala pohyb vľavo.



Obr. 5.22: Nerozhodná situácia pre UNS s použitou textúrou mreže na lopte

Celková úspešnosť UNS pri testoch zobrazených na obr. 5.22 bola o niečo vyššia ako 82%.

Obr. 5.23 poukazuje na aktiváciu bázových neurónov s použitím textúry šumu. Prvý prípad (ľavá strana) poukazuje na aktivácie v jednoduchom prostredí a druhý s aplikáciou textúry na stene.



Obr. 5.23: Aktivácie bázových neurónov s použitou textúrou šumu na lopte

V jednoduchom prípade (ľavá strana obr. 5.23) máme možnosť pozorovať podobne zašumený obrazec aktivácií ako je na stimule. UNS vykazovala schopnosť otáčania sa za loptou, ale nebola schopná upravovať vzdialenosť, pokiaľ nešlo o situáciu, v ktorej mala generovať cúvanie (60% úspešnosť v teste). K nejasnému stimulu bola ešte doplnená textúra, ktorá vytvára podobný, no nie rovnaký šum na aktivácie. UNS aj v tejto situácii javila už len snahu upravovať smer, i keď veľmi neisto, pokiaľ nebola lopta príliš ďaleko. Cúvanie v prípade potreby UNS opäť generovala stále správne (52% úspešnosť v teste).

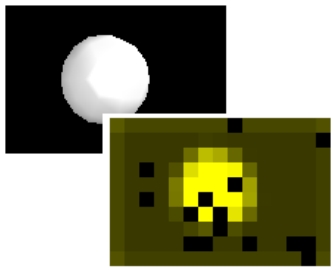
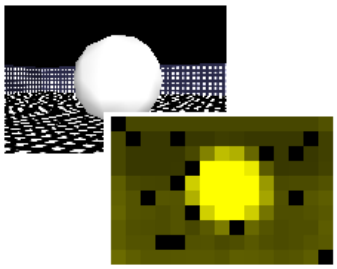
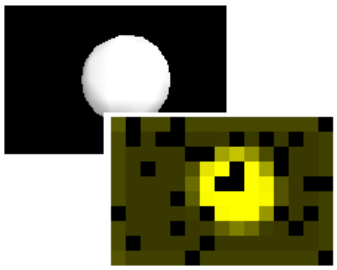
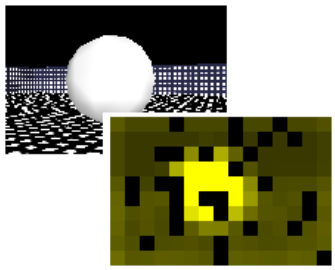
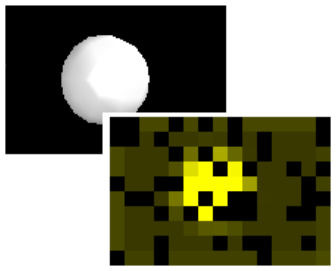
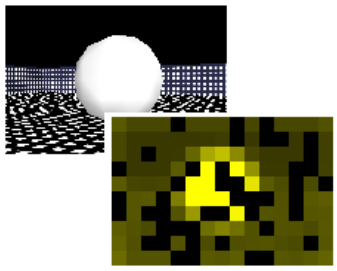
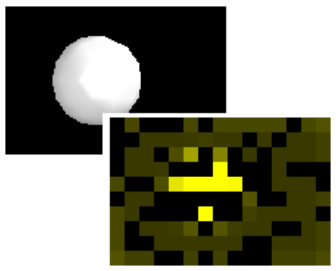
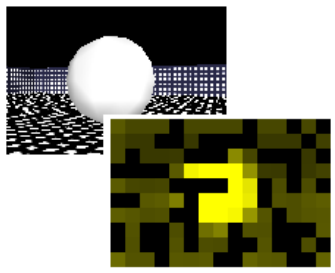
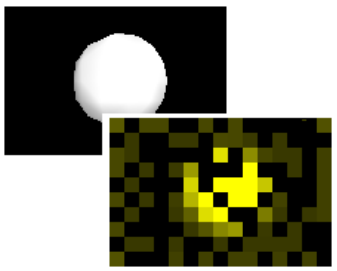
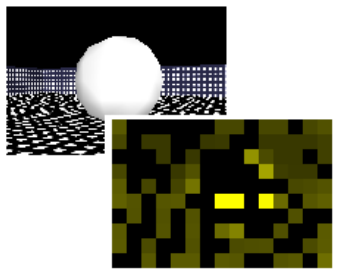
5.5.5 Nulovanie výstupov bázových neurónov

Doteraz sme sa venovali správaniu sa UNS v rôznych prípadoch použitia textúr na objektoch na scéne. Teraz sa posunieme iným smerom. Šum bude spôsobený poškodením skrytej vrstvy RBF siete. Zašumenými dátami (pre lineárnu vrstvu) rozumieme aj premazanie výstupov bázových neurónov nulou. Overíme schopnosť UNS zovšeobecniť informáciu i z takto poškodenou informáciou.

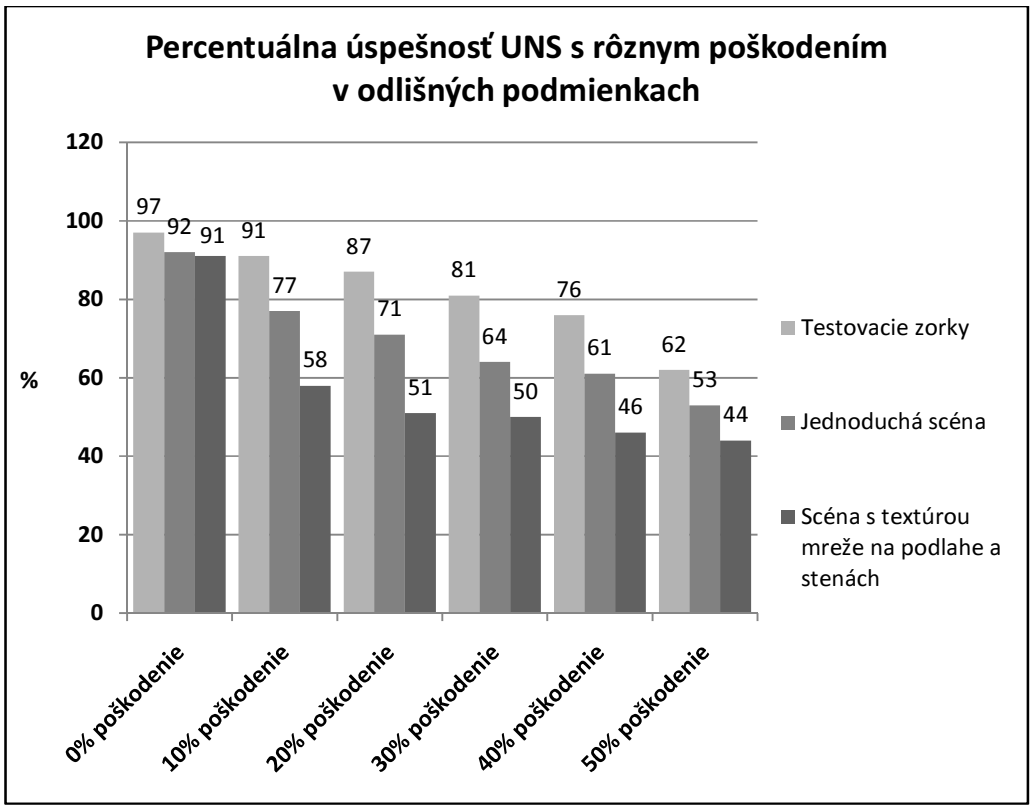
Na nulovanie vstupu bol použitý jednoduchý algoritmus, ktorý po ukončení výpočtu bázovej vrstvy generuje podľa zadaného počtu čísla (bez opakovania) z intervalu o veľkosti bázovej vrstvy. K danému číslu vyhľadá prislúchajúci bázový neurón a jeho výstup prepíše nulou. Takto upravené pole sa ďalej posúva lineárnej vrstve.

Nižšie uvedená tabuľka 5.1 zobrazuje príklady zobrazených aktivít bázových neurónov, ktoré boli poškodené v danom počte.

Tabuľka 5.1: Ilustrácia aktivít neurónov bázovej vrstvy po ich poškodení

10% poškodenie:		
20% poškodenie:		
30% poškodenie:		
40% poškodenie:		
50% poškodenie:		

Graf zobrazený na obr. 5.24 zobrazuje percentuálnu úspešnosť s poškodením bázovej vrstvy od 10% do 50% s posunom 10%. Modifikovaná sieť bola testovaná spôsobom opísaným v podkapitole 5.5.1, teda rovnakým, aký bol použitý v predchádzajúcich dvoch podkapitolách. Test bol vykonaný v jednoduchých podmienkach (čierna scéna) a zložitejších podmienkach (textúra čiernej mreže, rovnaká ako bola použitá v prípade zobrazenom na obr. 5.15). Tieto dva testy boli doplnené o test na testovacej množine použitej pri trénovaní. Každý z druhov testov bol vykonaný 10-krát pre každé poškodenie UNS.



Obr. 5.24: Úspešnosť UNS s rôznym poškodením v odlišných podmienkach

Najmenší pokles úspešnosti sa prejavil na testovacej množine, teda na ideálnych stimuloch, čo sme aj predpokladali. V tomto prípade sa 10% poškodenie dá prirovnať k úspešnosti UNS v 3D prostredí s použitým tieňom bez poškodenia bázovej vrstvy. Veľké problémy robili zhluky „mŕtvych“ neurónov, cez ktoré sa zvyčajne UNS nevedela dostať. (zľava doprava). Ak boli „zabité“ neuróny tak, že vytvorili kruh okolo stredu snímaného obrazu, UNS mala tendenciu generovať pohyb vpred (silná aktivácia

v strede s malou veľkostou sa javila ako vzdialená lopta), pričom zvykla prebehnúť cez guľu na opačnú stranu.

Najväčšie problémy z prostredia robila scéna s použitou textúrou. UNS mala tendenciu generovať pohyb tým smerom, kde bolo menej poškodených neurónov (prevážila silnejšia aktivácia). Išlo o situácie s väčším poškodením.

5.6 Testovanie vplyvu rýchlosti pohybu objektov a rýchlosť otáčania robota na efektívnosť UNS

Pripomenieme, že ovládanie robota pomocou UNS a lopty na scéne je uskutočňované v cykle (cyklus UNS - podkapitola 4.4). Vzhľadom na implementáciu nie je možné nezávisle snímať obraz z kamery robota a súčasne nezávisle pohybovať s loptou. Jediná možnosť, ako zvýšiť rýchlosť pohybu lopty, je úprava jej konštánt pohybu. Ich úpravou spôsobíme zväčšenie alebo zmenšenie diskrétneho kroku vykonávanej činnosti. Ak nastavíme veľké hodnoty konštánt pohybov objektov, budú sa na scéne presúvať skokovo, čo znamená, že ich pohyb nebude plynulý, ale trhaný.

Zmena rýchlosti pohybu lopty

Vykonali sme sériu testov, aby sme posúdili vplyv rýchlosti pohybu gule, rýchlosti pohybu robota a jeho rýchlosti otáčania. Vieme, že, rýchlosť otáčania neovplyvňovala výsledok uvedených testov. Rozhodli sme sa túto hodnotu nemeniť, a preto ju neuvádzame ani v nasledovných testoch.

1. **Test**

   V tomto teste sme nastavili nasledovné hodnoty konštánt:

   - rýchlosť pohybu robota = 0,9 f,
   - rýchlosť otáčania robota = 0,5 f,
   - rýchlosť pohybu gule = 0,9 f.

   Lopta sa pohybovala trhane a veľmi rýchlo. Navyše, robot sa otáčal s príliš veľkým skokom. Robot nedokázal vo svojom zornom poli udržať loptu a točil sa iba na mieste, ako keby loptu nevidel. Z uvedeného vyplýva, že rýchlosť pohybu robota bola nepodstatná.

2. **Test**

V tomto teste sme sa rozhodli preskúmať vplyv rýchlosti otáčania robota na výsledok simulácie. Oproti predchádzajúcemu testu sme zmenili iba rýchlosť otáčania robota a nastavili sme ju na nasledovnú hodnotu:

- rýchlosť otáčania robota = 0,05 f.

Vyhodnotením testu sme dospeli k záveru, že rýchlosť otáčania robota výrazne ovplyvňuje jeho schopnosti (očakávané reakcie na podnety). Pri takto nastavenej rýchlosti sa robot otáčal plynule, a nie trhane ako v predchádzajúcom teste. Keďže rýchlosť pohybu lopty bola vysoká, robot nestíhal sledovať jej pohyb.

Celkový výsledok testu je uspokojivý.

3. Test

Rozhodli sme sa zvýšiť rýchlosť otáčania robota a znížiť rýchlosť jeho pohybu a aj pohybu lopty. Hodnoty konštánt sme nastavili nasledovne:

- rýchlosť pohybu robota = 0,4 f,
- rýchlosť otáčania robota = 0,1 f,
- rýchlosť pohybu gule = 0,4 f.

Pri tomto testovaní robot preukázal problém s centrovaním lopty do svojho zorného poľa. Každé centrovanie lopty bolo sprevádzané kmitaním robota zo strany na stranu. Robot dokázal vycentrovať loptu vo svojom zornom poli, pokiaľ sa lopta dostala do správnej pozície. Pohyby robota pôsobili veľmi neisto. Príčinou uvedeného problému bola opäť vysoká rýchlosť otáčania robota.

4. Test

V záverečnom teste sme sa snažili nastaviť suboptimálne hodnoty konštánt. Podarilo sa nám ich nastaviť nasledovne:

- rýchlosť pohybu robota = 0,4 f,
- rýchlosť otáčania robota = 0,05 f,
- rýchlosť pohybu gule = 0,4 f.

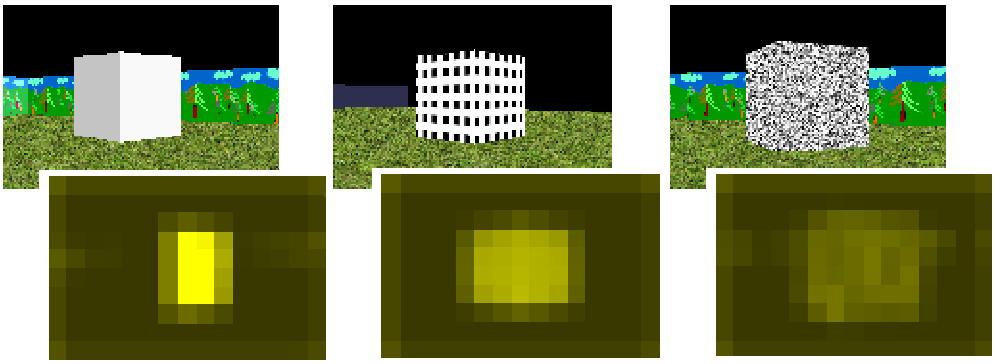
Rozhodli sme sa zmeniť iba rýchlosť otáčania robota, keďže hodnota tejto konštanty spôsobovala problémy. Rýchlosť pohybu objektov sme nezvyšovali. Pri ich vysokom nastavení sa objekty presúvali s veľkým diskrétnym krokom. Pohyb objektov pôsobil nesúvisle, až nereálne.

5.7 Doplňujúci test

Pri riešení zadania práce vznikla myšlienka preskúmania správania UNS v prípade nahradenia loptičky iným objektom, pričom by sa nastavenia váh nemenili.

Výmena lopty za kocku

Ako bolo opísané, natrénovaná UNS vykazovala veľmi dobré výsledky pri sledovaní aj zašumených objektov. Pri nahradení lopty bola UNS v podstate postavená pred analogickú situáciu ako predtým a dokázala sledovať objekt bež väčších problémov. Kocka mala šírku strany rovnakú ako bol priemer predtým použitej lopty. Nižšie uvedený obr. 5.29 dokumentuje aktivitu bázovej vrstvy na dané podnety.

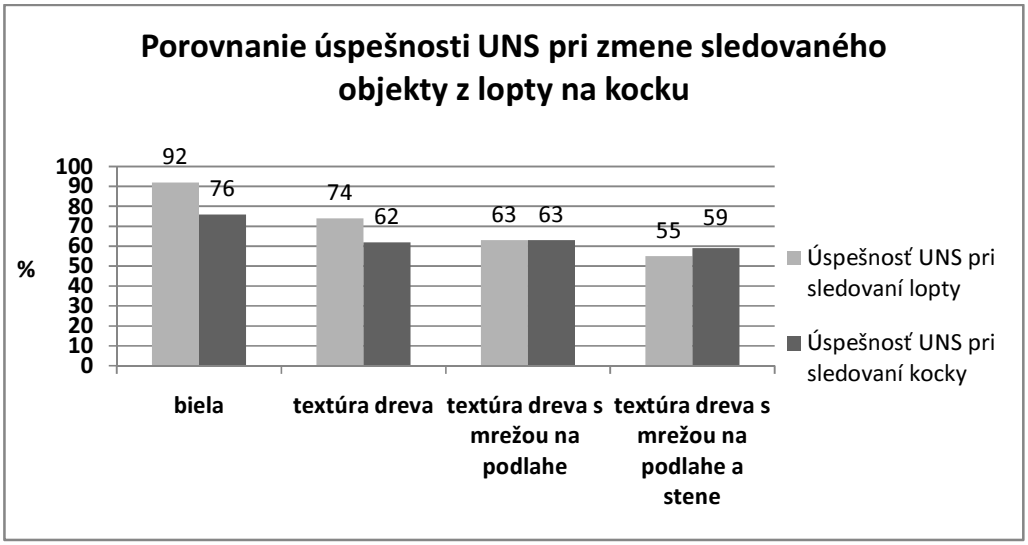


Obr. 5.25: Porovnanie aktivácií bázovej vrstvy na kocku

V prvom prípade, keď je kocka bez textúry, vidíme silnú aktiváciu na jej nezatienenú stranu. Problém tu vznikal s tieňom. Ak bol robot otočený na zatienenú stranu, tak UNS nemala možnosť nájsť silnejší odlesk, ako to bolo v prípade lopty, ale takýchto situácií sa nenašlo veľa. Na druhej strane, pri sledovaní osvetlenej strany mala možnosť sledovať rovný osvetlený povrch, ktorý zabezpečoval vyššiu úroveň aktivácie bázickej a aj výstupnej vrstvy.

Ďalšie dva zobrazené prípady aktivovali v UNS podobné správanie ako v ekvivalentoch s loptou.

V nasledujúcom grafe zobrazenom na obr. **5.26** máme možnosť vidieť porovnanie percentuálnej úspešnosti UNS v prípade sledovania loptičky a kocky v teste opísanom v podkapitole 5.5.1.



Obr. 5.26: Porovnanie úspešnosti UNS pri zmene sledovaného objekty z lopty na kocku

Na grafe vidíme, že UNS vykazovala o niečo menšiu percentuálnu úspešnosť v prípade sledovania kocky ako lopty hlavne v jednoduchších podmienkach. Z uvedeného, a to nielen z obr. 5.26, sa dá do určitej miery predpokladať, že UNS reaguje na zhluk aktívnych neurónov a tie považuje za záujmový objekt. Toto tvrdenie sa dá podporiť najmä v zložitejšej situácii, teda s použitím textúry na stenách a na podlahe. Vieme, že šírka hrany lopty je rovnaká ako priemer lopty, ale povrch, ktorý odráža svetlo je väčší ako v prípade lopty, a teda UNS v zložitejších podmienkach generuje lepšie výstupy na väčší podnet.

Záver

V práci sme realizovali simuláciu mobilného robota riadeného pomocou RBF siete. Dospeli sme k záveru, že vytváranie takéhoto druhu aplikácie nie je jednoduché. Je to proces, v ktorom je potrebné skĺbiť vedomosti a poznatky z viacerých vedných disciplín. Najdôležitejšie vedomosti, ktoré boli nevyhnutné k uskutočneniu práce pochádzali z teórie umelých neurónových sietí, kde sú obsiahnuté viaceré vedné disciplíny. Dôležité bolo porozumieť ich fungovaniu, aplikácií a v neposlednom rade aj spôsobom ich implementácie. Naprogramovaná umelá neurónová sieť by sama o sebe bez kontaktu s prostredím, v ktorom by mohla vykonávať akcie, nespĺňala požadované podmienky, a preto bolo rovnako podstatné nadobudnúť vedomosti aj zo senzomotorického riadenia používaného v mobilných robotoch.

Počas programovania sa vyskytlo niekoľko chýb, ktoré sa nám podarilo opraviť a dospeli sme k úspešnému ukončeniu implementácie jednoduchej simulácie robota riadeného RBF sieťou. Prostredie, určené na vykonávanie akcií robota, bolo do značnej miery zjednodušené, ale ako sme sa presvedčili po vykonaní testov, bolo stále postačujúce na demonštrovanie zadanej problematiky.

Uskutočnené testy potvrdili základnú vlastnosť umelých neurónových sietí, a to osvojovať si k predkladaným stimulom adekvátne výstupy pomocou učiaceho algoritmu. Hlavnou úlohou práce bolo otestovať správanie sa modelu so zašumenými dátami a správanie pri zmene rýchlosti pohybu robota a lopty na scéne.

Nami naprogramovaná UNS dokázala v jednoduchom prostredí sledovať loptičku takmer bez problémov. Jedine pri prihliadnutí na algoritmus ohodnocovania percentuálnej úspešnosti vidíme, že pri dorovnávaní vzdialenosti vznikal malý rozdiel medzi tým, čo UNS generovala ako výstup a tým, čo mala generovať podľa algoritmu ohodnocovania polohy lopty voči robotovi. Pri obyčajnom pozorovaní správania sa robota sa jeho vykonané akcie javili ako úplne bezchybné. Vyhodnocovanie chybného postavenia malo za následok použitie jemného tieňa na loptičke. Skreslenie ohodnotenia takéhoto druhu sme v princípe očakávali a môžeme ho považovať za zanedbateľné. Sila zovšeobecnenia sa prejavila najmä po pridaní textúr na povrch hracej plochy a na povrch lopty. UNS dokázala i napriek relatívne silne zašumeným dátam generovať požadované výstupy. Pokiaľ bola zachovaná biela farba loptičky UNS si vedela poradiť takmer s každým farebným prevedením prostredia. Markantné problémy sa prejavili až

v prípade použitia svetlých textúr prostredia. UNS vykazovala tendenciu sledovať smer pohybu lopty, ale strácala schopnosť jej vyhľadávania v prípade, keď nebola v zrnom poli, čo sme v danej situácií neočakávali. Pozorovali sme, že je tendencia UNS orientovať sa na zhluk aktívnych bázových neurónov. Ďalším prípadom zlého generovania pohybu bol podobný prípad. Išlo o situáciu, keď bola celá hracia plocha vo farebnom odtieni takmer zhodnom s bielou farbou, na ktorú bola UNS prioritne trénovaná. V takom prípade generovala pohyb vzad akoby bola lopta v tesnej blízkosti, čo sme aj predpokladali.

Dosiahnuté výsledky UNS v opísaných podmienkach boli uspokojivé, a to nás doviedlo k ďalšiemu druhu testovania so zašumenými dátami. Išlo o „umŕtvenie“ výstupov náhodne vybraných bázových neurónov v určitom počte. UNS dokázala i za takýchto podmienok v jednoduchom prostredí vykazovať správne výstupy za predpokladu, že nebola poškodená značná časť bázovej vrstvy. Čím väčší počet výstupov bázových neurónov bol nulovaný, tým horšia bola úspešnosť siete. Výsledky sa do značnej miery zhoršili v prípade, ak bola použitá textúra so svetlejšími odtieňmi. V mnohých prípadoch i napriek silne zašumeným dátam, či už pod vplyvom použitých textúr na objektoch alebo nulovaním výstupov bázových neurónov, UNS väčšinu problémov zvládala aspoň korigovaním natočenia smerom k lopte, alebo prípadným cúvaním v prípade potreby. Tento test definitívne ukázal schopnosti naprogramovanej RBF siete zovšeobecniť.

Počas fázy testovania vplyvu zašumenia dát na úspešnosť UNS bola pozorovaná tendencia UNS orientovať sa na zhluky svetlých bodov. Tu vznikla myšlienka na vymenenie sledovaného objektu lopty za kocku. Po uskutočnení testu môžeme definitívne konštatovať, že nami implementovaná UNS sa neorientuje až tak na tvar ako na hustotu aktivácií na bázovej vrstve.

Zadanie práce si vyžadovalo uskutočnenie testu správania sa UNS v prípade zrýchlenia pohybu objektov po scéne. Keďže v použitom programovacom prostredí je pohyb reprezentovaný opakovaným vykonávaním diskrétneho kroku s určitou veľkosťou, tento test nespĺňal príliš veľkú relevantnosť. Napriek všetkému môžeme konštatovať, že UNS vedela reagovať adekvátne výstupy, pokiaľ nebola rýchlosť otáčania robota nastavená na posun o veľmi veľký uhol. Tento test môžeme taktiež považovať za zvládnutý i keď nie veľmi relevantný. Tento druh testu by bolo vhodnejšie uskutočňovať v prostredí, kde sa nevykonáva diskrétny krok v pohybe.

Celkový pohľad na priebeh simulácie a výsledky simulácie hodnotíme ako veľmi uspokojivé. Do budúcnosti by bolo zaujímavé doplniť ešte nejaké ďalšie objekty na scénu za účelom otestovania schopnosti UNS vysporiadať sa aj s takýmto druhom problému. Prípadne prezentovaný model možno rozšíriť napr. o ešte jeden stupeň predspracovania obrazu, a to detekciu hrán. Takto predspracovaný obraz pre UNS by mohol zabezpečiť aktiváciu bázových neurónov len po obvode lopty, teda v špecifikovanejšom tvare. Týmto krokom by sa mohlo dať zamedziť tendencii UNS sledovať každý svetlejší objekt v zornom poli.

Zoznam bibliografických odkazov

[1] BRODAL, P.: Central Nervous System, The: Structure and Function. New York: Oxford University Press, 2004. 536 s. ISBN 0195165608.

[2] COVER, T.M. 1965. Geometrical and Statistical properties of systems of linear inequalities with applications in pattern recognition [online]. Hebb.mit.edu [cit. 2010.04.23]. Dostupné na internete < hebb.mit.edu/courses/9.641/2002/readings/Cover65.pdf >

[3] FERENCZI, T. 2008. Videnie mobilného robota pomocou wifi kamery [online]. Atpjournal.sk [cit. 2010.04.20]. Dostupné na internete: <www.atpjournal.sk/atpplus/archiv/2008_1/PDF/plus50_53.pdf>

[4] HAYKIN, S.: Neural Networks a Comprehensive Foundation. London: Prentice-Hall International, 1999. 823 s. ISBN 81-7808-300-0.

[5] LÚČNY, A. 2002. Ako sledovať pingpongovú loptičku [online]. Microstep-mis.com [cit. 2010.04.24]. dostupné na internete <www.microstep-mis.com/~andy/trojsten.ppt>

[6] MCCOULLOCH, W. S., PITTS, W. 1943. A logical calculus of the ideas immanent in nervous activity [online]. Soe.ucsc.edu [cit. 2010-04-28]. Dostupné na internete: <http://www.soe.ucsc.edu/classes/cmps130/Spring09/Papers/mculloch-pitts.pdf>

[7] PANDYA, A., S., MACY, R., B.: Pattern Recognition with Neural Networks in C++. Boca Raton: CRC Press, 1995. 410 s. ISBN: 0849394627.

[8] RUSS, J., C.: The Image Processing Handbook. Boca Raton: CRC Press, 1998. 771 s. ISBN 9780849325328.

[9] SELMAN, D.: Java 3D Programming. Greenwich: Manning Publications, 2002. 400 s. ISBN: 1930110359.

[10] SICILIANO, B., KHABIT O.: Springer Handbook of Robotics. New York: Springer, 2008. 1611 s. ISBN 978-3-540-23957-4.

[11] SONKA, M., HLAVAC, V., BOYLE, R.: Image Processing, Analysis, and Machine Vision. Toronto: Thomson Learning, 2008. 864 s. ISBN 0-4952-4438-4.

[12] THEODORIDIS, S., KOUTROUMBAS, K.: Pattern recognition. London: Elsevier Academic Press, 2003. 689 s. ISBN 0-12-685875-6.

[13] YILMAZ, A., JAVED, O. SHAH, M. 2006. Object Tracking: A Survey [online]. Cs.ucf.edu [cit. 2010.04.20]. Dostupné na internete <www.cs.ucf.edu/vision/public_html/papers/Object%20Tracking.pdf >