VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

KOHONENOVA SAMOORGANIZAČNÍ MAPA

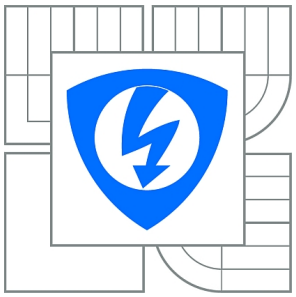
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. VIKTOR ŽÁČEK
AUTHOR

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

KOHONENOVA SAMOORGANIZAČNÍ MAPA

KOHONEN SELF-ORGANIZING MAP

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

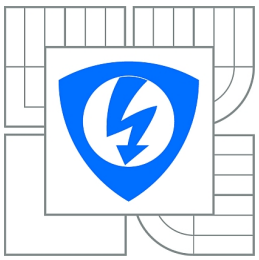
AUTOR PRÁCE — Bc. VIKTOR ŽÁČEK

AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. VÁCLAV JIRSÍK, CSc.

SUPERVISOR

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student:	Bc. Viktor Žáček	***ID:***	111170
Ročník:	2	***Akademický rok:***	2011/2012

NÁZEV TÉMATU:

Kohonenova samoorganizační mapa

POKYNY PRO VYPRACOVÁNÍ:

1. Seznamte se s problematikou samoorganizace se zaměřením na Kohonenovy samoorganizační mapy.
2. Navrhněte a naprogramujte Kohonenovou samoorganizační mapu.
3. Zaměřte se na využití Kohonenovy samoorganizační mapy pro sebelokalizaci mobilního robota.
4. Dosažené výsledky zhodnoťte.

DOPORUČENÁ LITERATURA:

Šíma J., Neruda R.: Teoretické otázky neuronových sítí. Matfyzpress, Praha 1996, ISBN 80-85863-18-9

Termín zadání:	6.2.2012	***Termín odevzdání:***	21.5.2012

Vedoucí práce: doc. Ing. Václav Jirsík, CSc.
Konzultanti diplomové práce:

doc. Ing. Václav Jirsík, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Práce se zabývá samoorganizačními mapami zejména pak Kohonenovou samoorganizační mapou, tvorbou aplikace realizující tvorbu a proces učení Kohonenovy mapy a využitím samoorganizační mapy pro sebelokalizaci robota.

KLÍČOVÁ SLOVA

Samoorganizační mapa, Kohonenova samoorganizační mapa, sebelokalizace robota

ABSTRACT

Work deal about self-organizing maps, especially about Kohonen self-organizing map. About creating of aplication, which realize creating and learning of self-organizing map. And about usage of self-organizing map for self-localization of robot.

KEYWORDS

Self-organizing map, Kohonen self-organizing map, self-localization of robot

ŽÁČEK, Viktor *Kohonenova samoorganizační mapa*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřící techniky, 2012. 72 s. Vedoucí práce byl doc. Ing. Václav Jirsík, CSc.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Kohonenova samoorganizační mapa“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce doc. Ing. Václavu Jirsíkovi, CSc. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne ....

(podpis autora)

OBSAH

1 ÚVOD

Tato práce se zabývá problematikou samoorganizačních map zejména pak Kohonenovy neuronové sítě a možností využití samoorganizačních map pro sebelokalizaci robota.

V kapitole 2 jsou zevrubně popsány různé typy samoorganizačních map, jejich rozdíly a výhody oproti Kohonenově samoorganizační mapě.

V kapitole 3 je podrobně rozebrán základní typ samoorganizační mapy, Kohonenova neuronová síť. Je zde popsána topologie sítě, trénovací a aktivační algoritmy a způsoby reprezentace sítě. V navazující kapitole 4 jsou popsány aplikace Kohonenovy samoorganizační mapy se zaměřením na segmentaci obrazu.

Prvním dílem praktické části práce bylo vytvoření programu KNN. Tento program byl navržen a použit pro účely výuky studentů v předmětu umělá inteligence. Umožňuje tvorbu, učení a aktivaci sítě při uživatelem definovaných parametrech sítě a učení. Program klade důraz na opakovatelnost experimentů, na názorném zobrazení učícího procesu a na grafické interpretaci výsledků naučené sítě. Program taktéž využil student Miloslav Fic pro svou bakalářskou práci. Funkce programu jsou popsány v kapitole 5, podrobný návod k programu je přiložen v příloze A.

Druhým dílem praktické části práce je využití samoorganizační mapy pro sebelokalizaci robota. V kapitole 6 je stručně popsána problematika průběžné lokalizace a mapování. Jednotlivé podkapitoly se zabývají snímáním okolí robota, extrakcí orientačních bodů z okolí robota a samotným výpočtem polohy robota.

V kapitole 7 je pak popsán samotný navržený algoritmus. Ten se skládá z předpracování dat z laserového skeneru, extrakce orientačních bodů pomocí samoorganizační mapy a následným výpočtem polohy na základě těchto orientačních bodů.

Na závěr jsou diskutovány výsledky dosažené tímto algoritmem.

2 SAMOORGANIZAČNÍ MAPY

Samoorganizační mapy (anglicky Self-Organizing Map, zkráceně SOM) jsou umělé neuronové sítě. Jejich charakteristickým rysem je učení bez učitele pomocí něhož je produkována nízko rozměrná (obvykle dvojrozměrná) reprezentace vstupního prostoru, mapa. Dalším charakteristickým rysem samoorganizačních map je funkce sousedství, která zajišťuje uchování topologického uspořádání vstupních dat ve výstupní mapě. Samo-oganizační mapy jsou jedním z nástrojů pro shlukovou analýzu, tedy dělení vstupních dat do několika tříd. Často bývají srovnávány s algoritmem K-means [28].

První samoorganizační mapu popsal profesor Teuvo Kohonen v letech 1981-1982, proto bývá označována jako Kohonenova síť. Ale zejména v anglicky psané literatuře je často označována obecně jako self-organizing map. Od té doby se objevila spousta dalších modifikovaných samoorganizačních map.

Hlavní vlastnost samoorganizačních map, tedy uspořádanost vstupně-výstupního mapování, je využívána v mnoha oblastech průmyslu [13].

2.1 Kohonenova samoorganizační mapa

Někdy také nazývána jako kohonenova neuronová síť, je základní samoorganizační mapou. Jedná se o dvojvrstvou plně propojenou neuronovou síť s topologicky uspořádanou výstupní vrstvou. Využívá učení bez učitele a funkce sousedství. Podrobně je rozebrána v kapitole 3.

2.2 LVQ

LVQ je anglická zkratka pro Learning Vector Quantization, česky Kvantování Vektorů Učením . Jedná se o předchůdce samoorganizačních map. Tato mapa slouží ke klasifikaci vstupních dat do několika kategorií. Byla taktéž vynalezena Teuvo Kohonenem. Narozdíl od samoorganizačních map využívá učení s učitelem. Jedná se o "reinforcement learning"(učení posilováním), toto učení nevyžaduje přesný požadovaný výsledek, ale z prostředí získává odezvu o výsledcích vygenerovaných výstupů, což zajišťuje jistou zpětnou vazbu.

Mapa je dvojvrstvá, ale její výstupní vrstva není nijak topologicky uspořádána, stejně tak není využita funkce sousedství. Každému výstupnímu neuronu je přiřazena třída objektů, které má klasifikovat. Adaptační algoritmus využívá soutěžní strategie učení zohledňující definované třídy objektů. Pokud je třída vstupního vzoru a třída vítězného výstupního neuronu stejná, váhy neuronu se přiblíží vstupnímu vzoru.

Pokud patří do tříd odlišných, váha neuronu se změní opačným směrem. Adaptace pokračuje, dokud všechny neurony správně neklasifikují vstupní data. V aktivační fázi je vždy aktivní pouze jeden neuron, jenž je aktivačnímu vzoru nejblíže. Tento neuron udává do jaké kategorie spadá aktivační vzor [31].

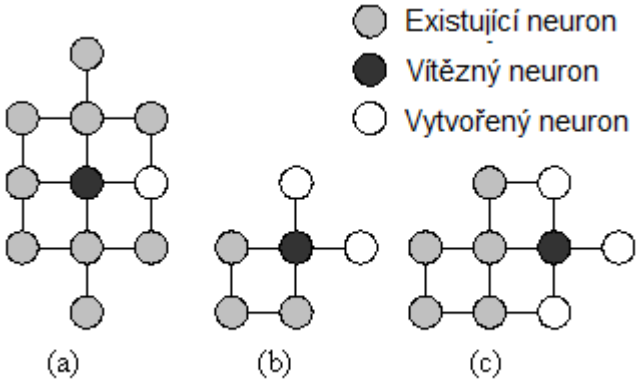
2.3 ASSOM

ASSOM je zkratka pro Adaptive-Subspace Self-Organizing Map (samoorganizační mapa s adaptivním podprostorem). Tato modifikace samoorganizační mapy extrahuje vztahy mezi vstupními daty pomocí filtrů, jenž jsou formovány automaticky v závislosti na sekvencích vstupních dat během učení. Hlavním rozdílem od Kohonenovy sítě je, že ve vstupní vrstvě sítě je místo obyčejných neuronů použito dvou vrstev neuronů: lineární vstupní vrstvy a vrstvy kvadratických neuronů [26]. Tato dvojvrstvá jednotka tvoří ony výše uvedené filtry. Nad maticí těchto filtrů je pak klasická výstupní vrstva samoorganizační mapy.

Tato síť se snaží odstranit nutnost předzpracování dat pomocí Gaborových a podobných filtrů [34],[13].

2.4 GSOM

Growing Self-Organizing Map (rostoucí samoorganizační mapa), řeší problém zvolení správné velikosti mapy. Při začátku adaptace začíná s minimem neuronů ve výstupní vrstvě (obvykle 4) a v průběhu učení vytváří nové. Metod růstu mapy existuje celá řada. Například doplňováním prázdných míst v okolí vítězného neuronu (obr. 2.1). Učící algoritmus je shodný s normální Kohonenovou mapou, startovní poloměr okolí se však volí podstatně nižší. Topologie sítě je stejná jako sítě Kohonenovy [2].



Obr. 2.1: Možné rozvoje mapy [14].

2.5 TASOM

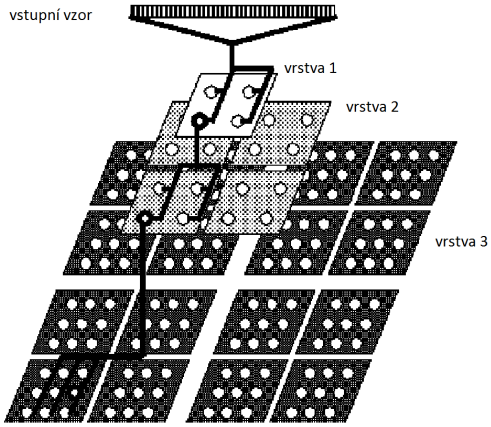
TASOM je zkratka pro Time Adaptive Self-Organizing Map (časově adaptivní samoorganizační mapa). Podobně jako Kohonenova neuronová síť je dvojvrstvá. Nicméně každý neuron má vlastní parametr učení a funkci sousedství, které se mění podle vstupních vzorů. Když se vzory blíží vahám neuronů, jsou učící parametry snižovány. Naopak čím jsou data rozdílnější od vah neuronu, tím více jsou učící parametry zvyšovány. Což vede k rychlejšímu přibližování vah k vstupním datům [24]. Použití vlastních parametrů učení a funkce sousedství zvyšuje výpočetní náročnost algoritmu. Avšak mapa konverguje mnohem rychleji než klasická Kohonenova síť. Uvádí se že TASOM je zhruba o 1/3 výpočetně náročnější než SOM, ale konverguje zhruba 5* rychleji, pro různé vstupní data se však rychlost konvergence může lišit [25].

2.6 HSOM

HSOM je zkratkou pro Hierarchical Self-Organizing Map (hierarchiká samoorganizační mapa). Tento typ využívá několik obyčejných samoorganizačních sítí, které jsou hierarchicky uspořádány do vrstev. Počátkem je kořenová mapa, každému neuronu v ní je v nižší vrstvě vytvořena samoorganizační mapa, další vrstvy jsou tvořeny obdobně. Na obrázku 2.2 je zobrazena třívrstvá HSOM s kořenovou mapou 2*2 neurony. V druhé vrstvě jsou tedy 4 mapy, každá o velikosti 2*2 neurony. V poslední vrstvě je 16 map o velikosti 3*3 neurony.

Učení probíhá od kořenové mapy k nižším vrstvám. První je naučena kořenová mapa na všechny trénovací data. Mapy v nižších vrstvách jsou pak učeny na trénovací vzory, které aktivují neuron, jenž je hierarchicky nad nimi (viz obrázek 2.2). HSOM vytváří lepší shluky než klasická samoorganizační mapa, díky zpřesňování shluků v podřízených vrstvách. Další výhodou je rychlost učení, naučení více malých map zabere méně času než naučení jedné veliké [30].

Mezi rozšíření této mapy patří takzvaná GHSOM, Growing Hierarchical Self-Organizing Map. Tato mapa je zpočátku tvořena pouze kořenovou mapou, přičemž mapa může růst jak do šířky tak do hloubky. Každá mapa může zvyšovat počet svých neuronů (jedná se o GSOM mapy) a každému neuronu může být vytvořena podřízená mapa v nižší vrstvě. Parametry pro růst mapy jsou pokrytí vstupního prostoru a kvantizační chyba shluků [30].



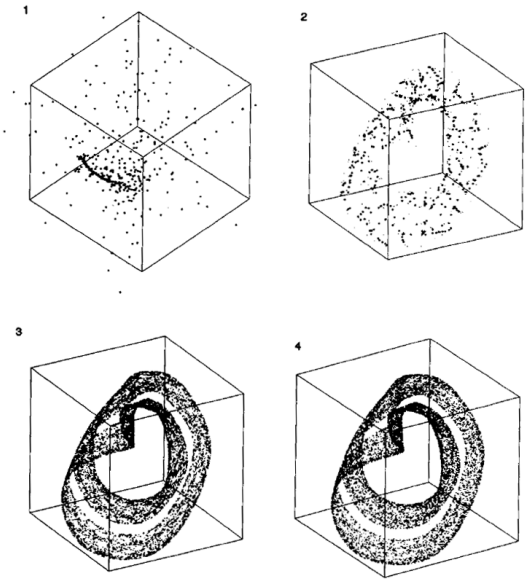
Obr. 2.2: Vrstvy HSOM [30].

2.7 LAMSTAR

LAMSTAR je zkratka pro Large Memory Storage and Retrieval neural network (Neuronové sítě pro uchovávání a správu objemných dat). Tuto mapu vynalezli Daniel Graupe, Hubert Kordylewsky a Nathan Schneider. Byla vynalezena speciálně pro aplikace pracující s velkým počtem dat s mnoha atributy, přičemž některé jsou zašuměné či úplně chybějící. Tato síť je velmi rychlá, je schopna růstu a zmenšování se, zapomínání, extrapolace a interpolace. Skládá se z velkého množství vstupních samoorganizačních map o různé velikosti, jenž zpracovávají jednotlivé podmnožiny z celkových dat předložených na vstup LAMSTARu, dále jednu nebo více samoorganizačních map, jenž slouží jako výstup LAMSTARu, a množství synapsí, jenž spojují vítězné neurony vstupních map s vstupními vrstvami výstupních map. Adaptační algoritmus mění jak váhy jednotlivých neuronů tak váhy synapsí mezi jednotlivými samoorganizačními mapami. Váhy jsou adaptovány pomocí učení posilováním [9].

2.8 Neural Gas

Neural Gas představili v roce 1991 Thomas Martinetz a Klaus Schulten. Neurony ve výstupní vrstvě nejsou nijak uspořádány. Využívá algoritmu soutěžního učení, svou váhu změní všechny neurony, přičemž vítěz své váhy změní nejvíce a se zmenšující se shodou se změna exponenciálně snižuje [18]. Název Neural Gas vychází z toho, jak se během učení neurony šíří datovým prostorem jako plyn (obr. 2.3). Existuje taktéž modifikace této sítě, která zvyšuje počet svých neuronů, Growing Neural Gas. Díky tomu je schopná lépe pokrýt vstupní prostor bez znalosti rozložení dat ve vstupním prostoru [10].



Obr. 2.3: Adaptace sítě Neural Gas v 3D prostoru [18].

3 KOHONENOVA SAMOORGANIZAČNÍ MAPA

Kohonenova samoorganizační mapa je softwarová metoda pro mapování více-dimenzionálního prostoru do méně-rozměrného prostoru, obvykle dvojrozměrného. Přičemž statistické nelineární vztahy mezi mnoho-dimenzionálními daty se konvertují v jednoduché geometrické vztahy zobrazené v málo rozměrném prostoru. Přesněji řečeno vzory, které jsou si blízké ve vstupním prostoru, způsobí v síti odezvu na neuronech, které jsou si blízké ve výstupním prostoru. Této vlastnosti je dosaženo využitím funkce sousedství.

Těchto vlastností může být taktéž využito pro shlukovou analýzu.

Principem funkce této neuronové sítě je naučení sítě učícími daty s tím, že není dán požadovaný výsledek. Síť se chová tak, že v průběhu učení přizpůsobuje váhy neuronů, aby výstupní neurony odpovídali učícím datům.

V aktivním režimu se po předložení dat aktivuje vždy pouze jeden výstupní neuron. Vstupní data, která jsou si blízká, způsobí odezvu na tom samém nebo na jiném blízkém neuronu. Neuron nebo skupiny sobě blízkých neuronů, pak reprezentují nějaký objekt nebo třídu objektů ze vstupního prostoru. Z rozložení takto naučené sítě se dají získat určité abstraktní informace.

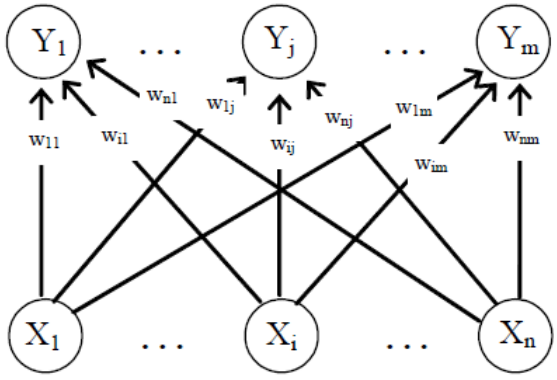
V praxi jsou samoorganizační mapy oblíbeny díky své malé výpočetní náročnosti. Možnosti využití Kohonenových map jsou popsány v kapitole 4.

3.1 Topologie

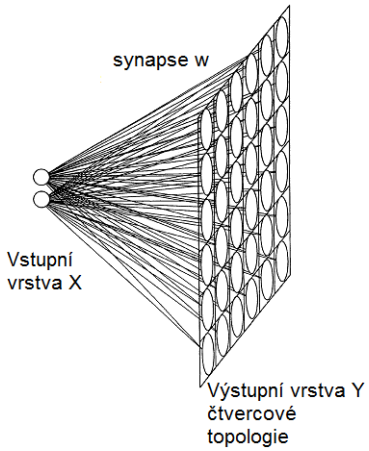
Jedná se o dvojvrstvou síť. Počet neuronů obsažený ve vstupní vrstvě udává dimenzi vstupních dat. Druhá vrstva, tedy výstupní vrstva, je uspořádána do určité topologické struktury. Nejčastěji dvojrozměrná mřížka s neurony uspořádanými do čtvercové nebo šestiúhelníkové struktury. Lze se také setkat s výstupní vrstvou ve tvaru jednorozměrné řady neuronů. Topologie výstupní vrstvy je důležitá z hlediska otázky okolí, tedy který neuron sousedí s kterým, viz kapitola 3.2. Všechny neurony ve výstupní vrstvě jsou navíc propojeny laterálními vazbami, viz kapitola 3.4.

Na obrázku 3.1 je zobrazena mapa z hlediska propojení vstupní a výstupní vrstvy. Síť se skládá z n neuronů ve vstupní vrstvě, $X_1, ..., X_n$. Z m neuronů ve výstupní vrstvě, $Y_1, ..., Y_m$. Přičemž každý výstupní neuron je propojen se všemi neurony ve vrstvě vstupní pomocí synapsí s váhou w. Dimenze vstupních dat je n. Každý neuron ve výstupní vrstvě je definován souborem vah $w_{i1}, ..., w_{in}$.

Na obrázku 3.2 je znázorněno propojení vstupní vrstvy X s topologicky uspořádanou výstupní vrstvou Y. Z hlediska výsledků sítě nás zajímá pouze topologicky uspořádaná vrstva výstupní,vstupní vrstva pouze udává dimenzi vstupního prostoru, proto bude dále zobrazována pouze výstupní vrstva sítě.



Obr. 3.1: Kohonenova mapa, pohled 1 [31].



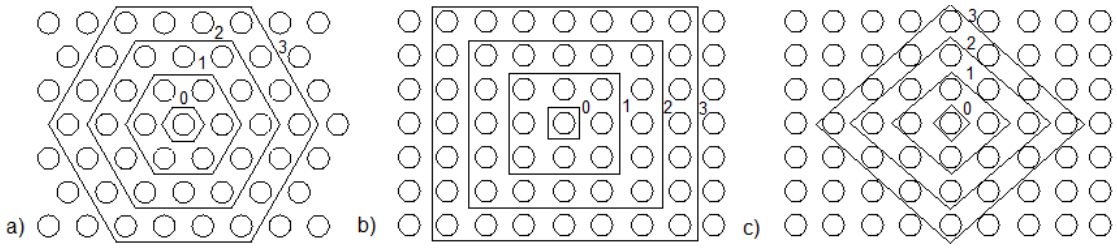
Obr. 3.2: Kohonenova mapa, pohled 2.

3.2 Funkce sousedství

Funkce sousedství má na chování Kohonenovy mapy zásadní dopad, zaručuje uspořádanost naučené mapy tak, že blízké data ve vstupním prostoru aktivují neurony jenž jsou si blízké ve výstupním prostoru. Aby byla tato vlastnost zachována musí být uvažované okolí nenulové.

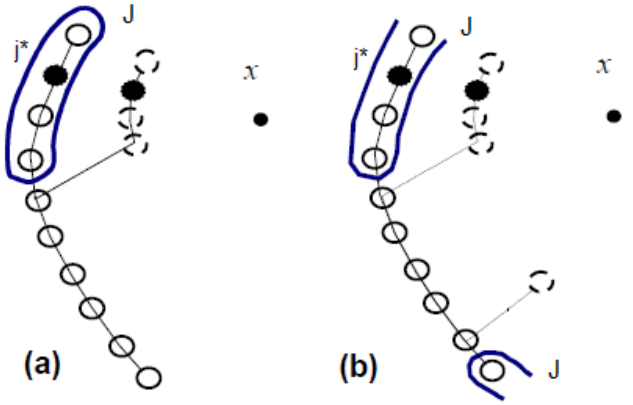
Funkce sousedství udává poloměr okolí ρ od daného neuronu, všechny neurony v tomto okolí jsou označeny jako sousedé onoho neuronu. Okolí může být ostré, tedy neuron buď je sousedem nebo není, nebo spojité, tedy že každému neuronu je přiřazena hodnota určující jeho náležení do okolí pomocí funkce příslušnosti. Tou bývá obvykle Gaussova funkce, nebo funkce Mexický klobouk (angl. mexican hat).

Na obrázku 3.3 je zobrazeno ostré okolí pro různé topologie a pojetí okolí. Číslo vedle ohraničení udává poloměr okolí ρ. Na obrázku 3.3 a) je mapa se šestiúhelníkovou strukturou, 3.3 b) a c) zobrazují mapu s čtvercovou topologií, na každém



Obr. 3.3: Okolí neuronu.

obrázku s jiným pojetím okolí. Obvyklejší je tvar sousedství zobrazeného na obrázku 3.3 b).



Obr. 3.4: Rozdíl mezi okolím a) čárové b) prstencové sítě [21].

Další vlastností okolí je, zda je limitováno koncem mapy a nebo pokračuje dál na protější straně mapy. Obvykle se to využívá u map tvořených pouze jednou řadou neuronů. Mapu, jejíž okolí je ohraničené koncem mapy, označujeme jako čárovou (angl. Linear). Mapu, jejíž okolí ohraničené není, jako prstencovou (angl. Annular). Rozdíl je zobrazen na obrázku 3.4. Učící vzor je označen x. Černě je vyznačen vítězný neuron $j*$, kolem něhož je okolí J, s poloměrem $\rho = 2$. Čárkovaně je pak naznačena změna mapy po jednom kroku adaptace.

Při použití neohraničeného okolí koncem mapy u dvojrozměrné mapy získáváme mapu toroidního tvaru.

3.3 Učení

Učící algoritmus vychází ze strategie soutěžního učení, jedná se tedy o učení bez učitele. Při učení postupně síti předkládáme jednotlivé tréninkové vzory. Po předložení tréninkového vzoru proběhne mezi neurony soutěž, vítězným neuronem je ten

jehož Eukleidovská vzdálenost od předloženého vzoru je nejmenší. Vítěznému neuronu jsou upraveny váhy tak, aby se přiblížily učícímu vzoru. Míra změny vah udává parametr učení $\mu \in (0, 1\rangle$. U ostatních neuronů je míra změn vah dána kromě μ jejich příslušností do okolí. Pokud je okolí ostré, neurony buď do okolí patří nebo ne. Ty které ano, změní váhy ve stejné míře jako vítězný neuron. Ty které ne, naopak nezmění své váhy vůbec. U okolí spojitého se velikost změny vah postupně snižuje, s tím jak roste vzdálenost od vítězného neuronu.

Aby síť konvergovala k určitému stavu, je obvyklé, že se μ snižuje s kroky adaptace. Obvyklou závislostí μ na krocích je lomená nebo exponenciální funkce.

Taktéž okolí mění s krokem svou velikost, na začátku je velké a postupně se snižuje až k nule. To zaručuje na začátku rovnoměrné rozprostření neuronů a na konci naopak přesné doučení jednotlivých neuronů na jim odpovídající vzory.

Algoritmus učení:

1. Inicializace počátečních hodnot vah sítě w, velikosti parametru učení μ a poloměru sousedství ρ (respektive například velikost parametru σ pro okolí definované Gaussovou funkcí).
2. Výběr a předložení učícího vzoru $x = (x_1, x_2, ..., x_n)$.
3. Pro každý výstupní neuron j vypočítat Eukleidovskou vzdálenost D(j) od učícího vzoru x:

$$D(j) = \sqrt{\sum_{i=1}^{n} (w_{ij} - x_i)^2}. \tag{3.1}$$

4. Najít neuron j^*, pro nějž je $D(j^*)$ nejmenší.
5. Aktualizace vah všech neuronů. $\Theta(j^*)$ je funkce příslušnosti do okolí, která závisí na vzdálenosti neuronu od vítězného neuronu j^*:

$$i = 1, ..., n$$

$$w_{ij}(t + 1) = w_{ij}(t) + \Theta(j^*)\mu[x_i(t) - w_{ij}(t)]. \tag{3.2}$$

6. Skok na krok 2, dokud nejsou předloženy všechny učící vzory.
7. Aktualizace parametru učení μ a poloměru okolí ρ.
8. Podmínka ukončení, pokud není splněna opakovat od kroku 2.

Ideálním stavem naučené sítě je, pokud každému učícímu vzoru odpovídá jeden výstupní neuron sítě [31].

3.4 Aktivace sítě

Pro Kohonenovu síť v aktivním režimu je charakteristické, že je vždy aktivní pouze jeden neuron. A to ten, který se nejvíce blíží vstupním datům. To je dosaženo propojením všech neuronů ve výstupní vrstvě laterálními vazbami.

Při aktivním režimu sítě se po předložení vstupních dat snaží všechny výstupní neurony zeslabit potenciál ostatních výstupních neuronů silou úměrnou jeho potenciálu (ten je tím větší, čím je neuron blíž vstupním datům). Neuron, který je vstupu nejblíže, tedy ostatní neurony utlumí a zůstane aktivní jen on sám (tkzv. laterální inhibice) [31].

Algoritmus aktivace:

1. Předložení vstupních dat $x = (x_1, x_2, ..., x_n)$.
2. Pro každý výstupní neuron j vypočítat Eukleidovskou vzdálenost od vstupních dat.
$$D(j) = \sum_{i=1}^{n} (w_{ij} - x_i)^2. \tag{3.3}$$
3. Najít neuron j^*, pro nějž je $D(j^*)$ nejmenší.
4. Neuron j^* je aktivní, ostatní neurony neaktivní.

3.5 Způsoby reprezentace výsledků

Získání informací z naučené Kohonenovy mapy je nejdůležitější a nejobtížnější částí praktické aplikace. Jak již bylo dříve řečeno, Kohonenova mapa dává velice abstraktní informace o souvislostech mezi daty. Existuje však několik způsobů, jak takové informace zvýraznit a vyčtení informací z mapy zjednodušit. V této kapitole jsou probrány hlavní metody prezentace výsledků z naučené Kohonenovy mapy.

3.5.1 U-Matrix

U-Matrix je zkratka pro Unified Distance Matrix (matice jednotných vzdáleností). Je to velice používaný a užitečný nástroj pro prezentaci výsledků dosažených samoorganizačními mapami. U-Matrix zobrazuje Eukleidovskou vzdálenost neuronu od jeho přímo sousedících neuronů pomocí škály šedi ve 2D-mapě nebo jako 3D krajinou mapu.

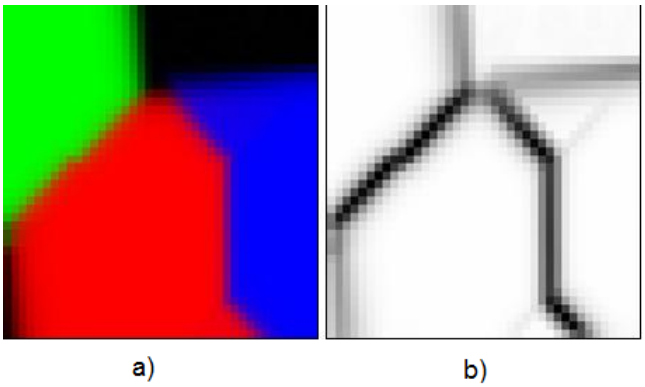
U-Matrix se vytváří z naučené samoorganizační mapy. Uvažujme neuron j a všechny neurony, které sním sousedí, označme jako množinu NN(j). Pak dle [29]:

$$U_h(j) = \sum_{m \in NN(j)} d[w(j) - w(m)], \tag{3.4}$$

kde $w(j)$ je vektor vah neuronu j, a $d[w(j) - w(m)]$ je Eukleidovská vzdálenost neuronu j od neuronu m, jenž přímo sousedí s neuronem j. Sečtená hodnota pro všechny neurony z množiny NN(j) udává celkovou vzdálenost neuronu j od jeho okolních neuronů. Tento výpočet provedeme pro všechny neurony, čímž získáme kompletní matici vzdáleností.

U-Matrix vykreslíme ve škále šedi nebo 3D mapě. Přičemž obvykle nejnižší vypočtená hodnota U_h je ve škále označena bílou (nebo nejmenší výškou v 3D mapě), a nejvyšší vypočtená hodnota černou (nejvyšší výškou v 3D).

Bílé plochy ('dna údolí'), označují skupiny neuronů které jsou si velmi blízké. Černé místa ('hory') naopak místa, kde se neurony velmi liší. Bílé plochy ohraničené černou linkou pak vytyčují jednotlivé skupiny neuronů, jež jsou si podobné.



Obr. 3.5: a) mapa 3 barev b) její U-Matrix [17].

Na obrázku 3.5 a) vidíme samoorganizační mapu naučenou na tři barvy pomocí modelu RGB, váhy jednotlivých neuronů jsou prezentovány odpovídající barvou v 2D mapě. Na obrázku 3.5 b) je U-Matrix této mapy. Jasně vidíme černě ohraničené bílé plochy, z nichž každá reprezentuje jednu barvu.

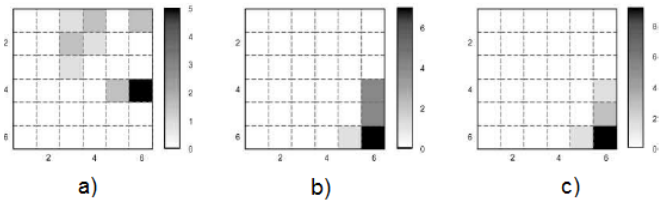
Pomocí U-Matrix jsme schopní rozčlenit samoorganizační mapu na oblasti, jenž odpovídají spolu souvisejícím vstupním datům, a zřetelně určit hranice mezi nimi. Jednotlivým oblastem je možno přiřadit název nebo význam. A následně pak v aktivním režimu vstupní data zařadit do příslušné kategorie na základě toho, kde jimi aktivovaný neuron leží.

3.5.2 Aktivační mapa

Aktivační mapa zobrazuje četnost aktivace neuronů ve výstupní mapě. Obvykle ve škále šedi, přičemž nejčastěji aktivovaný neuron je zobrazen černě a nejméně aktivované neurony bíle. Jednotlivým částem mapy může být opět přiřazen název

nebo určitý význam.Při aktivaci daty získanými v čase je možné čas zohlednit a vykreslit například stavovou trajektorii systému, z nějž data pocházejí.

Praktické využití je dobře ilustrováno v [1]. Zde byla aktivační mapa využita k vyhodnocení kvality mořské vody, měření složení vody bylo prováděno na několika stanicích. Mapa byla naučena na data ze všech stanic, aktivační mapy pak byli získány aktivací naučené mapy pomocí dat pouze z dané stanice. Na obrázku 3.6 vidíme tři aktivační mapy. Srovnáním těchto map lze na první pohled, říct které stanice měli podobnou kvalitu vody (b a c). Podrobnějším rozebráním výsledků je pak možné jednotlivým kvadrantům mapy přiřadit odpovídající charakteristické složení vody.



Obr. 3.6: Aktivační mapy kvality vody [1].

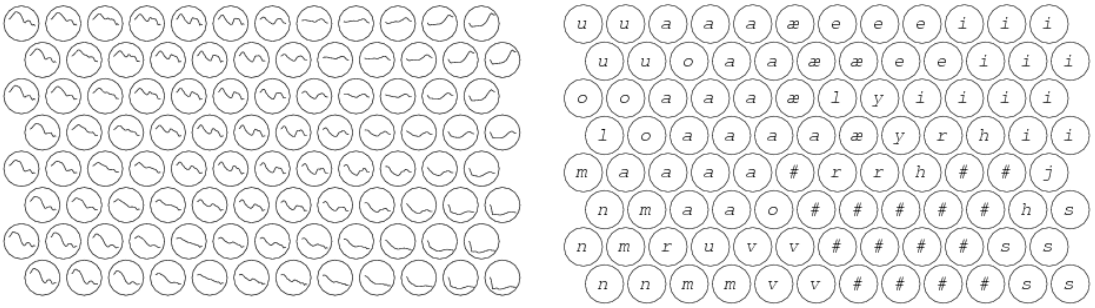
3.5.3 Dílčí rovina

Při tomto zobrazení je vybrána pouze jedna dílčí dimenze z celého vstupního prostoru a jí odpovídající váhy výstupních neuronů, jenž jsou zobrazeny ve škále šedi v dvojrozměrné mapě. Výsledná dílčí mapa má stejný formát jako původní Kohonenova mapa, jen je v ní zachycen pouze jeden ze vstupů. Celkem je tedy možno vytvořit tolik dílčích map, kolik je neuronů ve vstupní vrstvě mapy.

Tento způsob zobrazení se využívá s dalšími metodami reprezentace výsledků sítě. V [13] je popsáno využití tohoto zobrazení pro sledování stavu systému.

4 APLIKACE KOHONENOVÝCH MAP

Kohonenova neuronová síť byla prakticky poprvé využita na rozpoznání hlasu. Na obrázku 4.1 vlevo vidíme spektra Finských hlásek naučené v Kohonenově neuronové síti, vpravo přiřazení odpovídajících písmen neuronům. V dnešní době se samoorganizační mapy využívají v široké škále aplikací, jejichž výběr bude uveden níže.



Obr. 4.1: Mapa pro rozpoznání hlasu [19].

Typickým úkonem je rozpoznávání určitých vzorů ve vstupních datech. Některé z možných aplikací:

- rozpoznání hlasu [13],
- klasifikace mraků ze satelitních fotek,
- rozpoznávání příkazů mikroprocesoru, z elektromagnetických signálů [20],
- analýza signálů z mozku,
- hodnocení a předvídání kvality mořské vody [1],
- podoba fotky [4],
- rozeznání tváře [33],[13],
- analýza a rozpoznání textury [13].

Dalším úkolem, pro který jsou Kohonenovy mapy využívány, je organizace či vizualizace velkých databází dat. Příklady využití:

- mapa chudoby ve světě [11],
- organizace a hledání z velkých kolekcí dokumentů [12],
- vizualizace demografického profilu města [27].

Kohonenovy neuronové mapy se taktéž využívají pro zmenšení dimenze vstupních dat:

- Kvantování barev [6],
- generování a registrace značek z obrazu [23].

Dále je možné Kohonenovy mapy využít pro vizualizaci a analýzu systémů [13]:

- vizualizace a analýza stavové trajektorie,
- detekce poruchy,
- analýza a monitorování procesu.

4.1 Použití pro segmentaci obrazu

Modely některých tříd objektů jako lidské tváře, ručně psaného písma, či přírodních povrchů, jsou velmi těžce explicitně modelovatelné. Proto využití samoorganizačních map, jejichž učení probíhá bez učitele, může být přínosné.

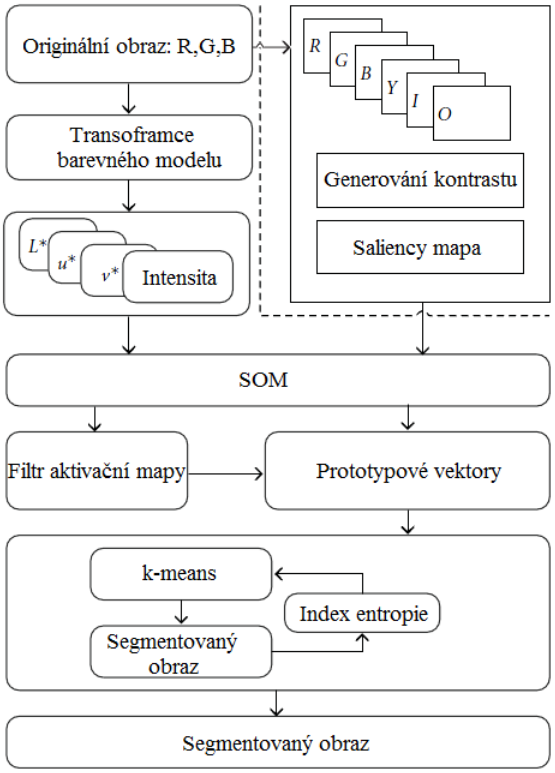
Pro většinu případů je nutno obraz před aplikací Kohonenovy mapy nějakou jinou metodou předzpracovat. Přitom pro každý specifický problém a pro každý typ obrázků jsou metody předzpracování jiné. Stejně tak jako metody použití samoorganizační mapy na obrázek.

Dále je uvedeno několik praktických příkladů využití Kohonenovy mapy na segmentaci obrazu.

4.1.1 Segmentace barevných obrazů s využitím k-means a saliency mapy

V článku [5] je popsána segmentace přírodních barevných obrazů s využitím samoorganizační mapy, k-means a saliency mapy. Samoorganizační mapa zde zajišťuje snížení počtu vstupních dat, které vstupují do k-means algoritmu. Z neuronů samoorganizační mapy jsou pomocí k-means algoritmu vytvořeny další shluky, jenž slouží k výsledné segmentaci obrazu.

V tabulce 4.1.1 je uvedena časová náročnost jednotlivých algoritmů. Přičemž SOM-K je algoritmus využívající samoorganizační mapu a k-means, SOM-KS využívá samoorganizační mapu, k-means a saliency mapu. Pro srovnání je uveden často používaný algoritmus JSEG. Časy byly změřeny během algoritmů na stejném PC, na stejných datech.



Obr. 4.2: Princip algoritmu [5].

Metoda	SOM-KS	SOM-K	k-means	JSEG
Čas [s]	16.1	7.8	33.7	<8.4

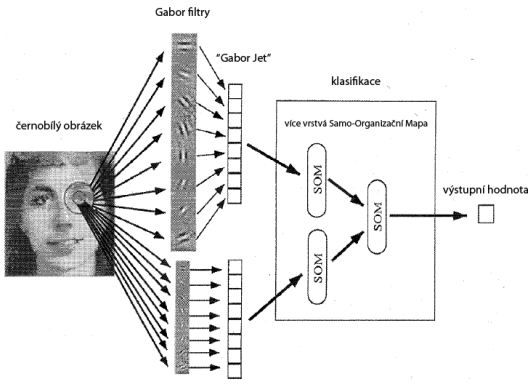
Tab. 4.1: Časová náročnost algoritmů SOM-K, SOM-KS, k-means a JSEG.

Z tabulky je zřejmé přínos samoorganizační mapy pro snížení celkové výpočetní náročnosti k-means algoritmu. Algoritmus k-means, který využil předzpracování pomocí samoorganizační mapy, byl více než třikrát rychlejší. Na obrázku 4.3 jsou ukázány výsledky segmentace.

Obr. 4.3: Výsledky segmentace při použití různých metod [5].

4.1.2 Rozeznávání lidských tváří

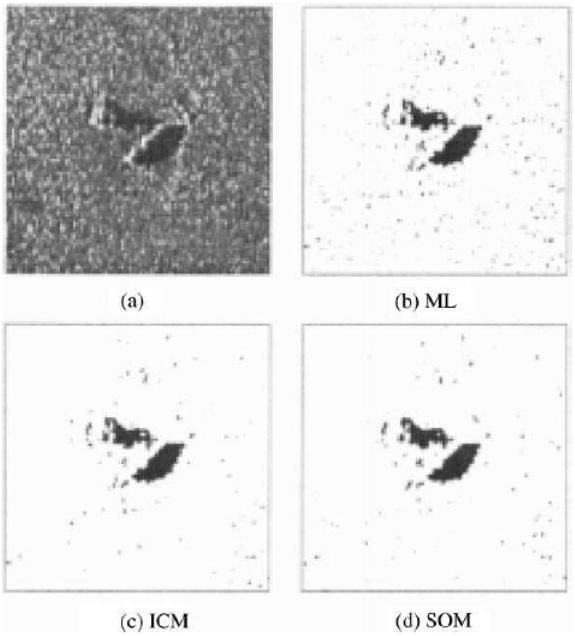
V [13] je popsána metoda rozeznávání lidských tváří. Bylo využito předzpracování pomocí 16-ti Gaborových filtrů, jenž byli aplikovány na všechny okna obrázku. Každá část obrázku je klasifikována sadou všech výstupů z filtrů. Tyto sady jsou pak zpracovány dvojvrstvou Kohonenovou mapou, která byla naučena obrazy lidských tváří a jiných přírodních objektů. Po zpracování všech části obrazu se z výstupů mapy sestaví histogram, na základě něhož jsme schopní klasifikovat objekt na obrazu. Na obrázku 4.4 je znázorněn postup zpracování obrazu.



Obr. 4.4: Rozeznání lidské tváře pomocí samoorganizační mapy [13].

4.1.3 Segmentace snímku ze sonaru

V [32]je popsáno využití pro segmentaci snímku ze sonaru. Do vstupní vrstvy mapy o devíti neuronech vstupuje výřez ze snímku o velikosti 3*3 body. Výstupní vrstva mapy má 100 neuronů, každý o devíti vahách. Mapa je naučena přímo na reálném sonarovém snímku. Po procesu učení se provede vyhodnocení neuronů, jenž odpovídají stínu a těm, které odpovídají odrazu. Následně je síť aktivována obrazem po oknech 3*3 body a výstup sítě je zakreslován do segmentovaného obrázku. Porovnání s jinými metodami segmentace viz obrázek 4.5.



Obr. 4.5: a)reálný sonarový snímek b)ML segmentace c)Markovova segmentace d)segmentace s použitím samoorganizační mapy [32].

5 PROGRAM KOHONENOVA NEURONOVÁ SÍŤ

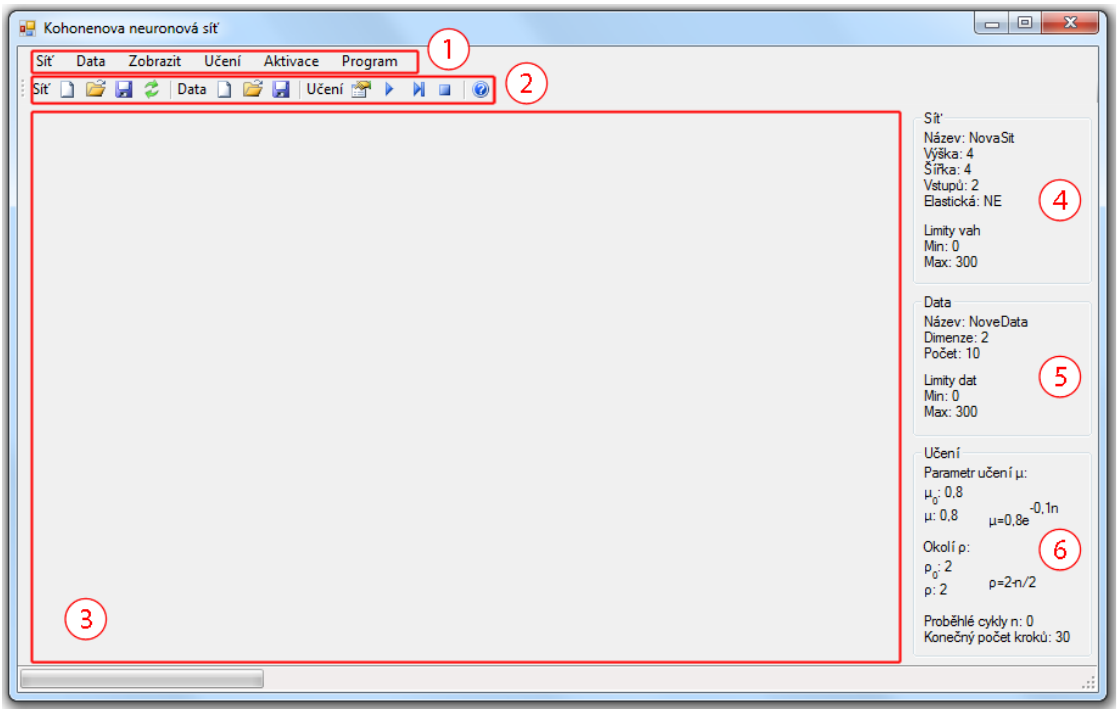
Program Kohonnenova Neuronová Síť, KNN, byl vytvořen pro potřeby výuky předmětu Umělá Inteligence. Proto byl kladen důraz na přátelskost uživatelského prostředí a bezproblémový chod aplikace. Hlavním cílem programu je studentům srozumitelně předvést, jak samo-organizační mapy fungují. To je docíleno grafickým zobrazením a podrobným krokováním učícího procesu.

Program využil pro svou bakalářskou práci student Miloslav Fic, na základě jeho požadavků byl program rozšířen o citlivostní analýzu.

Program byl napsán ve vývojovém prostředí Microsoft Visual Studio 2010, v jazyce C#.

5.1 Vizuální prostředí programu

Vizuální prostředí programu bylo navrženo s ohledem na přehlednost, ale aby přesto obsahovalo všechny podstatné informace a všechny důležité prvky byli snadno dostupné.



Obr. 5.1: Grafické prostředí programu.

1. Základní menu programu: obsahuje jednotlivé funkce programu.
2. Panel rychlého přístupu: obsahuje nejzákladnější funkce ovládání jako vytvoření nové sítě, uložení sítě/dat, inicializace sítě a podobně.
3. Zobrazovací plocha: zobrazuje síť a data zvoleným způsobem.
4. Informace o síti: zobrazuje informace o vytvořené síti, rozměry vstupní a výstupní vrstvy a podobně.
5. Informace o trénovacích datech: zobrazuje informace o načtených datech, počet vzorů, jejich dimenze aj..
6. Informace o nastavení učení: zobrazuje nastavení adaptace, parametr učení, velikost okolí a další.

5.2 Vytvoření mapy

Program umožňuje vytvoření Kohonenovy neuronové sítě čtvercové topologie, libovolných rozměrů s libovolným počtem vstupů. Dále je možno zvolit vlastnost elasticity, viz kapitola 3.2. Po vytvoření nové sítě jsou informace o ní zobrazeny v poli 4., viz obrázek 5.1.

Při vytvoření nové mapy je automaticky provedena náhodná inicializace vah, váhy jsou generovány v rozmezí hodnot minima a maxima sítě zadaného při vytváření sítě. Tyto minima a maxima však hrají roli pouze při inicializaci, při předložení dat, které jsou mimo tyto meze, mohou váhy neuronů taktéž přesáhnout tyto limity. Náhodnou inicializaci vah může uživatel kdykoliv spustit sám.

Kromě vytvoření nové mapy může uživatel načíst již vytvořenou mapu uloženou v souboru. A svou vytvořenou síť může taktéž do souboru uložit.

5.3 Učení

Program využívá algoritmus popsaný v kapitole 3.3.

Funkce sousedství je ostrá a časová závislost poloměru okolí ρ je lineárně klesající. Závislost okolí na čase:

$$\rho = \rho_0 - \frac{n}{j}, \tag{5.1}$$

kde n je počet proběhlých iterací, $j \in (0, \infty)$ je parametr udávající rychlost zmenšování okolí (udává počet iterací, po kterých se okolí sníží o jedna). Pro $j = 0$ je funkce snižování okolí s počtem iterací vypnuta. ρ_0 je počáteční poloměr okolí a ρ poloměr aktuální. Okolí může nabývat pouze celočíselných hodnot (dělení je celočíselné). ρ_0 a j jsou uživatelem voleny.

Funkce snižování parametru učení μ s časem je exponenciální.

$$\mu = \mu_0 e^{-kn}, \tag{5.2}$$

kde n je opět počet proběhlých iterací, parametr $k \in \langle 0, \infty)$ udává rychlost snižování μ, μ_0 je počáteční hodnota parametru učení a μ je aktuální hodnota parametru. Parametry k a μ_0 volí uživatel.

Posledním uživatelem nastavovaným parametrem je počet iteračních kroků.

Program umožňuje provést celou iteraci naráz, přičemž při vypnutém průběžném vykreslování jsou výsledky vypsány až po ukončení učení, při zapnutém průběžném vykreslování jsou výsledky vypisovány po každém kroku adaptace. To je však časově náročné. Dále je možnost adaptaci krokovat po jednotlivých fázích.

5.4 Aktivace mapy

Program využívá algoritmus popsaný v kapitole 3.4. Aktivace může být provedena několika způsoby v závislosti na rozměru vstupní vrstvy sítě. Nejzákladnějším způsobem je ruční zadání aktivačního vzoru. Další možností je načíst aktivační data ze souboru. Speciální možností je aktivovat síť se dvěma vstupy pomocí kursoru klikáním na zobrazovací plochu viz obrázek 5.2, po kliku je na dané místo přidán aktivační vzor, jehož atributy jsou dané souřadnicemi v místa kliku.

5.5 Způsoby zobrazení sítě

Program disponuje několika způsoby zobrazení sítě. Síť je zvoleným způsobem zobrazena v poli 3., viz obrázek 5.1. Každé z těchto zobrazení má jiné využití a přednosti. Pomocí těchto zobrazení jsou studentovi srozumitelně ukázány hlavní vlastnosti samo-organizační mapy.

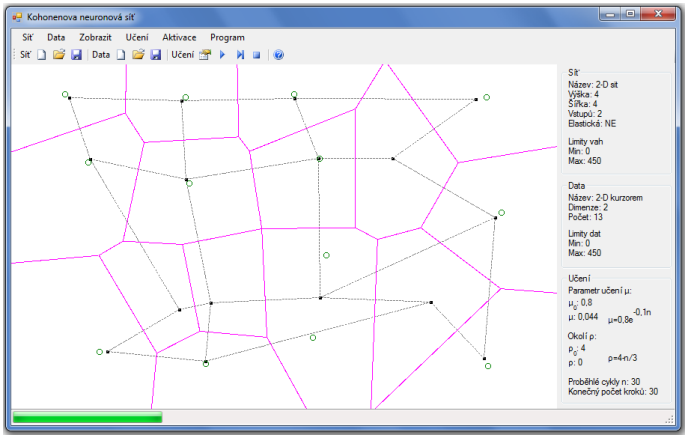
5.5.1 Textové zobrazení

Textové zobrazení umožňuje uživateli odečíst přesné hodnoty jednotlivých vah neuronů. Neurony jsou uspořádány stejně jako ve výstupní vrstvě mapy.

5.5.2 2D zobrazení

Program umožňuje zobrazit síť do dvojrozměrného prostoru (obr. 5.2). Toto zobrazení může být použito pouze pro síť se dvěma neurony ve vstupní vrstvě. Jednotlivé neurony výstupní vrstvy jsou reprezentovány body ve dvojrozměrném prostoru, jejich souřadnice jsou dány jejich váhami. Hranice, kterými neurony dělí vstupní

prostor na jednotlivé shluky, jsou zobrazeny fialovou čárou. Šedé přerušované čáry znázorňují vazby mezi přímo sousedícími neurony. Data jsou vykresleny jako zelené kola.

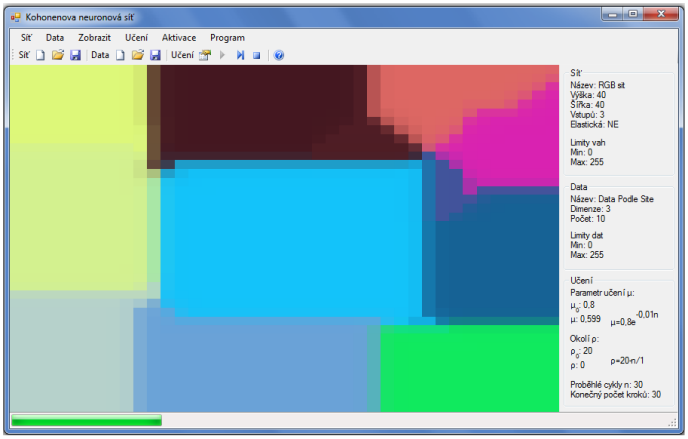


Obr. 5.2: 2D zobrazení dat, sítě a okolí neuronů.

Toto zobrazení slouží pro ilustraci procesu učení a aktivace mapy.

5.5.3 RGB zobrazení

Při RGB zobrazení je síť reprezentována pomocí palety barev (obr. 5.3). Toto zobrazení lze použít pro síť s třemi neurony na vstupu. Každá váha neuronu výstupní vrstvy definuje jednu složku z barevného modelu RGB. Neurony jsou zobrazeny jako obdélníky vybarvené danou barvou, přičemž rozložení obdélníku je stejné jako příslušných neuronů ve výstupní vrstvě.

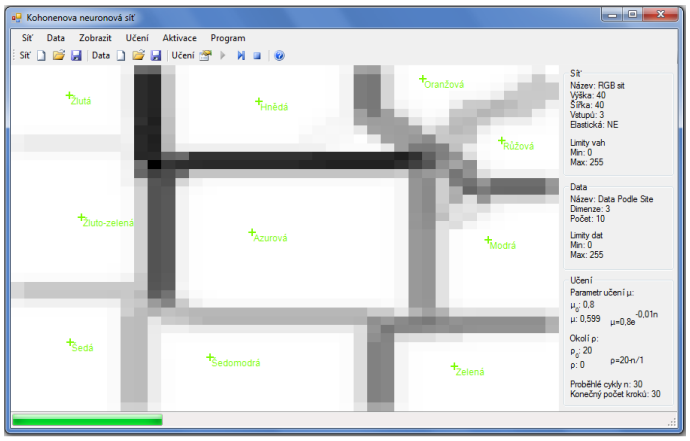


Obr. 5.3: RGB reprezentace mapy.

Toto zobrazení má za úkol prezentovat uspořádanost vstupně-výstupního mapování.

5.5.4 U-matrix

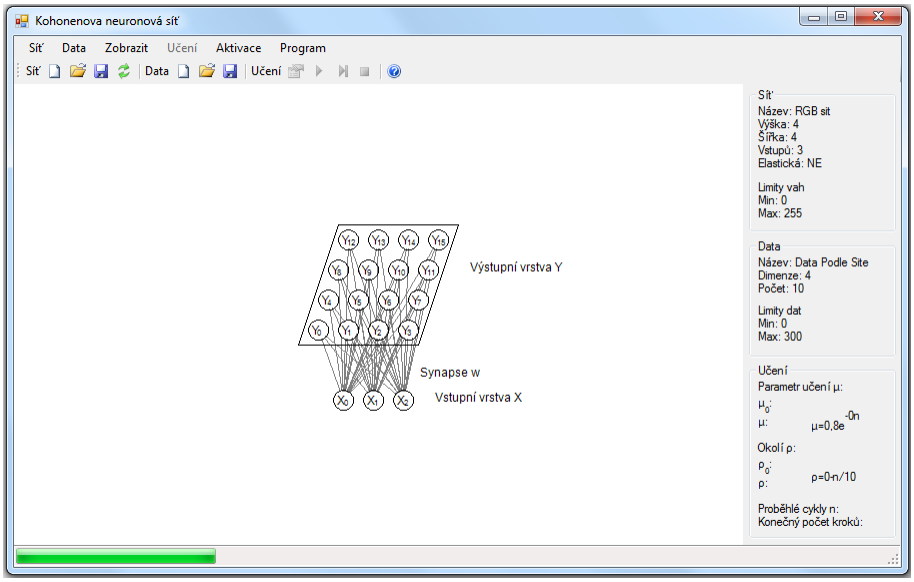
Program umožňuje zobrazit U-matrix sítě ve škále šedi. U-matrix je popsán v kapitole 3.5.1. Na obrázku 5.4 jsou mimo U-matrix zobrazeny i popisky, kterými si uživatel může mapu libovolně popisovat.



Obr. 5.4: U-matrix a ukázka značení.

5.5.5 Zobrazení vrstev sítě

Zobrazení vrstev mapy zobrazuje vstupní a výstupní vrstvu sítě a jednotlivé synapse mezi nimi.

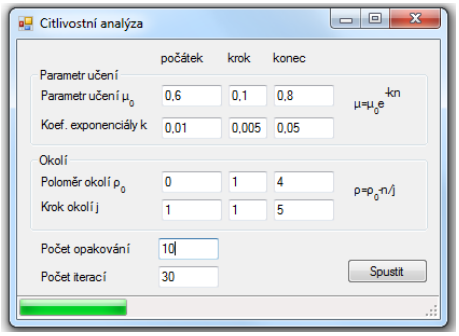


Obr. 5.5: Zobrazení vrstev sítě.

5.6 Citlivostní analýza

Modul citlivostní analýzy byl doprogramován na základě žádosti Miloslava Fice, který jej využil pro svou bakalářskou práci. Pro jeho správnou funkci je nutno mít nainstalovaný Microsoft Excel, verze 2007 nebo 2010, kompatibilita s jinými verzemi není zajištěna. Modul citlivostní analýzy umožňuje spustit velký počet trénovacích cyklů vytvořené sítě s různými parametry učení a jejich výsledky zapsat do Microsoft Excel souboru .xls.

Parametry, u kterých lze nastavit postupné měnění hodnoty, jsou: počáteční hodnota parametru učení μ_0, koeficient exponenciály k parametru učení, počáteční velikost okolí ρ_0, krok okolí j. Hodnoty, které jsou pevné pro všechny kroky cyklu, jsou počet iterací učícího algoritmu a počet opakování učení se stejnými parametry O. Okno nastavení citlivostní analýzy je ukázáno na obrázku 5.6.



Obr. 5.6: Nastavení citlivostní analýzy.

Výstupní soubor obsahuje hodnoty zvolených parametrů, chybu naučení sítě a výběrovou směrodatnou odchylku sítě. Chyba naučení sítě Δ_d udává průměrnou vzdálenost učících vzorů od nejbližšího neuronu:

$$\Delta_d = \frac{1}{O}\sum_{l=0}^{O}\frac{1}{N}\sum_{i=0}^{N} D_{li}(j*), \tag{5.3}$$

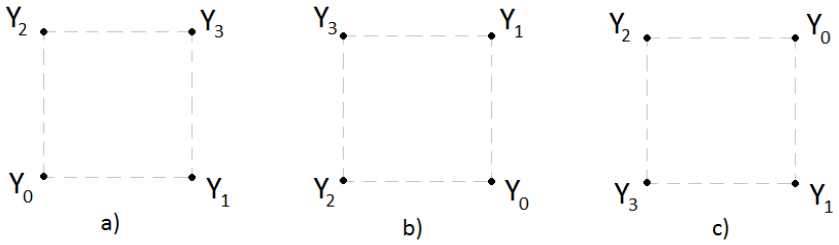
kde O je počet opakování učení, N je počet tréninkových vzorů, $D_{li}(j*)$ je vzdálenost i-tého učícího vzoru od nejbližšího neuronu $j*$ při l-tém opakování. Tato chyba postihuje schopnost sítě naučit se na předložená data při daných parametrech učení.

Výběrová směrodatná odchylka s sítě udává rozptyl jednotlivých neuronů při opakování procesu učení s náhodnou inicializací vah:

$$s = \frac{1}{O}\sum_{l=0}^{O}\sqrt{\frac{1}{m-1}\sum_{j=0}^{m}\sum_{i=0}^{n}(w_{lji} - \bar{w}_{lji})^2}, \tag{5.4}$$

kde O je počet opakování učení, n je rozměr vstupních dat, m je počet neuronů sítě, w je váha neuronu a $\bar{w}$ je průměrná váha neuronu. Problémem při výpočtu této

odchylky je různé natočení sítě při opakovaném učení. Byť jsou neurony přitaženy vždy na podobná místa, jedná se pokaždé o jiný neuron výstupní vrstvy. Při správném nastavení iteračního algoritmu je vždy zachován tvar sítě. Na obrázku 5.7 je problematika zjednodušeně zobrazena na síti se čtyřmi neurony, ve dvojrozměrném prostoru.



Obr. 5.7: Různé natočení mapy: a) původní mapa b) otočení mapy c) překlopení mapy přes úhlopříčku

Aby měla směrodatná odchylka sítě vypovídající hodnotu, bylo nutno toto vzít v potaz. Byl proto vytvořen jednoduchý algoritmus umožňující přeindexovat čtvercové sítě se dvěma vstupy tak, aby si neurony odpovídaly dle umístění v datovém prostoru.

6 SLAM

SLAM je zkratkou pro simultaneous localization and mapping, neboli průběžnou lokalizaci a mapování. Tato problematika se zabývá lokalizací robota v neznámém prostředí. Robot pro zjištění své polohy nevyužívá žádné vnější zdroje. Nemůže tedy využít například GPS navigaci, nebo různé navigační majáky či značky. Nejzákladnější metodou je odometrie. Při znalosti směru a rychlosti pohybu jsme schopni určit změnu polohy robota za jednotku času. Problémem je nepřesnost této metody. Chyba této metody je inkrementální a roste tedy s uraženou vzdáleností.

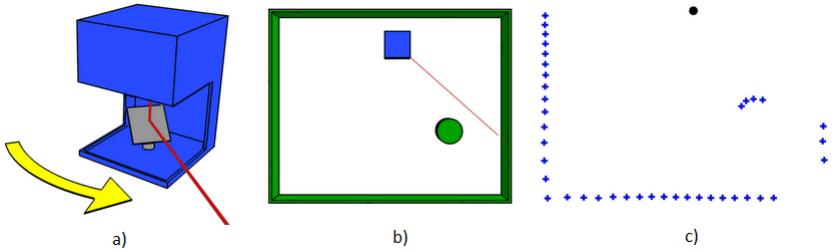
Pro zpřesnění lokalizace je vhodné využít další senzory: laserové skenery, ultrazvukové snímače, kamery. Pomocí těchto snímačů je určována změna polohy na základě změny snímaného obrazu vzhledem k obrazu předchozímu. Tato změna je obvykle vyhodnocována pomocí orientačních bodů v okolí, těmi můžou být například ostré hrany či rovné plochy. Na základě toho, jak se tyto orientační body posunuly, je určena změna polohy. Odometrie je pak obvykle využívána pouze k prvotnímu odhadu polohy, který je upřesněn v závislosti na informaci z ostatních snímačů.

Problematika SLAM je velmi obsáhlá, od snímání okolí robota přes zpracování dat a vyhodnocení dat až po samotný výpočet polohy robota [7].

6.1 Snímání okolí

Snímání okolí robota je nedílnou součástí problematiky sebelokalizace robota. Využívá se mnoho různých metod, mezi často používané patří snímání pomocí ultrazvukových snímačů, pomocí kamery, či pomocí laserových skenerů. Právě poslední jmenovaný způsob je stále běžnější. Problematika měření okolí laserem se nazývá LIDAR, Light Detection And Ranging. Měření vzdálenosti laserem je velmi rychlé a přesné. Nicméně hlavní výhodou pro účely robotiky je oproti ultra-zvuku či kameře velmi snadné zpracování měření. Nevýhodou je vyšší cena. Laserové detektory běžných vzdálenosti fungují na principu doby letu paprsku, paprsek je velmi úzký a měření je bodové.

Pro zachycení okolí robota se využívají PLS, laserové skenery vzdálenosti (proximity laser scanner). Skener může zachycovat pouze 2D rovinu, pak mluvíme o plošném laserovém skeneru, nebo 3D prostor. Laserový paprsek je obvykle rozmítán soustavou zrcadel (jedno pro 2D rovinu, dvou pro 3D rovinu), natáčených pomocí galvanometrů nebo se otáčí celá hlavice skeneru. Měření je prováděno po přesně definovaném kroku natočení. Jednotlivé odrazy pak vytvoří 2D/3D mrak bodů, který reprezentuje okolí skeneru.



Obr. 6.1: a) Plošný laserový skener. b) Okolí laseru. c) Změřené body [16].

Obvyklým výstupem ze skeneru jsou jednotlivé body popsané úhlem sejmutí a uraženou vzdáleností paprsku, transformaci těchto dat do kartézského souřadného systému lze provést dle rovnice:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} dcos(\theta)cos(\varphi) \\ dcos(\theta)sin(\varphi) \\ -dsin(\theta) \end{pmatrix}, \tag{6.1}$$

kde d je naměřená vzdálenost bodu, φ je vodorovný úhel laserového paprsku a θ úhel vertikální [3].

6.2 Získání orientačních bodů

Extrakce orientačních bodů je velmi důležitou částí SLAM. Jak bylo uvedeno v kaptiole 6, poloha robota je vyhodnocována na základě polohy orientačních bodů vůči robotu. Orientační body jsou určité význačné místa v okolí robota. Správný orientační bod by měl splňovat několik kritérií:

- Znovu pozorovatelnost: orientační bod musí být dobře znovu pozorovatelný. Při opakovaném průjezdu robota kolem orientačního bodu musí být opět rozpoznán. Taktéž musí být pozorovatelný z co největšího rozsahu úhlů.
- Rozpoznatelnost: orientační body nesmí být zaměnitelné.
- Stacionarita: orientační body se nesmí hýbat, nebo měnit svůj tvar a velikost.
- Četnost: orientačních bodů musí být dostatek, ale nesmí jich být příliš mnoho.

Splnění všech těchto podmínek je velmi obtížné a algoritmicky náročné.

Jedny ze základních metod pro extrakci orientačních bodů z laserového skenu jsou spiked landmarks (hrotové značky) a RANSAC (Random Sampling Consensus) [22].

Spiked landmarks

Spiked landmarks je nejjednodušší metoda pro extrakci orientačních bodů. Vyhledává objekty významně vyčnívající z okolí. Za orientační body označuje sousedící

body skenu, které se souřadnicemi liší o víc než zvolený práh. Jako orientační body jsou touto metodou označeny například nohy stolu, různé sloupky a podobné předměty. Volba prahu ovlivňuje počet nalezených orientačních bodů. Tento algoritmus selhává v prostředí bez úzkých ostře ohraničených předmětů. Vzhledem k jednoduchosti metody nejsou detekované orientační body příliš kvalitní (jsou snadno zaměnitelné [22]).

Obr. 6.2: Oranžové body jsou body rozpoznané jako orientační body [22].

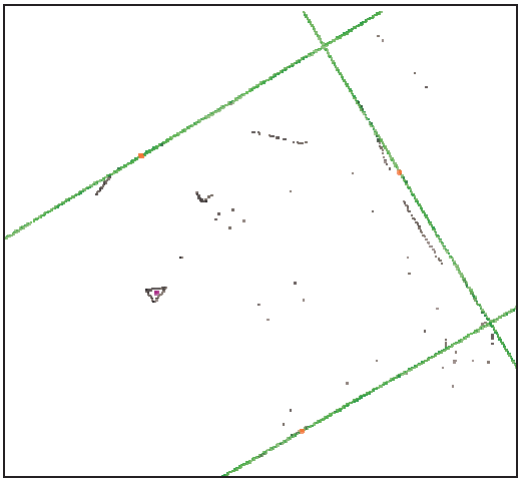
RANSAC

Metoda, která vyhledává ve skenu přímky. Tato metoda je vhodná zejména do interiéru budov, kde detekuje velmi dobře zdi. Metoda je odolná vůči výskytu osob na skenu, a nalezené orientační body jsou podstatně kvalitnější než body získané metodou spiked landmarks, ale je mnohem výpočetně náročnější.

Algoritmus metody [22]:

1. Načtení skenu a označení všech bodů jako nepřiřazené.
2. Náhodný výběr bodu ze množiny nepřiřazených bodů.
3. Výběr S náhodných bodů ležících v rozmezí D stupňů od bodu vybraného v předchozím kroku.
4. Proložení přímky body pomocí metody nejmenších čtverců.
5. Zjištění kolik nepřiřazených bodů leží na této přímce (s tolerancí X).
6. Pokud je počet bodů vyšší než práh C:
   (a) Výpočet nové přímky pouze pro body které leží na staré přímce.
   (b) Přidání této přímky mezi extrahované přímky.
   (c) Odstranění bodů, které leží na přímce z množiny nepřiřazených bodů.
7. Opakuj od kroku 2., dokud není splněna podmínka ukončení.

Podmínkou ukončení může být počet nalezených přímek, počet kroků algoritmu, počet nepřiřazených bodů. Výsledky algoritmu jsou laděny pomocí volby konstant S,D,X,C.



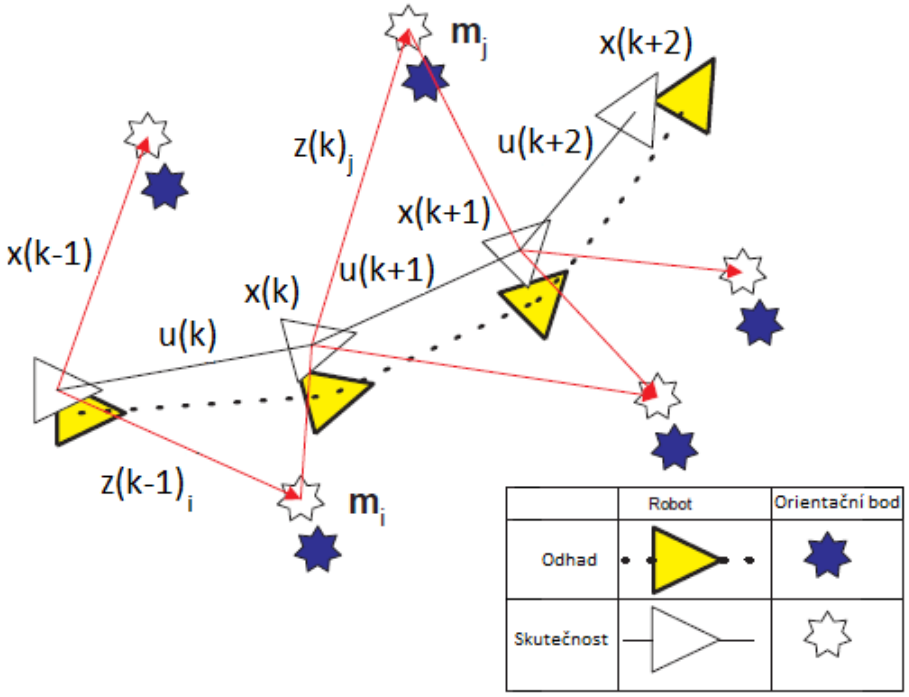
Obr. 6.3: Ukázka nalezených přímek v laserovém skenu [22].

Získání orientačních bodů pomocí neuronové sítě

V [8] je popsáno možné využití jednovrstvé neuronové sítě pro získání orientačních bodů při snímání okolí 16-ti ultrazvukovými snímači vzdálenosti. Základní myšlenkou je, že vhodné značky se podstatně liší od typického okolí robota. Jednovrstvá síť je naučena na predikování následující odezvy senzorů na okolí. Výstup sítě tedy odpovídá typické následující odezvě snímačů. Při vyhledávání orientačních bodů je naučené síti předložena minulá odezva senzorů a výstup sítě je porovnán s odezvou současnou, pokud chyba sítě překročí určitý práh, je současná pozice označena jako orientační bod. Hodnota prahu je počítána pomocí jednorozměrného Kalmanova filtru.

6.3 Výpočet polohy

V problematice SLAM je trajektorie robota a poloha orientačních značek určována online bez potřeby apriorní znalosti polohy. Základním problémem je, že je odhadována poloha jak robota tak orientačních značek. Pravá poloha není známa, proto se snažíme dosáhnout řešení, které bude v čase konvergovat ke skutečné poloze robota.



Obr. 6.4: Sebelokalizace robota [7].

Mějme mobilního robota, který se pohybuje v prostředí a snímá orientační body (obr. 6.4).

V čase k, jsou pak definovány následující proměnné [7]:

- $s(k)$: Stavový vektor popisující polohu a orientaci robota.
- $u(k)$: Vektor řízení aplikovaný v čase k-1, pro dosažení polohy $x(k)$ v čase k.
- m_i: Vektor popisující polohu i-tého orientačního bodu. Tato poloha je považována za neměnou.
- $z(k)_i$: Pozorovaná poloha i-tého orientačního bodu v čase k, soubor všech pozorování v čase k značíme $z(k)$.

Dále jsou definovány následující množiny:

- $S(0:k) = \{s(0), s(1), s(2), ..., s(k)\}$: Historie polohy robota.
- $U(0:k) = \{u(1), u(2), u(3), ..., u(k)\}$: Historie řízení.
- $m = \{m(0), m(1), m(2), ..., m(n)\}$: Množina všech orientačních bodů.
- $Z(0:k) = \{z(1), z(2), z(3), ..., z(k)\}$: Množina všech pozorování.

Pravděpodobnostní SLAM, definuje rozložení pravděpodobnosti stavu s jako [7]:

$$P(s(k)|Z(0:k), U(0:k), m), \tag{6.2}$$

které popisuje aposteriorní pravděpodobnost polohy $s(k)$ při znalosti historie řízení, množiny orientačních bodů a množiny všech pozorování. Dále je popsán model pozorování, který je dán pravděpodobnostním rozložením [7]:

$$P(z(k)|s(k), m), \tag{6.3}$$

které udává aposteriorní pravděpodobnost pozorování orientačních bodů $z(k)$ vzhledem k současné poloze $s(k)$ a poloze orientačních bodů m. Posledním pravděpodobnostním rozložením je pohybový model robota [7]:

$$P(s(k)|s(k-1), u(k)), \tag{6.4}$$

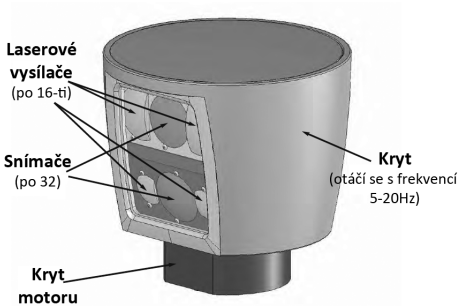
který udává aposteriorní pravděpodobnost polohy $s(k)$ vzhledem k předchozí poloze $s(k-1)$ a řídícímu vstupu $u(k)$. Nejčastěji používaným řešení pravděpodobnostního SLAMu patří extended Kalman filter (EKF). Nejdůležitější alternativou je Rao-Blackwellised particle filter neboli Fast-SLAM [7],[22].

7 NAVRŽENÝ ALGORITMUS

Algoritmus byl navržen a ozkoušen na datech z laserového 3D skeneru HDL-64E S2, které poskytl tým robotiků z ústavu Automatizace. Algoritmus využívá samoorganizační mapu pro extrakci orientačních bodů. Algoritmus se skládá z předzpracování dat z laserového skeneru, extrakci orientačních bodů pomocí samoorganizační mapy a z výpočtu polohy robota. Algoritmus byl naprogramován v jazyce C# a využívá některých upravených knihoven z programu KNN.

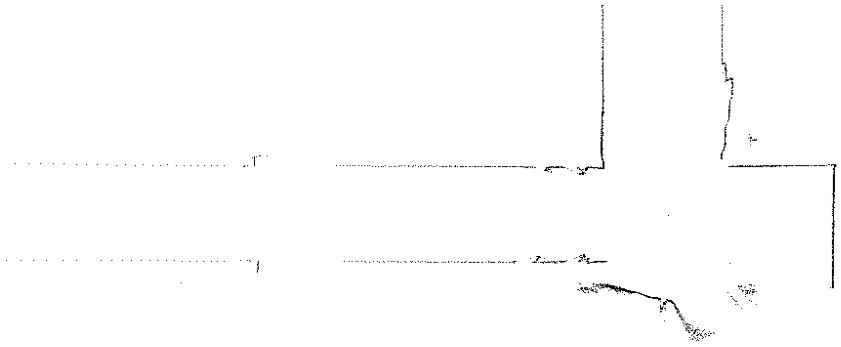
7.1 Předzpracování dat

Použitá data jsou 3D mrak bodů zapsaný v kartézských souřadnicích. Skener HDL-64E S2 snímá své okolí 64 lasery, přičemž každý má jiný sklon vůči vodorovné ose. Celá hlavice s lasery se otáčí dokola a měření probíhá po pevně daných krocích. Zorné pole snímače je 360° v horizontální rovině a 26.8° ve vertikální rovině (od +2° do -24.8°). V jedné otočce každý laser změří 4160 bodů, dohromady je pak 3D mrak tvořen 266240 body. Maximální snímací vzdálenost skeneru je 50-120m v závislosti na odrazivosti plochy. Minimální vzdálenost pro měření je 2.5m. Chyba měření vzdálenosti je menší než 2cm [15].



Obr. 7.1: Laserový 3D skener HDL-64E S2 [15].

Pro účely lokalizace byla ze 3D skenu vyříznuta pouze jedna rovina. Z 3D mraku bodů byly vybrány body odpovídající pouze jednomu laseru. A to laseru s nejmenším sklonem vůči vodorovné ose 0.987°, tento sklon byl zanedbán. Neupravený 2D sken je na obrázku 7.2.



Obr. 7.2: Neupravený sken.

7.1.1 Filtrace

Přesnost měření je velmi dobrá, nicméně vzhledem k hustotě bodů hraje velkou roli. Například body naměřené ve vzdálenosti 3m od skeneru jsou od sebe navzájem vzdáleny pouze 5mm, přitom chyba v měření vzdálenosti je až 2cm. Pro odstranění tohoto problému byli data vyfiltrovány průměrujícím filtrem podle rovnice:

$$i = 1, ..., N$$

$$p'_i = \frac{1}{M} \sum_{j=0}^{M-1} p_{i+j}, \tag{7.1}$$

kde p' je vektor zprůměrovaných bodů, p je vektor starých bodů a M je šířka průměrujícího okna. Každý bod je složen z x-ové a y-ové souřadnice. Vliv použití průměrujícího filtru ilustruje obrázek 7.3.

Obr. 7.3: Použití průměrujícího filtru.

7.1.2 Výběr bodů

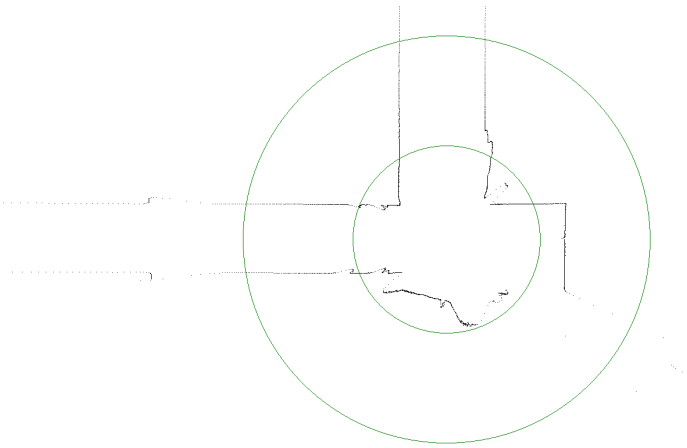
Při zpracování skenu je nutno vzít v potaz minimální vzdálenost d_{min}, pod kterou není schopen skener správně měřit. Ta je 2.5m, všechny body s nižší vzdáleností d jsou ignorovány. Dále je vhodné ignorovat i body příliš vzdálené. Hustota bodů s rostoucí vzdáleností klesá a tím klesá i rozpoznatelnost vzdálených objektů. Vybrána je proto část skenu, ve které je možno považovat hustotu bodů za homogenní. Vhodná maximální vzdálenost bodu je $d_{max} = 5 - 10m$. Rozhodnutí, zda bude bod zahrnut do výběru, se provádí na základě rovnic:

$$i = 1, ..., N$$

$$d_i = \sqrt{x_i^2 + y_i^2}, \tag{7.2}$$

$$d_{min} \leq d_i \leq d_{max}, \tag{7.3}$$

kde N je počet bodů skenu. Na obrázku 7.4 jsou limitní vzdálenosti označeny zelenými kruhy. Z obrázku je zřejmé odstranění chybných měření, která vznikají při malé vzdálenosti překážky.

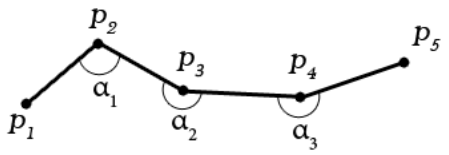


Obr. 7.4: Sken s vyznačenými limitními vzdálenostmi.

7.1.3 Výstupní data

Z takto upraveného skenu jsou vybírány okna o velikosti n bodů (obrázek 7.5), které vstupují do dalších částí algoritmu. Volba velikosti okna n, je velmi podstatná, pro malou velikost okna je algoritmus výpočetně nenáročný, ale kvalita orientačních bodů je malá. Orientační značky složené jen z několika bodů skenu jsou snadno zaměnitelné a jejich znovupozorovatelnost je malá. S rostoucí velikostí okna jsou

získány lepší orientační body, ale výpočetní náročnost roste. Vhodný rozsah parametru n je 5-15. Jednotlivé body v okně jsou popsány vzájemnými úhly, tento popis je invariantní vůči posunutí a natočení.



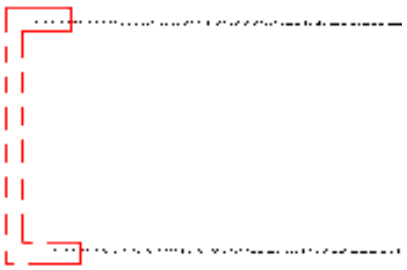
Obr. 7.5: Okno o velikosti n=5.

Vliv různé vzdálenosti mezi body je zanedbán, ve vybrané části skenu ji považujeme za homogenní. Je však nutno odstranit okna, jejichž rozptyl bodů je příliš veliký. Průměrná vzdálenost bodů v okně l_p nesmí přesáhnout daný práh l_{max}.

$$l_p = \frac{1}{n-1} \sum_{j=1}^{n-1} \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} \tag{7.4}$$

$$l_p \leq l_{max} \tag{7.5}$$

Vhodná velikost l_{max} byla stanovena na hodnotu 5 cm. Příkladem okna, které je nutno zamítnout, jsou okna přechodu mezi stěnami v chodbě (viz obrázek 7.6).



Obr. 7.6: Neplatné okno.

7.1.4 Zhrnutí

Algoritmus předzpracování dat se skládá z následujících kroků:

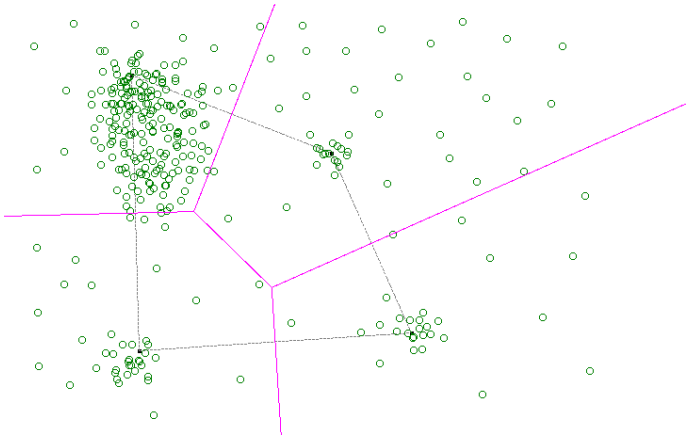
1. Filtrování bodů průměrujícím filtrem dle rovnice 7.1.
2. Odstranění bodů jejichž vzdálenost je menší než d_{min} nebo větší než d_{max}, dle rovnice 7.2.

3. Pro každý bod je vytvořeno okno o délce n.
4. Odstranění oken které nesplňují podmínky rovnice 7.5.

Výsledkem předzpracování dat je tedy N oken, o délce n bodů.

7.2 Extrakce orientačních bodů

Základní myšlenkou pro využití samoorganizační mapy pro extrakci orientačních bodů je ta, že za orientační bod můžeme považovat cokoliv, co je v daném prostředí dostatečně výjimečné. Je využita vlastnost Kohonenovy samoorganizační mapy pokrytí vstupního prostoru, kde nezávisí na počtu vzorů v jednotlivých shlucích, přitažený neuron je vždy jen jeden (viz obr 7.7). Shluk, ve kterém je malý počet vzorů, může být použit jako klasifikátor orientačních bodů.



Obr. 7.7: Pokrytí vstupního prostoru samoorganizační mapou.

Do samoorganizační mapy vstupují zpracované data ve formě oken o velikosti n bodů (kapitola 7.1). Mapa má na vstupu $n-2$ neuronů. Počet neuronů ve výstupní vrstvě udává počet shluků, na které bude rozdělen vstupní prostor. Protože však příliš jemné dělení vstupního prostoru není žádoucí, je vhodnou velikostí výstupní vrstvy například 4*4 neurony. Topologie výstupní vrstvy mapy byla zvolena čtvercová.

7.2.1 Naučení samoorganizační mapy

Algoritmus učení samoorganizační mapy byl probrán podrobně v kapitole 3.3. Trénovacími daty samoorganizační mapy jsou vzorové skeny prostředí, v němž očekáváme, že se bude robot pohybovat. Správné naučení sítě závisí na správném nastavení všech parametrů neuronové sítě. Byla použita ostrá funkce sousedství $\Theta(j)$ s lineárně se snižujícím poloměrem ρ v čase, viz rovnice 5.1. Vhodné hodnoty parametrů pro

mapu 4*4 jsou $\rho_0 = 2$, j=2-5. Pro malé mapy je takováto funkce okolí zcela dostačující. Funkce parametru učení μ je exponenciální, viz rovnice 5.2. Vhodné hodnoty parametrů jsou $\mu_0 = 0.8$,$k = 0.1$. Ukončovací podmínkou učení samoorganizační mapy je počet iterací N, pro uvedené hodnoty parametrů je vhodnou hodnotou $N = 15$.

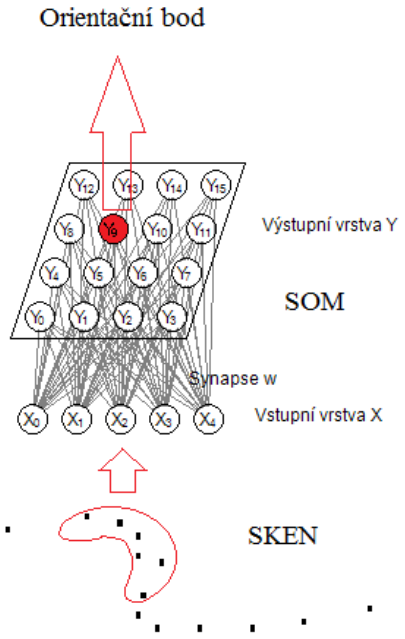
Po naučení sítě je vytvořena aktivační mapa. Algoritmus aktivace sítě je popsán v kapitole 3.4, aktivační mapa je popsána v kapitole 3.5.2. Data pro vytvoření aktivační mapy jsou stejná jako trénovací. V závislosti na četnosti aktivaci neuronů je vybrán neuron pro extrakci orientačních bodů. Neuron musí být aktivován alespoň třikrát pro každý vzorový sken, z neuronů jenž splňují tuto podmínku je pak vybrán neuron, jenž byl aktivován celkem nejméněkrát.

Algoritmus učení samoorganizační mapy:

1. Naučení sítě trénovacími daty.
2. Vytvoření aktivační mapy.
3. Výběr neuronu pro extrakci orientačních bodů podle počtu aktivací.

7.2.2 Aktivní režim

Orientační body jsou získány z dat pomocí aktivace samoorganizační mapy. Pokud je aktivován neuron, který byl vybrán pro extrakci, je předložené okno označeno za orientační bod (7.8).



Obr. 7.8: Princip extrakce orientačních bodů.

Orientační bod je popsán novým oknem o velikosti $3 * n$, počáteční bod nového okna je posunut o n bodů před počátečním bodem předloženého okna(obr. 7.9). Jinak řečeno, nové okno obsahuje n bodů před a za starým oknem. To je nutno provést kvůli odlišení orientačních bodů, protože všechny okna, která aktivují stejný neuron, jsou si velmi podobná.

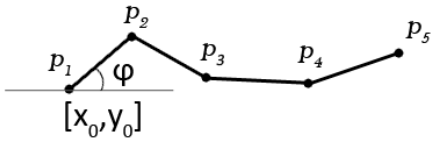
Obr. 7.9: Původní (červené) a rozšířené (zelené) okno.

Nově extrahovanému orientačnímu bodu je vypočtena průměrná relativní chyba δ vůči všem orientačním bodům dle rovnice:

$$j = 1, ..., M$$

$$\delta(j) = \frac{1}{3n} \sum_{i=1}^{3n} \left(\frac{|x'_i - x_{ij}|}{x_{ij}} + \frac{|y'_i - y_{ij}|}{y_{ij}} \right), \tag{7.6}$$

kde M je počet orientačních bodů v databázi, x' a y' jsou souřadnice extrahovaného orientačního bodu a x a y jsou souřadnice uložených orientačních bodů. Následně je vyhledána minimální chyba, neboli je zjištěno, kterému z uložených orientačních bodů je extrahovaný orientační bod nejpodobnější. Pokud je tato minimální chyba menší než práh $\delta(j)_{min} < p$, je orientační bod považován za shodný s již pozorovaným vzorem. V tom případě jsou souřadnice a natočení tohoto orientačního bodu předány algoritmu pro výpočet polohy robota. Pokud je chyba větší než práh, je orientační bod považován za nový a je uložen do databáze. Do databáze je uloženo okno bodů, souřadnice prvního bodu x_0, y_0 a úhel φ přímky mezi prvním a druhým bodem okna vůči ose x v souřadném systému robota (obr. 7.10). Vhodná hodnota prahu p se pohybuje v rozsahu 0.02-0.07 v závislosti na kvalitě měření.



Obr. 7.10: Orientační bod, s vyznačenými počátečními souřadnicemi a natočením.

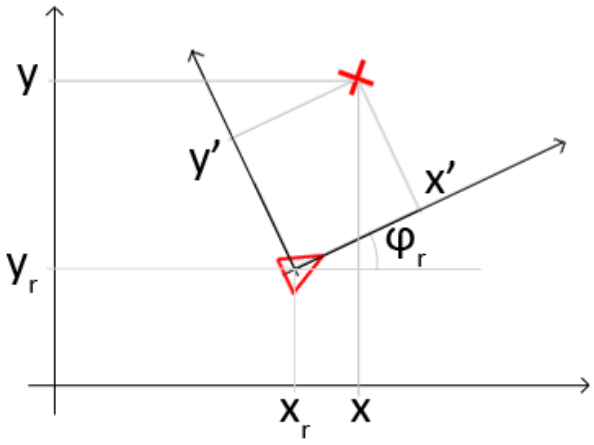
Transformace do hlavního souřadného systému je provedena dle rovnic:

$$x = x'cos(\varphi_r) - y'sin(\varphi_r) + x_r, \tag{7.7}$$

$$y = x'sin(\varphi_r) + y'cos(\varphi_r) + y_r, \tag{7.8}$$

$$\varphi = \varphi' + \varphi_r, \tag{7.9}$$

kde x_r, y_r, φ_r je momentální poloha a natočení robota v hlavním souřadném systému, x', y', φ' jsou souřadnice v souřadném systému s počátkem ve snímači robota, x, y, φ jsou souřadnice transformované do hlavního souřadného systému. Problematiku souřadných systémů zobrazuje obrázek 7.11.



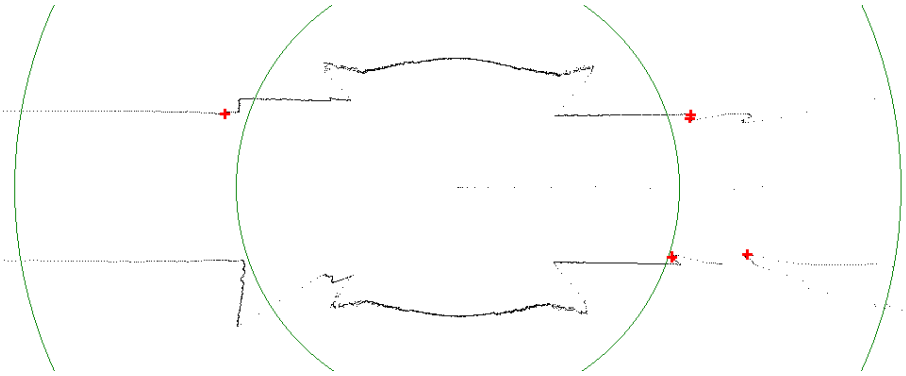
Obr. 7.11: Rozdílné souřadné systémy, červený trojúhelník značí robota.

Celý algoritmus extrakce orientačních bodů se dá shrnout do těchto bodů:

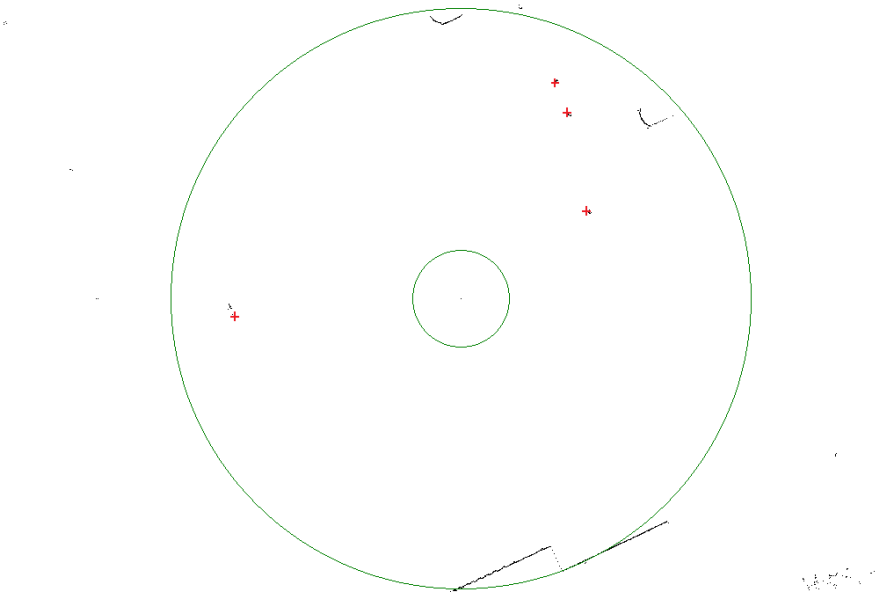
1. Výběr a předložení aktivačního vzoru.
2. Pokud je aktivován neuron pro extrakci orientačních bodů, pokračuj krokem 3, jinak zpět na krok 1.
3. Rozšíření okna na $3 * n$ bodů a posunutí počátečního bodu o $-n$ bodů.
4. Vypočtení chyby vůči orientačním bodům v databázi dle rovnice 7.6.
5. Nalezení nejmenší chyby $\delta(j)_{min}$.
6. Srovnání této chyby s prahem $\delta(j)_{min} < p$.
   (a) Chyba nižší než práh: už pozorovaný orientační bod, informace o jeho souřadnicích a natočení jsou předány algoritmu pro výpočet polohy.
   (b) Chyba vyšší než práh: nový orientační bod, uložení do databáze.

7.2.3 Zhrnutí

Algoritmus pro extrakci probíhá ve dvou fázích, nejdříve je nutno samoorganizační mapu naučit. Ve druhé fázi pak již získáváme samotné orientační body. V závislosti na použitých trénovacích datech je algoritmus schopný fungovat v různých prostředích. Ze vlastností samoorganizační mapy vyplývají i vlastnosti algoritmu extrakce orientačních bodů, algoritmus je výpočetně jednoduchý, a při správném nastavení učení deterministický. Na obrázku 7.12 jsou extrahované orientační body v interiéru budovy, lze vidět, že jako orientační body byly vybrány obzvláště ostré hrany. Na obrázku 7.13 jsou extrahované orientační body ve venkovním prostředí. Za orientační body v tomto prostředí byli vybrány sloupky a kmeny stromů.



Obr. 7.12: Extrahované orientační body ze skenu interiéru.



Obr. 7.13: Extrahované orientační body z venkovního skenu.

7.3 Výpočet polohy

Rovnice pro výpočet polohy vycházejí z teorie uvedené v kapitole 6.3.

Prvotní odhad polohy robota x v čase k je dán hodnotou získanou z odometrie robota, vychází z pohybového modelu 6.4. Který lze zapsat jako:

$$s(k) = f(s(k-1), u(k)) + v(k), \tag{7.10}$$

kde funkce f modeluje robotovy kinetické vlastnosti, a $v(k)$ je porucha. Při znalosti dat z odometrie můžeme za $f(s(k-1), u(k))$ dosadit přímo výstup odometrie:

$$s'(k) = s(k-1) + \Delta(k), \tag{7.11}$$

kde $s'(k)$ je odhad aktuální polohy robota pomocí odometrie, $s(k-1)$ je poloha robota v čase k-1, a $\Delta(k)$ je změna polohy změřená odometrií. Váha přiřazená tomuto odhadu je úměrná přesnosti odometrie.

Odhad polohy v závislosti na pozorování orientačních bodů je dán rovnicemi:

$$\varphi_{ri} = \varphi^* - \varphi, \tag{7.12}$$

$$x_{ri} = x^* - (xcos(\varphi_{ri}) - ysin(\varphi_{ri})), \tag{7.13}$$

$$y_{ri} = y^* - (xsin(\varphi_{ri}) + ycos(\varphi_{ri})), \tag{7.14}$$

$$s_i = \begin{pmatrix} x_{ri} \\ y_{ri} \\ \varphi_r i \end{pmatrix}, \tag{7.15}$$

kde x, y, φ jsou souřadnice a natočení orientačního bodu naměřené při pozorování, x^*, y^*, φ^* jsou souřadnice a natočení orientačního bodu uložené v databázi, $x_{ri}, y_{ri}, \varphi_{ri}$ jsou odhadnuté souřadnice robota.

Váha pozorování je stanovena v závislosti na poloze robota. Vycházíme z rovnice:

$$w_{s_i(k)} = h(s'(k), s_i(k)), \tag{7.16}$$

kde $w_{s_i(k)}$ je váha pozorování, h je funkce hodnotící pravděpodobnost pozorování. Tato funkce odstraňuje nereálná pozorování, vzdálenost mezi robotem a orientačním bodem musí být v daném rozsahu. Pokud není jedná se o špatně přiřazený orientační bod.

$$h(s', s_i) = \begin{cases} 1 & : \quad d_{min} \leq d(s', s_i) \leq d_{max} \\ 0 & : \quad d(s', s_i) < d_{min} \vee d(s', s_i) > d_{max} \end{cases} \tag{7.17}$$

Výsledná poloha je určena váženým průměrem všech odhadů:

$$s(k) = \frac{w_{s'}(k)s'(k) + \sum_{i=0}^{M} s_i(k)w_{s_i}(k)}{w_{s'}(k) + \sum_{i=0}^{M} w_{s_i}(k)}, \tag{7.18}$$

kde M je počet pozorování.

7.4 Výsledky

Algoritmus byl otestován na datech poskytnutý týmem robotiků z ústavu Automatizace. Celkem bylo k dispozici 35 skenů z interiéru budovy a 74 venkovních skenů. Venkovní skeny nebyli vhodné pro sebelokalizaci v dvojrozměrném prostoru kvůli způsobu, jakým byli sejmuty. Skener byl v úrovni 1.8m nad terénem, při výběru vodorovného paprsku bylo v jeho dráze jen minimum překážek a to ještě velmi vzdálených. Vybrané venkovní skeny byli použity pro vyzkoušení extrakce orientačních bodů, viz obrázek 7.13.

Algoritmus sebelokalizace byl tedy ozkoušen jen na skenech interiéru.

7.4.1 Znovu pozorování místa

V obdržených datech bylo několik skenů stejného místa ze stejné pozice. Na těchto skenech byla ověřována znovu rozpoznatelnost orientačních bodů. 3D mrak bodů je vždy trochu jiný, i když snímáme stejné místo. Proto je důležité stanovit, s jakou odchylkou je algoritmus schopný vypočíst polohu stejného místa s jiným skenem.

Zde byla odhalena největší nevýhoda metody a to zaměnitelnost orientačních bodů. Vybraný popis pomocí oken z jednotlivých bodů se ukázal jako nevhodný. Okolí jednotlivých orientačních bodů je si velmi podobné, protože se obvykle jedná o rovné zdi. Aby bylo obsaženo širší okolí, které může obsahovat další charakteristické prvky prostředí, je nutno použít širší okno, tím zároveň ale roste výpočetní náročnost. Nicméně i při použití velkého okna byla míra zaměněných orientačních bodů vysoká. V tabulce 7.1 je znázorněna závislost chyby na velikosti okna. Použita byla síť 4*4, $\rho_0 = 2$, krok okolí $j = 5$, $\mu_0 = 0.8$, koeficient exponenciály parametru učení $k = 0.1$, počet iterací N=15.

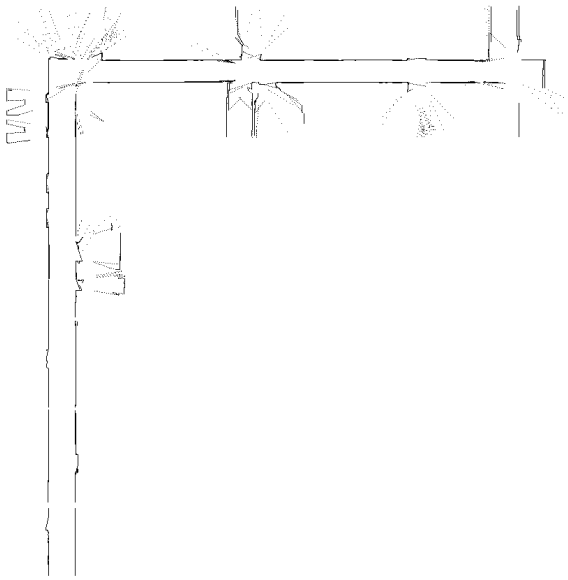
velikost okna n	chyba Δ [m]
5	1,28
10	1,02
15	0,92
30	0,44

Tab. 7.1: Tabulka chyby polohy v závislosti na velikosti okna n

7.4.2 Průběžná lokalizace

Velkým problémem pro testovaní průběžné lokalizace byly příliš vzdálené skeny. Pro dobré ověření algoritmu by bylo třeba skenů s podstatně menším posunem mezi jednotlivými měřeními. Což ústilo v neschopnost algoritmu najít při větších posunech

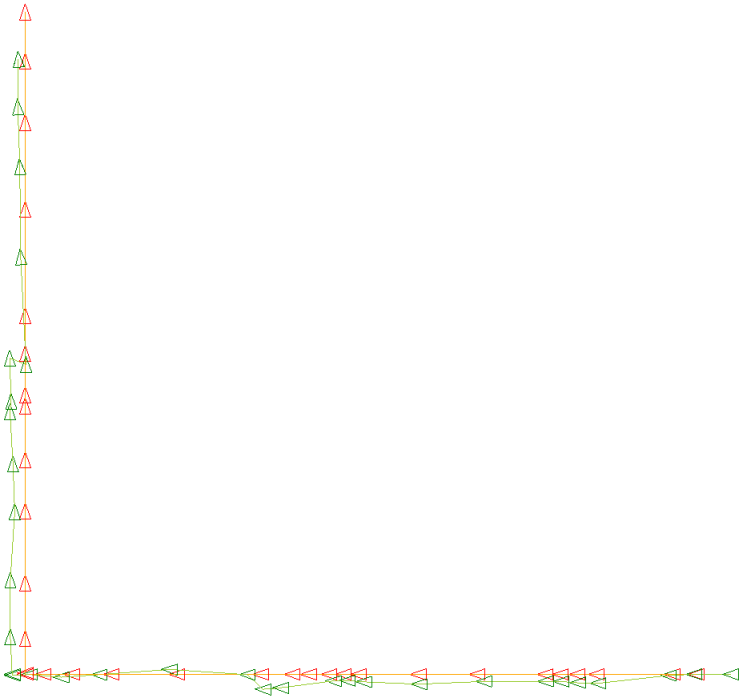
skenu společné orientační body. Všechny skeny poskládané do sebe jsou zobrazeny na obrázku 7.14. Vzhledem ke způsobu jakým byli skeny změřeny, nebyli k dispozici údaje z odometrie. Údaje z odometrie byli proto vytvořeny uměle, zatížené chybou 10%.



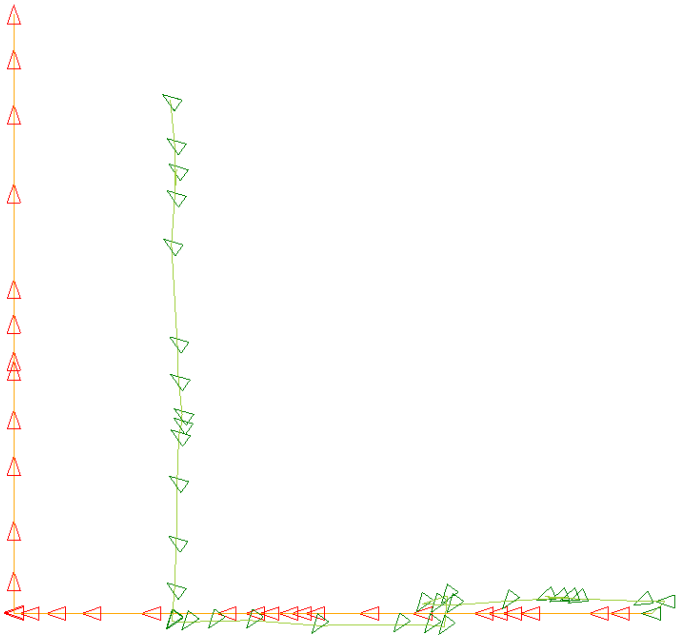
Obr. 7.14: Veškeré skeny.

Samotný algoritmus byl nespolehlivý, výsledky se diametrálně lišily v závislosti na nastavení algoritmu a výběru trénovacích dat. Pro srovnání rozdílných výsledků algoritmu jsou uvedeny dva různé pokusy. Oba použily stejně nastavený algoritmus, ale každé byly naučeny na jiná vzorová trénovací data. Koeficienty algoritmu: velikost okna n=15, síť 4*4 neurony, $\rho_0 = 2$, krok okolí $j = 5$, $\mu_0 = 0.8$, koeficient exponenciály parametru učení $k = 0.1$, počet iterací N=15.

První pokus je zobrazen na obrázku 7.15, přesné hodnoty vypočtené a skutečné polohy robota jsou uvedeny v tabulce 7.2. První pokus dosáhl konečné chyby 2.38 m. Druhý pokus je zobrazen na obrázku 7.16 a podrobně popsán v tabulce 7.3. Chyba v posledním kroku lokalizace dosáhla 9.86m, tedy více než čtyřnásobku pokusu jedna. Z obrázku 7.16 jsou zřejmé místa, kde došlo k zaměnění orientačních bodů a tím významnému zvýšení chyby lokalizace.



Obr. 7.15: Sebelokalizace pokus 1, červeně je vyznačena skutečná poloha, zeleně poloha vypočtená.



Obr. 7.16: Sebelokalizace pokus 2, červeně je vyznačena skutečná poloha, zeleně poloha vypočtená.

reálné hodnoty			vypočtené			chyba
x [m]	y [m]	φ	x [m]	y [m]	φ	Δ [m]
36,28	0,00	180,00	36,28	0,00	180,00	0,00
34,55	0,00	180,00	34,45	0,00	179,29	0,09
33,35	0,00	180,00	33,17	-0,05	179,61	0,19
29,56	0,00	180,00	29,66	-0,44	179,72	0,45
28,60	0,00	180,00	28,64	-0,42	179,78	0,42
27,78	0,00	180,00	27,81	-0,36	180,62	0,36
27,02	0,00	180,00	27,03	-0,35	179,88	0,35
23,61	0,00	180,00	23,95	-0,35	179,76	0,49
20,66	0,00	180,00	20,71	-0,49	178,97	0,49
17,68	0,00	180,00	17,95	-0,37	178,86	0,46
16,92	0,00	180,00	17,13	-0,31	178,19	0,37
16,21	0,00	180,00	16,44	-0,33	178,62	0,40
15,18	0,00	180,00	13,79	-0,67	178,05	1,54
14,36	0,00	180,00	12,92	-0,75	178,86	1,62
12,79	0,00	180,00	12,13	-0,02	178,30	0,65
8,61	0,00	180,00	8,24	0,24	178,83	0,44
5,32	0,00	180,00	4,72	-0,02	179,60	0,61
3,35	0,00	180,00	2,83	-0,14	180,43	0,54
1,91	0,00	180,00	1,26	-0,01	179,95	0,65
1,05	0,00	180,00	0,40	-0,06	179,42	0,65
1,04	0,06	90,00	0,39	0,00	89,57	0,65
1,04	1,78	90,00	0,29	1,88	90,35	0,76
1,04	4,57	90,00	0,30	4,71	89,89	0,76
1,04	8,16	90,00	0,52	8,13	90,05	0,52
1,04	10,72	90,00	0,43	10,50	90,10	0,65
1,04	13,42	90,00	0,28	13,13	91,09	0,81
1,04	13,97	90,00	0,33	13,64	91,00	0,78
1,04	16,02	90,00	0,27	15,80	91,18	0,81
1,04	17,93	90,00	1,08	15,49	94,16	2,44
1,04	23,23	90,00	0,81	20,86	93,97	2,37
1,04	27,57	90,00	0,77	25,36	93,09	2,23
1,04	27,57	90,00	0,77	25,36	93,66	2,23
1,04	30,63	90,00	0,67	28,35	93,76	2,31
1,04	33,10	90,00	0,70	30,74	93,21	2,38
1,04	33,10	90,00	0,70	30,74	93,88	2,38

Tab. 7.2: Sebelokalizace pokus 1.

reálné hodnoty			vypočtené			chyba
x [m]	y [m]	φ	x [m]	y [m]	φ	Δ [m]
36,28	0,00	180,00	36,28	0,00	180,00	0,00
34,55	0,00	180,00	37,04	0,66	205,17	2,58
33,35	0,00	180,00	35,76	0,68	204,91	2,51
29,56	0,00	180,00	32,16	0,81	203,99	2,72
28,60	0,00	180,00	31,13	0,87	203,20	2,68
27,78	0,00	180,00	30,39	0,93	202,22	2,77
27,02	0,00	180,00	31,70	0,88	233,01	4,76
23,61	0,00	180,00	28,38	0,69	233,54	4,82
20,66	0,00	180,00	25,30	0,46	233,60	4,66
17,68	0,00	180,00	24,43	0,49	234,20	6,77
16,92	0,00	180,00	23,61	0,53	233,93	6,71
16,21	0,00	180,00	24,99	0,97	234,30	8,84
15,18	0,00	180,00	24,84	-0,73	233,99	9,69
14,36	0,00	180,00	24,05	-0,65	233,09	9,71
12,79	0,00	180,00	22,35	-0,61	233,98	9,58
8,61	0,00	180,00	17,83	-0,67	234,03	9,24
5,32	0,00	180,00	14,22	-0,39	233,36	8,91
3,35	0,00	180,00	12,11	-0,40	232,70	8,77
1,91	0,00	180,00	10,66	-0,48	233,14	8,75
1,05	0,00	180,00	9,80	-0,44	233,66	8,77
1,04	0,06	90,00	9,79	-0,39	143,78	8,76
1,04	1,78	90,00	9,92	1,39	144,62	8,89
1,04	4,57	90,00	10,02	3,98	144,33	9,00
1,04	8,16	90,00	10,06	7,29	144,25	9,06
1,04	10,72	90,00	10,16	9,84	144,52	9,16
1,04	13,42	90,00	10,31	10,49	145,10	9,72
1,04	13,97	90,00	10,34	11,01	145,46	9,76
1,04	16,02	90,00	10,14	12,91	144,65	9,62
1,04	17,93	90,00	10,07	14,99	144,19	9,50
1,04	23,23	90,00	9,75	20,38	143,66	9,17
1,04	27,57	90,00	10,04	24,53	144,04	9,50
1,04	27,57	90,00	9,93	23,07	144,50	9,97
1,04	30,63	90,00	9,93	25,95	145,44	10,04
1,04	33,10	90,00	9,69	28,38	145,02	9,86
1,04	33,10	90,00	9,69	28,38	145,61	9,86

Tab. 7.3: Sebelokalizace pokus 2.

8 ZÁVĚR

Byl vytvořen program pro výuku studentů v předmětu Umělá inteligence, program byl použit při výuce a neobjevily se žádné vážné nedostatky, program byl stabilní a studentům nabízel všechny potřebné funkce pro názorné předvedení samoorganizačních map. Program byl doplněn o modul citlivostní analýzy, jenž slouží pro podrobné prozkoumání vlivu parametrů mapy a učení na výsledku učení. Program byl napsán v jazyce C#.

V druhé části práce byl navržen algoritmus pro sebelokalizaci robota. Cílem bylo prozkoumat možnosti využití samoorganizační mapy pro sebelokalizaci robota.

Data pro algoritmus byla dodána robotiky z Ústavu automatizace. Data byla sejmuta pomocí 3D skeneru HDL-64E S2. Pro problematiku sebelokalizace byla z 3D skenů vyříznuta pouze jedna 2D rovina. Data byla vyfiltrována, popsána tak, aby byla invariantní vůči posunu a natočení, a předána algoritmu pro extrakci orientačních bodů.

Extrakce orientačních bodů byla realizována pomocí samoorganizační mapy, základní myšlenka byla ta, že za organizační bod je možno považovat cokoliv co je dostatečně výjimečné v prostředí. Principem bylo vytvoření shluků pomocí samoorganizační mapy (každý shluk zastupuje jeden neuron výstupní vrstvy mapy), a vybrání shluku, pod nějž spadající data byla označena za orientační body. Výhodou algoritmu byla extrakce orientačních bodů bez apriorní znalosti tvaru orientačních značek. Algoritmus je možno naučit na různá data a tak je schopen pracovat v různých prostředích. Metoda extrakce orientačních bodů pomocí samoorganizační mapy byla také prezentována na soutěži EEICT.

Extrahované orientační značky byly použity pro sebelokalizaci robota. Kvalita orientačních bodů byla otestována jednak při znovu pozorování stejného místa a jednak pro průběžnou sebelokalizaci. Byl diskutován vliv velikosti okna na přesnosti lokalizace a vliv rozdílných trénovacích dat při souběžné lokalizaci.

Celkově byl algoritmus příliš závislý na výběru trénovacích dat a parametrech algoritmu, a proto nepříliš vhodný pro reálné použití.

LITERATURA

[1] Aguilera, P.; Frenich, A.; Torres, J.; aj.: Application of the kohonen neural network in coastal water management: methodological development for the assessment and prediction of water quality. *Water Research*, ročník 35, č. 17, 2001: s. 4053 – 4062.

[2] Alahakoon, D.; Halgamuge, S.; Srinivasan, B.: Dynamic self-organizing maps with controlled growth for knowledge discovery. *Neural Networks, IEEE Transactions on*, ročník 11, č. 3, may 2000: s. 601 –614.

[3] Burian, F.; Zalud, L.; Florian, T.; aj.: Unified Storage for Laser Scanner Data. 2012.

[4] Chesnut, C.: Self Organizing Map AI for Pictures. [online], 2004.
URL `http://www.generation5.org/content/2004/aiSomPic.asp`

[5] Chi, D.: Self-Organizing Map-Based Color Image Segmentation with k-Means Clustering and Saliency Map. *ISRN Signal Processing*, ročník 2011, 2011.

[6] Dekker, A. H.: Optimal Colour Quantization using Kohonen Neural Networks. In *Network: Computation in Neural Systems*, 1993.

[7] Durrant-Whyte, H.; Bailey, T.: Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms. *IEEE ROBOTICS AND AUTOMATION MAGAZINE*, ročník 2, 2006: str. 2006.

[8] Fleischer, J.; Marsland, S.: Learning to autonomously select landmarks for navigation and communication. In *Proceedings of the seventh international conference on simulation of adaptive behavior on From animals to animats*, ICSAB, Cambridge, MA, USA: MIT Press, 2002, ISBN 0-262-58217-1, s. 151–160.

[9] Graupe, D.: Large Memory Storage and Retrieval (LAMSTAR) network. [online], 1999.
URL `http://www.freepatentsonline.com/5920852.pdf`

[10] Holmström, J.: *Growing Neural Gas Experiments with GNG, GNG with Utility and Supervised GNG*. Diplomová práce, Uppsala University, 2006.

[11] Kaski, S.: World Poverty Map. [online], 1997.
URL `http://www.cis.hut.fi/research/som-research/worldmap.html`

[12] Kohonen, T.: WEBSOM - Self-Organizing Maps for Internet Exploration. [online], 1999.
URL `http://websom.hut.fi/websom/`

[13] Kohonen, T.; Oja, E.; Simula, O.; aj.: Engineering applications of the self-organizing map. *Proceedings of the IEEE*, ročník 84, č. 10, oct 1996: s. 1358 –1384.

[14] Kolektiv autorů: Growing self-organizing map. [online], 2011.
URL `http://en.wikipedia.org/wiki/Growing_self-organizing_map`

[15] Kolektiv autorů: HDL-64E S2 and S2.1. [online], 2011.
URL `http://www.velodynelidar.com/lidar/products/manual/63-HDL64E%20S2%20Manual_Rev%20D_2011_web.pdf`

[16] Kolektiv autorů: LIDAR. [online], 2011.
URL `http://en.wikipedia.org/wiki/LIDAR`

[17] Kolektiv autorů: Self-organizing map. [online], 2011.
URL `http://en.wikipedia.org/wiki/Self-organizing_map`

[18] Martinetz, T.; Berkovich, S.; Schulten, K.: 'Neural-gas' network for vector quantization and its application to time-series prediction. *Neural Networks, IEEE Transactions on*, ročník 4, č. 4, jul 1993: s. 558 –569.

[19] McKinstry, J.: Kohonen network. [online], 2007.
URL `http://www.scholarpedia.org/article/Self-organizing_map`

[20] Quisquater, J.-J.; Samyde, D.: Automatic code recognition for smart cards using a Kohonen neural network. In *Proceedings of the 5th conference on Smart Card Research and Advanced Application Conference - Volume 5*, CARDIS'02, 2002, s. 6–6.

[21] Reyes-Aldasoro, C. C.: Image Segmentation with Kohonen Neural Network Self-Organisning Maps. [online], 2004.
URL `http://www.cs.jhu.edu/~cis/cista/446/papers/SegmentationWithSOM.pdf`

[22] Riisgaard, S.; Blas, M. R.: SLAM for Dummies. [online], 2004.
URL `http://ocw.mit.edu/courses/aeronautics-and-astronautics/16-412j-cognitive-robotics-spring-2005/projects/1aslam_blas_repo.pdf`

[23] Sabisch, T.; Ferguson, A.; Bolouri, H.: Automatic Landmark Extraction using Self-Organising Maps. In *IEE/BMVA Proceedings of the 1st annual conference on medical image understanding and analysis (MIUA'97)*, 1997, s. 157–160.

[24] Shah-Hosseini, H.; Safabakhsh, R.: The Time Adaptive Self-Organizing Map with neighbrohood functions for bilevel tresholding. [online], 2000.
URL http://www.acims.arizona.edu/CONFERENCES/ais2000/Papers/ConfPap/a039shah_hosseinih.pdf

[25] Shah-Hosseini, H.; Safabakhsh, R.: A TASOM-based algorithm for active contour modeling. *Pattern Recognition Letters*, ročník 24, č. 9–10, 2003: s. 1361 – 1373.

[26] Su, M.-C.: Quadratic Neurons. [online].
URL http://cilab.csie.ncu.edu.tw/course/cluster/Quadratic%20Neurons.pdf

[27] Takatsuka, M.: An application of the Self-Organizing Map and interactive 3-D visualization to geospatial data", The. In *Proceedings of the 6 th International Conference on GeoComputation*, 2001.

[28] Trosset, M. W.: Representing Clusters: K-Means Clustering, Self-Organizing Maps, and Multidimensional Scaling. [online], 2008.
URL http://www.stat.indiana.edu/files/TR/TR-08-03.pdf

[29] Ultsch, A.: U*-matrix: a tool to visualize clusters in high dimensional data. Technická Zpráva 36, Philipps-University Marburg, Germany, 2003.
URL http://www.mathematik.uni-marburg.de/~databionics/en/downloads/papers/ultsch03ustar.pdf

[30] Vicente, L.; Vellido, A.: Review of Hierarchical Models for Data Clustering and Visualization. [online], 2004.
URL http://www.lsi.us.es/redmidas/Capitulos/LMD20.pdf

[31] Volná, E.: *Neuronové sítě 1.* Ostravská univerzita v Ostravě, 2008.

[32] Yao, K.; Mignotte, M.; Collet, C.; aj.: Unsupervised segmentation using a self-organizing map and a noise model estimation in sonar imagery. [online], 1999.
URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.38.5285&rep=rep1&type=pdf

[33] Yuriy, C.: Kohonen's Self Organizing Maps in C++ with Application in Computer Vision Area. [online], 2007.
URL http://www.codeproject.com/KB/graphics/som.aspx

[34] Zhang, B.; Fu, M.; Yan, H.; aj.: Handwritten Digit Recognition by Adaptive-Subspace Self-Organizing Map (ASSOM). *IEEE Trans. on Neural Networks*, ročník 10, 1999: s. 10–939.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

SOM Self-Organizing map, samoorganizační mapa

KNN Kohonenova neuronová síť

LVQ Learning Vector Quantization

ASSOM Adaptive-Subspace Self-Organizing Map

GSOM Growing Self-Organizing Map

TASOM Time Adaptive Self-Organizing Map

HSOM Hierarchical Self-Organizing Map

LAMSTAR Large Memory Storage and Retrieval neural network

U-Matrix unified distance matrix

ρ poloměr okolí

J okolí

j neuron

$j*$ vítězný neuron

x učící vzor

w váha synapsí

μ parametr učení

SLAM simultaneous localization and mapping

LIDAR Light Detection And Ranging

PLS proximity laser scaner

RANSAC random sampling consensus

EKF extended Kalman filter

s stavový vektor polohy robota

u vektor řízení

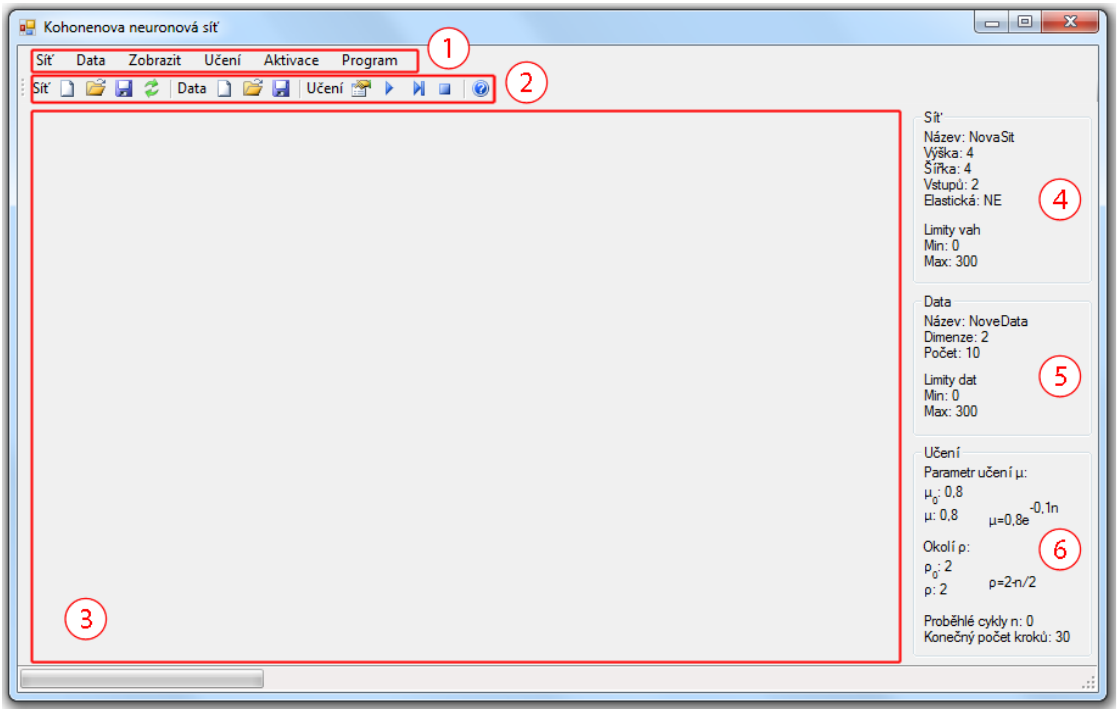
m množina orientačních bodů

z	vektor pozorování
δ	relativní chyba
Δ	absolutní chyba

SEZNAM PŘÍLOH

A POPIS PROGRAMU KOHONENOVA NEURONOVÁ SÍŤ

A.1 Vizuální prostředí programu



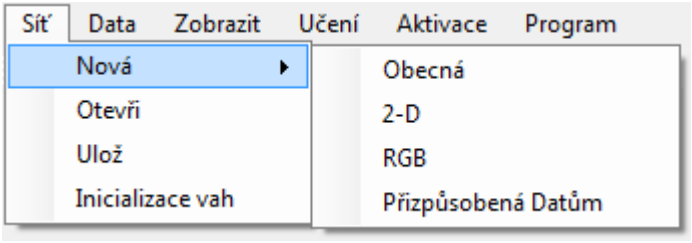
Obr. A.1: Grafické prostředí programu.

1. Základní menu programu: obsahuje jednotlivé funkce programu.
2. Panel rychlého přístupu: obsahuje nejzákladnější funkce.
3. Zobrazovací plocha: zobrazuje síť a data zvoleným způsobem.
4. Informace o síti: zobrazuje informace o vytvořené síti.
5. Informace o trénovacích datech: zobrazuje informace o načtených datech.
6. Informace o nastavení učení: zobrazuje nastavení adaptace.

A.1.1 Základní menu programu

Položka Síť

Položka Síť slouží k vytvoření, načtení, ukládání a inicializaci neuronové sítě.

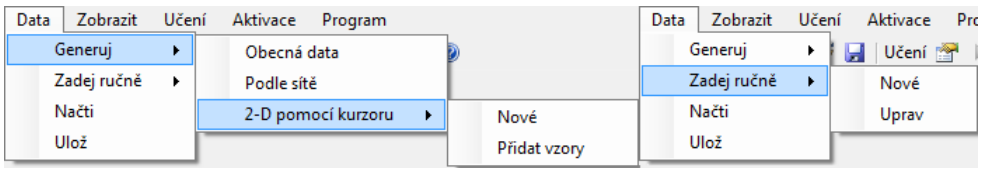


Obr. A.2: Rozbalené menu Síť.

- Nová:
  - Obecná: Otevře okno Vytvoření nové sítě bez jakýchkoliv omezení, více viz podkapitola A.1.3.
  - 2-D: Otevře okno Vytvoření nové sítě, počet vstupů je fixně nastaven na 2.
  - RGB: Otevře okno Vytvoření nové sítě, počet vstupů je fixně nastaven na 3, minimum na 0 a maximum na 255.
  - Přizpůsobená Datům: Tato volba se otevře pouze pokud jsou načteny nějaké trénovací data. Otevře okno Vytvoření nové sítě, počet vstupů a limity sítě jsou fixně nastavené podle trénovacích dat.
- Otevři: Otevře okno pro načtení sítě ze souboru.
- Ulož: Zpřístupní se pokud je vytvořena nebo načtena síť. Otevře okno pro uložení sítě do souboru.
- Inicializace vah: Náhodně nastaví váhy neuronů v rozmezí minima a maxima sítě.

Položka Data

Položka Data slouží ke generaci, načítání a ukládání trénovacích dat.



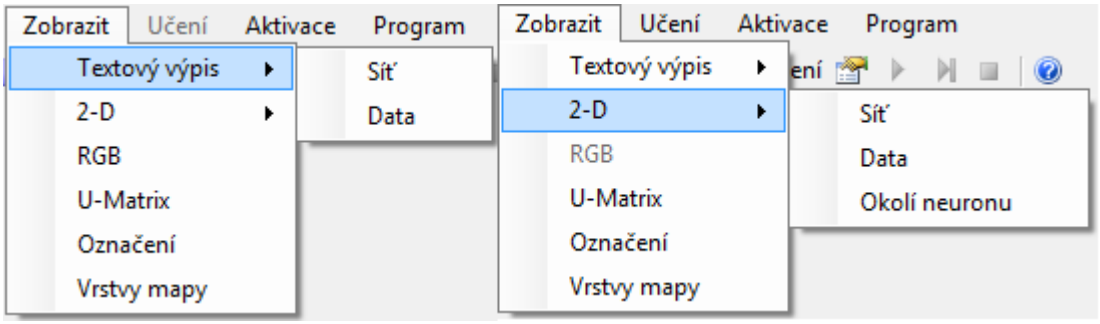
Obr. A.3: Rozbalené menu Data.

- Generuj:
  - Obecná data: Otevře okno Generování Dat bez jakýchkoliv omezení, více viz podkapitola A.1.4.
  - Podle sítě: Zpřístupní se po vytvoření sítě. Otevře okno Generování Dat, dimenze a limity dat jsou nastaveny fixně podle sítě.

  - 2-D pomocí kurzoru:
    * Nové: Začne generování 2D dat (stávající data jsou smazány), klikáním na zobrazovací plochu se přidávají 2D trénovací data odpovídající místu kliknutí. Data jsou průběžně zobrazována. Generování se ukončí stiskem klávesy Esc.
    * Přidat vzory: Stejné jako možnost Nové, s tím že stávající 2D data jsou ponechána a generované vzory se k nim přidávají. Přístupné pouze pokud jsou současná data dvojrozměrná.
- Zadej ručně:
  - Nové: Otevře okno pro zadání dat ručně, stávající data jsou smazána.
  - Uprav: Otevře okno pro upravení stávajících dat.
- Načti: Otevře okno pro načtení trénovacích dat ze souboru.
- Ulož: Otevře okno pro uložení trénovacích dat do souboru.

Položka Zobrazit

Položka Zobrazit slouží k výběru typu zobrazení neuronové sítě a trénovacích dat na zobrazovací ploše. Je možno mít aktivní jen jeden režim zobrazování.



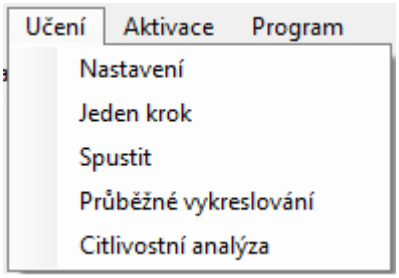
Obr. A.4: Rozbalené menu zobrazit.

- Textový výpis: Aby byli položky v tomto podmenu přístupné musí být síť, nebo data vytvořeny.
  - Síť: Zapne nebo vypne vypisování váhy neuronů sítě do zobrazovací plochy.
  - Data: Zapne nebo vypne vypisování trénovacích data do zobrazovací plochy.
- 2-D: Pokud jsou vytvořeny vykreslí 2D reprezentaci trénovacích dat a vah neuronů sítě do zobrazovací oblasti. V síti jsou vyznačené propojení s přímo sousedícími neurony. Trénovací data jsou zobrazena zelenými kruhy, neurony černými tečkami, aktivní neuron červenou tečkou a spojení s bezprostředně sousedícími neurony šedou přerušovanou čárou. Položky v tomto menu jsou

přístupná pouze pokud má síť dva vstupy, respektive trénovací data mají dimenzi dva.
  - Síť: Zapne nebo vypne zobrazení 2D reprezentace sítě.
  - Data: Zapne nebo vypne zobrazení 2D reprezentace trénovacích dat.
  - Okolí neuronu: Vykreslí fialovou čarou hranice oblastí jednotlivých neuronů. Každá oblast obsahuje datový prostor který je nejblíže danému neuronu.
- RGB: Zobrazí reprezentaci výstupní vrstvy sítě. A to tak, že každý neuron je zastoupen obdélníkem, jenž je vybarven barvou, která odpovídá jeho vahám. Ty musí být 3 a mít hodnoty 0-255, barva je pak dána vzetím vah jako hodnot RGB barevného prostoru. Pokud jsou váhy mimo rozsah je zobrazená barva bílá. Aktivní neuron je zobrazen červeně.
- U-matrix: Zobrazí reprezentaci U-matrixu výstupní vrstvy sítě pomocí škály šedi. Černá barva zobrazuje největší vzdálenost mezi sousedícími neurony, bílá nejmenší. Aktivní neuron je zobrazen červeně.
- Označení: Zapne nebo vypne zobrazení uživatelem vytvořených značek, pro textový výpis se značky nezobrazují.
- Vrstvy mapy: Zobrazí náhled na obě vrstvy sítě. Funkce je přístupná pouze pokud je síť vytvořená.

Položka Učení

Položka Učení slouží k nastavení a spouštění adaptace neuronové sítě.



Obr. A.5: Rozbalené menu Učení.

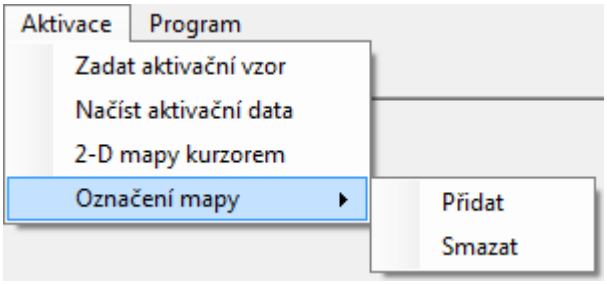
- Nastavení: Otevře okno pro nastavení adaptace, více viz podkapitola A.1.5.
- Jeden krok: Provede jeden krok adaptačního algoritmu, lze taktéž spustit klávesou mezerník. Při zaplém 2D zobrazení je krok rozložen na dílčí části:
  1. Výběr trénovacích dat, vybraný vzor je červeně zvýrazněn.
  2. Vyhodnocení nejbližšího neuronu a jeho okolí. Nejbližší neuron je zvýrazněn červeně, neurony spadající do jeho okolí a topologické spojnice mezi nimi oranžově.

3. Změna vah. Dále se pokračuje krokem 1.

- Spustit: Spustí celou adaptaci, zobrazení sítě je zaktualizováno až po proběhnutí adaptace.
- Průběžné vykreslování: Zapne nebo vypne mód průběžného vykreslování. Při zapnutém módu je po spuštění adaptace zobrazení sítě aktualizováno po každém kroku.
- Citlivostní analýza: Otevře okno citlivostní analýzy.

Položka Aktivace

Položka Aktivace slouží k aktivaci mapy. Tato položka je zpřístupněná jen pokud je vytvořená mapa.

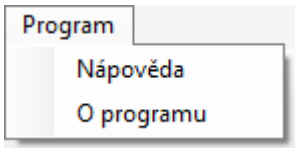


Obr. A.6: Rozbalené menu Aktivace.

- Zadat aktivační vzor: Otevře okno pro zadání aktivačního vzoru z klávesnice.
- Načíst aktivační data: Otevře okno pro načtení souboru. Data obsažená v souboru musí mít stejnou dimenzi jako má síť vstupů. Po načtení se otevře okno s výběrem jednotlivých vzorů pro aktivaci.
-
- 2-D mapy kurzorem: Funkce je přístupná pouze pokud má mapa dva vstupy. Aktivační data se přidávají kliknutím na zobrazovací plochu, aktivovaný neuron je červeně zvýrazněn. Mód aktivace se ukončí buď stiskem klávesy Esc, nebo opětovným kliknutím na položku v menu.
- Označení mapy:
  - Přidat: Otevře okno pro přidání značky na zobrazovací plochu. Značka je tvořena zeleným křížkem a názvem značky. Značka se umisťuje po stisknutí tlačítka přidat, kliknutím na zobrazovací plochu.
  - Smazat: Smaže veškeré značky.

Program

Skrze menu Program je možný přístup k základním informacím o programu a k nápovědě.

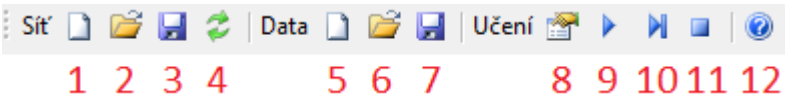


Obr. A.7: Rozbalené menu Program.

- Nápověda: Otevře nápovědu v HTML formátu.
- O programu: Zobrazí základní informace o programu.

A.1.2 Panel rychlého přístupu

Panel rychlého přístupu obsahuje nejzákladnější funkce, jejich vlastnosti jsou odpovídající těm z menu. Jedinou výjimkou je tlačítko zastavit, pomocí něhož se zastaví adaptace, to je užitečné zejména při zapnutém průběžném vykreslování, při němž je proces adaptace časové náročný.

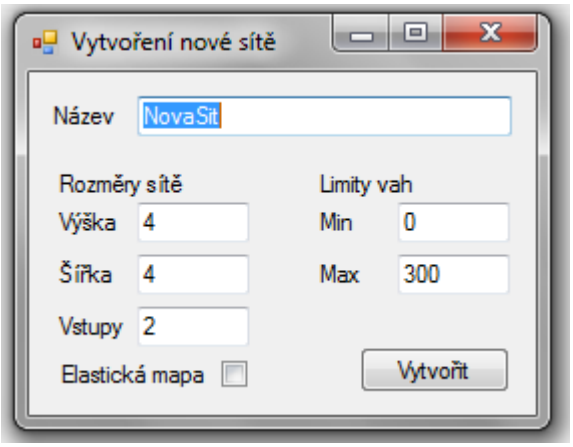


Obr. A.8: Panel rychlého přístupu.

1. Vytvoření nové obecné sítě.
2. Otevření sítě ze souboru.
3. Uložení sítě do souboru.
4. Inicializace vah sítě.
5. Generování obecných dat.
6. Načtení dat ze souboru.
7. Uložení dat do souboru.
8. Nastavení učení.
9. Jeden krok adaptace.
10. Spustit adaptaci.
11. Zastavit adaptaci.
12. Nápověda.

A.1.3 Vytvoření mapy

Program umožňuje vytvoření Kohonenovy neuronové sítě čtvercové topologie libovolných rozměrů s libovolným počtem vstupů. Dále je možno zvolit vlastnost elasticity. Po vytvoření nové sítě jsou informace o ní zobrazeny v poli 4. viz obrázek A.1.



Obr. A.9: Okno vytvoření nové sítě.

Při vytvoření nové mapy je automaticky provedena náhodná inicializace vah, váhy jsou generovány v rozmezí hodnot minima a maxima sítě zadaného při vytváření sítě. Tyto minima a maxima však hrají roli pouze při inicializaci, při předložení dat které jsou mimo tyto meze mohou váhy neuronů taktéž přesáhnout tyto limity. Inicializaci vah může uživatel taktéž spustit sám.

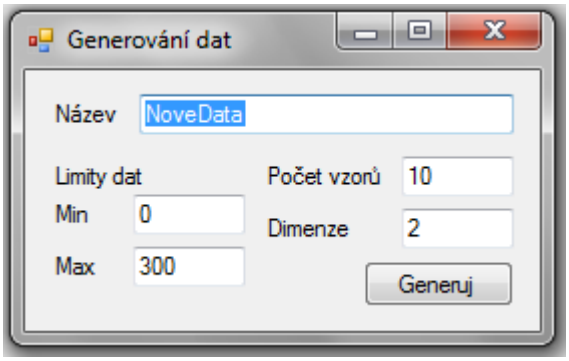
Kromě vytvoření nové mapy může uživatel načíst již vytvořenou mapu uloženou v souboru. A svou vytvořenou síť může taktéž do souboru uložit.

Všechny funkce obstarávající práci se sítí jsou v hlavním menu pod položkou Síť, popsáno v podkapitole A.1.1.

A.1.4 Generování trénovacích dat

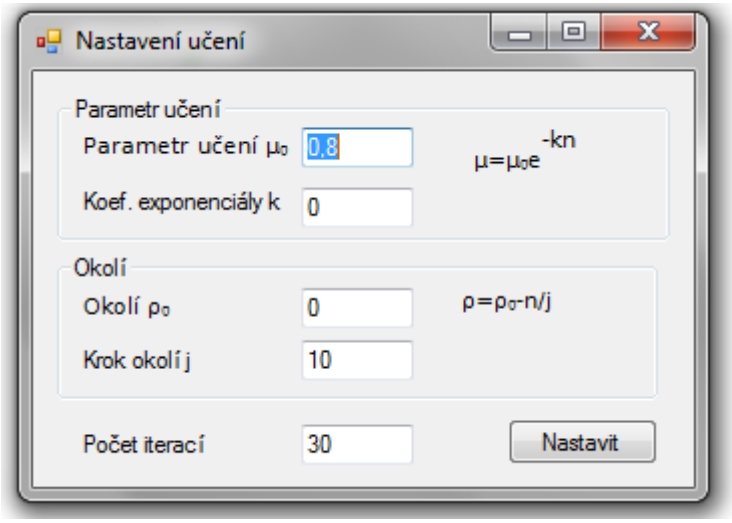
Program zajišťuje možnost generování náhodných dat o libovolné dimenzi, meze těchto hodnot se opět dají nastavit. Další možností generace dat je pomocí kurzoru, tato metoda generuje 2-D data na základě toho kam uživatel kliká. Dále může uživatel vytvořit data ručně pomocí klávesnice. Data je možno načíst ze souboru nebo do souboru uložit.

Funkce pracující s trénovacími daty jsou pod položkou Data v hlavním menu, popsáno v podkapitole A.1.1.



Obr. A.10: Okno generování trénovacích dat.

A.1.5 Učení



Obr. A.11: Nastavení adaptace.

Funkce sousedství je ostrá a časová závislost poloměru okolí ρ je lineárně klesající. Závislost okolí na čase:

$$\rho = \rho_0 - \frac{n}{j}, \tag{A.1}$$

n je počet proběhlých iterací, $j \in (0, \infty)$ je parametr udávající rychlost zmenšování okolí (udává počet iterací, po kterých se okolí sníží o jedna). Pro $j = 0$ je funkce snižování okolí s počtem iterací vypnuta. ρ_0 je počáteční poloměr okolí a ρ poloměr aktuální. Okolí může nabývat pouze celočíselných hodnot (dělení je celočíselné). ρ_0 a j jsou uživatelem voleny.

Funkce snižování parametru učení μ s časem je exponenciální.

$$\mu = \mu_0 e^{-kn}, \tag{A.2}$$

n je opět počet proběhlých iterací, parametr $k \in \langle 0, \infty)$ udává rychlost snižování μ, μ_0 je počáteční hodnota parametru učení a μ je aktuální hodnota parametru. Parametry k a μ_0 volí uživatel.

Posledním uživatelem nastavovaným parametrem je počet iteračních kroků.

Program umožňuje provést celou iteraci naráz, přičemž při vypnutém Průběžném vykreslování jsou výsledky vypsány až po ukončení učení, při zapnutém průběžném vykreslování jsou výsledky vypisovány po každém kroku adaptace. To je však časově náročné, proto je uživateli k dispozici tlačítko zastav (viz položka 10. na obrázku A.8). Dále je možnost adaptaci krokovat po jednotlivých fázích. Tyto funkce jsou přístupné z menu, popsaného v podkapitole A.1.1.

A.2 Datové formáty

Pro uložení sítě a dat se využívá textového zápisu do souboru. Soubory pro síť mají příponu .knn, pro trénovací data .dat. Nicméně je možno načíst soubor s odpovídajícím formátováním nehledě na příponu. Soubory lze editovat pomocí běžného textového editoru, například poznámkovým blokem.

A.2.1 Formát sítě .knn

Číslo řádku	Obsah řádku
1.	Název sítě
2.	Počet vstupů
3.	Výška mapy
4.	Šířka mapy
5.	Minimální hodnota vah
6.	Maximální hodnota vah
7.	1/0 (1 pokud je mapa elastická, 0 pokud není)
8+	w_{m1} w_{m2}.... w_{mn}

Tab. A.1: Formát souboru sítě .knn.

A.2.2 Formát dat .dat

Číslo řádku	Obsah řádku
1.	Název trénovacích dat
2.	Dimenze trénovacích dat
3.	Počet trénovacích vzorů
4.	Minimální hodnota dat
5.	Maximální hodnota dat
6+	x_{m1} x_{m2} ... x_{mn}

Tab. A.2: Formát souboru sítě .dat.

A.3 Struktura adresářů

V kořenovém adresáři programu je hlavní spustitelný soubor KNN.exe a složky do kterých se implicitně ukládají výsledky programu:

- Citlivostni: do tohoto adresáře se ukládají výsledky citlivostní analýzy.
- Data: v tomto adresáři jsou uchovávány uložená data.
- Net: v tomto adresáři jsou uchovávány uložené sítě.

Poslední složkou v kořenovém adresáři programu je složka help, v níž jsou uloženy soubory nápovědy programu.