VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

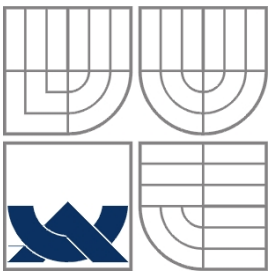
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

# DETEKCE HRAN POMOCÍ NEURONOVÉ SÍTĚ

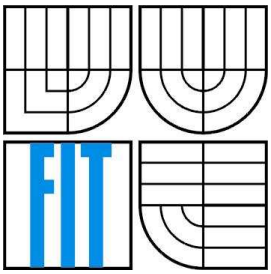
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE SOŇA JAMBOROVÁ
AUTHOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

# DETEKCE HRAN POMOCÍ NEURONOVÉ SÍTĚ

NEURAL NETWORK BASED EDGE DETECTION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

SOŇA JAMBOROVÁ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MIROSLAV ŠVUB

BRNO 2009

# Abstrakt

Tato práce se zabývá návrhem a implementací softwaru pro detekci hran v obraze pomocí neuronové sítě. Definuje nezbytné základní pojmy v této problematice. Zaměřuje se hlavně na přípravu obrazových informací pro detekci pomocí neuronové sítě. Popisuje a porovnává různé přístupy k využití implementovaného softwaru na syntetické a reálné množině obrázků, včetně experimentů.



# Abstract

This work is about suggestion and implementation of the software for detection of edges in images using neurons network. It defines basic terms for this topic and focusing mainly at preperation imaging imformation for detection using nerons network. Describing and comparing different aproachings for using implemented software on synthetic and real set of images, including experiments.



# Klíčová slova

Neuronová síť, neuron, dopředná neuronová síť, zpětné šíření chyby, CImg knihovna, FANN knihovna, LoG, DoG, detekce hran, obraz, RGB, konvoluce, mediánový filtr, Gaussův filtr, Gaussův šum, prahování

# Keywords

Neurons network, neuron, forward neuron network, Back-Propagation Algoritmus, CImg library, FANN library, LoG, DoG, detection of edges, images, RGB, convolution, Median filter, Gaussian filter, tresholding, Gaussian noise


# Citace

Jamborová Soňa: Detekce hran pomocí neuronové sítě, bakalářská práce, Brno, FIT VUT v Brně, 2009

# Detekce hran pomocí neuronové sítě

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Miroslava Švuba. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

……
Soňa Jamborová
18. 5. 2009

## Poděkování

Děkuji vedoucímu bakalářské práce Ing. Miroslavu Švubovi za udělené rady a pomoc při vypracovávání této práce.

# Obsah

# Úvod

Aniž si to uvědomujeme, každou minutu našeho života určujeme, kde je jaký objekt (židle, stůl, mobil, …). Dá se říci, že určujeme hranu objektu. Často tato informace rozhoduje o našem dalším jednání. Může třeba ovlivnit rozhodnutí, jestli si sedneme na židli, které chybí noha. Dále se informace o hraně objektu hodí v situacích jako např. čtení, hledání tužky na stole, otvírání okna, vyhnutí se překážce na ulici, orientaci v prostoru atd. To jsou důvody, proč detekce hran patří mezi důležité úkony při rozpoznávání obrazu.

Jak jsem již uvedla, naše tělo detekuje hrany celý život. Starají se o to hlavně dva smysly: hmat a zrak. Zrak zpracuje více informací než hmat. Zkusili jste někdy najít tužku se zavřenýma očima? Jde to těžko. Zrak občas zklame (přehlédneme se), po hmatu ji najdeme vždycky, ale déle to trvá. Tělo v obou případech k detekci využívá nervovou soustavu, jejíž základní jednotkou je nervová buňka (neuron).

Nervová soustava má úžasnou vlastnost aproximace různých funkcí. Lidstvo se snaží popsat matematicky činnosti, jež denně využívá, ale tento popis bývá často velmi složitý, zatímco neurony živého organismu obdobné problémy řeší ve zlomku vteřiny. Některé funkce jsou vrozené (reflexy), jiné se musí člověk naučit. Jak to vypadá, detekce hran pomocí neuronové sítě není nic nového. Je tu od zrození prvních organismů. Je na čase se pokusit o její převedení do digitální podoby. Abstrakcí nervové soustavy pro technické účely je neuronová síť.

Cílem mé práce je experimentace s neuronovou sítí pomocí softwaru, který si sama za tímto účelem vytvořím. Jeho návrh a implementace je popsána v odpovídajících kapitolách dokumentu. Předmětem experimentu bude stanovení optimálních parametrů pro co nejúčinnější nasazení softwarových modulů. Do těchto parametrů můžeme zařadit vhodný výběr testovací množiny, struktury neuronové sítě a úpravy obrázků pro korelaci nedostatků již natrénované sítě. Upozorním na nedostatky nebo přednosti jednotlivých postupů. Pro měření výkonnosti sítě využiji náhodně vygenerované syntetické obrázky, zatímco pro ověření schopností natrénované neuronové sítě v praxi využiji reálné digitální fotografie. Avšak druhá z jmenovaných množin vstupních dat se nedá zcela objektivně zhodnotit. Také nastíním možnosti detekce hran pomocí jiných algoritmu.

# 1 Umělá neuronová síť

Umělá neuronová síť je výpočetní model, který našel své uplatnění v umělé inteligenci. Má svůj vzor v biologické nervové soustavě. Ta má za úkol zachytit a zpracovat podměty působící na organismus a zajistit odpovídající reakci na ně. Základním prvkem nervové soustavy je neuron.

## 1.1 Historie

Existují dva zásadní důvody pro realizaci umělých neuronových sítí:

- pochopit lidský mozek,
- napodobit funkce lidského mozku.

V roce 1943 pánové Warren McCulloch a Walter Pittse přišli s jednoduchým modelem umělého neuronu. Podařilo se jim dokázat, že nejjednodušší typ modelu neuronu, může počítat libovolnou aritmetiku nebo funkci. V roce 1949 vyšla kniha *The Organization of Behavior* od Donalda Hebba. Hebb byl k jejímu napsání inspirován myšlenkou podmíněných reflexů pozorovaných u živočichů. Přinesla návod na učící pravidlo pro umělé synapse. Celkově se však v 40. a 50. letech 20. století nepodařilo dosáhnout zásadního pokroku ve vývoji neuronových sítí.

V roce 1957 byl zobecněn model neuronu Frankem Rosenblattem na tzv. perceptron. Jedná se o pevnou architekturu jednovrstvé sítě, která měla $m$ vstupů a $n$ výstupů a počítá s reálnými čísly. Rosenblatt navrhl učící algoritmus pro perceptron a spolu s Charlesem Wigthmanem sestrojil první neuropočítač pro rozpoznávaní znaků (A, B, C, D, …). Kniha Perceptrons od Marvina Minského a Seygmoura Parperta poukázala na nemožnost preceptoru počítat funkci XOR. To pozastavilo přísun financí a pro období 1967–1982 nastává stagnace v tomto oboru.

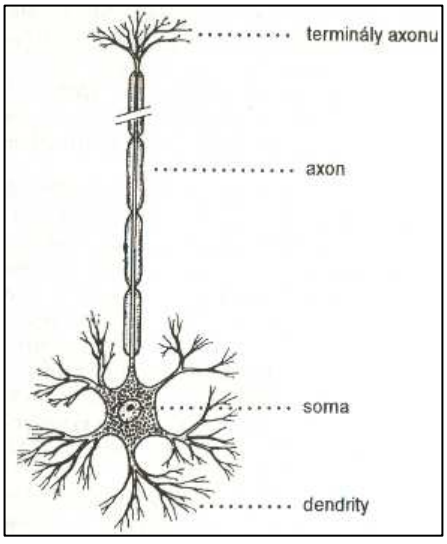
V 80. letech 20. století začíná velký rozvoj neuronových sítí. Začaly se rozvíjet cyklické a vícevrstevné sítě. Pro ně byl vyvinut učící algoritmus zpětného šíření chyby tzv. backpropagation, který je používán v 80 % neuronových sítí. Čímž se podařilo vyřešit problém, který byl naznačen Minským, a realizovat funkci XOR. Ta podle něho byla nerealizovatelná. V roce 1987 byla založena mezinárodní společnost pro výzkum neuronových sítí INNS. Vzniká řada časopisů věnujících se této problematice. Od té doby jsou systémy založeny na neuronových sítí nasazovány do praxe.

## 1.2 Biologický neuron

Neuron je základní stavební prvek neuronové sítě. Je volen jako parafráze k biologickému neuronu. U biologického neuronu (*Obrázek 1*) rozlišujeme:

- tělo neuronu,
- výběžky neuronu:
  - dendrity – dostředivé dráhy, přebírají informaci z axonů jiných neuronů;
  - axony – odstředivé dráhy, neuron má pouze jeden, posílá informaci dalším neuronům.

Spojení axonu a dendritu se nazývá synapse. Synapse přiřazuje propustnost určité dráze v nervové síti. Šíření



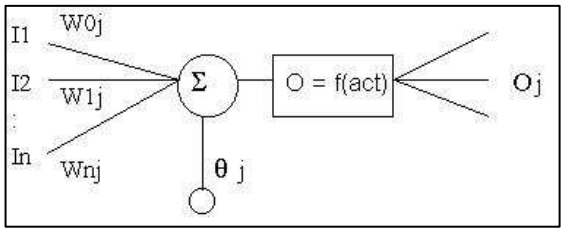

***Obrázek 1:*** *Schéma biologického neuronu*

vzruchu v biologickém neuronu se děje na základě akčního potenciálu, který je v neurální membráně a práh šíření je dán propustností synapse.

# 1.3 Umělý neuron

Umělý neuron (*Obrázek 2*) je sestaven podle neuronu biologického. Jeho funkčnost je však více přizpůsobena matematické specifikaci. Rozlišujeme:

- tělo neuronu ( $f_{act}(x)$ ) – aktivační funkce neuronu;
- spoje mezi neurony – každé dva neurony v síti mohou mít maximálně jeden spoj:
  - vstupy (I) – napojeny na výstupy předcházejících neuronů, mají určitou váhu ( $w$ );
  - výstupy (O) – nesou výsledek aktivační funkce, přivádí ho na vstup dalším neuronům.

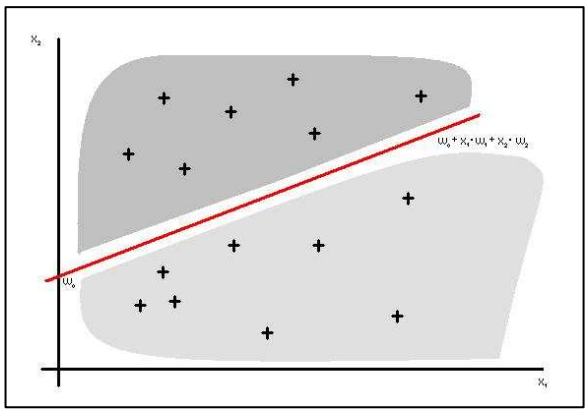



***Obrázek 2:*** *Schéma umělého neuronu*

Synapse zde nejsou přímo konstruovány, proto je každému spoji přiřazena váha. Ta má za úkol zvýraznit, případně utlumit jednotlivé spoje neuronů. Každý vstup je vynásoben touto váhou a výsledky mezi sebou sečteny. Součet slouží jako vstup pro přenosovou funkci, která rozhodne, zda je dostatečně velký, aby mohl být vzruch dál nesen.

Umělý neuron je dán relací $N : \Re^n \rightarrow \Re$ . Tohoto vztahu je dosaženo následující strukturou neuronu, u které popisujeme:

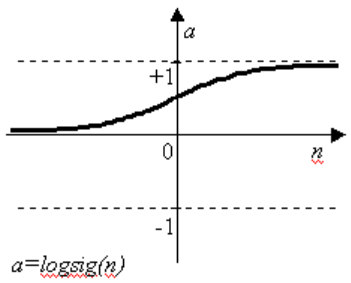
- vektoru vstupů: $\vec{X} = (I_0; I_1; ...; I_n)$; kde $I_0 = 1$ ;
- vektoru vah vstupů: $\vec{W} = (w_0; w_1; ...; w_n)$;
- aktivační funkcí: $f_{act}(\sum_{i=0}^{n} I_i \cdot w_i)$ ;
- výstupem: O.



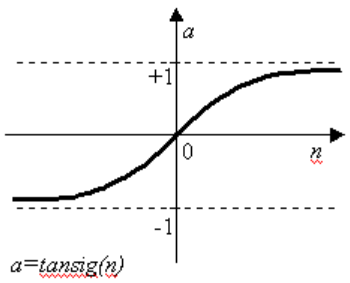
***Obrázek 3:*** *Schéma rozdělení bodů roviny do dvou množin neuronem s dvěma vstupy*

Lze říci, že jádro aktivační funkce neuronu jde zapsat též jako $w_0 + x_1 \cdot w_1 + ... + x_n \cdot w_n$ . Tento výraz je součástí rovnic nadrovin (pro $n = 2$ to je rovnice přímky, pro $n = 3$ to je rovnice roviny, atd.). Z toho vyplývá situace zakreslená na obrázku 3.

Aktivační funkce může nabývat různých tvarů. Rozeznáváme funkci skokovou, lineární, sigmoidní. Každá z těchto funkcí má různé modifikace.




**Obrázek 4:** *Klasická sigmoidní funkce*

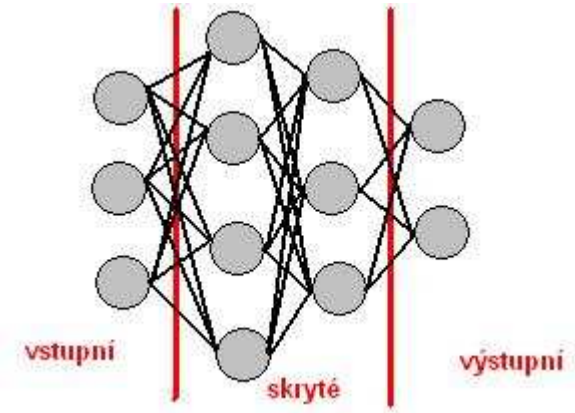



**Obrázek 5:** *Symetrická sigmoidní funkce*

## 1.4 Dopředná neuronová síť

Patří k nejpoužívanějším neuronovým sítím. Neurony jsou v ni uspořádány v několika vrstvách, které jsou řazeny za sebou. Dopředná neuronová síť je dána relací $NN : \Re^m \rightarrow \Re^n$; kde počet vstupních neuronů (vstupní vrstva) je $n$ a počet výstupních neuronů (výstupní vrstva) je $m$. Vícevrstvá síť obsahuje taktéž kromě vstupních a výstupních neuronů i neurony skryté (jejich počet označme jako $k$), jež mohou být uspořádány do dalších vrstev. Propojeny jsou pouze neurony sousedících vrstev. Spoje jsou jednosměrné a nemají žádnou zpětnou vazbu. Dopředná síť má minimálně dvě vrstvy a to vstupní a výstupní vrstvu.

Návrh uspořádání sítě není jednoznačně dán. Topologie sítě je dána specifičností úlohy, pro kterou je navrhována. Většinou návrh vychází ze simulačních výpočtů a experimentů. Řada úloh má uspokojivý efekt už při použití dvouvrstvé sítě.




**Obrázek 6:** *Schéma sítě o 4 vrstvách*

Učící algoritmus zpětného šíření chyby (Back Propagation Algoritmus) byl navržen pro dobřednou síť. Na vstupy sítě ($I$) se opakovaně zadávají hodnoty z tzv. trénovací množiny ($T$) a hledá se optimální nastavení vektoru vah, při které bude chyba výstupu neuronové sítě ($E$) minimální. Formálně se dá trénovací množina zapsat jako $T = \{(I;O); I \in \Re^n; O \in \Re^m\}$. Poté je možné minimální chybu výstupu matematicky formulovat následujícím způsobem $E = \sum_{i=1}^{|T|} (NN(I_i) - O_i)^2$. Jeden průchod testovací množiny neuronovou sítí se nazývá epocha. Ukončení procesu učení, lze po dosažení:

- stanovené minimální hodnoty chyby,
- stanoveného počtu epoch trénování,
- stanoveného minimálního rozdílu dvou po sobě jdoucích hodnot chyby trénování,
- při minimální chybě trénování.

Volba je na zkušenosti uživatele a specifikaci řešené úlohy.

# 2 Zpracování obrazu

V této kapitole se pokusím vysvětlit metodiky zpracovávání obrazu a vysvětlit důležité pojmy spojené s detekcí hran pomocí neuronové sítě. Kapitola určuje v jakém kontextu budou tyto pojmy dále využívány.

## 2.1 Obraz

Obraz je obecně jakékoliv grafické vyjádření. Zpravidla bývá ztvárněn ve dvou dimenzích, jako například fotografie, kresby. Bývá zachycen optickými přístroji (fotoaparáty, kamerami, …), přírodními objekty (oko, světločivná skvrna, …), přičemž se využívá různých optických jevů (rozhraní dvou prostředí, odraz a lom světla atd.). Takto získaný obraz lze dále zpracovávat nervovou soustavou nebo počítačem – využíváme digitální obraz.

Obraz je dán spojitou funkcí $I:\langle 0,w\rangle\times\langle 0;h\rangle\to C$, kde $w$ je šířka obrazu, $h$ je výška obrazu, $C$ je barevný prostor. Pro digitální obraz se však užívá diskrétní obrazová funkce $I:\{x\in N;0\le x\le w\}\times\{y\in N;0\le y\le h\}\to C$.

### 2.1.1 Barevný prostor

Existuje pouze jeden barevný prostor, který může být daný různou interpretací.

**Šedotónový diskrétní barevný prostor** ($C_G$ - jas, intenzita) je dán pouze jednou dimenzí, která je reprezentována hodnotami z množiny $C_G=\{b\in N;0\le b\le 255\}$. Obrazová funkce, která se promítá do $C_G$, znázorňuje tzv. obrázek v stupni šedi.

**Model RGB** je způsob znázornění barevného prostoru. Můžeme ho definovat jako $C_{RGB}=C_G^3$. Je to aditivní způsob míchání třech barevných složek červené (red), zelené (green) a modré (blue). Na výsledné barvě se každá složka podílí určitým množstvím. Pro digitální reprezentaci se často používá 24 b, kdy každá složka má k dispozici 8 b. Což v decimální soustavě je interval 0–255 a pro hexadecimální soustavu 00–FF. Celkem jsme tak schopni zobrazit něco přes 16 milionů odstínů barev. Zvolený odstín se uvádí v hexadecimálním tvaru následovně RRGGBB, kde R, G, B je hexadecimální číslice pro danou složku. Pro názornost uvedu pár barev:

**Tabulka 1:** *Reprezentace barevného prostoru modelem RGB*

| BARVA | R | G | B |
|---|---|---|---|
| ČERVENÁ | FF | 00 | 00 |
| ZELENÁ | 00 | FF | 00 |
| MODRÁ | 00 | 00 | FF |
| BÍLÁ | FF | FF | FF |
| ČERNÁ | 00 | 00 | 00 |

### 2.1.2 Intenzita v modelu RGB

Každá ze tří složek modelu RGB má jiný podíl na výsledné intenzitě pixelu. Experimentálně se zjistilo, že největší podíl na intenzitě má zelená složka (55 %), dále červená (30 %) a nejmenší podíl

má složka modrá (15 %). Podíl se mírně liší podle použitého zdroje. Výpočet intenzity pixelu se často využívá při převodu barevného obrázku na stupně šedi.

# 2.2 Filtrace obrazu

Při snímání obrazu (fotografování, skenování, …) dochází k různým anomáliím, označovaným jako šum. Na digitální fotografii, můžeme zaznamenat, že každý pixel má jinou hodnotu, které si však jsou docela blízké. Zašumění obrazu pomáhá vytvořit realistickou podobu synteticky vzniklých obrazů. Avšak při digitálním zpracování reálných obrazů může způsobovat nestabilitu metod.

Rozeznáváme 2 typy šumu:

- náhodný šum – označován též jako nezávislý šum, bílý šum či sůl a pepř;
- Gaussův šum – označován jako závislý šum, je dán Gaussovým rozložením náhodných čísel.

## 2.2.1 Konvoluce

Konvoluce je důležitou operací při lineárním zpracování obrazu. Konvoluce dvourozměrných spojitých funkcí $f$ a $h$ je definovaná jako $f(x,y) * h(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x-a, y-b)h(a,b)dadb$.

Protože budeme používat zvláště digitální obraz, jež je definovaný jako diskrétní funkce ( $I$ ), bylo by vhodné se zmínit o diskrétním tvaru konvoluce, jež zní:

$$I(x,y) * h(x,y) = \sum_{i=-k}^{k}\sum_{j=-k}^{k} I(x-i, y-j)h(i,j).$$

V obou případech je funkce $h(x,y)$ nazývána jádrem konvoluce. To má vliv na váhy jednotlivých prvků konvoluční matice.

## 2.2.2 Gaussův filtr

Gaussův filtr je efektivní lineární metoda pro odstranění šumu. Je to obdobná technika jako odstranění šumu obyčejným průměrováním. Metoda je založena na váženém průměru s využitím Gaussova rozložení náhodných čísel. Při použití se nemění celková světlost obrazu a dochází k rozmazání hran.

Při aplikaci je využito konvoluční masky s konvolučním jádrem $h_g(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{-x^2+y^2}{2\sigma^2}}$ .

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

**Obrázek 7:** *Konvoluční matice pro okolí 3×3*

### 2.2.3 Mediánový filtr

Mediánový filtr je nelineární metoda pro odstranění šumu zvláště typu pepř a sůl. Pro každý pixel vypočítá medián zvoleného okolí. Je vhodné volit okolí s lichým počtem prvků.

Medián souboru je statistický údaj, který říká, že 50 % prvků souboru je menší než tato hodnota a 50 % prvků souboru je větší než tato hodnota. Základním algoritmem výpočtu je seřazení prvků souboru podle velikosti hodnot a vybrání prostředního z nich. Z tohoto důvodu je vhodné volit lichý počet prvků, kde je právě jeden prostřední prvek. Medián lépe reprezentuje hodnotu daného souboru. Dokáže eliminovat ojedinělá maxima či minima mnohem lépe než průměr. Výpočet mediánu je náročný na čas. Závisí na výběru řadícího algoritmu.

Při použití mediánového filtru dochází k odstranění šumu a zvýraznění hran, přitom ostré špice rohů se zaoblí.

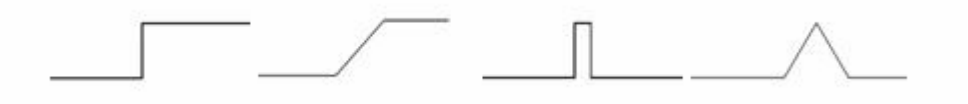
# 3 Detekce hran

Detekce hran jsou metody, které dokážou odhalit hranu objektu zachyceného na obrázku. Má časté použití v počítačovém „vidění“. Využívá se pro rozpoznání obsahu v obraze, 3D rekonstrukce scény, problém korespondence, sledování.

Detekce hran je taková funkce, která všechny body obrazu rozdělí do dvou množin, ty co jsou na hraně a co leží mimo hranu.

## 3.1 Definice hrany

Z geometrického hlediska je hrana průsečnice jednotlivých stěn objektu. Její viditelnost záleží na úhlu pohledu. Ohraničuje objekty v 2D grafice. Avšak pomocí obrazové funkce ($I$) definujeme hranu jako body obrazu, u kterých se hodnota jasu prudce mění (Obrázek). Z toho důvodu je hrana reprezentována velikostí a směrem. Směr růstu funkční hodnoty vystihuje gradient.



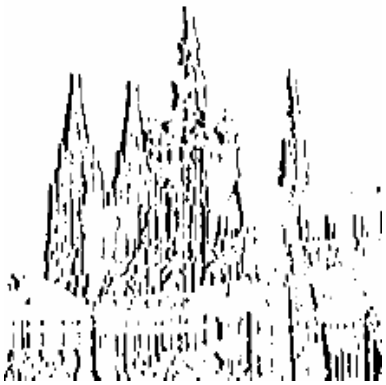
**Obrázek 8:** *Projev hrany v obrazové funkci*

## 3.2 Hledání hran pomocí derivací

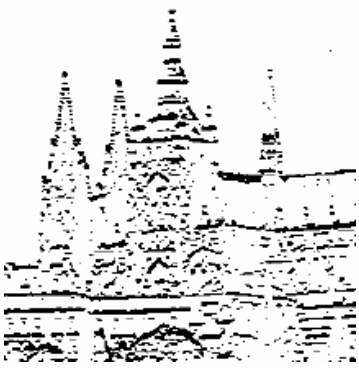
Metody založené na první derivaci hledají extrémy hranové funkce. Derivaci lze vypočítat zvlášť pro řádky nebo sloupce. Velikost gradientu ($G$) pak získáme dle předpisu $G(i,j)=\sqrt{G_R(i,j)^2+G_S(i,j)^2}$, kde $G_R$ je gradient řádku a $G_S$ je gradient sloupce. Výpočet gradientu může probíhat jako konvoluční filtrování. Jedná se například o operátor Prewittové, Sobelův operátor, Robinsonův operátor, Kirschův operátor. Většinou jsou citlivé pouze na jeden směr hrany podle natočení konvoluční matice.

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

**Obrázek 9:** *Operátor Prewittové (matice)*



**Obrázek 10:** *Operátor Prewittové (severní hrana)*



**Obrázek 11:** *Operátor Prewittové (východní hrana)*

$$\begin{bmatrix} 3 & 3 & 3 \\ 3 & 0 & 3 \\ -5 & -5 & -5 \end{bmatrix}$$

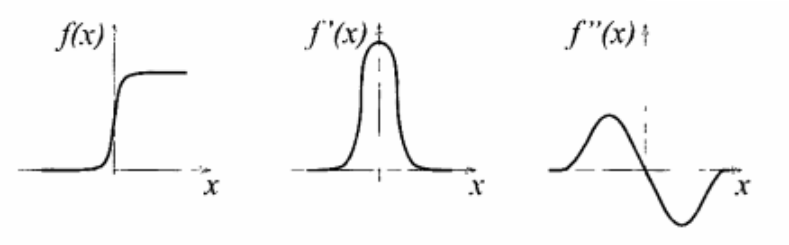
**Obrázek 12:** *Kirschův operátor*

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{bmatrix}$$

**Obrázek 13:** *Robinsonův operátor*

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

**Obrázek 14:** *Sobelův operátor*




**Obrázek 15:** *Průběh obrazové funkce, její 1. derivaci a její 2. derivaci*

Extrémy 1. derivace lze brát jako nulové body 2. derivace. Při hledání extrémů pomocí metod založených na 2. derivaci odpadá složité hledání extrémních hodnot. Stačí se zaměřit na hledání nulových hodnot.

Rovnice metody LoG (Laplacian of Gaussian) využivá následující vztah:

$$\nabla^2[G(x,y,\sigma) * f(x,y)] \triangleq [\nabla^2 G(x,y,\sigma)] * f(x,y)$$

Konvoluční jádro ($\nabla^2 G$) uvedeného vztahu lze analyticky přepočítat. Poté lze vytvořit konvoluční masku pro operátor LoG (tzv. Mexický klobouk). Konvoluce masky s obrazovou funkcí je citlivá na nulu, která označuje hranu. S rostoucím rozptylem Gaussovy funkce rostou požadavky na velikost konvoluční masky (např. $\sigma = 4$ potřebuje masku o šířce 40 pixelů). Zároveň s rostoucí $\sigma$ dochází k potlačení méně významných hran.

$$\begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

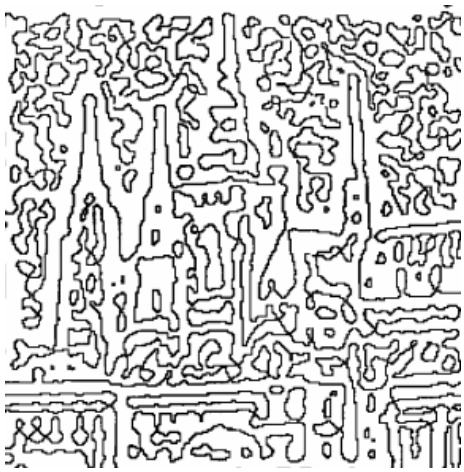
**Obrázek 16:** *Konvoluční maska metody LoG*

Metoda DoG (diference of Gaussian) dobře aproximuje operátor LoG. Obrázek je dvakrát filtrován maskami o různých hodnotách rozptylu Gaussovy funkce, které se od sebe odečtou.

Při použití čistě metod LoG a DoG dochází k efektu, jež se nazývá „talíř špaget“. Protože 2. derivace může dosáhnout extrému s hodnotou 0, což naruší efekt detekce hrany a hrana má tendenci se uzavírat. Toto lze ošetřit, když si v okolí označeného bodu ohlídáme znaménka. Musí dojít k výměně znamének, jak vyplývá z obrázku 15. Toho dosáhneme ještě jednou filtrací o velikosti matice $2\times2$, která odstraní body, v jejichž okolí jsou stejná znaménka.

Je potřeba dodat, že nelze hledat přímo nulové body, ale hodnoty, jež jsou blízké nule.

**Obrázek 17:** *Obrázek po filtraci DoG* *(*$\sigma_1 = 0{,}10; \sigma_2 = 0{,}09$*)*



**Obrázek 18:** *Prahováni obrazku DoG*

**Obrázek 19:** *Prahování obrázku DoG (po filtraci maticí 2x2)*

**Obrázek 20:** *Prahování obrázku LoG (po filtaci maticí 2x2)* - $\sigma = 0{,}20$

## 3.3 Prahování

Prahování je vždy poslední krok při detekci hrany. Po průchodu zvoleným detekčním operátorem mají hrany hodnoty blízké očekávané hodnotě, ne však totožné. Prahování určuje hranice tohoto intervalu. Závisí na tom správné odhalení hran. Pokud je práh příliš nízký, dochází k zašumění. Místa, která nejsou na hraně, ale mají podobné rysy v okolí, jsou označena jako hrana. Je-li však práh moc vysoký, dochází ke ztrátám méně výrazných hran.

# 4 Návrh

Praktická část bakalářské práce má za úkol vyvinout softwarové zázemí pro detekci hran pomocí neuronové sítě. Přitom by program měl klást na uživatele malé nároky ohledně znalosti daného problému. Jediné, o co by se měl uživatel starat, je pouze správné nastavení parametrů, nikoliv se zabývat vlastnostmi potřebných materiálů pro správnou detekci hran. Samozřejmě pokročilému uživateli by měl nabídnou určitou variabilitu.

Stejně jako klasické metody detekce hran se neuronová síť zabývá klasifikací okolí pixelu. Nepoužívají se však žádná konvoluční jádra, nýbrž pouze přepočet vstupního vektoru, jehož velikost je odvozena od velikosti okolí pixelu, pomocí vah spojů mezi neurony na vektor výstupní. Protože se jedná o detekci hran v obraze, bude jako nejčastější zdroj obrazových informací volen syntetický obrázek, který má pro detekci neocenitelnou vlastnost, a to, že lze u něho jasně určit hranu. Množiny syntetických a reálných obrázků lze sjednotit řadou filtrů. Pro přiblížení syntetického obrázku reálnému obrázku lze využít zašumění a naopak pro přiblížení reálného obrázku syntetickému můžeme využít například mediánový filtr. Toho se dá využít jak při sestavování neuronové sítě, tak při vlastní detekci hran. Nakonec je potřeba určit úspěšnost detekce.

## 4.1 Dekompozice návrhu

Metodika práce s neuronovou sítí se dá shrnout do pěti fází. Tyto fáze představují uzavřené činnosti, které dohromady umožní efektivně využít neuronovou síť. To je důvod, proč jsem se rozhodla vytvořit celkem 5 modulů formou konzolových aplikací. Každý z nich by měl představovat jednu z dále uvedených fází.

Navržené aplikace umožňují sdružování do různých celků pomocí dávkových souborů. Lze si vybrat pouze jednu fázi a s tou pracovat, dokud nepřinese maximální zlepšení celé detekce. Toto rozhodnutí jsem učinila na základě náplně mé práce, kterým je experimentace s nastavením jednotlivých modulů. Z experimentů bych měla být schopna zodpovědět následující otázky:

- jak vybrat vhodnou množinu obrázků;
- kolik bodů z obrázku je vhodné vybírat;
- jakou zvolit aktivační funkci neuronu;
- kolik neuronů by mělo tvořit neuronovou síť;
- jak nastavit prahy výstupu z neuronové sítě;
- kdy má síť optimální výsledky.

Uvedené otázky budou zodpovězeny v příslušné kapitole. Je však potřeba při návrhu nechat v těchto věcech potřebnou dávku variability na uživateli. To zajistí volba parametrů přes příkazovou řádku.

Nyní bych se raději zmínila o problémech, na které jsem narazila již při samotném návrhu. Např.:

- jak zvolit způsob tvorby obrázků;
- jakým způsobem vybrat body pro trénování sítě;
- čím by se daly vylepšit výsledky detekce;
- jak změřit výsledky dosažené sítí;
- kompatibilita mezi moduly.

Podobných problémů je mnohem více. Bohužel také odpovědi na uvedené otázky mohou výrazně ovlivnit celkový výkon neuronové sítě. Následující podkapitoly bych věnovala možnostem realizace

softwarového zázemí a přiblížení důvodů, proč jsem se rozhodla pro následující řešení nastíněných problémů.

## 4.2 Princip komunikace modulů

Ještě než začnu s popisem úkolů jednotlivých fází (modulů), ráda bych nastínila způsob komunikace mezi jednotlivými moduly. Protože detekce hran je hlavně o obrazových informacích, zvolila jsem jako komunikační jednotku formát souboru BMP (bitmap picture). Na volbu tohoto formátu mě přivedla grafická knihovna CImg.h, která pro použití více užívaného formátu JPEG potřebuje ještě instalaci dalších knihoven. Formát BMP je zvolen pro reprezentaci:

- trénovací množiny obrázků,
- informací o hranách jednotlivých objektů v obrázcích,
- množiny obrázků, na něž bude použita neuronová síť,
- výsledného označení hran.

Jistou záštitu pro jednotné pojmenování souborů tohoto typu a také jistou zárukou kontability modulů bude knihovna bakalarka.h, v níž budou uvedeny řetězce využívané pro pojmenování jednotlivých souborů a funkce pro přepočet reprezentace barevného prostoru kvůli rozsahům vstupů do neuronové sítě dle zvolené aktivační funkce neuronu.

Další soubory, které se budou používat pro komunikaci, vychází z implementace knihovny FANN (floatfann.h), kterou jsem zvolila pro realizaci vlastní neuronové sítě a práce s ní (trénovaní, spouštění, …). Jedná se o textové soubory „test.data“ a „hrana.net“. Soubory slouží pro reprezentaci zvolené trénovací množiny (vstupy a požadovaný výstup na síť) a vlastní natrénované sítě.

Z důvodů, jež uvedu později jsem se rozhodla, že detekci hran budu provádět na základě okolí jednotlivých pixelů, jež bude mít velikost $5 \times 5$. Na tomto rozhodnutí závisí celá řada realizací, jež budou popsány níže.

## 4.3 Výběr obrázků

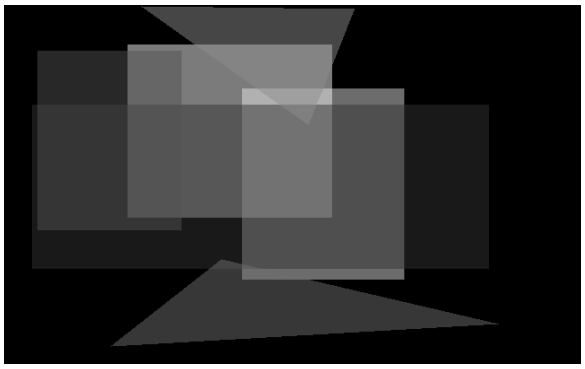
Důležité je si uvědomit, jakou roli hraje použití obrázků v procesu detekce hran pomocí neuronové sítě. Nesmíme zapomenout na to, že obrázky zde nebudou jen pro detekci hran, ale potřebujeme pomocí nich natrénovat i neuronovou síť na očekávaný výstup. Při práci s programem můžeme využit dva typy digitálních obrázků (uměle vytvořené a snímané).

### 4.3.1 Syntetické obrázky

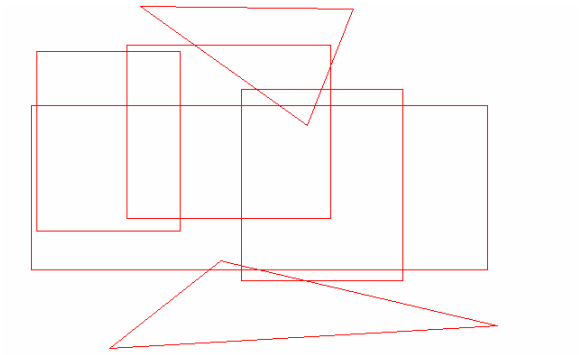
Syntetické obrázky jsou uměle vytvořeny na počítači za pomoci grafických editorů. Tato skupina obrázků se vyznačuje:

- nízkým nebo vůbec žádným šumem,
- jasně danou hranici dvou různobarevných ploch,
- matematickým popisem hran.

Na obrázky nebyly uplatněny žádné filtry, které mají přispět k jejich reálnému vzhledu. Detekce hran pouze na syntetické množině by byla značně jednodušší než na množině reálné. Nemusí se počítat s různými anomáliemi v hodnotách pixelu. I pouhým okem lze přesně určit, kde na obrázku se nachází hrana.



**Obrázek 21***: Syntetický obrázek*



**Obrázek 22:** *Šablona obrázku (znázorňuje hrany)*

**Obrázek 23:** *Maximální zoom hrany Obrázek pomocí aplikace Prohlížeč obrázků a faxů*

**Obrázek 24:** *Maximální zoom plochy Obrázek pomocí aplikace Prohlížeč obrázků a faxů*

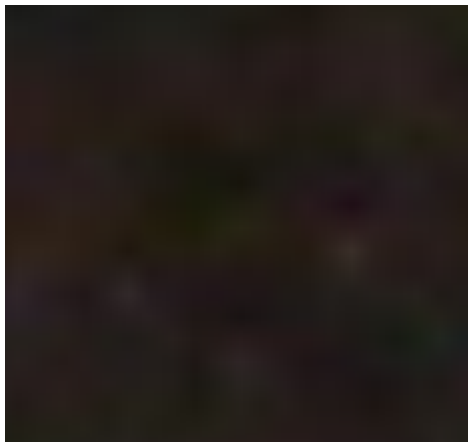
### 4.3.2 Reálné obrázky

Obrázky, které jsou snímané pomocí elektronických zařízení (fotoaparát, skener, …). Jedná se o skupinu obrázků, které se vyznačují:

- vysokým šumem,
- rozmazanou hranicí,
- složitou aproximací funkce určující hranu,
- kolísáním hodnot intenzity pixelu.

Záleží na kvalitě snímání (výběr metody snímání, osvětlení okolí, …). Reálné snímky jsou často matematicky nedokonalé. Manuálně se těžko určuje přesná hranice a matematická aproximace je složitá (křivky, …). Dochází ke zkreslení informací o odstínu sousedních pixelů (šum). Vidíme-li jednobarevnou plochu, neznamená to, že se skládá z pixelů stejného odstínu. To souvisí i s rozmazáním hran. Máme-li rozhraní dvou sobě blízkých odstínů u syntetického obrázku, lze je snadno detekovat, zatímco to samé rozhraní v reálném obrázku splývá. Dochází k rozplynutí hrany do okolí a je horší určit, kde končí jeden odstín a začíná druhý. Vesměs se jedná o efekty, které jsou zapříčiněny nedokonalostí snímajícího zařízení nebo objektu, ale zároveň bez tohoto efektu by se nám snímky nezdály reálné. Snad protože lidské oko také nedokonale zpracovává obrazy (např. možnost realizace videa pomocí sekvence obrázků).


**Obrázek 25:** *Příklad reálného obrázku*



**Obrázek 26:** *Zoom plochy cedule na obrázku 25 pomocí aplikace Prohlížeč obrázků a faxů*

**Obrázek 27:** *Zoom hrany písmena A z cedule na obrázku 25 pomocí Prohlížeče obrázků a faxů*

### 4.3.3 Obrázky použitelné pro detekci

U obrázků použitelných pro detekci budeme chtít určit hrany. Množina obrázků může být tvořena ze syntetických i reálných obrázků. Syntetické obrázky často představují ideální případy. Vše je dokonale přesné. Nenajdeme žádné anomálie. Detekce hran na této množině je podstatně jednodušší než na množině reálných obrázků. Zároveň to však znamená, že detekce, která dává uspokojivé výsledky na syntetické množině, nemusí dávat ty samé nebo odpovídající výsledky na množině reálné, a platí to i naopak. Každá metoda zohledňuje jiné odchylky a jejich záměna může mít fatální dopad na výsledek detekce.

### 4.3.4 Obrázky pro trénování

Pro obrázky určené na tvorbu trénovací množiny je důležité mít možnost určit s požadovanou přesností, zda pixel patří na hranu nebo mimo ni. Požadovaného efektu lze dosáhnout buď pomocí vhodně zvoleného detektoru klasické metody (viz kapitola 3.2) nebo zaznamenáním polohy hran do pomocné struktury.

### 4.3.5 Návrh aplikace

Pro grafické prvky jsem zvolila knihovnu CImg.h. Hlavním úkolem při návrhu aplikace je generovat syntetické obrázky, na kterých by se mohla natrénovat činnost neuronové sítě. Aby mohly být vygenerované obrázky použity, je potřeba někde uložit hranice objektů v nich obsažených. Pro tento účel jsem zvolila obrázek (šablonu) s bílým pozadím, ve kterém jsou červenou barvou označeny hrany. Ke každému rozložení objektů nebo barevné kombinaci (inverze pixelů) je generována i příslušná šablona. Aby bylo dosaženo lepšího výsledku při detekci hran, je možno použít filtr, který vnese do obrázku Gaussovský šum.

Objekty jsou do obrázku zaneseny pomocí generátoru rovnoměrného rozložení. Tímto rozložením je určena poloha objektu, tvar i zbarvení. Zdrojový kód je připraven na přidání i jiných objektů než je jenom trojúhelník a čtverec. Výběr tvaru je určen pomocí větvení příkazem switch( ). Při generování obrazce je zapisováno jak do vlastního obrázku, tak i do šablony. Do obrázku je zakreslen plný obrazec se zvolenou barvou plochy a do šablony jsou zakresleny pouze červené obrysy obrazce stejného tvaru. Propustnost pozadí při vykreslování obrazce do obrázku zajistí více hran různé orientace a velikosti, než plné zakreslení na stejném prostoru.

Gaussovský šum je do obrázku vnášen pomocí generátoru Gaussova rozložení, jež je implementován v knihovně bakalarka.h. Obraz je brán pixel po pixelu a ke každé hodnotě intenzity je přičteno číslo vygenerované dle Gaussova rozložení. Toto rozložení je specifické tím, že hodnoty blízké nule jsou generovány mnohem častěji než hodnoty vzdálenější. Je možné zvolit střední hodnotu i rozptyl Gaussova rozložení.

Inverze obrázku je pouze doplněk intenzity pixelu do maximální hodnoty barevného prostoru, což je číslo reprezentované hodnotou B_MAX. Popsaný algoritmus je použit postupně na každý pixel v obrázku. Inverze je užitečná pro natrénování detekce na opačnou orientaci hrany.

## 4.4 Výběr trénovací množiny

Výběr trénovací množiny je relace vstupů do neuronové sítě a jim přiřazené výstupy. Volba trénovací množiny je důležitý moment v detekci hran pomocí neuronové sítě. Podle zvolené trénovací množiny se bude chovat naše síť. Někdy i menší odchylky od požadovaného vzoru chování, můžou způsobit zhoršení rozpoznání hran o několik procent.

Účelem zmíněné fáze projektu je najít takový způsob výběru trénovací množiny, aby byl program, jež ji bude reprezentovat, schopen bez bližší specifikace uživatelem, vybrat právě takovou množinu, při které bude mít síť relativně vysoké procento úspěšnosti při odhalování hran.

### 4.4.1 Volba okolí pixelu

Volba okolí pixelu je jeden z faktorů, který má vliv na úspěšnost detekce a zasahuje do topologie neuronové sítě. Tím pádem ovlivňuje i tvorbu trénovací množiny. Jedná se hlavně o velikost zvoleného okolí. Můžeme brát v úvahu buď:

- izolovaný pixel,
- 4 okolí,
- různě velké matice o rozměrech $2\times2$, $3\times3$, $4\times4$, $5\times5$ atd.

Je potřeba mít na zřeteli, že při větším okolí může docházet k splynutí dvou hran, které jsou blízko sebe. Okolí by tedy mělo být dostatečně malé a zároveň by mělo být schopno pokrýt přímé okolí pixelu ($3\times3$).

Je dobré si zvolit model, kdy volený pixel je uprostřed matice. Všechny směry tak mají stejnou váhu při určování, zda je pixel na hraně. Například při posunutí okolí pixelu více nahoru by mohlo dojít k potlačení vodorovných hran, nebo by se hrana detekovala, ale došlo by k jejímu posunu ve svislém směru.

Okolí 3×3 je již dostatečné pro detekci hrany. Avšak uvedená varianta by mohla být náchylnější k detekci hrany tam, kde se jedná pouze o šum. Má menší statistický základ. Stabilnější výsledky by mělo zaručovat okolí 5×5. Tady by již ke špatné detekci hrany mělo docházet zřídka (má větší statistický soubor). Je třeba si uvědomit, že čím větší okolí, tím širší čárá bude hranu označovat. U okolí 5×5 bude tloušťka dobře nalezené hrany 5 pixelů.

### 4.4.2 Způsob výběru trénovací množiny

Úkolem fáze je výběr správných bodů pro natrénování neuronové sítě. Výsledek má vliv na celou funkčnost detekce hran. Množina by měla obsahovat hlavní rysy obrázků, na které bude síť aplikovaná. Neuronová síť působí vlastně jako násobička jednotlivých vstupů, které mají koeficienty zvoleny tak, aby při vstupech podobných vlastností ukázala obdobný výstup. Proto bude potřeba zajistit, aby v trénovací množině bylo zastoupeno okolí rozhraní s:

- různým sklonem hran,
- různou orientací a velikostí vektoru hrany.

Uživatel však bude běžně používat program na rozličné soubory obrázků. Předem tedy nebude znát společné rysy. Mým cílem je navrhnout takový algoritmus, aby výběr okolí byl co nejvyváženější. Důsledkem je platnost těchto pravidel:

- Od počtu zastoupení hrany určitých parametrů v trénovací množině se odvíjí, jak bude síť na ni reagovat (čím více vzorků, tím lepší natrénovanost).
- Musí být ve stejném poměru zastoupeny všechny možnosti pro výstup sítě (jinak by se ztratila schopnost sítě diferencovat jednotlivé případy).
- Algoritmus by měl být schopen ze stejných trénovacích obrázků najít obdobné trénovací množiny, pro možnost srovnání výsledků, mělo by platit následující tvrzení: $T_1 \subset T_2 \wedge |T_1| < |T_2|$.

Všechny tyto vlastnosti by měly platit současně. Je třeba říci, že uživateli by měla být ponechána možnost ovlivnit výslednou natrénovanost sítě skladbou trénovacích obrázků. Proto jakákoliv podružná kontrola vybraných bodů implementována do programu, která by mohla zamezit uživateli v ovlivnění trénovanosti sítě, je nežádoucí, i když celkově povede k lepším výsledkům detekce hran.

### 4.4.3 Návrh aplikace

Aplikaci tvoří hlavně algoritmus pro výběr bodů testovací množiny a algoritmus pro zápis okolí o velikosti 5×5 vybraného pixelu, tak aby bylo pokryto, co nejvíce možností výskytu hrany a také jistá míra podobnosti vybraných množin na stejných obrázcích.

Výběr bodů do testovací množiny je zajištěn výběrem řádků, ve kterých jsou nalezeny všechny hrany. Rozložení vybraných řádků je dáno na základě aritmetické posloupnosti:

$$a_{i+1} = a_i + d \text{ ; kde } d \text{ je konstantní}$$

Pro vytvoření této posloupnosti je potřeba znát počáteční řádek $a_0$ a $d$. Specifika použité aritmetické řady vyplývají z ošetření, aby nedocházelo k výběru mimo hranice obrázku a aby následné okolí pixelu bylo celé v obrázku. Z toho plyne, proč pro každý člen platí:

$$a_i \in \{y \in N; 2 \le y \le \max Y - 3\}$$

a je toho dosaženo příkazem *ai = ai%obr.height* a pokud i přesto je $a_i$ mimo interval

$$\text{tzn. } a_i \in \{y \in N; 0 \le y \le 2 \wedge \max Y - 3 \le y \le \max Y - 1\},$$

pak se i-tý člen vynechá a platný je až jeho následovník i+1.

Pro dobrou natrénovanost sítě je potřeba, aby množina bodů na hraně měla stejnou velikost jako množina bodů mimo hranu. Pro zajištění různobarevnosti vybraných bodů mimo hranu, je i zde zvolena číselná posloupnost s náhodně voleným krokem v rozmezí 50–100 pixelů. Každý pixel v řádku je podroben testu, zda náleží hraně nebo zda je beze zbytku dělitelný zvoleným krokem. Samozřejmě vyšší prioritu kontroly má náležitost k hraně. Výběr z obrázku je ukončen tehdy až je dostatečný počet prvků v obou množinách, případné přebývající prvky jsou odstraněny.

Dalším důležitým úkolem fáze je zápis vybraných množin do textového souboru test.data. Tento soubor musí být vytvořen v souladu s pravidly danými knihovnou floatfann.h pro funkci *fann_train_on_file(…)*. První řádek souboru obsahuje:

„počet bodů v množině“ *mezera* „počet vstupů“ *mezera* „počet výstupů“

V případě vybírání stoprvkové množiny z každého obrázku při 50 obrázcích (počet bodů ve výsledné množině násobeno konstantou 16), to je zápis:

„80000 25 1“

Následuje zápis jednotlivých vstupů a očekávaných výstupů. Je prováděn následujícím způsobem (uváděno po řádcích):

- vstupy (např. 0.262745 0.286275 0.239216 0.278431 0.239216 0.294118 0.223529 0.278431 0.262745 0.294118 0.192157 0.231373 0.294118 0.215686 0.32549 0.992157 0.945098 0.976471 0.968627 0.992157 0.992157 0.976471 1 0.992157 1);
- výstupy (např. 1);
- vstupy (např. 0.898039 0.976471 0.952941 0.968627 1 0.992157 1 1 0.992157 1 1 1 0.952941 0.960784 1 0.992157 0.968627 0.929412 0.992157 0.952941 0.952941 0.992157 0.976471 1 0.992157);
- výstupy (např. 0).

Pro komplexnější natrénování jsem zvolila metodu otočení matice okolí, tím se získá natrénovanost sítě na více sklonů. Totéž lze udělat s inverzní maticí. To způsobí, že síť reaguje i na hranu s opačnou orientací a zase ve všech směrech. To máme čtyři otočení normální matice a čtyři otočení inverzní matice, tím jsem z jedné hrany získala osm různých hran. Jak jsem se již zmínila, proto, aby bylo možné síť dobře natrénovat, je potřeba vzít stejné množství i bodů mimo hranu. Protože prosté opakování vstupů by mohlo trénování sítě degradovat, zvolila jsem stejný postup a matici okolí pixelu mimo hranu jsem též otočila a invertovala.

## 4.5 Trénování sítě

Trénování sítě je fáze, ve které dochází k aplikování trénovací množiny na neuronovou síť určité topologie. Při vytvoření neuronové sítě jsou použity stejné váhy u všech spojů neuronů. Využitím trénovacího algoritmu a aplikací trénovací množiny je síť schopna změnit váhy spojů tak, aby síť dávala chtěný výsledek. Tím, že je modul samostatný, dává možnost oddělit trénování sítě od vlastního použití a zkrátit tak dobu potřebnou k detekci hran na minimum, pokud máme připravenou natrénovanou síť.

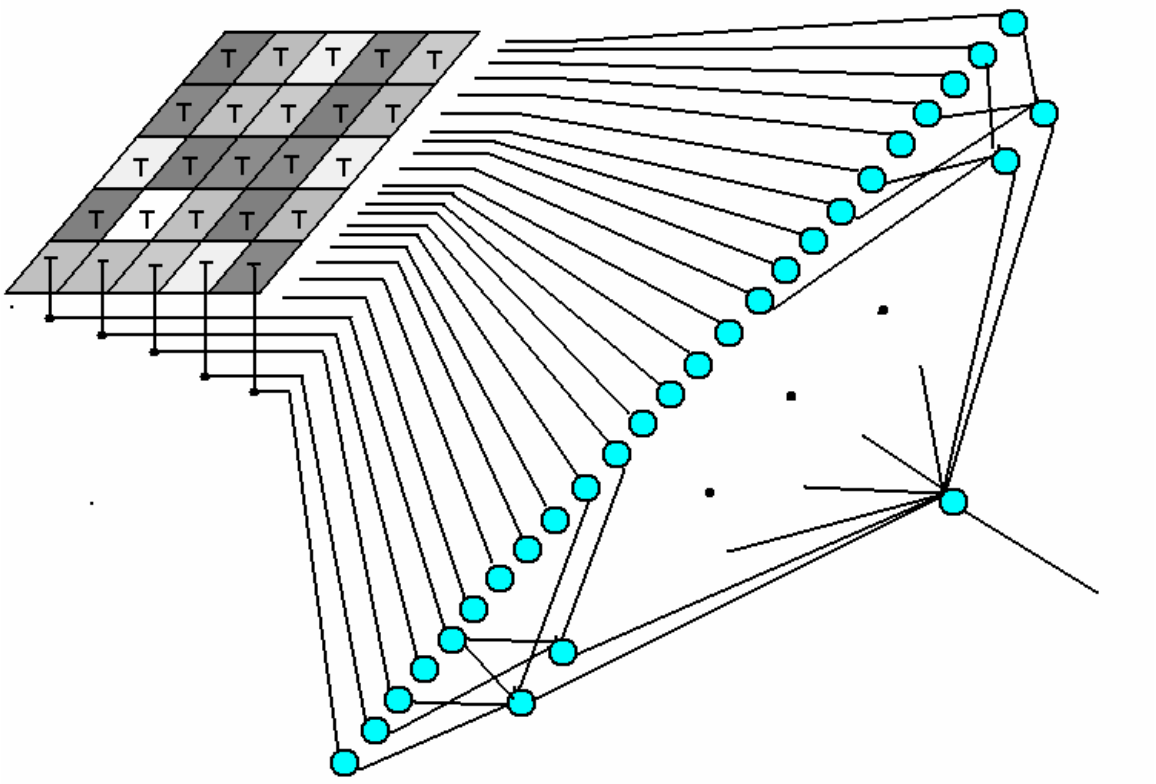
### 4.5.1 Návrh neuronové sítě

Abychom mohli tvořit trénovací množinu pro neuronovou síť, měli bychom znát topologii neuronové sítě. Při návrhu neuronové sítě hraje zásadní roli volba velikosti okolí pixelu. V našem případě se jedná o okolí velikosti 5x5, tudíž obsahuje 25 pixelů. Pro svoji práci jsem si zvolila vícevrstvou dopřednou síť. V literatuře se často uvádí jako výchozí bod pro všechny typy úloh a až v případě nedostatečné aproximace je doporučováno volit jinou topologii sítě. Aby byla síť schopna zpracovat úlohu detekce hran, musíme zvážit, jakým stylem zvolíme hodnoty vstupu. Nabízí se nám dvě varianty:

- vstupem je hodnota jednotlivé barevné složky pixelu;
- vstupem je hodnota intenzity pixelu.

Každá varianta má své specifikum.

První varianta je závislá na použitém barevném modelu. Při každé změně by se musela měnit i topologie sítě. Každý barevný model má různý počet složek, ze kterých se vypočítá výsledná barva. Pak by musela neuronová síť aproximovat i funkci pro výpočet intenzity. Došlo by k prodloužení času potřebného k natrénování sítě. Množství vstupů sítě je v tomto případě závislé na počtu jednotlivých složek barevného modelu a na počtu pixelů ve zvoleném okolí.



**Obrázek 28:** *Schéma propojení okolí pixelu s neuronovou sítí (varianta 2)*

Druhá varianta je mnohem více odolnější vůči změně barevného modelu. Intenzita pixelu se většinou udává v procentech. Vzorec pro výpočet intenzity není z hlediska výpočetní jednotky složitý. Jedná se o součet podílu každé složky na výsledné intenzitě. Například pro model RGB má rovnice tvar: $I = 0{,}299 \cdot R + 0{,}557 \cdot G + 0{,}144 \cdot B$. Co se týká počtu vstupů do neuronové sítě, má druhá varianta pouze tolik vstupů, kolik je pixelů v okolí.

Dále je třeba si zvolit výstup neuronové sítě. To by mělo odpovídat počtu kategorií, do kterých chceme jednotlivé okolí zařadit. Neuronová síť je schopna vstup přiřadit tolika kategoriím, kolik je $2^{pocetVstupu}$. Pro detekci hran se nabízí kategorizace do následujících skupin:

- okolí obsahuje hranu,
- okolí obsahuje plochu.

Řešením jednoduché exponenciální rovnice nám vyjde, že postačí pouze 1 výstup, abychom byli schopni tuto úlohu efektivně řešit.

Po určení vstupní a výstupní vrstvy vícevrstvé neuronové sítě zbývá vyřešit, kolik skrytých vrstev by měla obsahovat a kolik neuronů by mělo být v každé z nich. Tady je to už s přesným určením horší. Po vzoru využití perceptronu (jednovrstvá neuronová síť) na rozpoznání znaků, jsem se rozhodla pro jednu skrytou vrstvu a stanovení počtu neuronů bude předmětem experimentovaní.

Naše neuronová síť bude obsahovat 25 vstupních neuronů, 1 výstupní neuron a 1 skrytou vrstvu o neznámém počtu neuronů.

### 4.5.2 Trénování sítě

Pro každý typ neuronové sítě je zvolen jiný algoritmus učení. Pro mnou zvoleným typ sítě „dopředná síť“ je uváděn algoritmus zpětného šíření (viz 1.4).

### 4.5.3 Návrh aplikace

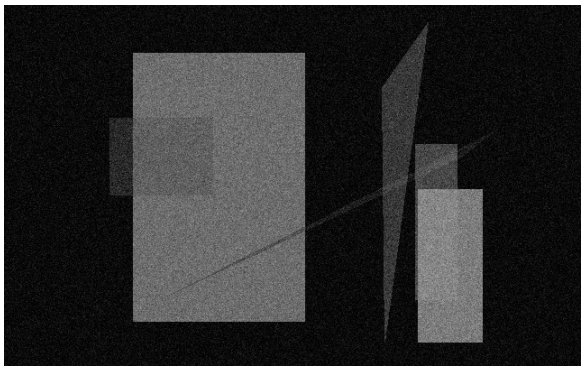
V aplikaci není potřeba řešit složitější algoritmy. Jedná se pouze o využití možností, jež nabízí knihovna floatfann.h. Dle návodu v tutoriálu knihovny je potřeba navolit topologii sítě a spustit její trénování ze souboru test.data. Poté je síť uložena do souboru „hrana.net“.

## 4.6 Úprava obrázku a vlastní detekce hran

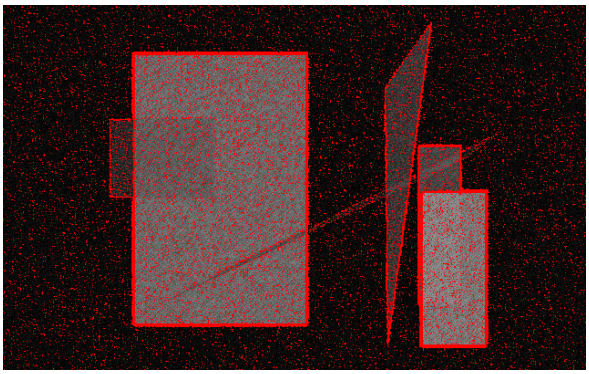
Jde o fázi, kdy se zúročí výsledky z fází minulých. Máme natrénovanou síť a nyní jde už jen o její použití, jak bylo nastíněno v předešlých kapitolách či podkapitolách. Existují techniky, jak vylepšit vlastnosti obrázku pro účely detekce.

### 4.6.1 Úprava obrázků

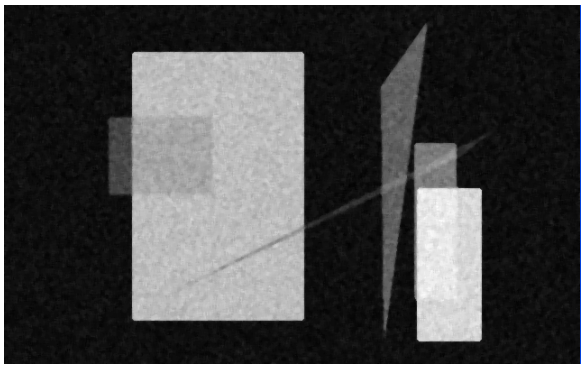
Protože ne vždy se natrénování sítě vydaří nebo je potřeba detekovat jiný soubor obrázků než pro který byla síť natrénována, lze vylepšit kvalitu detekce pomocí filtrů. Například může být použit mediánový filtr, který dokáže zvýraznit hrany v obrázku. Jeho vlastnosti byly popsány v kapitole 2.2.3. Avšak vyhlazením obrázku by mohlo dojít k odstranění přirozeného šumu daného gaussovým rozložením, na který je síť naučena. Proto by bylo vhodné ještě před vlastní detekcí vrátit kousek toho „přirozeného“ šumu pomocí Gaussova filtru. Samozřejmě není vyloučeno použití filtrů podobných vlastností.



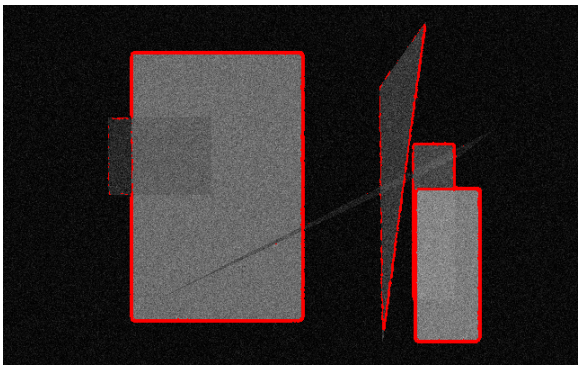
**Obrázek 29:** *Vygenerovaný obrázek*



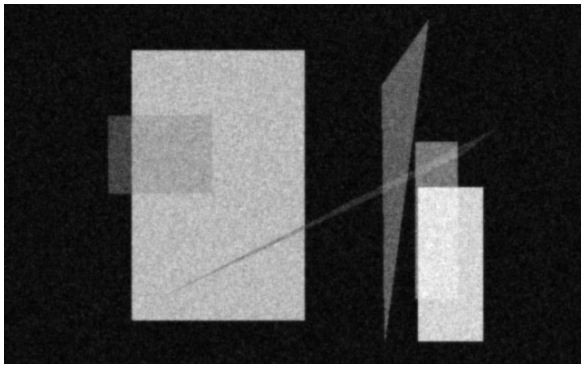
**Obrázek 30:** *Detekce hran (75,28 %)*



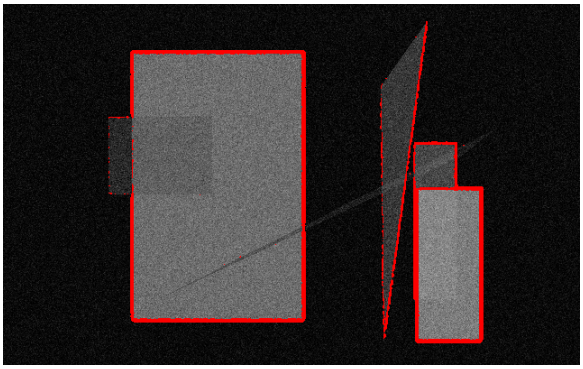
**Obrázek 31:** *Mediánový filtr o matici 5x5*



**Obrázek 32:** *Detekce hran s Mediánovým filtrem (63 %)*



**Obrázek 33:** *Gaussův filtr o rozptylu 1,0*



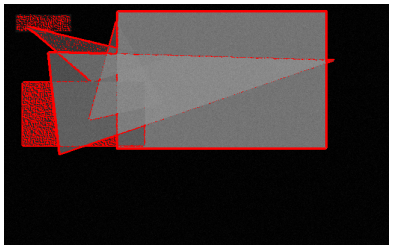
**Obrázek 34:** *Detekce hran s Gaussovým filtrem (62,8 %)*

### 4.6.2 Vlastní detekce hran

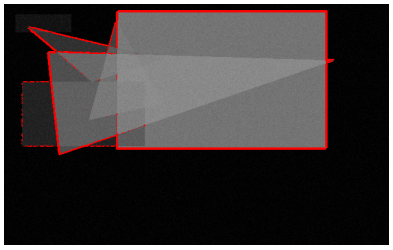
Natrénovaná neuronová síť bude použita na celý obrázek, ve kterém mají být hrany detekovány. Obrázek bude testován po jednotlivých pixelech. Na vstupy sítě vždy bude přivedeno okolí aktuálního pixelu a pokud bude stanoveno, že jde o hranu, pixel změní svoji barvu na barvu označující hranu, jinak se použije barva označující plochu (např. pixel si ponechá svoji hodnotu).

Neuronová síť má na výstupu rozsah hodnot v určitém intervalu. Ten je daný oborem hodnot aktivační funkce. Hranice intervalu říkají, zda je prvek na hraně nebo mimo hranu. Bohužel většinou se síť nenatrénuje dostatečně přesně, aby na výstupu byly pouze hodnoty hranic intervalu. Je třeba zavést tzv. prahování (viz kapitola 3.3). Jeho účelem je nalézt hodnotu tvořící hranici, kde se už

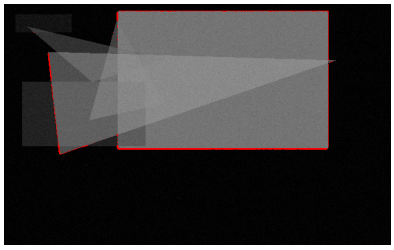
neprojeví šum. Stanovení optimálního prahu detekce hran je jeden z cílů experimentů popsaných v následujících kapitolách.



**Obrázek 35:** *Hladina 0,5*



**Obrázek 36:** *Hladina 0,7*



**Obrázek 37:** *Hladina 1,0*

### 4.6.3 Návrh aplikace

Aplikace má za úkol pouze využití neuronové sítě na zadané obrázky a jejich případnou úpravu. Opět nebylo potřeba definovat vlastní funkce. Mediánový a Gaussův filtr jsou implementovány v knihovně CImg.h a o práci s neuronovou sítí se postarají funkce z knihovny floatfann.h. Jde jen o správné seřazení těchto funkcí.

Nejdříve bychom měli načíst neuronovou síť, kterou jsme již natrénovali v předchozí fázi. Tuto síť beze změny použijeme na všechny obrázky v daném souboru. Budeme postupně načítat jeden obrázek za druhým a pokud bude požadována úprava, použijeme určený filtr. Ještě než začneme s detekcí, je potřeba změnit barvu obrázku na stupně šedi, protože hrana je funkcí intenzity. Potom procházíme obrázek pixel po pixelu a dané okolí uložíme v požadovaném formátu do vstupního pole sítě. Funkce pro běh neuronové sítě nám vrátí požadované číslo podle daného intervalu aktivační funkce neuronu. Uživatel musí určit, zda je toto číslo (práh) dostatečně malé pro znázornění všech hran a dostatečně velké pro odfiltrování šumu. Zvolení optimální hodnoty prahu je součástí experimentování.

Postupně jsou do dalšího obrázku uloženy pixely ležící na hraně. Jako pozadí se dá zvolit černá barva nebo vlastní obrázek. Vše je uloženo do nového obrázku vysl*.bmp a pokračuje se dalším obrázkem.

## 4.7 Vyhodnocení výsledků

Kapitola má vnést do detekce hran míru objektivnosti. Zvolená úloha, jejíž řešením se zabývám, je do jisté míry subjektivní. Jak jsem se už v úvodu zmínila, každý živý organismus si sám detekuje obraz a protože každý člověk je svým způsobem originál (např.: každý máme jinou sítnici oka, existují oční vady zkreslující vidění), pak lze chápat, že může být u jedinců trochu pozměněna citlivost na hranu. Myslím si, že i na většinu syntetických obrázků připravených mým programem by měli různí lidé různou představu o tom, kde hrana je. Pro účely tohoto projektu je však důležité stanovit měření úspěšnosti podle objektivních skutečností. Počítač nemůže do dat vnést sémantiku, vždy je nutno udat šablonu chování, tím je v našem případě označení hrany v pomocné struktuře u syntetických obrázků.

Pokud už máme pevně definováno, co bereme za hranu a co ne (pomocná struktura u syntetických obrázků), je třeba určit metodu měření úspěšnosti detekce hran. Dá se říci, že pro nás mají význam dvě metriky:

- Metrika 1 - kolik procent hran bylo detekováno;
- Metrika 2 - kolik procent z detekovaných hran je opravdu hranou (šum – působí rušivě).

Obě metriky mají neocenitelnou výpovědní hodnotu. Z prvního odstavce lze zdůvodnit tvrzení, že u první metriky můžeme být benevolentnější než u metriky druhé, kde budeme vyžadovat přesnost 99–100 %.

### 4.7.1 Návrh aplikace

Tato část je důležitá pro určení kvality výsledku. Jejím základním úkolem je vypočítat zadané metriky pro každý obrázek. K vypočítání metriky potřebujeme jak vlastní obrázek, tak jeho šablonu. Pro posouzení úspěšnosti si definujme dvě metriky.

Metrika 1 určuje, kolik procent skutečných hran odhalila detekce. Je mírou úspěšnosti odhalení skutečných hran. Algoritmus metriky porovnává označené body z obrázku s body ze šablony. Shodují-li se, inkrementuje se počítadlo. Po každém označeném pixelu v šabloně se inkrementuje celkový počet bodů. V závěru se vydělí počítadlo s celkovým počtem bodů a získáme procentuální zastoupení. Může nastat situace, kdy síť označí všechny pixely v obrázku (dojde k zašumění). Potom bude úspěšnost vyhodnocena jako 100%, i když nebyla označena pouze hrana. Pro rozlišení této situace zavádím ještě druhou metriku.

Metrika 2 určuje, kolik procent z detekovaných hran je reálných. Je to míra šumu detekce. Algoritmus této metriky porovnává označené body ze šablony s okolím těchto bodů v obrázku. Je-li v okolí nalezena červená barva, je inkrementováno počítadlo. Po podělení celkovým počtem detekovaných hranových pixelů, získáme procentuální zastoupení správně detekovaných hran. Doplněk do 100 % určuje, kolikaprocentní je šum detekce.

# 5 Implementace softwaru

Kapitola by měla posloužit jako manuál k použití jednotlivých aplikací. Naleznete tu návod na obsluhu softwaru.

Pro implementaci jsem si zvolila programovací jazyk C/C++ a jako vývojové prostředí Microsoft Visual C++ 2008 Express Studio. Program je primárně určen pro operační systém Windows XP, ve kterém byl vyvíjen a testován.

Software obsahuje 5 modulů (generátor.exe, test_mnozina.exe, trenovani.exe, detekce.exe, metrika.exe), které odpovídají fázím naznačeným v kapitole 4. Modulové řešení má výhodu při spouštění dávkových souboru (*.bat). Dá se také jednoduše spojit nějakým vizuálním prostředím, které zatím není implementováno. Jednotlivé moduly představím v následujících kapitolách.

## 5.1 Knihovny

Kromě standardních knihoven jazyka C++ jako jsou string, vector, iostream, fstream, apod., jsem pro implementaci zvolila další dvě knihovny pro práci s grafickými objekty a neuronovou sítí. Také jsem si vytvořila vlastní knihovnu, která do jisté míry zaručuje kompatibilitu jednotlivých modulů. Ráda bych se zmínila o jednotlivých knihovnách a k čemu jsou využity.

### 5.1.1 CImg.h

Jedná se o volně dostupnou C++ grafickou knihovnu. Obsahuje jeden hlavičkový soubor s otevřeným zdrojovým kódem. Poskytuje široké zázemí pro přístup, práci a filtraci obrázků. Je přenositelná na různé typy operačních systémů a podporuje řadu grafických formátů.

### 5.1.2 Floatfann.h

Tato knihovna mi pomáhá implementovat neuronovou síti. Dokáže vytvořit dopřednou neuronovou síť libovolné topologie a pomocí algoritmu zpětného šíření ji natrénuje. Pro uložení topologie sítě a trénovací množiny využívá vlastní formáty textového souboru. Pro reprezentaci vah a šířeného signálu pomocí spojů mezi neurony používá datového typu float. Nabízí spoustu funkcí pro nastavení a práci s neuronovou sítí. Hlavní využití této knihovny je zvláště v modulech trenovani.exe a detekce.exe.

### 5.1.3 Bakalarka.h

Tvoří zázemí pro kompatibilitu jednotlivých modulů. Má za úkol ohlídat, aby při spuštění v dávkovém souboru probíhalo vše v pořádku a došlo ke sjednocení názvů výstupu.

## 5.2 Modul 1

Modul slouží především pro generování syntetických obrázků ve stupni šedi. Obraz je tvořen z náhodně generujících obrazců (trojúhelníků a čtverců) vyplněných náhodně volených odstínech šedi. Obrazce mají nastavenou průhlednost na 50 %. Vzniká tak daleko větší spektrum hran.

### 5.2.1 Návod na použití modulu

Modul je konzolová aplikace, která přijímá parametry přes příkazovou řádku. Tabulka 2 ukazuje, jak se dá ovlivnit běh programu.

**Tabulka 2:** *Parametry modulu 1(generátor.exe)*

| PARAMETR | IMPLICITNÍ HODNOTA | CO PARAMETR NASTAVÍ |
|---|---|---|
| -n | 6 | Počet obrazců v generované sérii obrázků |
| -o | 50 | Počet obrázků v generované sérii |
| -g | false | Obrázky v sérii jsou zašuměny podle gaussovského rozložení |
| -m | 0 | Střední hodnota pro gaussovské rozložení |
| -s | 5 | Rozptyl pro gaussovké rozložení |
| -i | false | Ke každému rozložení objektů je generován i obrázek s inverzním zbarvením (doplňkem do 255) |
| -h nebo -help | - | Výpis parametrů |

Aplikace generuje sérii obrázků podle výše zmíněných parametrů. Názvy obrázků jsou ovlivněny obsahem proměnných n_obr a n_sabl v knihovně bakalarka.h. Implicitní nastavení obsahu těchto proměnné n_obr je „obr“ (pro obrázek ve stupních šedi) a pro n_sabl je „sabl“ (pro obrázek obsahující pouze hrany). Názvy obrázků jsou vygenerovány následujícím způsobem:

n_obr + pořadí + prip nebo n_sabl + pořadí + prip

např. obr10.bmp nebo sabl10.bmp

Šablona a obrázek patří k sobě, obsahuje-li stejné pořadí. Pořadí je řada čísel 0, 1, 2, 3, …, (počet obrázků v sérii – 1).

## 5.3 Modul 2

Slouží pro výběr trénovací množiny. Do souboru test.data zapisuje okolí vybraných pixelů společně s rozhodnutím, zda toto okolí obsahuje hranu nebo nikoliv. Jako vstup slouží syntetické obrázky z modulu 1. Jsou potřeba jak obrázky obr*.bmp, tak sabl*.bmp.

### 5.3.1 Návod na použití modulu

Modul je konsolová aplikace, která přijímá parametry přes příkazovou řádku. Tabulka 3 ukazuje, jak se dá ovlivnit běh programu.

**Tabulka 3:** *Parametry modulu 2 (test-mnozina.exe)*

| PARAMETR | IMPLICITNÍ HODNOTA | CO PARAMETR NASTAVÍ |
|---|---|---|
| -m | 100 | Počet bodů na hraně vybraných z jednoho obrázku |
| -o | 50 | Počet obrázků, ze kterých se má množina vytvořit |
| -s | false | Hodnoty jsou uváděny v intervalu <-1;1> (implicitně int.<0;1>) |
| -od | 177 | Řádek obrázku, od kterého se začne výběr bodů |

| | | |
|---|---|---|
| -po | 59 | Určí krok výběru řádku |
| -h nebo -help | - | Výpis parametrů |

Aplikace vybírá trénovací body po řádcích. Najde všechny hrany na řádku a z nich pak vybere výslednou množinu. Pro vizuální kontrolu výběru testovacích množin je každý obrázek uložen pod názvem test*.bmp (kde * určuje číslo obrázku), ve kterých jsou červeně označeny body, které byly vybrány.

Pro správný běh modulu je potřeba zajistit, aby obrázky, ze kterých se provádí výběr množiny, byly pojmenovány od obr0.bmp, obr1.bmp, …, obr49.bmp (pro implicitní nastavení parametrů).

# 5.4 Modul 3

Modul složí pro vytvoření a natrénování neuronové sítě. Vyžaduje v místě spuštění soubor test.data (se vstupy a výstupy v daném intervalu podle zvolené aktivační funkce). Výstupem modulu je natrénovaná neuronová síť uložená v souboru hrana.net.

## 5.4.1 Návod na použití modulu

Modul je konsolová aplikace, která přijímá parametry přes příkazovou řádku. Tabulka 4 ukazuje, jak se dá ovlivnit běh programu.

**Tabulka 4:** *Parametry modulu 3 (trenovani.exe)*

| **PARAMETR** | **IMPLICITNÍ HODNOTA** | **CO PARAMETR NASTAVÍ** |
|---|---|---|
| -n | 10 | Počet bodů v prostřední vrstvě |
| -e | 10000 | Počet epoch, po jejichž proběhnutí se trénování ukončí |
| -be | 1000 | Počet epoch, po kterých se vypíše stav trénování |
| -s | false | Přepínač mezi symetrickou a stupňovitou (implicitní) aktivační funkcí neuronu |
| -er | 0.1 | Určí, při jaké chybě sítě se trénování ukončí |
| -h nebo -help | - | Výpis parametrů |

Modul využívá třívrstvou dopřednou neuronovou síť. Lze měnit počet neuronů v prostřední vrstvě. Je však třeba dbát na to, že počet neuronů ovlivní velikost trénovací množiny. Dále se může měnit rozsah hodnot, které jsou uloženy v souboru test.data a to podle toho, jakou aktivační funkci právě používáme:

- symetrickou sigmoidní funkci: vstupy $\langle -1;1 \rangle$, výstupy $\langle -1;1 \rangle$ - rychlejší, méně přesná;

- klasickou sigmoidní funkci: vstupy $\langle 0;1 \rangle$, výstupy $\langle 0;1 \rangle$ - pomalejší, více přesná.

# 5.5 Modul 4

Jedná se o modul, jež provádí vlastní detekci hrany. Máme-li natrénovanou síť uloženou v souboru hrana.net a obrázky, ve kterých chceme detekovat hrany uloženy pod názvy obr0.bmp, obr1.bmp, atd., nebrání nám nic v jejím spuštění. Detekované hrany se objeví v obrázcích nazvaných vysl0.bmp, vysl1.bmp atd..

### 5.5.1 Návod na použití modulu

Modul je konsolová aplikace, která přijímá parametry přes příkazovou řádku. Tabulka 5 ukazuje, jak se dá ovlivnit běh programu.

**Tabulka 5:** *Parametry modulu 4 (detekce.exe)*

| PARAMETR | IMPLICITNÍ HODNOTA | CO PARAMETR NASTAVÍ |
|---|---|---|
| -o | 50 | Počet obrázků, ve kterých se bude provádět detekce |
| -c | 0.7 | Hranice pro určení hrany |
| -p | false | Přepínač určující pozadí detekovaných hran (černá x vlastní obrázek) |
| -s | false | Vstup a výstup v intervalu <-1;1> (implicitně int.<0;1>) |
| -m | 0 | Použití mediánového filtru na obrázek před detekcí (implicitně vypnuto), Hodnota okolí pro použití filtru. |
| -g | 0 | Použití gaussova filtru na obrázek před detekcí (implicitně vypnuto), Hodnota rozptylu pro použití filtru. |
| -h nebo -help | - | Výpis parametrů |

Aplikace pouze detekuje hrany s využitím neuronových sítí. Podle parametrů nabízí před detekcí úpravu obrázků tak, aby došlo k co nejlepší detekci. Aplikace použije stejné parametry na celý soubor obrázků určený k detekci.

## 5.6 Modul 5

Tento modul je určen pro kontrolu výsledků detekce hran na syntetických obrázcích, které mají k dispozici šablonu. Metrika je dvojího typu: kolik hran je detekováno (metrika 1) a nakolik je detekce správná (metrika 2).

### 5.6.1 Návod na použití modulu

Modul je konsolová aplikace, která přijímá parametry přes příkazovou řádku. Tabulka 6 ukazuje, jak se dá ovlivnit běh programu.

**Tabulka 6:** *Parametry modulu 5 (metrika.exe)*

| PARAMETR | IMPLICITNÍ HODNOTA | CO PARAMETR NASTAVÍ |
|---|---|---|
| -o | 50 | Počet obrázků, ve kterých se bude provádět detekce |
| -vm1 | false | Přepínač vypne metriku 1 |
| -vm2 | false | Přepínač vypne metriku 2 |
| -h nebo -help | - | Výpis parametrů. |

# 6 Experimentální část

V této kapitole bych ráda shrnula možnosti volitelného nastavení jednotlivých modulů. Při kterých hodnotách dosahují nejlepších výsledků, případně určit chování jednotlivých parametrů a nastínit jejich vliv na detekci hran pomocí neuronové sítě. Tak, aby uživatel, jež bude chtít vytvořený software využívat, měl představu, jak navolit parametry právě na jeho soubor obrázků, určených pro detekci.

V kapitole 4 jsem parametry, jež mohou ovlivnit výsledek sítě, rozdělila do dvou skupin. Jednu skupinu jsem optimalizovala přímo při návrhu a implementaci softwaru. Druhá skupina ponechává uživateli volnou ruku při použití jednotlivých modulů. Právě tato skupina bude předmětem experimentů.

Každý experiment bude zdokumentován na přiloženém CD/DVD.

## 6.1 Vytyčení cíle experimentální části

Jednotlivé experimenty se vztahují vždy k již zmíněným fázím detekce hran pomocí neuronové sítě. V kapitole 4 jsou pojmenovány průběhy jednotlivých fází, ze kterých můžeme snadno odvodit účely jednotlivých experimentů.

U volby trénovacích obrázků není mnoho faktorů, které mohou být ovlivněny uživatelem. Zaměřím se pouze na porovnání několika souborů trénovacích obrázků a jejich vliv při detekci na reálném a syntetickém souboru, u nichž budu detekovat hranu. Sledovanými parametry budou:

- zatížení detekce na jednu hranu;
- vliv na úspěšnost odhalení hrany.

U výběru trénovací množiny ze souboru obrázků bych se zaměřila hlavně na:

- optimální počet trénovacích bodů;
- rozdíly při výběru trénovacích bodů z většího množství obrázku s minimem bodů vybraných z jednoho obrázku nebo raději volit menší rozsah trénovacího souboru obrázku, ale s maximálním počtem vybraných bodů z jednoho obrázku;
- vliv správného rozptylu při výběru řádků.

Pokud se jedná o vlastní trénovací fázi je zde možnost experimentovat s:

- velikostí chyby trénovanosti neuronové sítě,
- volbou aktivační funkce.

Při vlastní detekci by se dal stanovit:

- vliv filtru na výsledek sítě,
- vliv volby citlivosti prahu detekce,
- volba aktivační funkce.

Při fázi měření výsledků již není s čím experimentovat. Modul pro stanovení úspěšnosti detekce by měl odvádět stabilní výkony.

## 6.2 Experiment 1: Vliv různých trénovacích souborů obrázků

Jedná se o experiment určený pro zjištění vlivu výběru trénovacích souborů obrázků na výslednou detekci a netrénovanost sítě. Experiment se bude skládat ze tří různých trénovacích souborů, jednoho syntetického a reálného souboru obrázků určených pro detekci. Trénování bude probíhat při nastavení výběru 60 trénovacích bodů z každého obrázku. Neuronová síť se bude skládat z 10 neuronů a aktivační funkce bude klasická sigmoidní. Detekce bude primárně postavena na hladině prahu 0,5; 0,6 a 0,7. Obrázek se před detekcí nebude upravovat.

Při vyhodnocování budu sledovat zatížení detekce na hrany, určitého směru a vliv na úspěšnost odhalení hrany.

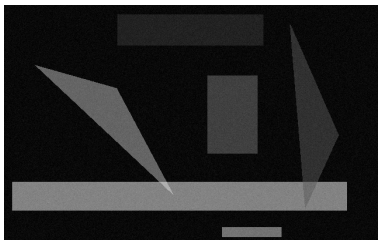
### 6.2.1 Popis charakteristik souborů obrázků

Všechny syntetické obrázky budou zašuměny Gaussovým rozložením při střední hodnotě 0 a roztylu 10.

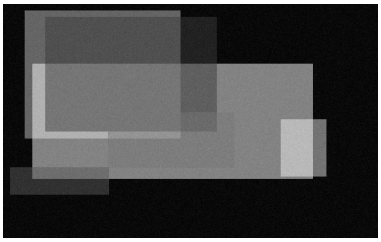
**První vstupní soubor** se bude skládat ze třiceti obrázků, ve kterých se budou objekty navzájem minimálně dotýkat. Dojde tak k nízkému výskytu různých hran v obrázku.

**Druhý vstupní soubor** se bude skládat ze třiceti obrázků, ve kterých budou hlavně objekty jednoho typu (čtverec).

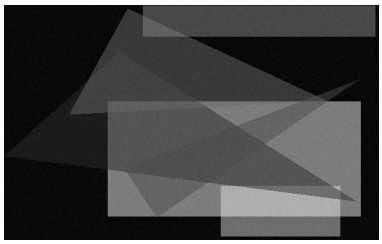
**Třetí vstupní soubor** bude o velikosti třiceti obrázků, jež běžně modul generuje.



**Obrázek 38:** *Obrázek 1. souboru*



**Obrázek 39:** *Obrázek 2. souboru*



**Obrázek 40:** *Obrázek 3. souboru*

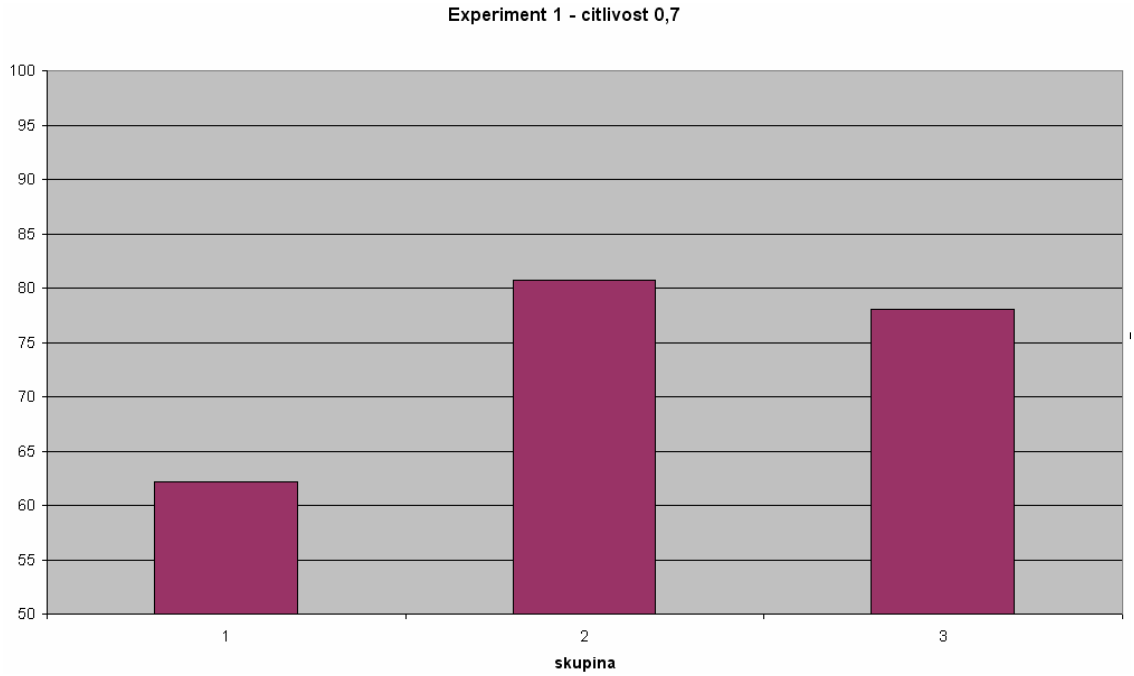
**Syntetický soubor pro detekci** bude vytvořen obdobně jako „třetí vstupní soubor“. Avšak bude vytvořeno při jiném spuštění příslušného modulu. Počet takto vygenerovaných obrázků bude omezen na 10.

**Reálný soubor pro detekci** bude obsahovat různé hrany z fotografií a 3 obrázky budov.

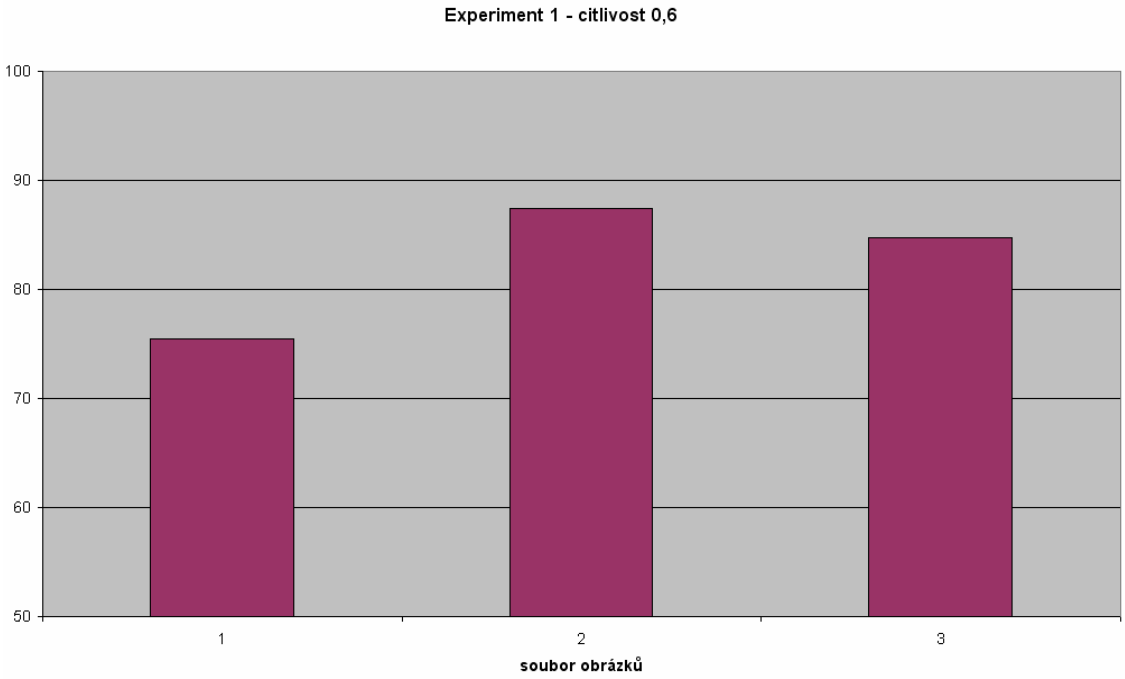
### 6.2.2 Postup experimentu

Sestavila jsem si dávkový soubor pro operační systém Microsoft Windows XP. Nejdříve se provede detekce hran pro syntetický a reálný soubor za pomoci využití prvního vstupního souboru, podruhé se provede pro druhý vstupní soubor a nakonec pro třetí. U množiny syntetických obrázků se změří kvalita výstupu a výsledky budou zakresleny do grafu. Pro srovnání uvedu z reálné množiny pár obrázků.

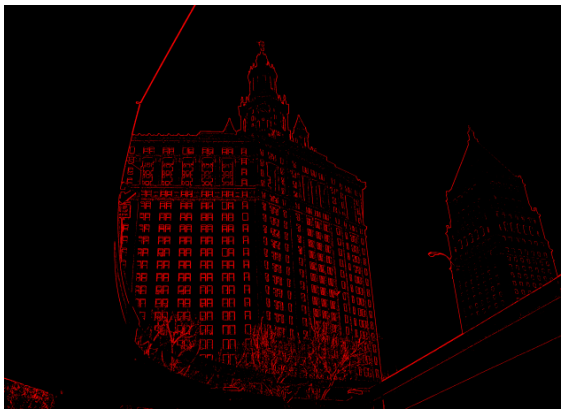
### 6.2.3 Výsledky experimentu



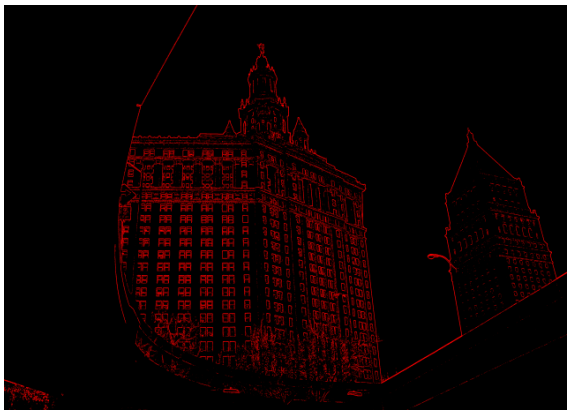

**Graf 1**



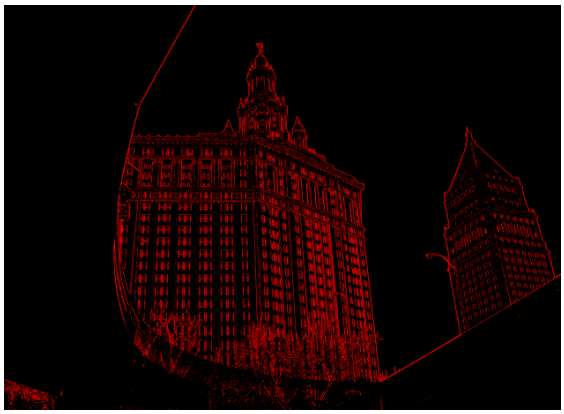

**Graf 2**



**Obrázek 41:** *Skupina 1*



**Obrázek 42:** *Skupina 2*



**Obrázek 43:** *Skupina 3*

### 6.2.4 Závěr experimentu

Nejlepší detekce hrany s minimálním šumem u všech tří souborů obrázků má práh 0,6. Při hladině 0,5 dochází u třetího souboru až k 40% šumu. Naproti tomu hladina 0,7 má sice nulový šum, ale také detekce hrany je až o 5-13 % horší. Projevilo se to zejména u 1. souboru, kde není dostatečné zastoupení typů hran. Nejvyšší trénovanosti dosahuje druhá skupina.

Při větším počtu testů nad syntetickou množinou by se možná dalo usoudit, že na tvaru hrany v trénovacích obrázcích nezáleží, důležité je postihnutí různých orientací a velikostí hrany. Ovšem na reálném obrázku vidíme, že nejlepší detekce je při natrénování třetí skupinou. Je postihnuto více detailů. U ostatních skupin je vždy vynechán určitý typ hrany.

## 6.3 Experiment 2: Vliv množství trénovacích bodů na úspěšnost detekce hran

Trénovací body mohou být z hlediska uživatele vybírány dvěma způsoby. Dávat přednost:

- obecnosti před specifičností: velká množina trénovacích obrázků a z každého obrázku je vybrán malý počet bodů;
- specifičnosti před obecností: malá množina trénovacích obrázků a z každého obrázku je vybran velký počet bodů – větší přizpůsobení určitým obrázkům.

V tomto experimentu bych ráda nastínila chování sítě v obou případech.

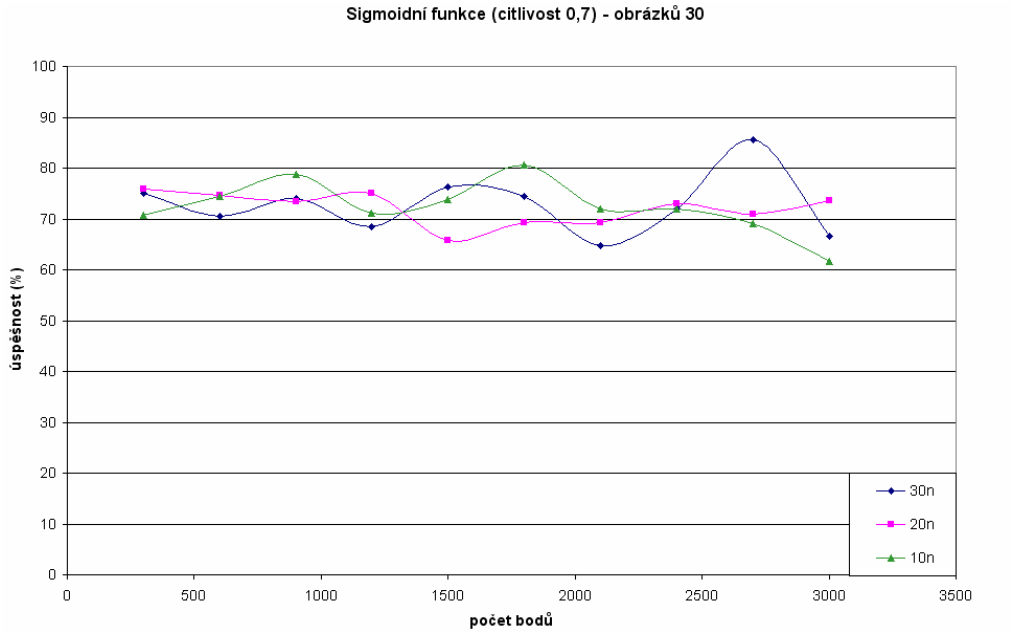
### 6.3.1 Postup experimentu

Pro neuronové sítě s 10, 20 a 30 neurony ve skryté vrstvě budu volit trénovací množinu tak, že:

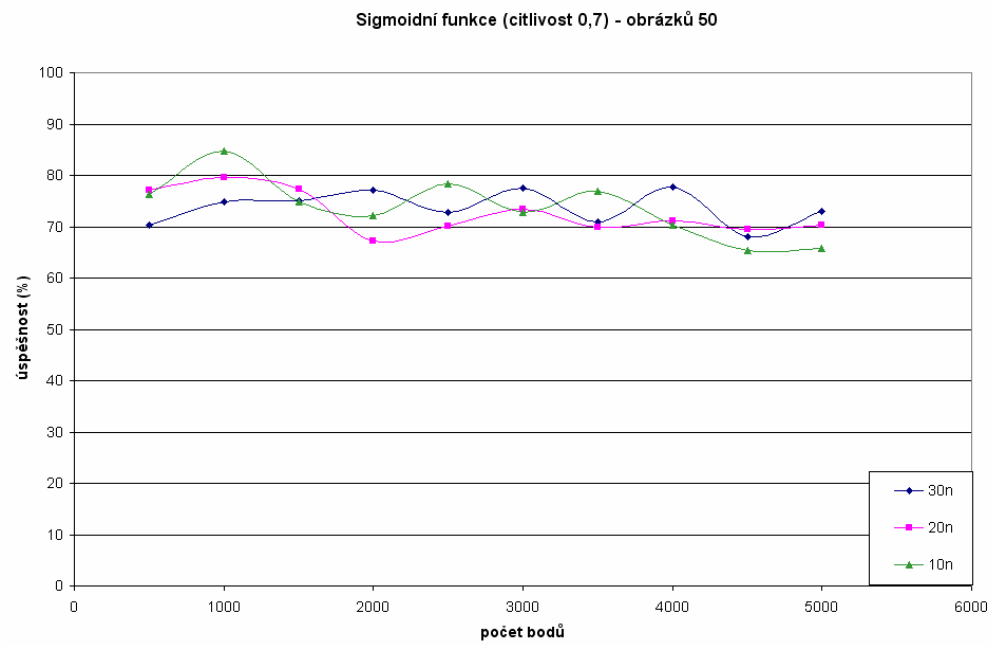
- počet vybíraných bodů z obrázků bude stabilní, měnit se bude počet obrázků;
- počet obrázků bude stabilní, měnit se bude počet vybíraných bodů z obrázků.

Vždy ten variabilní parametr bude volen z množiny $\{10,20,...,100\}$ a stabilní parametr bude vybírán z množiny $\{30,50\}$. Výsledky budou vhodně zaznamenány do grafů. Prahování bude probíhat na hladině 0,7. U některých vzorků může nastat šum v detekci hran. Na to však při tak velkém počtu trénování nebudu brát ohled.

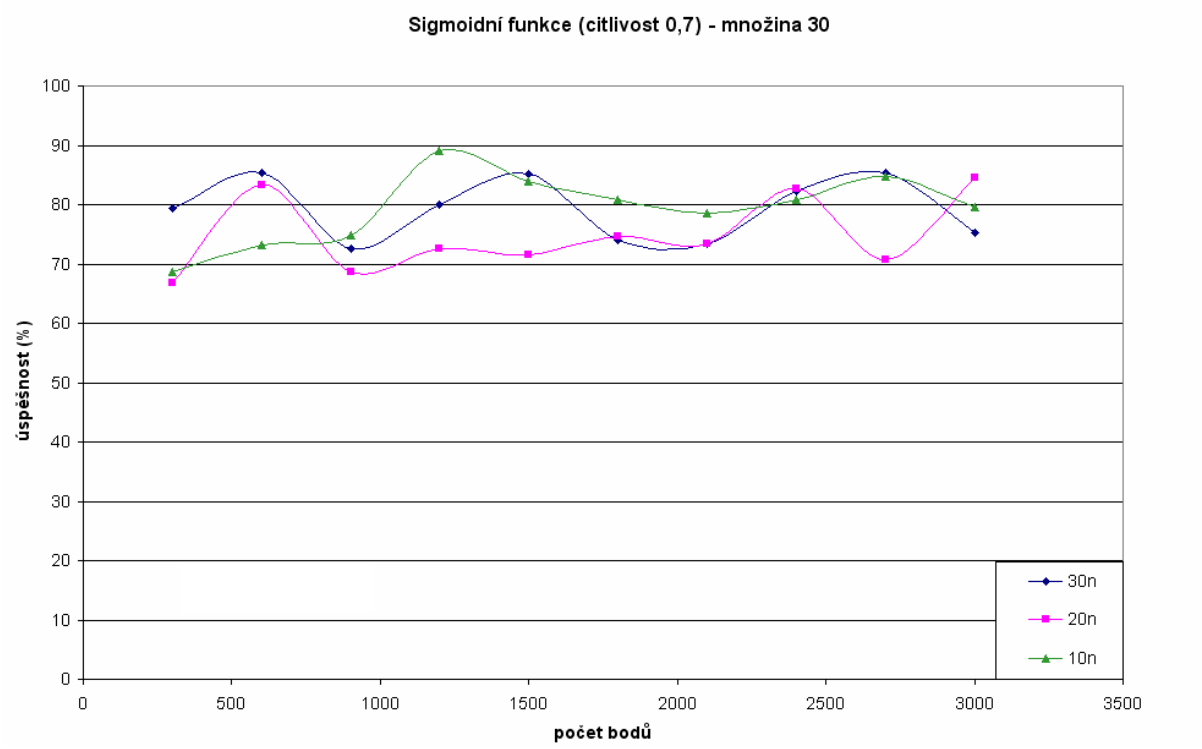
### 6.3.2 Výsledky experimentu



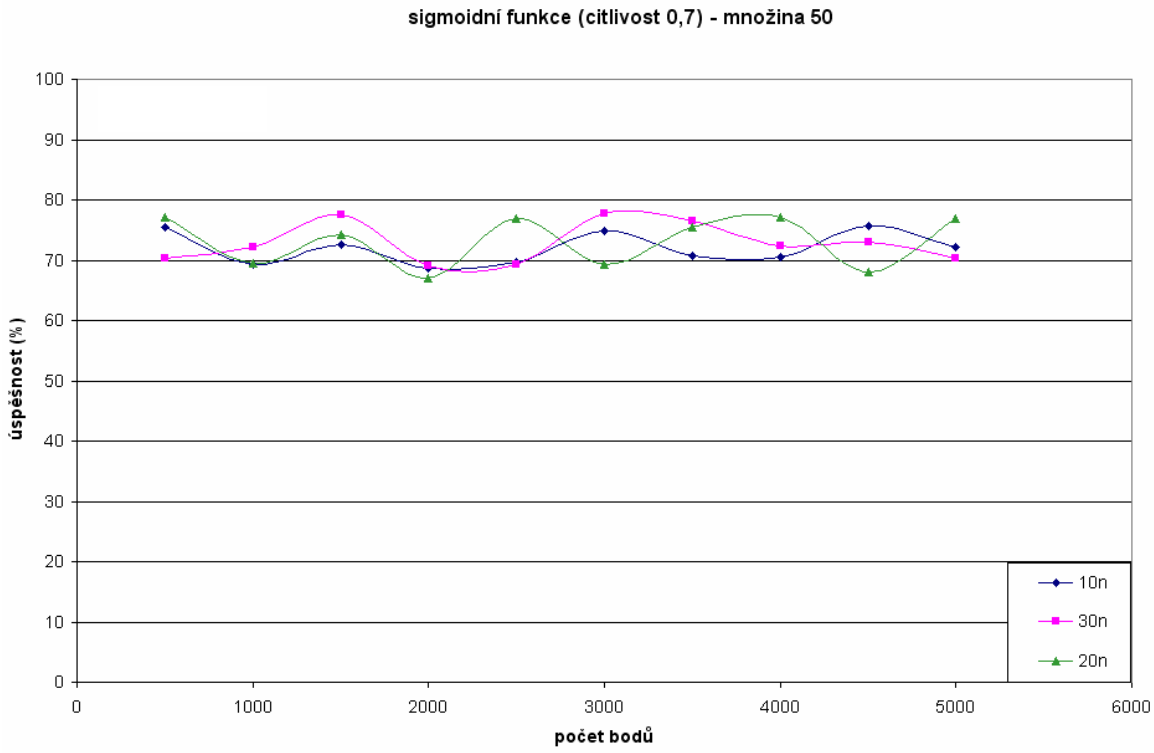

**Graf 3**




**Graf 4**


Sigmoidní funkce (citlivost 0,7) - množina 30
úspěšnost (%)
počet bodů
30n
20n
10n

**Graf 5**


sigmoidní funkce (citlivost 0,7) - množina 50
úspěšnost (%)
počet bodů
10n
30n
20n

**Graf 6**

### 6.3.3 Závěr experimentu

Z grafů je patrné, že úspěšnost při rozpoznávání hran na syntetické množině náhodně zvolených 10 obrázků je ustálena v rozmezí 70-80 %. Ovšem jednotlivé průběhy úspěšnosti v rámci neuronové sítě s příslušným počtem skrytých neuronů nabývají různých tvarů. Grafy naznačují, že i když jsou stabilní počty množin nebo počty obrázků, dochází k podobným jevům.

Při menším počtu stabilních prvků je vždy větší variabilita úspěšnosti. Se zvyšujícím se počtem stabilních prvků dochází k rovnoměrnosti průběhu funkcí. V grafu 4 a grafu 3 lze u neuronových sítí s počtem skrytých neuronů 10 a 30 pozorovat jistou periodicitu. Při každém novém natrénování dochází ke zlepšení či zhoršení schopnosti detekovat hrany. Zatímco v grafu 3 se s rostoucími prvky množiny vybraných bodů odchylka zvyšuje, u grafu 4 zůstává odchylka rovnoměrná, zvláště pro velikost množiny vybraných bodů v rozmezí 30 až 80 bodech.

Neuronová síť s počtem skrytých neuronů 20 se většinou v určitém intervalu počtu trénovacích bodů ustálí na hladině podobné úspěšnosti. Většinou se jedná o dolní hranici průběhu. Ustálení pro 20 skrytých neuronů proběhne:

- na hladině kolem 73 % úspěšnosti pro množinu 30 bodů ze 40 až 70 obrázků,
- na hladině kolem 70 % úspěšnosti pro množinu 50 až 100 bodů z 50 obrázků,
- na hladině kolem 75 % úspěšnosti pro množinu 10 až 40 bodů ze 30 obrázků,
- na hladině kolem 70 % úspěšnosti pro množinu 60 až 90 bodů ze 30 obrázků.

Neuronová síť se skrytými 30 neurony si zachovává svoji periodicitu až na případ, ve kterém se hodnota ustálí na 72 % úspěšnosti a to při množině 50 bodů ze 70 až 100 obrázků.

Taktéž neuronová síť s 10 skrytými neurony si uchovává svoji periodicitu až na případ, při kterém dojde k ustálení na hladině 80 % úspěšnosti. Jedná se o volbu trénovací množiny tvořenou 30 vybranými body ze 60 až 80 obrázků.

## 6.4 Experiment 3: Volba aktivační funkce

V aplikacích jsou implementovány dva druhy sigmoidní funkce:

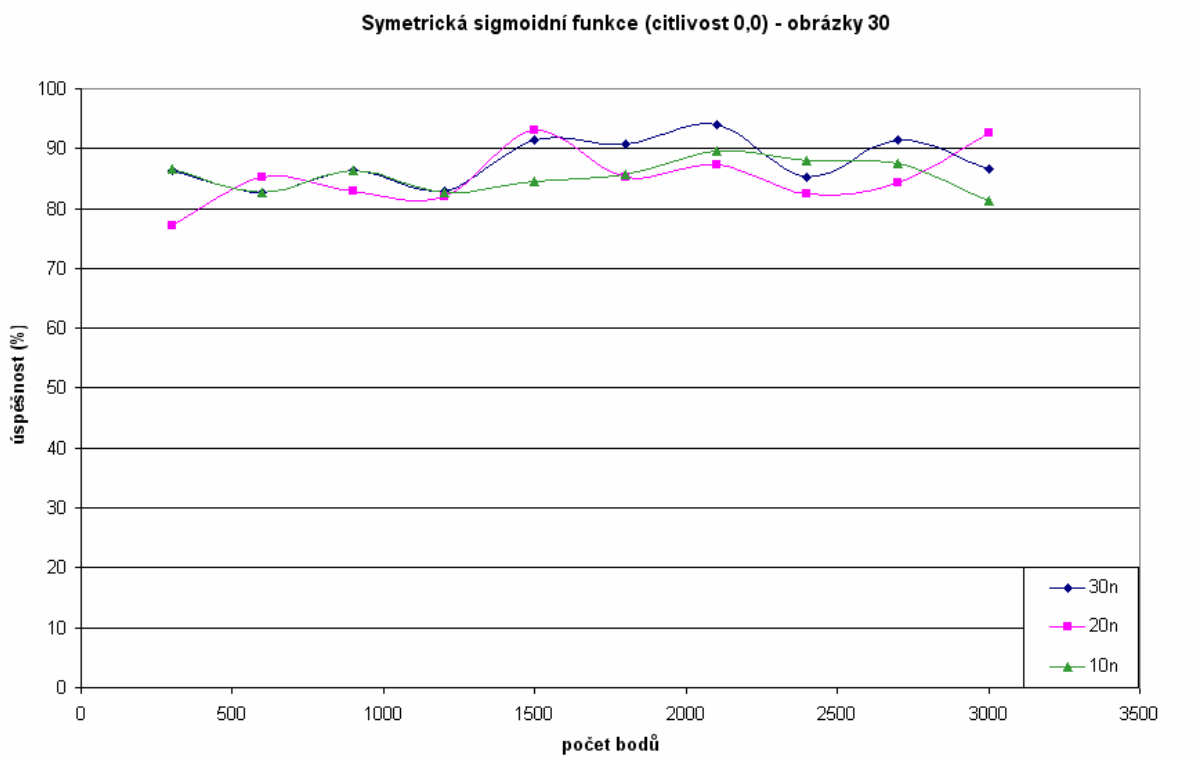
- klasická,
- symetrická.

Úkolem experimentu je zjistit, jak se od sebe tyto funkce liší. Proto bude ještě jednou vykonán Experiment 3, akorát s jinou aktivační funkcí.

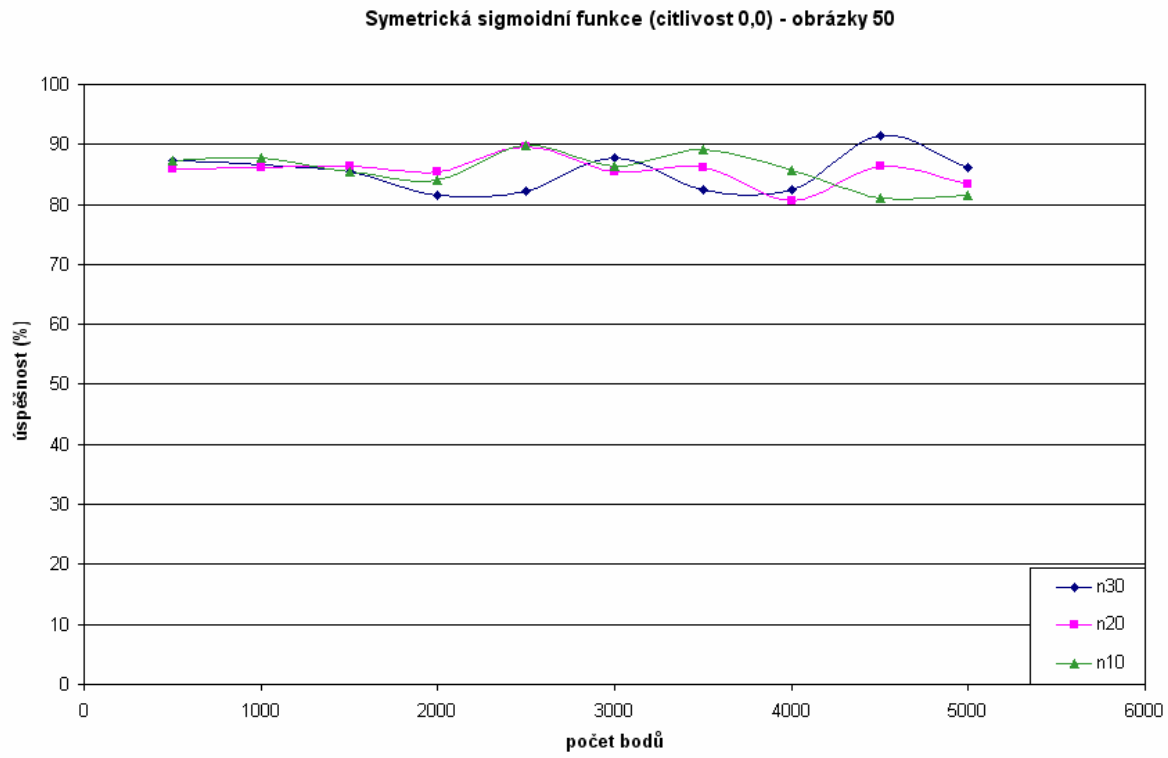
### 6.4.1 Postup experimentu

Zopakuji experiment č. 2, ale tentokrát využijí symetrickou variantu funkce. Zapíši výsledky do grafů. Prahování bude probíhat na hladině 0,0. Zase se může objevit šum při detekci.

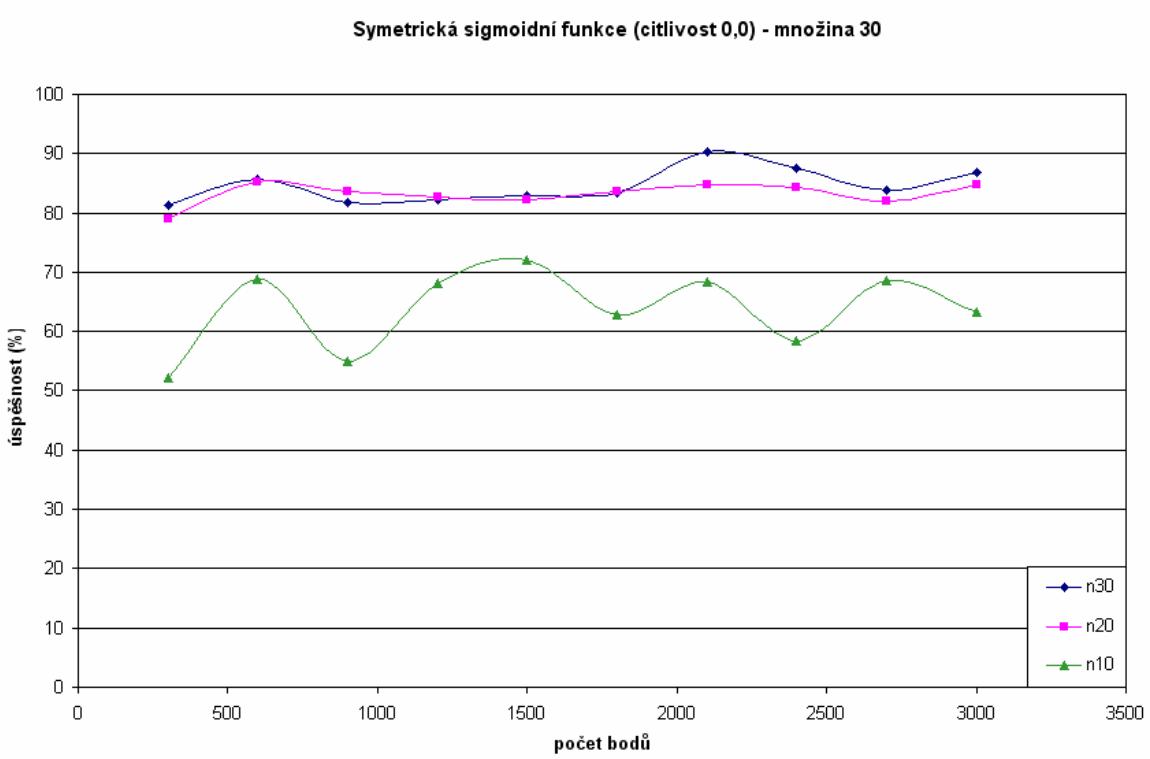
### 6.4.2 Výsledky experimentu



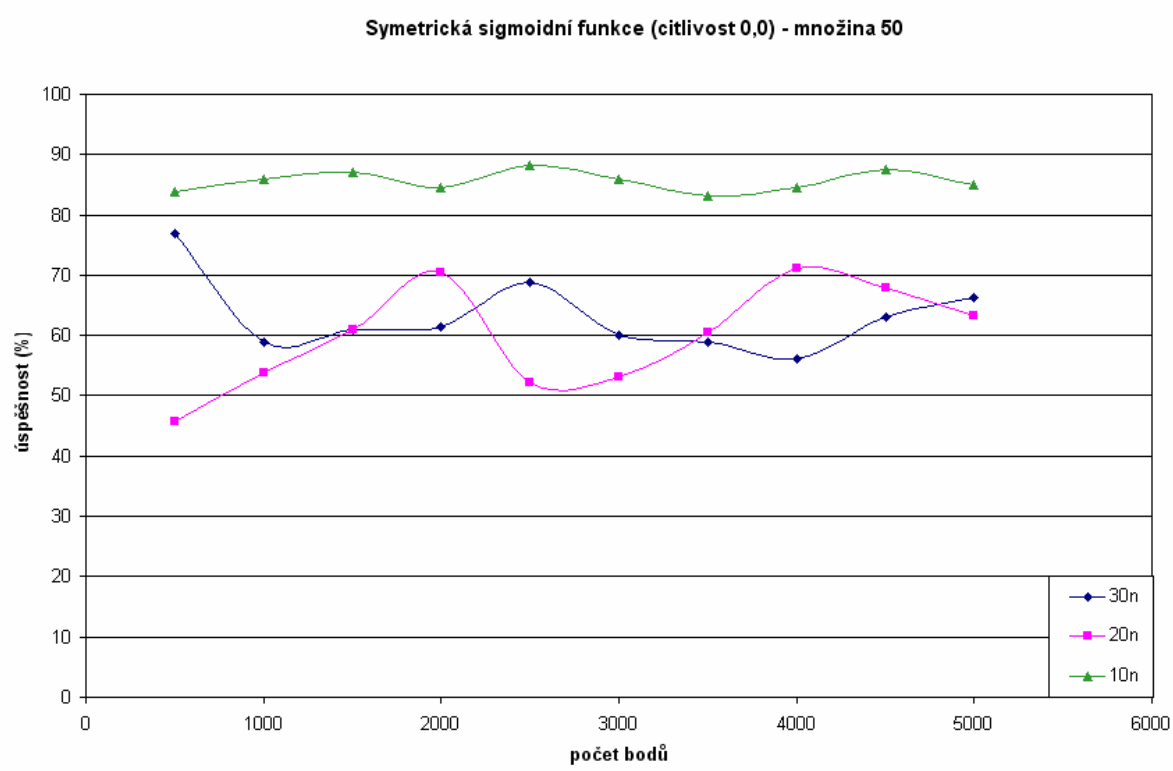

**Graf 7**




**Graf 8**




**Graf 9**




**Graf 10**

### 6.4.3 Závěr experimentu

Jak je vidět na grafech v podkapitole 6.4.2, u symetrické sigmoidní funkce mnohem větší rozdíly v daném natrénování než u funkce klasické sigmoidní. Zde můžeme říci, že symetrická funkce je mnohem stabilnější pro všechny počty skrytých neuronů, pokud trénování provádíme na určité množině obrázků a jen měníme počty vybraných bodů. Dokonce v tomto případě dochází k mnohem větší úspěšnosti detekce hrany než u klasické sigmoidní funkce.

Při variabilním počtu obrázků a s výběrem bodů o stabilním počtu, dochází k tomu, že detekce neuronovou sítí začíná být silně závislá na počtu skrytých neuronů.

Pokud bych měla uvést parametry pro nejlepší natrénovaní neuronové sítě nezávisle na počtu skrytých neuronů, volila bych trénovací množinu o velikosti 50 obrázků a z každého obrázku vybrala 10–50 bodů.

## 6.5 Experiment 4: Vliv volby prahu při určování hrany

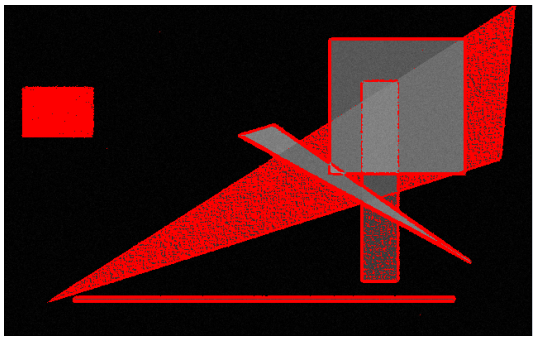
Protože u sigmoidní funkce docházelo k nejlepšímu natrénování při volbě trénovací množiny ze 30 obrázků s výběrem 70 bodů na obrázek a také u ni docházelo k šumu detekce hran. Rozhodla jsem se prahovat právě na této množině jak pro symetrickou, tak pro klasickou sigmoidní funkci

### 6.5.1 Postup experimentu

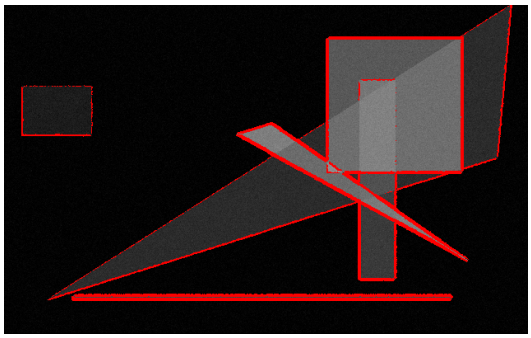
Na syntetickou množinu obrázků se použije detektor hran s již natrénovanými sítěmi s různou aktivační sítí. Výsledky experimentu budou obrazově zpracovány. Vždy bude uvedena hladina, kdy ještě je detekován šum a kdy už je čistá detekce, na příkladě jednoho rozložení obrázku.

### 6.5.2 Výsledky experimentu

Pro 10 neuronů s klasickou sigmoidní aktivační funkcí ve skryté vrstvě sítě:

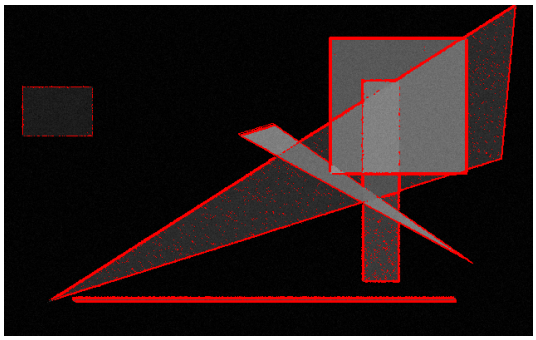


**Obrázek 44:** *Hladina 0,5 a úspěšnost 90,97 %*

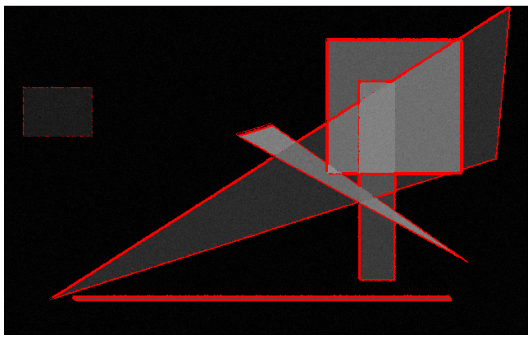


**Obrázek 45:** *Hladina 0,6 a úspěšnost 84,66 %*

Pro 10 neuronů se symetrickou sigmoidní aktivační funkcí ve skryté vrstvě:

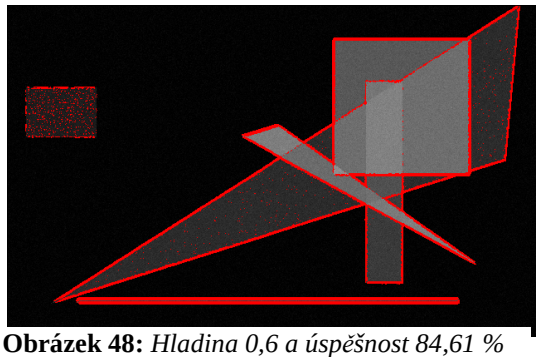


**Obrázek 46:** *Hladina 0,0 a úspěšnost 87,85 %*

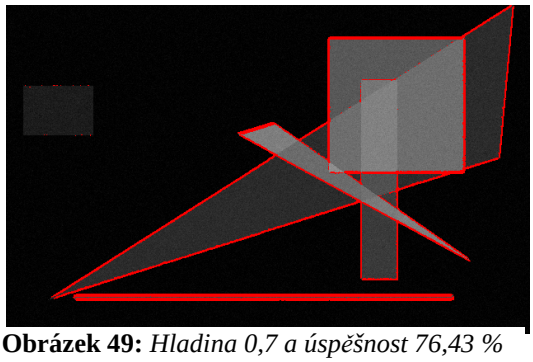


**Obrázek 47:** *Hladina 0,1 a úspěšnost 83,51 %*

Pro 20 neuronů s klasickou sigmoidní aktivační funkci ve skryté vrstvě:

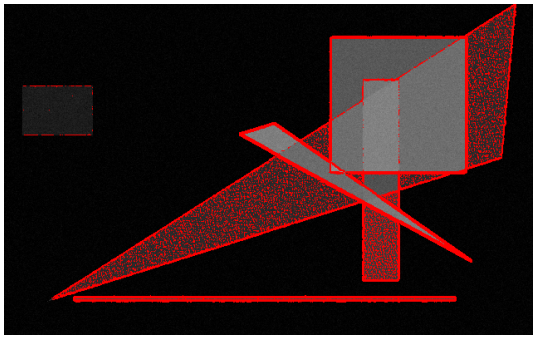


**Obrázek 48:** *Hladina 0,6 a úspěšnost 84,61 %*

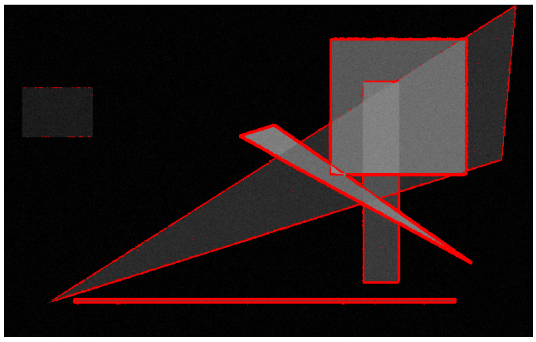


**Obrázek 49:** *Hladina 0,7 a úspěšnost 76,43 %*

Pro 20 neuronů se symetrickou sigmoidní aktivační funkci ve skryté vrstvě:

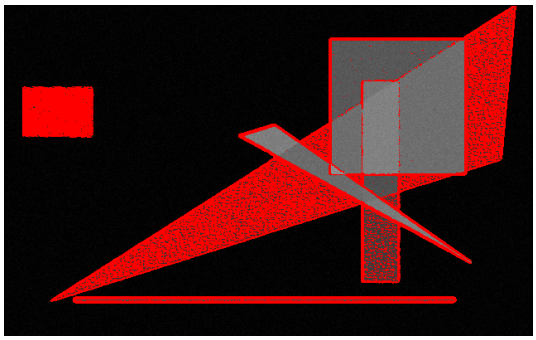


**Obrázek 50:** *Hladina -0,1 a úspěšnost 90,45 %*

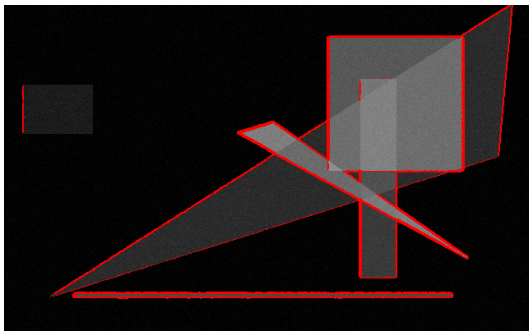


**Obrázek 51:** *Hladina 0,0 a úspěšnost 84,40 %*

Pro 30 neuronů s klasickou sigmoidní aktivační funkcí ve skryté vrstvě:

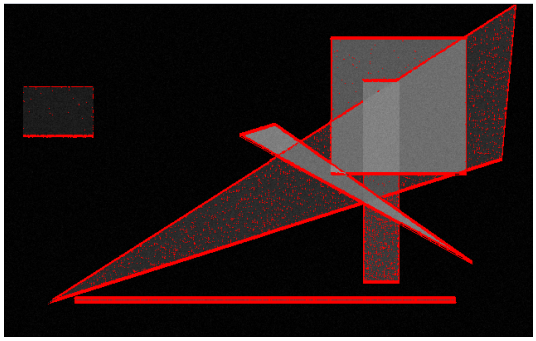


**Obrázek 52:** *Hladina 0,5 a úspěšnost 88,74 %*

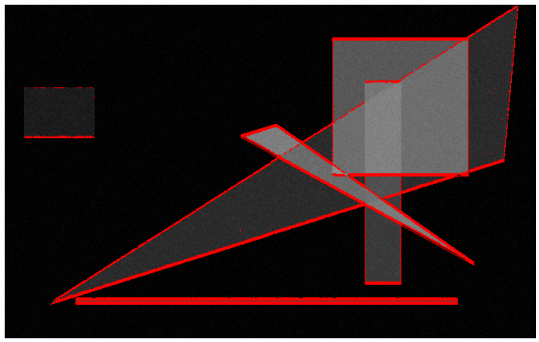


**Obrázek 53:** *Hladina 0,7 a úspěšnost 73,73 %*

Pro 30 neuronů se symetrickou sigmoidní aktivační funkcí ve skryté vrstvě:



**Obrázek 54:** *Hladina 0,1  úspěšnost 89,61 %*



**Obrázek 55:** *Hladina 0,3 a úspěšnost 86,26 %*

### 6.5.3 Závěr experimentu

Hladina práhování je velmi nestabilní. V experimentu byla dodržena stejná trénovací množina a stejný obrázek, na kterém byly hrany detekovány s různou citlivostí. Přesto můžeme zjistit výrazné odchylky ve správné volbě hladiny pro detekci.

Dalo by se říci, že s vyšším počtem neuronů ve skryté vrstvě roste i interval, ve kterém dochází k výskytu minimálního šumu (90–99 %). Je vidět, že práh u klasické sigmoidní funkce lze volit z okolí hodnoty 0,7 a u symetrické sigmoidní funkce volíme z okolí hodnoty 0,1.

Při zvyšování hodnoty prahu rapidně klesá úspěšnost detekce hran.

# 7 Závěr

Detekce hran pomocí neuronové sítě je existenčně závislá na výběru trénovací množiny. Což je nevýhoda pro nezkušeného uživatele, ale zároveň dává výhodu pro použití stejného postupu pro citlivost detekce na různé hrany (jen vodorovná, v šikmém směru, …). Stačí jen změnit procentuální zastoupení hrany v trénovací množině.

Aby uživatel dosáhl dostatečných výsledků, aniž by musel složitě vybírat trénovací množinu, je potřeba volit trénovací obrázky s co největším počtem hran a dostatečnou množinu vybíraných bodů z obrázků (okolo 50 bodů na obrázek).

Při experimentování jsem zjistila, že jsou rozdíly při použití různých aktivačních funkcí. U klasické sigmoidní funkce není potřeba brát zřetel na to, zda je množina vybírána spíše podle počtu obrázků nebo podle počtů vybíraných bodů z jednoho obrázku. Avšak při použití symetrické sigmoidní funkce velmi záleží na typu složení trénovací množiny. Je lepší navyšovat počet vybíraných bodů a nechat stabilní počet obrázků (tzn. raději jít cestou specializace trénovací množiny). Jinak dojde k velkému rozdílu úspěchu při detekci. Klasická funkce vykazuje jistou stabilitu. Při některých množinách může mít symetrická funkce výrazně navrch, u jiných může být těžce podprůměrná oproti klasické sigmoidní funkci. Co se týká rychlosti, symetrická funkce je výrazně rychlejší při trénování sítě.

Stanovení citlivosti detekce na hranu není jednoduchá záležitost. Každá trénovací množina má svá specifika a může se stát, že u některé detekce je síť odolnější vůči šumu a optimální výsledky u klasické sigmoidy dosahuje již od prahu 0,5. Ovšem ne vždy se detekce vydaří. Proto bych standardně doporučovala volit práh 0,7 a v případě nedostatečné detekce práh snižovala. U symetrické funkce to je trochu složitější. Není jednoduché stanovit práh, hrají tam významnou roli setiny citlivosti. Pro tento případ bych doporučovala volit spíše práh 0,0.

Okolí $5\times5$ pixelů, jež používám při detekci může hrany vyznačovat velice silně. Avšak u reálných obrázků nedosahuje šířka čáry velkých rozměrů. Patrnější to je při použití mediánového filtru, jež zvýrazní hranu a dochází k lepší detekci. Zde se projevuje zaoblení hran a čára označující hranu v plné šířce 5 pixelů. To může způsobit ztrátu detailů.

Síť u reálných obrázků dosahuje docela slušné kvality již při trénovanosti kolem 70 %. Což bývá standard při náhodném trénování.

Neuronová síť je jen další variantou při detekci hran. Má obdobné použití jako konvoluční jádro klasických operátorů pro detekci hran. Je použita na určité okolí (podle počtu vstupů do neuronové sítě). Může však být mnohem úspěšnější. Natrénování neuronové sítě je dost časově náročné. Vlastní detekce s příslušnou natrénovanou sítí má již mnohem menší časovou náročnost.

# Literatura


[1] Mařík, V., Štěpánková, O., Lažanský, J., kolektiv: Umělá inteligence 4. Academia, 2003. ISBN 80-200-1044-0.

[2] Drábek, O., Seidl, P., Taufer, I.: Umělé neuronové sítě – základy teorie a aplikace (3). Internetový zdroj: http://www.chemagazin.cz/Texty/CHXVI_1_cl3.pdf.

[3] Drábek, O., Seidl, P., Taufer, I.: Umělé neuronové sítě – základy teorie a aplikace (6). Internetový zdroj: http://www.chemagazin.cz/Texty/CHXVI_6_cl9.pdf.

[4] Historie neuronových počítačů a sítí. Internetový zdroj: http://www.fi.muni.cz/usr/jkucera/pv109/2000/xneudert.html.

[5] Fast Artificial Neural Network Library. Internetový zdroj: http://leenissen.dk/fann/index.php.

[6] The CImg Library – C++. Internetový zdroj: http://cimg.sourceforge.net/.

[7] Sonka, M., Hlavac, V., Boyle, R.: Image Processing, Analysis, and Machina Vision. Thomson Learning, Toronto, 2007. ISBN 049508252X.

# Seznam příloh