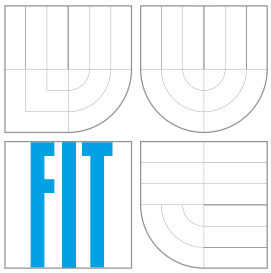
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DETEKCE DOPRAVNÍCH ZNAČEK

TRAFFIC SIGN DETECTION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — TOMÁŠ ŤAPUŠKA
AUTHOR

VEDÚCI PRÁCE — Ing. MICHAL HRADIŠ
SUPERVISOR

BRNO 2008

Abstrakt

Táto bakalárska práca sa zaoberá detekciou dopravných značiek v obraze. Popisuje známe postupy, ich výhody a nevýhody. Je v nej uvedený popis implementácie systému pre detekciu dopravných značiek. V poslednej kapitole sú uvedené testy prevedené na systéme s využitím testovacej sady, ktorú som zostavil a anotoval.

Kľúčové slová

detekcia dopravných značiek, detekcia rohov, klasifikácia, segmentácia farieb, diskrétna kovolúcia, gaussova funkcia

Abstract

This bachelor's thesis is about traffic sign detection in picture. There are written some known methods, their advantages and disadvantages. There is present implementation of the system for traffic sign detection. There are present in the last chapter some tests that were done on the system with using testing set, which was created specialy for this purpose.

Keywords

traffic sign detection, corner detection, classification, color segmentation, discrete convolution, gaussian function

Citácia

Tomáš Ťapuška: Detekce dopravních značek, bakalářská práce, Brno, FIT VUT v Brně, 2008

Detekce dopravních značek

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Michala Hradiša. Uviedol som všetky literárne pramene odkiaľ som čerpal.

.....
Tomáš Ťapuška
6. mája 2008

Poďakovanie

Ďakujem vedúcemu bakalárskej práce Ing. Michalovi Hradišovi za ochotu, konštruktívnu kritiku a odborné vedenie pri písani tejto práce.

Obsah

Úvod

Dopravné značky sú neoddeliteľnou súčasťou vedenia tak motorového, ako aj nemotorového vozidla po dopravných komunikáciach. Usmerňujú dopravu, obmedzujú maximálnu rýchlosť, snažia sa zabrániť zbytočným kolíziam a informujú vodiča o možných nebezpečenstvách a prekážkach na ceste. Pre vodiča je veľmi dôležité pozorne sledovať situáciu na ceste, pričom mu často unikajú informácie, ktoré dopravné značky poskytujú a to môže spôsobiť nežiadúci chaos, v horšom prípade dopravnú nehodu a následne straty na životoch. Z tohto dôvodu je potrebné dokázať vodiča včas upozorniť na príslušnú dopravnú značku, prípadne prispôsobiť jazdu tomuto dopravnému nariadeniu. Toto je jedna z vecí, prečo sa snažiť o nájdenie a rozpoznanie dopranej značky pomocou počítača. Ďalšie využitie je možné v systémoch *tempomat*, ktorý udržiava vodičom nastavenú rýchlosť. Tempomat by bol schopný po nasadení systému pre rozpoznanie dopravných značiek upraviť rýchlosť v tom prípade, ak by to dopravné značenie nariaďovalo. Iným dôvodom môže byť aj autonómne vedenie motorového vozidla tak, aby boli dodržané dopravné predpisy a pravidlá cestnej premávky. Týmto problémom sa zaoberá projekt *Darpa Urban Challenge*[1].

Štruktúra práce

V mojej práci sa budem venovať návrhu systému pre detekciu dopravných značiek v obraze. Prácu som rozdelil na 2 hlavné kapitoly – kapitola 1 *Metódy detekcie dopravných značiek* a kapitola 2 *Návrh riešenia*. V kapitole 1 sa budem snažiť priblížiť známe postupy pri detekcii dopravných značiek a ich rôzne riešenia, následne v kapitole 2 porovnať výhody a nevýhody jednotlivých prístupov a vysvetliť prečo som si vybral práve to riešenie, ktoré som naimplementoval. V tejto kapitole bude uvedený aj mnou navrhnutý systém pre detekciu dopravných značiek. Následne v kapitole 3 bude môj naimplementovaný systém otestovaný a budú tu vyhodnotené výsledky a zhodnotená jeho spoľahlivosť.

[1] http://www.darpa.mil/GRANDCHALLENGE/

Kapitola 1

Metódy detekcie dopravných značiek

Aby boli dopravné značky ľahko rozoznatelné vodičom pri vedení motorového vozidla musia mať jednoznačnú výpovednú hodnotu. Nemôžu byť zložité na pochopenie. Preto sa ich tvorcovia snažili podať informáciu, ktorú dopravná značka nesie, pomocou farieb, tvaru, prípadne sú doplnené jednoduchým piktogramom alebo nápisom. Tieto ich vlastnosti je možné výborne využiť pri ich hľadaní v obraze. Ako sa uvádza v rôznych zdrojoch (nápríklad aj v [4]), je vhodné použiť pri detekcii dopravných značiek v obraze nasledujúce kroky:

1. Segmentácia farieb
    - Prahovanie farebných zložiek v RGB farebnom priestore
    - Prahovanie farebných zložiek v HSI farebnom priestore
    - RG farebný priestor
2. Detekcia rohov
    - Detekcia hrán
    - Nájdenie rohov pomocou konvolučných masiek
    - Hľadanie rohov pomocou algoritmu Adaboost
3. Rozpoznanie útvaru
    - Trojuholníkové dopravné značky
    - Obdĺžnikové a kruhové dopravné značky
4. Klasifikácia dopravných značiek
    - Normalizácia nájdenej dopravnej značky
    - Tample matching

V tejto kapitole bude následne v jednotlivých sekciách vysvetlený každý krok s najčastejšie sa používanými technikami. Každá technika má svoje výhody a nevýhody a preto je len na programátorovi, ktorú z uvedených metód si vyberie.

1.1 Segmentácia farieb

Dopravné značky sú charakteristické svojou jednoznačnou a jasnou farebnosťou. Preto je v rámci čo najefektívnejšieho spracovania obrazu dobré predspracovať si obraz tak, že ponecháme iba pixely, ktorých farebná informácia sa aspoň približuje tej, ktorú značky môžu niesť. Pre ďalšie spracovanie nám stačí binárny obraz (obraz zložený iba z dvoch farieb). Toto je možné urobiť rôznymi spôsobmi. My si vysvetlíme tri najpoužívanejšie. Takto môže vyzerať obrázok po segmentácii farieb:

Obrázok 1.1: Vľavo: originálny obrázok, vpravo: obrázok po segmentácii farieb

1.1.1 Prahovanie farebných zložiek v RGB farebnom priestore

Ako sa píše v [4], najintuitívnejší farebný priestor je RGB, kde každý bod je reprezentovaný tromi farbami – Red (červená), Green (zelená), Blue(modrá). Potom je možné rozhodnúť o tom či daný bod spĺňa, alebo nespĺňa podmienky pomocou funkcie:

$$\begin{aligned} g(x,y) &= k_1 \begin{cases} R_a \leq f_r(x,y) \leq R_b \\ G_a \leq f_g(x,y) \leq G_b \\ B_a \leq f_b(x,y) \leq B_b \end{cases} \\ g(x,y) &= k_2 \quad \text{v ostatnych pripadoch} \end{aligned} \tag{1.1}$$

Kde $f_r(x,y)$, $f_g(x,y)$, $f_b(x,y)$ sú funkcie vracajúce hodnotu farebnej zložky na pozícii x, y, R_a, G_a, B_a sú dolné medze a R_b, G_b, B_b sú horné medze, ktoré ohraničujú hodnotu jednotlivých farebných zložiek na nami zvolenú oblasť záujmu. Ak bude hodnota farebnej zložky v medziach, bude bodu na súradniciach x, y priradená hodnota k_1, v opačnom prípade k_2.

Táto metóda je veľmi rýchla, jednoznačná, jednoduchá na pochopenie a implementáciu. Ale má svoje zásadné nevýhody. Najväčšou nevýhodou je veľká citlivosť na zmeny osvetlenia scény, čo môže spôsobiť nemalé nepríjemnosti keďže v reálnom svete sa svetelné podmienky menia veľmi často, či už zmenou počasia alebo striedania dňa a noci.

1.1.2 Prahovanie farebných zložiek v HSI farebnom priestore

Podľa [4], je odpoveďou na citlivosť zmien osvetlenia v obraze farebný priestor *HSI* (Hue, Saturation, Intenzity). *HSI* je vyjadrenie farebnej informácie tak ako ju vníma ľudské oko. „Hue“ predstavuje samotnú informáciu o farebnom odtieni (červená, zelená, modrá, ...), „Saturation“ predstavuje sýtosť farby a „Instensity“ jej intenzitu. Keďže väčšinou

je farebná informácia obrázka uložená v *RGB* farebnom priestore, je pred samotnou segmentáciou farieb potrebné previezť obrázok do *HSI* farebného priestoru. Potom je už možné pokračovať s vhodným určením dolnej a hornej medze podobne ako v 1.1.1.

Pri tejto metóde už nie je problém s rôznou intenzitou osvetlenia, ale nebolo to zadarmo. Predspracovanie obrazu prevodom do *HSI* farebného priestoru je veľmi výpočtovo náročná operácia. Pri prevode sa využívajú goniometrické funkcie a odmocniny, ktoré spôsobujú túto časovú a výpočtovú náročnosť prevodu. Tým sa nám môže proces detekcie veľmi spomaliť, preto je dôležité rozhodnúť sa, či nám to stojí za túto časovú stratu, alebo nie. Ďalšou možnosťou je hľadať inú alternatívu.

Prevod do HSI farebného priestoru

Ako sa uvádza v [6] je pred samotným prevodom potrebné znormalizovať hodnoty RGB:

$$r = \frac{R}{R+G+B} \quad g = \frac{G}{R+G+B} \quad b = \frac{B}{R+G+B} \tag{1.2}$$

Potom je každá normalizovaná zložka *HSI* získaná z :

$$h = \cos^{-1}\left\{\frac{\frac{1}{2}\cdot[(r-g)+(r-b)]}{\sqrt{[(r-g)^2+(r-b)(g-b)]}}\right\} \quad h \in [0,\pi] \text{ pre } b \leq g \tag{1.3}$$

$$h = 2\pi - \cos^{-1}\left\{\frac{\frac{1}{2}\cdot[(r-g)+(r-b)]}{\sqrt{[(r-g)^2+(r-b)(g-b)]}}\right\} \quad h \in [\pi,2\pi] \text{ pre } b > g \tag{1.4}$$

$$s = 1 - 3\cdot min(r,g,b); \quad s \in [0,1] \tag{1.5}$$

$$i = \frac{(R+G+B)}{3\cdot 255}; \quad i \in [0,1] \tag{1.6}$$

Hodnoty h, s a i následne konvertujeme do intervalov $h \in [0,360]$, $s \in [0,100]$ a $i \in [0,255]$:

$$H = \frac{h\cdot 180}{\pi}, \quad S = s \cdot 100, \quad I = i \cdot 255 \tag{1.7}$$

1.1.3 RG farebný priestor

RG farebný priestor je podľa [12] farebný priestor, ktorý využíva iba dve farebné zložky, red (červenú) a green (zelenú). Tento formát sa na zobrazovanie už veľmi nepoužíva. Prestal sa používať najmä kvôli tomu, že tento systém nie je schopný vytvoriť *bielu* farbu a veľa farieb zobrazených v tomto farebnom priestore je skreslených. Žiadna farba obsahujúca *modrú* farebnú zložku nemôže byť zobrazená v *RG* správne.

Ale ako sa píše v [2], je *RG* farebný priestor vhodný pri segmentácii farieb v obraze. Nie je síce tak schopný eliminovať svetelné podmienky ako *HSI* (viď 1.1.2), ale viac ako *RGB* (viď 1.1.1). Jeho najväčšou výhodou je jeho nízka výpočtová náročnosť. Na prevod do *RG* stačia dva jednoduché výpočty pre každý bod obrazu.

$$r = \frac{R}{R+G+B}; \qquad g = \frac{G}{R+G+B} \tag{1.8}$$

Potom na určenie oblasti, ktorá najviac spĺňa podmienky pre farbu značky je možné použiť *dvojrozmernú Gaussovu funkciu*. Bez Gaussovej krivky by sme museli vymýšlať opäť vhodné hranice kedy sa jedná ešte o požadovanú farbu a kedy už nie. Použitie tejto krivky nám pomôže určiť jej hranice a pre zistenie príslušnosti k požadovanej farbe stačí zistiť či farebná informácia skúmaného bodu spadá pod *dvojrozmernú Gaussovu funkciu*. Čím bližie k jej stredu tým pravdepodobnejšia je farebná zhoda.

2D Gaussova funkcia

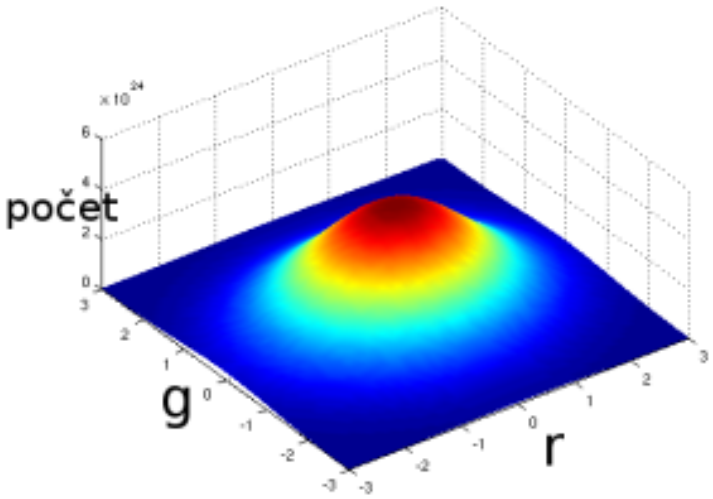
2-rozmerná Gaussova funkcia ako sa uvádza v [10] je reprezentovaná funkciou:

$$f(x,y) = A\ e^{-\left(\frac{(x-x_O)^2}{2\sigma_x^2}\right)-\left(\frac{(y-y_O)^2}{2\sigma_y^2}\right)} \tag{1.9}$$

Kde koeficient A je amplitúda (maximálna výchylka), x_0, y_0 je stred a σ_x, σ_y sú x-ová a y-ová roztiahnutosť 2-rozmernej Gaussovej krivky.
Existuje aj všeobecný tvar funkcie, ktorý je vyjadrený ako:

$$f(x,y) = A\ e^{-\left(a(x-x_o)^2+b(x-x_o)(y-y_o)+c(y-y_o)^2\right)} \tag{1.10}$$

Kde matica $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$ je zložená z kladných zložiek.



Obrázok 1.2: RG farebný priestor zobrazený pomocou 2-rozmernej Gaussovej krivky

1.2 Detekcia rohov

Ďalšou špecifickosťou dopravných značiek je ich tvar. Každý druh dopravnej značky predstavuje iný geometrický tvar. Väčšinou všetky geometrické tvary sú jednoznačne určené svojimi vrcholmi. Preto sa pri hľadaní geometrických tvarov v obraze využíva detekcia rohov. Ako sa uvádza v [4], existujú 2 metódy pre nájdenie rohov v obraze a jedna menej používaná, ale o to zaujímavejšia metóda:

1. Detektor rohov, ktorý potrebuje predspracovanie obrazu. Jedná sa o rozdelenie obrazu do viacerých oblastí pomocou hrán. V takomto prípade je potrebné najskôr nájsť hrany, potom rozdeliť obraz do oblastí a nakoniec detekovať rohy. Táto metóda je veľmi výpočtovo a aj implementačne náročná.

2. Detektor, ktorý pracuje priamo na vstupnom obrázku. Tento typ detektoru je závislý na gradiende obrázku. Zmeny v intenzite a smere gradiendu sú kritéria pre označenia vstupného bodu ako roh. Ani jedna z doterajších metód nedokázala rozlíšiť *roh* podľa typu, čo ako uvidíme ďalej je veľmi potrebné pri detekcii dopravných značiek. Preto je vhodné použiť menšiu úpravu na tento detektor a to tak, že bude obrázok *konvoluovaný* s vhodnou *maskou* prislúchajúcou určitému typu rohu.

3. Detektor využivajúci algoritmus *Adaboost*. Táto metóda je robustná, dokáže rozlišovať typy rohov a je aj veľmi rýchla. Jej jedinou nevýhodou je veľká náročnosť na vytvorenie vhodného *klasifikátoru* pre každý roh.

1.2.1 Detekcia hrán

Vychádzajúc z [15], je hrana vektorová veličina, ktorá je určená veľkosťou a smerom. Hrana indikuje body v obraze, v ktorých dochádza ku zmenám. Operátory pre detekciu a ohodnotenie hrán v obraze vychádzajú z parciálneho diferenciálneho operátoru. Veľkosť gradiendu ∇ je daný vzťahom:

$$|\nabla g(x,y)| = \sqrt{\left(\frac{\partial g}{\partial x}\right)^2 + \left(\frac{\partial g}{\partial y}\right)^2} \tag{1.11}$$

Druhou zložkou gradiendu je smer Ψ

$$\Psi = \tan^{-1}\left(\frac{\frac{\partial g}{\partial y}}{\frac{\partial g}{\partial x}}\right), \quad \text{pre} \quad \frac{\partial g}{\partial x} \neq 0 \tag{1.12}$$

Niekedy sa zaujímame iba o veľkosť gradiendu a jeho smer nás nezaujíma. V takomto prípade sa často používa *Laplaceov operátor*, ktorý aproximuje druhú deriváciu a je nemenný voči rotácii. *Laplaceov operátor* ∇^2 sa vypočíta podľa vzťahu:

$$\nabla^2(x,y) = \frac{\partial^2 g(x,y)}{\partial x^2} + \frac{\partial^2 g(x,y)}{\partial y^2} \tag{1.13}$$

Hranové operátory je možné rozdeliť do dvoch skupín:

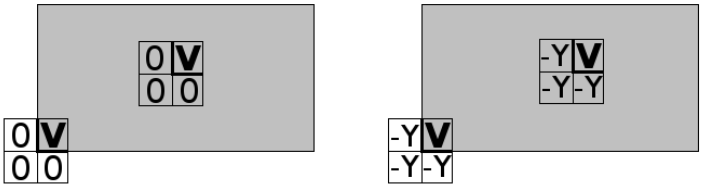
1. Skupina operátorov využíva diskrétnu konvolúciu. Operátory, ktoré sa nezaujímajú o rotáciu sa realizujú jedinou konvolučnou maskou (napríklad *Laplaceov operátor*), naopak operátory zaujímajúce sa o rotáciu používajú niekoľko masiek. Smer gradiendu sa odhaduje pomocou nájdenia masky, ktorá odpovedá najväčšej veľkosti gradiendu.

2. Skupina operátorov, ktorá označí miesta ako hrany tam, kde druhá derivácia obrazovej funkcie prechádza nulou (napríklad *Cannyho operátor*).

Z prvej skupiny operátorov je najznámejší a najobľúbenejší *Laplaceov gradiendný operátor* ∇^2. Ako už vieme je necitlivý voči otočeniu, udáva len veľkosť hrany a nie jej smer. Dve najpoužívanejšie konvolučné jadrá (jeden pre 4-okolie a druhý pre 8-okolie):

$$h_{4-okolie} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \qquad h_{8-okolie} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{1.14}$$

Laplaceov gradiendný operátor je veľmi citlivý na šum, pretože aproximuje 2. deriváciu. Ak je vstupný obraz zašumený je nepoužitelný.



Obrázok 1.3: Intuitívna konvolučná matica pre 90°

Z druhej skupiny operátorov je známy *Cannyho hranový operátor*.Jeho základom je hľadanie priechodu druhej derivácie nulou. Nie je citlivý na šum a dokáže pracovať aj so značne zašumeným obrazom. Výsledkom Cannyho detektoru je aj veľkosť a aj smer hrany. Bližšie informácie o *Cannyho hranovom operátore* je možné nájsť v [9].

1.2.2 Nájdenie rohov pomocou konvolučných masiek

Ako sme si už uviedli skôr, pri detekcii dopravných značiek je dobré vedieť aký typ rohu sme našli. Pre tento účel je vhodné využiť konvoluovanie obrazovej funkcie pomocou vhodne zvolených masiek. Ale čo je to tá maska a ako ju získať? Ako sa uvádza v [4], sú to korelačné filtre a ideálny obraz po filtrácii by mal mať rovnakú úroveň šedi vo vnútri objektu a nulovú mimo objektu. Potom by mohla maska pre 90° roh vyzerať ako na obrázku 1.3. Po konvolúcii s touto maskou dáva kladné hodnoty v mieste výskytu rohu, ostatné sú nulové. Ale táto maska nie je ideálna pretože získané hodnoty sú rovnaké v mieste rohu ako aj vo vnútri objektu. Keď zmeníme nulové hodnoty matice na záporné, pozadie ostane s nulovými výslednými hodnotami, maximum bude v rohu objektu a bude možné ho aj lokalizovať. Táto maska funguje oveľa lepšie, ale je použitelná iba na ideálnom vstupnom obraze. V skutočnom obraze nebude roh tak jasne definovaný, ale môže sa tam vyskytnúť aj zašumenie. Aby sme toto minimalizovali je potrebné zvoliť vhodné hodnoty kladných a záporných zložiek masky.

Nech je roh umiestnený do počiatku súradnicového systému tak, že X-ová os bude rameno uhla tvoreného hranami tohto uhla. Potom funkcia popisujúca stupeň šedi bude:

$$I(x,y) = \begin{cases} A & \text{ak } x > 0 \ \wedge \ -mx < y < mx \\ 0 & inak \end{cases} \tag{1.15}$$

Ak $n(x,y)$ je šum, potom funkcia popisujúca stupeň šedi okolo rohu je:

$$F(x,y) = I(x,y) + n(x,y) \tag{1.16}$$

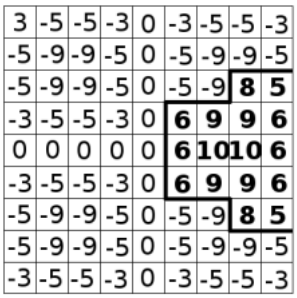
Funkcia $g(x,y)$ pre vhodný hranový operátor musí spĺňať nasledujúce kritéria:

- Výsledok konvolúcie $O(x,y) = F(x,y) * g(x,y)$ musí byť najväčší v mieste rohu.
- $g(x,y)$ nesmie byť citlivý na šum.

Keď predchádzajúce kritéria zapíšeme matematicky dostaneme rovnice:

$$g(x,y) = c_1 \sin\frac{m\pi x}{W} \cdot \left[-(e^{zW} + e^{-zW}) + (e^{zy} + e^{-zy})\right] \tag{1.17}$$

$$g(x,y) = c_2 \sin\frac{n_1\pi x}{W} \cdot \sin\frac{n_2\pi y}{W} \tag{1.18}$$



Obrázok 1.4: Konvolučná matica pre 60° roh

Kde W je veľkosť masky a c_1, c_2, n_1, n_2, m a z sú konštanty. Pre tú časť masky, kde sa nachádza roh platí funkcia 1.17 a pre zvyšok 1.18. Konštanty c_1 a c_2 násobia všetky hodnoty masky takže ich hodnota je ľubovoľná. Výsledok rovnice 1.17 bude vždy záporný pre $y \in (0, W)$. Ale keďže vzťah 1.17 predstavuje časť konvolučnej matice kde sa nachádza roh, musí jej výsledok byť vždy kladný. Preto zvolíme konštantu $m = -1$. Pre získanie záporných hodnôt pre časť masky prislúchajúcu pre pozadie (vzťah 1.18) musia byť konštanty n_1 a n_2 opačných znamienok, preto môžeme zvoliť $n_1 = 1$ a $n_2 = -1$. Autori metódy nenašli žiadnu zákonitosť, podľa ktorej by sa dala určiť hodnota z, ale zistili, že jej zvyšovanie zvyšuje schopnosť filtrovania šumu. Preto bola experimentálne zvolená hodnota $z = 0,2$. Z hodnôt, ktoré boli použité v príklade, je možné vytvoriť konvolučnú masku o rozmeroch 9×9 pre 60° roh, ktorú je možné vidieť na obrázku 1.4. Ako už bolo spomínané vyššie na detekciu dopravných značiek je potrebné viac druhov rohov. Konvolučné matice pre jednotlivé typy rohov sú uvedené v prílohe A.

Keď už máme vytvorené konvolučné matice pre jednotlivé typy rohov môžeme začať so získavaním rohov z obrazu. Budeme postupovať podľa nasledujúcich bodov:

- Obraz konvolujeme jednotlivými konvolučnými maskami.
- Výsledné hodnoty porovnáme s určeným prahom, ak sú väčšie ponecháme, ostatné nastavíme na hodnotu pozadia.
- Z výsledných hodnôt ponecháme iba najväčšie hodnoty vo svojom okolí. Veľkosť okolia môže byť zvolená podľa potreby.

Diskrétna konvolúcia

Ako sa uvádza v [14], diskrétna konvolúcia sa často používa v algoritmoch spracovania digitálneho obrazu. Vzorec diskrétnej konvolúcie má tvar:

$$f(x,y) * h(x,y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} f(x-i, y-j) \cdot h(i,j) \tag{1.19}$$

Kde funkcia $h(i,j)$ sa nazýva *konvolučné jadro*, pri diskrétnej konvolúcii nazývané aj *konvolučná maska*. Každý bod prekrývajúci sa s maskou vynásobíme koeficientom z masky a potom hodnoty sčítame, výslednú hodnotu zapíšeme namiesto bodu prekrývajúceho sa so stredom masky.

Non-maxima suppression

Vychádzajúc z [3], *non-maxima suppression* (potlačenie nemaxím), je algoritmus, v ktorom sa prehľadáva okolie určitého bodu a ponecháva sa iba maximálna intenzita. Táto metóda sa najčastejšie používa pri hľadaní rohov a hrán na zamedzenie viacnásobných detekcií. Algoritmus pozostáva z:

1. Máme bod (x, y), kde x,y sú jeho súradnice a $I(x, y)$ je intenzita tohto bodu.
2. Vypočítame gradiend obrázku a jeho veľkosť v bode (x, y).
3. Vypočítame veľkosť gradiendu bodov v blízkom okolí bodu (x, y)
4. Ak nie je tento bod maximum, potom nie je rohový (resp. hranový) bod.

1.2.3 Hľadanie rohov pomocou algoritmu Adaboost

Podľa [5] bol tento algoritmus prvý krát formulovaný v roku 1995 pánmi *Yoav Freund* a *Robert E. Schapire*. Adaboost je skratka zo slóv *Adaptive* (adaptívny) a boosting (posilnenie). Je to metóda strojového učenia. Ako tránovaciu sadu požaduje vstup $(x_1, y_1) \ldots (x_m, y_m)$, kde $x_i \in X$ sú dáta a $y_i \in Y = -1, 1$ je údaj o tom, či je príslušníkom požadovanej skupiny, alebo nie. Ďalej je potrebná množina základných (tzv. slabých) klasifikátorov, ktoré sú schopné klasifikovať lepšie ako náhodne. Na začiatku majú nastavené počiatočné váhy, ktoré sa v priebehu trénovania menia podľa toho, či klasifikátor určil výsledok správne alebo nie. Po ukončení trénovania vzniká zo slabých klasifikátorov jeden silný.

Takýmto spôsobom je možné natrénovať všetky druhy rohov. Táto metóda je náročná hlavne na natrénovanie dobrého klasifikátoru. Ďalej sa ponúka otázka, či nepoužiť *adaboost* na nájdenia a klasifikovanie celej dopravnej značky. Pre bližšie informácie viz [5].

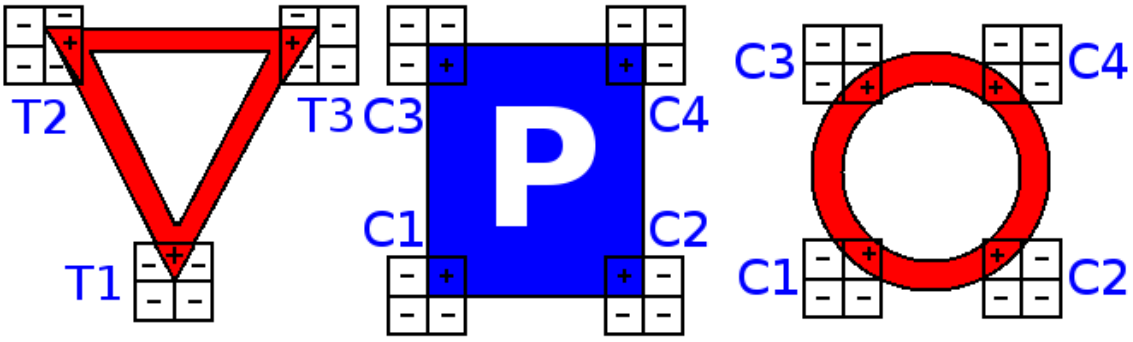
1.3 Rozpoznanie útvarov

Dopravné značky sú charakteristické svojim tvarom. Toto využijeme pri rozpoznaní útvaru (v našom prípade dopravnej značky). Ako sa uvádza v [4], môžeme ich rozdeliť do dvoch skupín podľa typov rohov, ktoré budeme u nich očakávať:

- *Trojuholníkové* (výstražné) dopravné značky. Sú zložené z troch typov 60° rohov. Ako je vidieť na obrázku 1.5, sú to dolný vrchol T1, horný ľavý (T2) a pravý (T3) vrchol. Ak máme opačne orientovaný trojuholník sú aj rohy orientované opačne (jeden hore a dva naspodu).
- Obdĺžnikové (informačné) a kruhové (zákazové, príkazové) dopravné značky. Sú zložené zo štyroch 90° rohov. Ako je vidieť na obrázku 1.5, sú to dolný ľavý (C1) a pravý (C2) roh a horný ľavý (C3) a pravý (C4) roh.

1.3.1 Trojuholníkové dopravné značky

Ako bolo uvedené vyššie, dopravné značky trojuholníkového typu je možné nájsť pomocou troch rohov. Tieto rohy, ako sa píše v [4], musia spolu skladať rovnostranným trojuholník. Keďže pracujeme v reálnom svete, pri zosnímaní dopravnej značky bude táto značka určite pootočená, nebude to nikdy úplne presný rovnostranný trojuholník, ale budeme musieť



Obrázok 1.5: Typy rohov pre rozličné geometrické tvary

počítať aj so sploštením, prípadne s natiahnutím trojuholníku. Pre nájdenie výstražnej dopravnej značky je potrebné vykonať nasledujúce kroky.

1. Nájdeme všetky typy rohov, ktoré sú pre trojuholník dôležité (T1, T2, T3).
2. Podľa vzájomnej polohy nájdených rohov určíme či sa naozaj jedná o výstražnú dopravnú značku. Budeme postupovať podľa nasledujúceho algoritmu:
   I. Nájdeme spodný roh (typ T1).
   II. Od rohu T1 budeme hľadať roh typu T2. Keďže trojuholník nebude ideálny, určíme si oblasť v ktorej by sa daný vrchol mohol nachádzať. V rovnostrannom trojuholníku a v ideálnom prípade by sme roh T2 hľadali pod sklonom $-60°$. Potom môže byť oblasť záujmu sklon od $-68°$ do $-52°$. Ak je T2 nájdený pokračujeme ďalším krokom, ak nie vrátime sa späť na krok I
   III. Druhú oblasť nášho záujmu vytvoríme podobne ako v kroku II, len sklon bude od $52°$ do $68°$. Ďalej musíme oblasť zúžiť na výrez, ktorý je približne v rovnakej horizontálnej pozícii ako vrchol T2. V tejto oblasti budeme hľadať roh typu T3. Ak budeme úspešný máme kompletný trojuholník, ak nie vrátime sa na bod II.

1.3.2 Obdĺžnikové a kruhové dopravné značky

Budeme postupovať podobne ako pri trojuholníkových značkách, vychádzajúc z [4]. Rozdiel bude v typoch rohov a oblastiach záujmu. Pre obdĺžnikové značky sú typické $90°$ rohy. Konkrétne sú to dolný ľavý (C1) a pravý (C2) roh a horný ľavý (C3) a pravý (C4) roh.

1. Nájdeme všetky typy rohov, ktoré sú pre obdĺžnik dôležité (C1, C2, C3, C4).
2. Pri študovaní vzájomnej polohy rohov budeme postupovať podobne ako pri trojuholníku. Budeme postupovať podľa nasledujúceho algoritmu:
   I. Nájdeme ľavý spodný roh (typ C1).
   II. Napravo od rohu C1 hľadáme roh C2. Bude sa nachádzať približne v rovnakej horizontálnej polohe ako C1. Akú veľkú toleranciu zvolíme, záleži na nás (v [4] sa uvádza $\pm 5°$), prípadne sa doladí pri testovaní. Ak nebol roh C2 nájdený vrátime sa na krok I, inak pokračujeme ďalej.

III. Po nájdení pravého horného rohu (C2) budeme hľadať pravý dolný roh (C3), ktorý má približne rovnakú vertikálnu polohu ako C2. Pri úspešnosti pokračujeme ďalej, inak sa vrátime späť na krok II.

IV. Posledný roh, ľavý dolný (C4), budeme hľadať v oblasti, ktorá má približne rovnakú horizontálnu polohu ako roh C3 a vertikálnu ako roh C1. Ak sa v tejto oblasti nachádza ľavý dolný roh máme kompletný obdĺžnik, inak sa vrátime na krok III.

Kruhové dopravné značky

Pre nájdenie kruhu, ktorým sú špecifické asi najpoužívanejšie zákazové a príkazové dopravné značky je množtvo možností. Ale keďže sme používali doteraz konvolučné masky ostaneme pri tom aj teraz. Vytvoriť konvolučnú masku, aby detekovala časť kružnice dokážeme. Problém je v tom, že by týchto masiek muselo byť veľa, aby sme dokázali detekovať rôzne veľké kruhy. Ale ako je uvedené v [4], je možné využiť už vytvorených masiek a to tých istých ako pri detekcii obdĺžnika. Tieto masky by nám mali dokázať detekovať roh v 45°, 135°, 225°, 315° kruhu. Kladná časť masky bude opäť označovať nájdený roh na obvodovej kružnici a záporné hodnoty pozadie. Najväčšou výhodou tejto metódy je, že nemusíme konoluovať obraz znovu, ale kruhové dopravné značky detekujeme už pri detekcii obdĺžnikových dopravných značiek.

1.4 Klasifikácia dopravných značiek

Po úspešnej segmentácii farieb, nájdení rohov a rozpoznaní geometrických útvarov máme dopravnú značku nájdenú. Teraz nám ostáva túto značku identifikovať. Pre správnu klasifikáciu nájdenej dopravnej značky je potrebné mať jej obraz normalizovaný na určitú veľkosť. Potom môže prebehnúť samotná klasifikácia. Pre klasifikáciu existuje veľa metód ako napríklad *adaboost* (viz sekciu 1.2.3), *template matching*, ...

1.4.1 Normalizácia nájdenej značky

V našom prípade sa normalizáciou myslí zmena veľkosti na rovnakú veľkosť, ako bude vzorová značka, pomocou ktorej sa bude klasifikovať. Pre zmenu veľkosti existujú rôzne metódy, napríklad:

- *Interpolácia najbližšieho suseda* (Nearest-neighbor interpolation) – Ako sa uvádza v [11], je to najjednoduchšia a najrýchlejšia metóda interpolácie, často využívaná v praxi. Algoritmus jednoducho zoberie hodnotu bodu najbližšieho suseda a o ostatných susedov sa nezaujíma.

- *Bilineárna interpolácia* (Bilinear interpolation) – Vychádzajúc z [8], je bilineárna interpolácia rozšírením *lineárnej*. Základná myšlienka tohto algoritmu je urobiť lineárnu interpoláciu v jednom smere a potom v druhom. Predpokladajme, že chceme zistiť hodnotu funkcie f v bode $P = (x, y)$. Vychádzame z toho, že vieme hodnotu funkcie f v bodoch $Q_{11} = (x_1, y_1)$, $Q_{12} = (x_1, y_2)$, $Q_{21} = (x_2, y_1)$, $Q_{22} = (x_2, y_2)$. Ako prvú urobíme interpoláciu v x-ovom smere:

$$f(R_1) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21}) \quad \text{kde} \quad R_1 = (x, y_1) \tag{1.20}$$

$$f(R_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22}) \quad \text{kde} \quad R_2 = (x, y_2) \tag{1.21}$$

Ďalej urobíme interpoláciu v y-ovom smere:

$$f(P) \approx \frac{y_2 - y}{y_2 - y_1} f(R_1) + \frac{y - y_1}{y_2 - y_1} f(R_2) \tag{1.22}$$

Z toho vyplýva vzťah pre $f(x, y)$:

$$\begin{aligned} f(x,y) \approx & \frac{f(Q_{11})}{(x_2 - x_1)(y_2 - y_1)}(x_2 - x)(y_2 - y) \\ & + \frac{f(Q_{21})}{(x_2 - x_1)(y_2 - y_1)}(x - x_1)(y_2 - y) \\ & + \frac{f(Q_{12})}{(x_2 - x_1)(y_2 - y_1)}(x_2 - x)(y - y_1) \\ & + \frac{f(Q_{22})}{(x_2 - x_1)(y_2 - y_1)}(x - x_1)(y - y_1) \end{aligned} \tag{1.23}$$

Výsledok bilineárnej interpolácie nie je závislý od poradia jednotlivých interpolácii.

- *Bikubická interpolácia* (Bicubic interpolation) – Podľa [7], je bikubická interpolácia rozšírením *kubickej*. Používa sa najmä vtedy, keď nie je rozhodujúci čas, ale kvalita. Obrázky zväčšené pomocou bikubickej interpolácie sú hladšie a majú menej interpolačných artefaktov. Pre túto interpoláciu sa využíva *Lagrangeov polynóm*, *kubický spline* alebo kubická konvolúcia. Viac informácii je možné nájsť v [7].

1.4.2 Template matching

Vychádzajúc z [13], je *template matching* (šablónové spájanie) technika využívajúca sa v spracovaní digitálneho obrazu. Používa sa na nájdenie malých častí v obrázku, ktoré sa zhodujú so šablónou. Môže byť využitá pri navigácii mobilných robotov, hľadanie hrán v obraze, alebo vo výrobe ako časť kontroly kvality.

Existuje viac prístupov k tejto metóde:

- Základnou metódou je prechádzanie celého obrázku pixel po pixeli a porovnávanie so šablónou. Táto metóda je síce jednoduchá na pochopenie a implementáciu, ale je veľmi pomalá. Najviac sa využíva v zariadeniach, ktoré majú na *template matching* špecializovaný hardware.

- Spôsob ako zrýchliť algoritmus, je rozdeliť obrázok na viac malých obrázkov. Vytvorí sa takzvaná obrazová pyramída (pre obrázok 128×128 px je to 64×64, 32×32, 16×16, ...). Potom sa *template matching* vykoná najskôr na malých obrázkoch, postupne na väčších až po skutočnú veľkosť, pričom oblasť kde sa porovnáva je určená z podobrázkov.

Kapitola 2

Návrh riešenia

Podľa zadania bolo potrebné preštudovať základy spracovania obrazu a klasifikácie, vytvoriť si prehľad o súčasných metódach detekcie dopravných značiek. Navrhnúť a implementovať systém pre detekciu dopravných značiek v obraze a tento následne otestovať na reálnych dátach. Základy spracovania obrazu a prehľad aktuálnych metód pre detekciu dopravných značiek boli uvedené v kapitole 1. V tejto kapitole sa budem zaoberať samotným návrhom systému a jeho implementáciou. Bude tu uvedené aký postup pri detekcii dopravných značiek som zvolil a prečo som zvolil práve tú metódu, ktorú som vybral.

2.1 Analýza a návrh riešenia problému

Dopravné značky boli navrhované a vyrobené tak, aby boli pre vodiča ľahko rozoznateľné pri vedení motorového vozidla. Vodič ich dokonca musí dokázať rozoznávať intuitívne, bez toho, aby ho rozptylovali pri šoférovaní. Preto by mali mať značky svoju typickú sýtu farbu a tvar. Tieto ich vlastnosti využijem aj pri zostavovaní a tvorbe systému pre detekciu dopravných značiek v obraze.

Dopravné značky je možné rozdeliť podľa tvaru a farby tak ako je vidieť v tabuľke 2.1.

	Farba značky	**Tvar značky**	**Príklad**
Zákazové dopravné značky	červená	kruh	
Príkazové dopravné značky	modrá	kruh	
Výstražné dopravné značky	červená	trojuholník	
Informatívne dopravné značky	modrá, zelená, biela	obdĺžnik	

Tabuľka 2.1: Prehľad tvarov a farieb dopravných značiek.

Podľa vizuálnych vlastností a typov dopravných značiek a po preštudovaní súčasných metód v tejto oblasti som sa rozhodol zvoliť takúto štruktúru systému:

1. *Segmentácia farieb* – tu vyžijem práve už spomínanú farebnú špecifickosť dopravných značiek. Pokúsim sa oddeliť potecionálnu značku od zvyšku scény obrazu pomocou segmentácie farieb. Táto metóda značne urýchli ďalšiu prácu s obrázkom. Na segementáciu farieb je možné využiť viacej prístupov, ako som už spomínal v sekcii 1.1. Niektoré sú rýchlejšie iné pomalšie, niektoré viac citlivé na zmenu svetelných podmienok, iné zas menej citlivé. Každá metóda má svoje výhody aj nevýhody:

   - *Prahovanie farebných zložiek v RGB farebnom priestore*: RGB farebný priestor je najznámejší a najintuitivnejší. Keďže v takomto farebnom priestore môžeme očakávať vstupné dáta , nie je potrebné robiť žiadnu konverziu a preto je táto metóda jedna z najrýchlejších. Na druhej strane je tento farebný priestor veľmi citlivý na zmenu svetelných podmienok.
     výhody: ľahká na implementovanie a pochopenie, nízka výpočtová a časová náročnosť
     nevýhody: veľmi citlivé na zmeny svetelných podmienok v obraze
   - *Prahovanie farebných zložiek v HSI farebnom priestore*: HSI farebný priestor je vyjadrenie farebnej informácie tak, ako ju vníma ľudské oko. Ako sa uvádza v [6], je tento farebný priestor odpoveďou na zmeny svetelných podmienok v obraze. Podľa uvedeného by sa mohlo javiť, že máme ideálny farebný priestor pre našu situáciu, ale opak je pravdou. Vychádzajúc z [6], je potrebné najskôr vstupnú farebnú informáciu previezť do HSI, tento proces je ale veľmi výpočtovo náročný. Táto skutočnosť spôsobuje nežiadúce veľké spomalenie.
     výhody: nie je citlivá na zmeny svetelných podmienok
     nevýhody: metóda zložitá na implementovanie, má vysokú výpočtovú a časovú náročnosť
   - *Prahovanie farebných zložiek v RG farebnom priestore*: Podľa [12] sa tento farebný model už dlho nepoužíva na zobrazovanie farebnej informácie najmä kvôli jeho neschopnosti zobraziť v obraze modrú farbu. Napriek tomu sa veľmi často vužíva v oblasti počítačového videnia a spracovania obrazu. Je to farebný priestor podobný RGB, preto aj prevod nie je veľmi náročný, viď sekcia 1.1.3.
     výhody: je menej citlivá na zmeny svetelných podmienok ako RGB, jednoduchá na implementáciu, má nízku výpočtovú a časovú náročnosť
     nevýhody: je viac citlivá na zmeny svetelných podmienok ako HSI

   Po zvážení výhod a nevýhod, ktoré tieto prístupy ponúkajú som sa rozhodol zvoliť segmentáciu farieb pomocou *RG farebného modelu* s využitím dvojrozmernej Gaussovej funkcie (viď sekcia 1.1.3). Rozhodol som sa tak najmä kvôli tomu, že ponúka rýchlosť vyššiu ako pri *HSI farebnom modeli* a jeho citlivosť na zmenu svetelných podmienok nie je až tak vysoká ako pri *RGB farebnom modeli*.

2. *Nájdenie rohov* – v tejto časti systému využijeme ďalšej špecifickosti a tou je tvar dopravných značiek. Ako je vidieť v tabuľke 2.1 môžu byť značky troch geometrických tvarov – kruhové, trojuholníkové a obdĺžnikové. Ako som písal aj v kapitole 1.2 a uvádza sa v [4] je pre detekciu geometrických útvarov vhodné využiť toho, že obsahujú špecifické druhy rohov a ich rozpoloženie. Existuje viac prístupov ako hľadať rohy v obraze:

   - *Nájdenie rohov pomocou detekcie hrán*: Táto metóda nie je veľmi šťastná. Pred samtoným hľadaním rohov je potrebné najskôr v obraze nájsť hrany, potom

obraz rozdeliť na segmenty a nakoniec hľadať rohy. Toto robí túto metódu veľmi výpočtovo a implementačne náročnou. Ďalšou nevýhodou tejto metódy je jej neschopnosť hľadať rohy konkrétneho typu, čo je pri detekcii dopravných značiek veľmi potrebná vlastnosť. Najmä kvôli jej výpočtovej náročnosti sa táto metóda prakticky nepoužíva.

- *Nájdenie rohov pomocou konvolučných masiek*: Táto metóda pracuje priamo na vstupnom obraze a tým šetrí veľa času pri výpočtoch. Je výpočtovo veľmi rýchla hoci implementačne náročnejšia. Pomocou tejto metódy je tiež možné hľadať konkrétny typ rohu. Jeho nevýhodou sa môže zdať nutnosť výroby masiek, ale tieto sú vyrobené iba raz pri tvorbe programu, ďalej ich už máme uložené. Pre aplikáciu konvolučných masiek sa využíva diskrétna konvolúcia, viď sekcia 1.2.2. Niektoré typy masiek viď príloha A.
- *Nájdenie rohov pomocou algoritmu adaboost*: Táto metóda, podobne ako predchádzajúca, pracuje priamo na vstupnom obrázku. Je pomocou nej tiež možné nájsť konkrétny typ rohu. Jej nevýhodou je oveľa väčšia zložitosť pri tvorbe klasifikátoru pre rôzne typy rohov. Bližšie informácie o tomto algoritme viď sekcia 1.2.3.

Po zvážení výhod a nevýhod, ktoré jednotlivé metódy ponúkajú som sa rozhodol pre hľadanie rohov v obraze využiť metódu pomocou *konvolučných masiek*. Rozhodol som sa tak preto, že táto metóda je rýchla a poskytuje detekciu konkrétnych typov rohov, čo metóda pomocou *detekcie hrán* neponúka. Pred *adaboost*om som ju uprednostil najmä kvôli zložitosti prípravy klasifikátorov.

K tomu aby som mohol použiť metódu konvolučných masiek si potrebujem vyrobiť masky pre rôzne typy rohov. Podľa tvaru značky ich môžme rozdeliť na:

- *trojuholník*: Obsahuje tri vrcholy. V ideálnom prípade je dopravná značka trojuholníkového tvaru, rovnostranný trojuholník. Rovnostranný trojuholník má všetky uhly 60°. Keďže ja budem pracovať s reálnymi dátami, budem pripúšťať aj miernu odchýlku od ideálneho rovnostranného trojuholníka. Pre nájdenie trojuholníku v obraze je potrebné nájsť horný (prípadne dolný, ak je opačne otočený trojuholník) roh a ľavý a pravý dolný (respektíve horný) roh.
- *obdĺžnik*: Obsahuje štyri v ideálnom prípade 90° rohy. Takisto ako pri trojuholníku bude prípustná určitá odchýlka od ideálneho pravého uhla, pretože pracujem s reálnym a nie s ideálnym obrazom. I keď obdĺžnik obsahuje štyri rôzne rohy, je to v podstate ten istý typ len inak otočený, takže mi stačí vyrobiť jednu konvolučnú masku pre 90° roh a tú potom vhodne otočiť, aby som získal ostatné masky.
- *kruh*: Aj keď kruh vo svojej podstate rohy neobsahuje je možné na jeho obvode, ako sa uvádza v [4], nájsť pomocou konvolučných masiek rohy rovnaké ako pri obĺžniku. Takýmto prístupom ušetríme jedno prehľadávanie obrazu, keďže pri hľadaní rohov pre obdĺžnik nájdeme zároveň aj tie pre kruh.

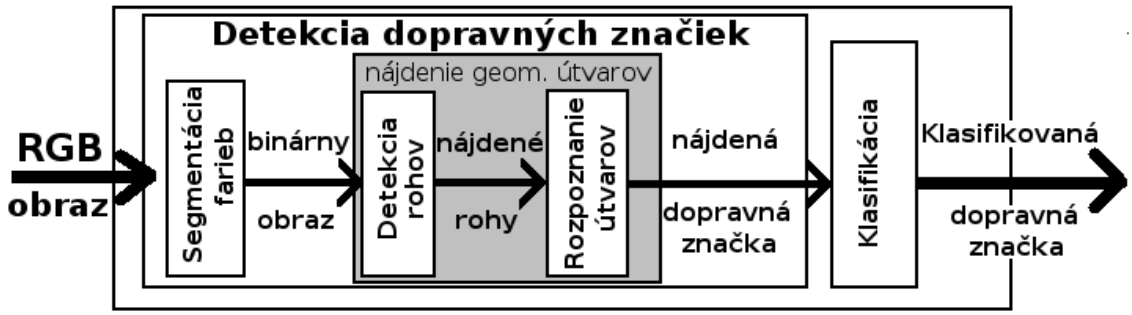
Pre lepšiu názornosť viď obrázok 1.5, kde sú znázornené jednotlivé typy rohov. Potrebné konvolučné masky k nájdeniu rohov vyrobené podľa [4] sú uvedené v prílohe. Po prevedení diskrétnej konvolúcie s konvolučnou maskou dostaneme veľa bodov, ktoré môžu byť s väčšou alebo menšou pravdepodobnosťou rohami. Tieto su zhluknuté do určitého okolia závislého poväčšine na veľkosti detekovaného rohu. Pre od-

stránenie nežiadúcich detekcií a ponechanie iba najlepšej som sa rozhodol použiť metódu *non-maxima suppression*.

3. *Rozpoznanie geometrického útvaru* – po nájdení všetkých potrebných rohov potrebujem zistiť či som pomocou nich schopný zostaviť nejaký požadovaný geometrický útvar (rovnostranný trojuholník, obdĺžnik, alebo kruh). Algoritmus ako postupovať pri rozpoznávaní geometrických útvarov je uvedený v sekcii 1.3. Tento algoritmus vychádza z predpokladu, že každá dopravná značka je určitého geometrického tvaru a ten je zložený z usporiadanej n-tice rohov. Po úspešnom ukonční algoritmu pre nájdenie gemometrických útvarov je možné predpokladať, že som s veľkou pravdepodobnosťou detekoval dopravnú značku.

4. *Klasifikácia dopravnej značky* – po úspešnom nájdení dopravnej značky je vhodné určiť akú dopravnú značku som detekoval. Na klasifikáciu existuje v dnešnej dobe veľa prístupov. Asi najznámejší je algoritmus *adaboost*. Je to sofistikovaný a robustný algoritmus, pomocou ktorého je možné klasifikovať prakticky všetko. Ale mojim cieľom nebolo použitie klasifikátorov pri samotnej detekcii dopravnej značky a využitie robustného adaboostu na samotné určenie názvu dopravnej značky je podľa môjho názoru zbytočné. Bolo by potrebné vyrobiť klasifikátor samostatne pre každú dopravnú značku. Z týchto dôvodov a aj preto, že v zadaní nebolo vyžadované dopravnú značku klasifikovať som sa rozhodol využiť algoritmus *template matching*. Je to algoritmus ako sa píše v [13], ktorý sa využíva pri rozpoznávaní malých častí v obrázku. Viac o template matchingu je uvedené v sekcii 1.4.2.

Ako vstupné dáta bude systém požadovať obrázok v niektorom zo štandarných formátov (bmp, jpeg, png, ...). Jeho výstupom bude v prípade čisto detekcie pôvodný obrázok s vyznačenou dopravnou značkou a na štandardný výstup sa vypíšu súradnice ľavého horného a pravého dolného vrcholu obdĺžnika, ktorý vyznačuje nájdenú dopravnú značku. V prípade, keď bude požadovaná aj klasifikácia bude výstupom pôvodný obrázok s vyznačenou dopravnou značkou a popisom o akú dopravnú značku sa jedná, na štandardný výstup sa vypíšu súradnice rovnako ako pri detekcii a navyše aj názov dopravnej značky. Celkový návrh systému je zobrazený na obrázku 2.1.



Obrázok 2.1: Návrh systému pre detekciu dopravných značiek

2.2 Popis implementácie

Implemetácia navrhnutého systému bola urobená podľa prevedenej analýzy. Bolo ptrebné vybrať si vhodný programovací jazyk. Zvažoval som viac možností, či už *matlab*, *java*, prípadne niektorí zo skriptovacíh jazykov (python, perl, ...). Vedúcim práce mi bola odporučená knižnica *OpenCV*, ktorú je možné využiť v skriptovacom jazyku *python*, alebo v programovacom jazyku *C/C++*. Keďže interpretované jazyky, ku ktorým patrí *python*, sú oveľa pomalšie ako kompilované, ku ktorým patrí *C/C++*, rozhodol som sa použiť pre implementáciu môjho systému jazyk *C/C++* s knižnicou *OpenCV*.

2.2.1 Knižnica OpenCV

OpenCV (Open source Computer Vision) je knižnica pôvodne vyvinutá firmou Intel. Ako už z názvu vyplýva, je to voľne šíritelná knižnica s otvorenými zdrojovými kódmi, takže môže každý prispieť k jej vylepšeniu. Ako sa píše v [1], knižnica *OpenCV* je napísaná v jazyku C/C++, je možné ju teda v ňom aj použiť, ale je možnosť použiť túto knižnicu aj v jazyku python. Je použiteľná ako pod systémom *Linux*, tak aj *Windows* a MacOS X. Jej najväčšie využitie je v oblasti počítačového videnia v reálnom čase. Príklady použitia *OpenCV* knižnice v praxi: rozpoznávanie objektov, segmentácia obrazu, rozpoznávanie tváre v obraze,rozpoznávanie gést, ... Túto knižnicu využívam najmä na načítanie vstupného obrázku a zobrazenie výsledku.

2.2.2 Popis jednotlivých častí systému

Ako je vidieť na obrázku 2.1, rozhodol som sa rozdeliť systém na tri základné časti. Vstupom do systému je RGB obrázok, v ktorom chceme detekovať dopravnú značku:

- *Segmentácia farieb*: Je to prvá časť systému, takže pracuje priamo so vstupným obrázkom. Farebná škála, ktorá je požadovaná, aby v obrázku zostala je pomocou RG farebného modelu a dvojrozmernej gaussovej krivky uložená v súbore určujúcom súradnice jej stredu , jej rozptyl v oboch smeroch a minimálnu hodnotu. Názov súboru sa zadáva pri spustení programu, ak nie je zadaný použije sa implicitne súbor `red.rg`. Tento súbor je vytvorený a dodaný spolu zo systémom ako demonštračný príklad, ale je možné vytvoriť si vlastný pomocou programu `rg`, ktorý je tiež súčasťou. Výstupom tejto časti systému je binárny obraz, kde biela farba je pre požadovanú farbu a šedá pre zvyšok. Šedá farba bola zvolená kvôli najlepším odozvám pri hľadaní rohov pomocou konvolučných masiek.

  Program `rg` sa spúšťa s parametrom, ktorý určuje názov obrázka, z ktorého budeme vyberať požadovanú farebnú škálu. Po vybratí požadovanie farebnej škály program vytvorí v RG farebnom priestore dvojrozmernú gaussovu krivku (viď obrázok 1.2) a jej parametre uloží do súboru. Tento proces stačí urobiť raz, prípadne ak by sme sa chceli v budúcnosti orientovať na inú farebnú škálu, je možnosť vytvoriť nový súbor.

- *Nájdenie geometrických útvarov*: Táto časť systému je rozdelená na dve podčasti:

  - *nájdenie rohov*: Vstupom je binárny obraz, v ktorom budú hľadané rohy. Na vyhľadanie sa použije diskrétna konvolúcia s konvolučnými maskami pre každý typ rohu zvlášť. Po prevedení konvolúcie je jeden roh označený viac krát, čo je nežiadúce. Toto je možné odstrániť pomocou metódy *non-maxima supression*. Je to metóda pomocou, ktorej sa v určitej oblasti nájde maximum to je ponechané

a ostatné hodnoty sa nastavia na nulu. Potom sa uložia súradnice nájdeného rohu samostane pre každý typ zvlášť.

  - *rozpoznanie útvarov*: Ako vstup požaduje súradnice rohov jednotlivých typov. Prevedie sa algoritmus na rozpoznanie geometrických útvarov uvedený v sekcii 1.3. Po úspešnom nájdení sú súradnice ľavého horného a pravého dolného rohu obĺžnika ohraničujúceho nájdený geometrický útvar, pravdepodobne dopravnú značku, buď použité na zvýraznenie nájdenej dopravnej značky a vypísanie na štandardný výstup, alebo použité ďalej pri klasifikácii.

- *Klasifikácia dopravných značiek*: Dopravnú značku som úspešne našiel, zostáva ju iba klasifikovať. Aby som mohol previesť samotnú klasifikáciu je potrebné znormalizovať výrez obsahujúci nájdenú značku, ktorý je ohraničený súradnicami získaným z detekcie na veľkosť zhodnú so vzorom, s ktorým sa bude porovnávať.

  - *normalizácia nájdenej značky*: Vstupom pre normalizáciu je výrez nájdenej značky v RGB farebnom priestore. Normalizácia, v mojom prípade zväčšenie (respektíve zmenšenie) výrezu, je prevedená pomocou algoritmu *nearest-neighbor interpolation*. Po prevedení normalizácie je výsledkom pôvodný výrez s rozmermi 85px × 85px, ktorý je aj jej výstupom.
  - *klasifikácia*: Ako vstup požaduje normalizovanú dopravnú značku, ktorá sa má klasifikovať a vzor pomocou, ktorého sa má klasifikovať. Pre lepšiu prácu som sa rozhodol použiť pre zadávanie vzorov *XML súbor*, v ktorom je uvedená relatívna cesta k vzorovému súboru a presný názov dopravnej značky. Zo *XML súboru* sa vyberajú postupne vzory a prevádza sa template matching so vstupným normalizovaným výrezom, obsahujúcim dopravnú značku. Po prevední template matchingu dostanem výsledok, na koľko percent sa daný vzor zhoduje s výrezom. Potom je vybratý najlepší, ktorý musí zároveň spĺňať aj minimálnu hranicu zhody a podľa neho sa určí presný názov dopravnej značky vo výreze. Výstupom sú súradnice ohraničujúce klasifikovanú dopravnú značku a jej presný názov.

2.2.3 Obmedzenia systému

Ako každý systém aj tento má svoje obmedzenia. Pri jeho tvorbe bola použitá knižnica `OpenCV`, takže pri kompilácii zo zdrojových súborov je nutné mať nainštalovanú túto knižnicu. Pre samotné používanie vopred zkompilovaných zdrojových textov to už nie je potrebné. Systém bol testovaný len na operačnom systéme Linux (konkrétne aj na servery merlin), takže jeho plnú funkčnosť môžem zaručiť iba pod týmto operačným systémom.

Vstupné obrázky sú načítané pomocou funkcie `cvLoadImage` z knižnice `OpenCV`. Funkcia podporuje tieto formáty súborov: Windows Bitmap (BMP, DIB), JPEG súbory (JPEG, JPG, JPE), Portable Nerwork Graphics (PNG), prenosný formát obrázku (PBM, PGM, PPM), Sun rastre (SR, RAS), TIFF súbory (TIF, TIFF).

Z časových dôvodov bola do systému naimplementovaná iba detekcia a klasifikácia zákazových dopravných značiek. Ale systém je navrhnutý tak, aby bolo možné jednoduché rozšírenie jeho funkčnosti aj pre iné typy dopravných značiek.

Kapitola 3

Testovanie systému

Po naimplementovaní systému bolo potrebné previesť otestovanie či systém pracuje správne. Pri testovaní systémov, ktoré sa zaoberajú či už detekciou alebo rozpoznávaním objektov v obraze sa skúmajú tieto faktory:

- *detection rate (dr)* – percentuálne vyjadrenie koľko z požadovaných objektov systém správne detekoval.
- *false positives (fp)* – počet nesprávne nájdených objektov v obraze.
- *false negatives (fn)* – počet detekcií, ktoré nemali byť detekované, ale systémom boli označené ako hľadané objekty.
- *false detections (fd)* – súčet *false positives* a *false negatives*.

Pre testovanie je potrebná testovacia sada, ktorú som nedokázal nikde zohnať. Preto som sa rozhodol vytvoriť svoju vlastnú testovaciu sadu pre detekciu zákazových dopravných značiek.

3.1 Testovacia sada

Môj systém požaduje ako vstup obrázok vo formáte, ktorý dokáže knižnica `OpenCV` načítať. Tieto formáty sú uvedené v sekcii 2.2.3. Rozhodol som sa zvoliť formát JPG, kvôli jeho menšej veľkosti, a rozmery obrázkov 800px × 566px. Obrázky boli fotené za plnej premávky a za bieleho dňa, ale pri rôznych svetelných podmienkach. Snažil som sa fotiť dopravné značky, na ktoré svietilo slnko aj tie, ktoré boli v tieni. Testovacia sada bola vyrábaná najmä pre detekciu zákazových dopravných značiek, pretože môj systém, ako bolo uvedené v sekcii 2.2.3, v jeho súčasnej podobe dokáže detekovať iba zákazové dopravné značky.

Po nafotení dopravných značiek bolo potrebné ich anotovať. Anotáciu som sa rozhodol urobiť tak, že k testovacej sade je pridaný textový súbor, v ktorom sú informácie o každom obrázku zvlášť na riadku. Informácie sú oddelené bodkočiarkou a popis ku každému obrázku je na samostatnom riadku pre lepšie spracovanie automatickým skriptom.
Príklad anotačného súboru:

```
nazov suboru;pocet znaciek;x1;y1;x2;y2;Nazov;
1.jpg;2;544;50;640;145;339;239;415;325;Stop;Zakaz vjazdu;
2.jpg;1;232;45;456;354;Zakaz zastavenia;
```

V ktorom x1,y1 sú súradnice ľavého horného rohu a x2,y2 pravého dolného rohu, obdĺžnika ohraničujúceho dopravnú značku. Počet značiek predstavuje počet anotovaných dopravných značiek v obrázku. Názov značky je braný do úvahy iba ak je robený test pre klasifikáciu. Je potrebné, aby anotačný súbor bol ukončený prázdnym riadkom.

3.2 Výsledky testovania

	dr	**fp/obrázok**	**fn/obrázok**	**fd/obrázok**
detekcia	77,52 %	2,19	0,29	2,48
1. klasifikácia	62,96 %	0,33	0,37	0,70
2. klasifikácia	55,56 %	0,31	0,44	0,75

Tabuľka 3.1: Testy prevedené na systéme detekcie a klasifikácie dopravných značiek.

V tabuľke sú uvedené výsledky mnou prevedených testovaní. Pri detekcii je síce úspešnosť relatívne dosť vysoká, ale aj hodnota nesprávne označených detekcií je veľmi vysoká. Nesprávne detekcie by sa dali čiastočne odstrániť prísnejšími kritériami na farbu dopravnej značky. Prípadne zmenšiť povolené maximálne posunutie rohov, jeden vzhľadom na ostatné, alebo maximálne sploštenie kruhu na elipsu. Pomocou týchto úprav sa ale potom znižuje úspešnosť nájdených značiek, keďže dopravné značky môžu mať rôzne odtiene červenej farby a pri odfotografovaní z uhla môžu nadobudnúť tvar elipsy. Preto som sa rozhodol urobiť kompromis medzi úspešnosťou a nesprávnymi detekciami. Výsledkom tohto kompromisu sú výsledky v tabuľke 3.1.

Ďalšou možnosťou ako znížiť počet zlých detekcií je využiť klasifikácie, týmto sa odstráni väčšina nežiadúcich detekcií. V tabuľke sú výsledky od dvoch rôznych testov. *Prvá klasifikácia* bola prevedená so vzormi, ktoré boli vybrané z reálnych obrázkov a nebolo im zmenené pozadie. Pri *druhej klasifikácii* boli použité vzory, ktoré boli síce z reálnych obrázkov, ale pozadie dopravnej značky bolo konštantne, pre všetky vzory, nastavené na bielu farbu. Ako je z výsledkov testovania vidieť, pri klasifikácii pomocou metódy *template matching*, je lepšie keď je pozadie vzorov členité ako konštantnej farby. Pri využití takejto jednoduchej klasifikácie som dosiahol to, čo som potreboval, odstránil som prebytočné detekcie. I keď je pravda, že klasifikácia niekedy zle zaradila značku a pridelila jej iný význam. Pre presnejšiu a citlivejšiu klasifikáciu by bolo potrebné využiť sofisitkovanejšej metódy akou je napriklád algoritmus *adaboost*.

Záver

Cieľom mojej práce bolo naštudovať známe metódy pre detekciu dopravných značiek a naimplementovať systém, ktorý by zvládol značky v obraze detekovať. Prácu som sa snažil písať tak, aby jej porozumel aj človek, ktorý sa s podobnou problematikou nikdy nestretol. V kapitole 1 som priblížil ako sa v dnešnej dobe k tejto problematike pristupuje. Sú v nej popísané rôzne prístupy k detekcii dopravných značiek. Niektoré sú viac, iné menej preferované, ale jednoznačne sa ani pri jednej nedá povedať, ktorá metóda je úplne zlá a ktorá tá najlepšia. Preto je vždy na programátorovi akou cestou sa vyberie

Po preštudovaní problematiky som sa rozhodol naimplementovať môj systém tak ako je uvedené v kapitole 2. V tejto kapitole som sa snažil priblížiť prečo som si vybral tú metódu, ktorú som naimplementoval. Snažil som sa priblížiť ako som konkrétnu metódu implementoval. Následujúca kapitola sa zaoberala testovaním systému. Testovania som robil aj priebežne počas implemetácie, kde som odhalil niekoľko chýb, ktoré sa mi podarilo odstrániť. V tejto kapitole sú uvedené výsledky testov prevedených na mnou vytvorenej a anotovanej testovacej sade. Z výsledkov testovania je zrejmé, že na systéme je ešte čo zlepšovať.

Na systéme by bolo možné v budúcnosti zlepšiť samotnú detekciu, odstránením falošných detekcií, ktoré spomaľujú celý proces a sú nežiadúce. Ďalej by bolo možné pridať detekiu aj iných ako zákazových dopravných značiek a pri klasifikácii použiť sofistikovanejšiu metódu ako je template matching, čím by sa určite zvýšila úspešnosť dobre klasifikovaných dopravných značiek. Asi posledným prípadným rozšírením by mohla byť možnosť výberu medzi spracovávaním statického obrázku a videa.

Literatúra

[1] OpenCV Wiki. [online], [citované 23.Apríl 2008].
URL <http://opencvlibrary.sourceforge.net/>

[2] CHANG, C.-Y.; TU, Y.-C.; CHANG, H.-H.: *Adaptive Color Space Switching Based Approach for Face Tracking*, ročník 4233/2006, kapitola Neural Information Processing. Berlin / Heidelberg: Springer, 2006, ISBN 978-3-540-46481-5, s. 244–252, [citované 18.Apríl 2008].
URL
<http://www.springerlink.com/content/g2472610t6766r43/fulltext.pdf>

[3] DEVERNAY, F.: A Non-Maxima Suppression Method for Edge Detection with Sub-Pixel Accuracy. Technická Zpráva RR-2724, [citované 26.Apríl 2008].
URL <ftp://ftp.inria.fr/INRIA/publication/publi-pdf/RR/RR-2724.pdf>

[4] de la ESCALERA, A.; MORENO, L. E.; SALICHS, M. A.; aj.: Road Traffic Sign Detection and Classification. *IEE Transactions on Industrial electronics*, ročník 44, č. 6, December 1997: s. 848–859, ISSN 0278-0046, [citované 17.Apríl 2008].
URL <http://turan.uc3m.es/uc3m/dpto/IN/dpin04/IE3tie97.pdf>

[5] FREUND, Y.; SCHAPIRE, R.: A short introduction to boosting. [online], 1999, [citované 19.Apríl 2008].
URL <citeseer.ist.psu.edu/freund99short.html>

[6] KUMAR, S.: RGB to HSI Color Conversion. [online], [citované 17.Apríl 2008].
URL <http://web2.clarkson.edu/class/image_process/RGB_to_HSI.pdf>

[7] Wikipedia: Bicubic interpolation – Wikipedia, The Free Encyclopedia. 2008, [Online; citované 21.Apríl 2008].
URL <http://en.wikipedia.org/w/index.php?title=Bicubic_interpolation&oldid=205331932>

[8] Wikipedia: Bilinear interpolation – Wikipedia, The Free Encyclopedia. 2008, [Online; citované 21.Apríl 2008].
URL <http://en.wikipedia.org/w/index.php?title=Bilinear_interpolation&oldid=207086541>

[9] Wikipedia: Canny edge detector – Wikipedia, The Free Encyclopedia. 2008, [Online; citované 19.Apríl 2008].
URL <http://en.wikipedia.org/w/index.php?title=Canny_edge_detector&oldid=198396334>

[10] Wikipedia: Gaussian function – Wikipedia, The Free Encyclopedia. 2008, [Online; citované 18.Apríl 2008].
URL <http://en.wikipedia.org/w/index.php?title=Gaussian_function&oldid=205528837>

[11] Wikipedia: Nearest-neighbor interpolation – Wikipedia, The Free Encyclopedia. 2008, [Online; citované 20.Apríl 2008].
URL <http://en.wikipedia.org/w/index.php?title=Nearest-neighbor_interpolation&oldid=205573540>

[12] Wikipedia: RG color space – Wikipedia, The Free Encyclopedia. 2008, [Online; citované 18.Apríl 2008].
URL <http://en.wikipedia.org/w/index.php?title=RG_color_space&oldid=186039897>

[13] Wikipedia: Template matching – Wikipedia, The Free Encyclopedia. 2008, [Online; citované 21.Apríl 2008].
URL <http://en.wikipedia.org/w/index.php?title=Template_matching&oldid=195657056>

[14] Wikipedie: Konvoluce – Wikipedie, Otevřená encyklopedie. 2008, [Online; citované 19.Apríl 2008].
URL
<http://cs.wikipedia.org/w/index.php?title=Konvoluce&oldid=2260113>

[15] ŠONKA, M.; HLAVÁČ, V.: *Počítačové vidění*. Praha: Grada a.s., 1992, ISBN 80-85424-67-3.

Dodatok A

Konvolučné masky

-6	-11	-11	-6	0	**4**	**7**	**7**	**4**
-11	-18	-18	-11	0	**8**	**13**	**13**	**8**
-11	-18	-18	-11	0	**10**	**17**	**17**	**10**
-6	-11	-11	-6	0	**12**	**19**	**19**	**12**
0	0	0	0	0	**12**	**20**	**20**	**12**
-6	-11	-11	-6	0	-6	-11	-11	-6
-11	-18	-18	-11	0	-11	-18	-18	-11
-11	-18	-18	-11	0	-11	-18	-18	-11
-6	-11	-11	-6	0	-6	-11	-11	-6

Tabuľka A.1: ľavý dolný 90° roh

-6	-11	-11	-6	0	-6	-11	-11	-6
-11	-18	-18	-11	0	-11	-18	-18	-11
-11	-18	-18	-11	0	-11	-18	-18	-11
-6	-11	-11	-6	0	-6	-11	-11	-6
-6	-11	-11	**12**	**12**	**12**	-11	-11	-6
-11	-18	-18	**19**	**20**	**19**	-18	-18	-11
-11	-18	**17**	**19**	**20**	**19**	**17**	-18	-11
-6	-11	**10**	**12**	**12**	**12**	**10**	-11	-6
0	**12**	**20**	**20**	**12**	**20**	**20**	**12**	0

Tabuľka A.2: horný 60° roh

-6	-11	-11	-6	0	-6	**7**	**7**	**4**
-11	-18	-18	-11	0	-11	**13**	**13**	**8**
-11	-18	-18	-11	0	**10**	**17**	**17**	**10**
-6	-11	-11	-6	0	**12**	**19**	**19**	**12**
0	0	0	0	0	**12**	**20**	**20**	**12**
-6	-11	-11	-6	0	-6	-11	-11	-6
-11	-18	-18	-11	0	-11	-18	-18	-11
-11	-18	-18	-11	0	-11	-18	-18	-11
-6	-11	-11	-6	0	-6	-11	-11	-6

Tabuľka A.3: ľavý dolný 60° roh

Tieto konvolučné masky boli použité z [4]. Ich rotáciou je možné získať ostatné typy rohov.