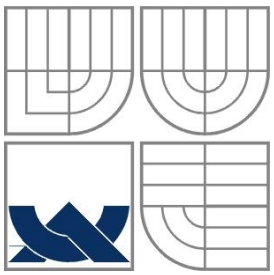
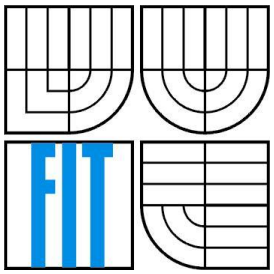


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ROZPOZNÁNÍ STAVU HRY SCRABBLE

SCRABBLE GAME STATE RECOGNITION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JIŘÍ STANĚK

VEDOUCÍ PRÁCE
SUPERVISOR

ING. PAVEL SVOBODA

BRNO 2013

Abstrakt

Tato práce se zabývá problematikou zpracování obrazu, detekce a rozpoznávání objektů v obraze a OCR (Optical Character Recognition). Popisuje návrh a implementaci aplikace pro rozpoznání stavu hry Scrabble využívající některé algoritmy z této oblasti. Je zde popsána detekce hrací desky a jednotlivých písmen v obraze. Dále klasifikace detekovaných písmen a tvorba slov. Výsledná aplikace využívá knihovny OpenCV pro zpracování obrazu a LIBSVM pro klasifikaci písmen. Aplikace byla testována na vlastní sadě fotografií.

Abstract

This thesis deals with the image processing, object detection and recognition and OCR (Optical Character Recognition). It describes the design and implementation of an application for Scrabble game state recognition using some algorithms in this area. It describes the detection of the game board in the image and the letters on it, classification of detected letters and word formation. The final application uses the OpenCV library for image processing and the LIBSVM library for letter classification. The application was tested on a custom set of photos.

Klíčová slova

zpracování obrazu, Houghova transformace, Scrabble, OCR, klasifikace, LIBSVM, OpenCV, detekce hrací desky, detekce písmen

Keywords

image processing, Hough transform, Scrabble, OCR, classification, LIBSVM, OpenCV, game board detection, letters detection

Citace

Staněk Jiří: Rozpoznání stavu hry Scrabble, bakalářská práce, Brno, FIT VUT v Brně, 2013

Rozpoznání stavu hry Scrabble

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Pavla Svobody.
Další informace mi poskytl Ing. Lukáš Polok.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jiří Staněk
15. května 2013

Poděkování

Rád bych poděkoval vedoucímu práce Ing. Pavlovi Svobodovi za jeho trpělivost a vstřícný přístup. Bez jeho cenných rad a pomoci by tato práce nemohla vzniknout. Také bych chtěl poděkovat Ing. Lukášovi Polokovi za poskytnutý dataset využitý při trénování klasifikátoru.

Obsah

1 Úvod

V dnešní době zažíváme rychlý rozvoj technologií, hlavně počítačů. S tím souvisí rozvoj počítačového vidění a zpracování obrazu. S počítačovým viděním se setkáváme ve všech oblastech každodenního života. Ať už je to při lékařských vyšetřeních, v moderních automobilech, kamerovém zabezpečení různých objektů, ale také v různých aplikacích, například pro mobilní zařízení.

Cílem této práce je prostudovat metody používané v počítačové grafice pro zpracování obrazu a detekci objektů. Dále metody pro rozpoznávání a klasifikaci jednotlivých znaků abecedy. Z těchto metod jsou poté vybrány ty, které jsou potřebné při návrhu a realizaci aplikace pro rozpoznání stavu hry Scrabble. Tato aplikace má za úkol nalézt na vstupu, což je fotografie hrací desky, všechna zahraná slova.

Pro realizaci zmíněné aplikace je třeba uvědomit si jednotlivé kroky, které nás dovedou k cíli. Na fotografii je většinou nějaké pozadí, které nás nezajímá a naopak by nás rušilo v dalších krocích. Proto si popíšeme, jak detekovat samotnou hrací desku. Dále potřebujeme získat jednotlivé čtverečky s písmeny. Popíšeme si, jakým způsobem detekovat a procházet všechny čtverečky na hrací ploše a jak rozhodnout, zda čtvereček obsahuje písmeno, či nikoliv. Nalezená písmena musíme klasifikovat, tj. rozpoznat o jaké písmeno se jedná. K tomu poslouží klasifikační metoda Support Vector Machines.

V kapitole 2 si popíšeme hru Scrabble, abychom věděli, jak vypadá hrací deska a jednotlivá písmena a také, jakým způsobem jsou tvořena slova, abychom je pak mohli detekovat.

V kapitole 3 budou popsány použité metody pro zpracování obrazu, detekci objektů a rozpoznání a klasifikaci jednotlivých znaků abecedy.

Kapitola 4 pojednává o samotné aplikaci. Nejprve si popíšeme návrh aplikace, tj. jakým způsobem použijeme popsané metody, abychom se dopracovali k cíli. Dále v této kapitole bude popsána implementace výsledné aplikace. Popíšeme si použité knihovny (OpenCV a LIBSVM) a implementaci některých metod.

Testování výsledné aplikace je uvedeno v kapitole 5. Zde budou vyhodnoceny výsledky nejdůležitějších částí aplikace, což jsou detekce a klasifikace písmen. Bude poukázáno na chyby v aplikaci a na její kritické části.

V Závěru bude uvedeno shrnutí dosažených výsledků a otázka budoucí práce.

2 Hra Scrabble

Hru hrají dva až čtyři hráči na čtvercové hrací desce, obsahující 15x15 políček, každé pro jedno písmeno. Hra obsahuje 100 písmen, z nichž hráči skládají slova. Každé písmeno má bodové ohodnocení. Ve hře jsou 2 žolíci, za které není žádný bod. Bodové ohodnocení je od 1 bodu do 10 bodů a je založeno na frekvenci výskytu písmen v daném jazyce.

2.1 Pravidla

Před začátkem hry se hráči dohodnou na slovníku, který se bude používat. Jsou povolena všechna slova, která jsou součástí daného jazyka (slova převzatá, zastaralá, hovorová, slangová,…). Výjimku tvoří názvy, zkratky, předpony a přípony stojící osamoceně a slova vyžadující pomlčku nebo apostrof.

Postup hry je následovný:

1. První hráč vytvoří slovo ze dvou nebo více svých písmen a umístí jej na hrací desku tak, aby jedno z těchto písmen bylo umístěno na prostředním čtverečku hrací plochy. Není povoleno umísťovat slova po diagonále.
2. Po umístění slova je spočítáno bodové ohodnocení tohoto slova a hráč si vytáhne tolik nových písmen, kolik jich právě zahrál (musí jich mít před sebou sedm).
3. Hra pokračuje po směru hodinových ručiček. Druhý hráč a poté i každý další přidá jedno nebo více písmen k písmenům již umístěným na hrací desce tak, aby vytvořil alespoň jedno nové slovo. Všechna nová písmena musí být umístěna v jednom řádku nebo sloupci hrací desky. Takto umístěná písmena musí tvořit slova se všemi písmeny na hrací desce, kterých se dotýkají (jako v křížovce). Hráč dostává body za všechna slova, která takto vytvořil nebo upravil.
4. Nová slova mohou být vytvořena:
    - Přidáním jednoho nebo více písmen k písmenům již umístěným na hrací desce.
    - Umístěním slova v pravém úhlu ke slovu, které je na hrací desce. V novém slově musí být použito jedno z písmen, které je již umístěno na hrací ploše.
    - Umístěním celého slova paralelně ke slovu již umístěnému na hrací ploše tak, že dotýkající se písmena musí rovněž tvořit slova.
5. Žádné písmeno nesmí být přesunuto nebo nahrazeno poté, co bylo zahráno a ohodnoceno.
6. Žolíci: Ve hře jsou dva žolíci (bílý čtvereček), kteří mohou být zahráni místo jakéhokoliv písmene. V případě umístění žolíka musí hráč určit, které písmeno zastupuje. Toto písmeno bude zastupovat až do konce hry.
7. Hráč může využít své kolo k výměně všech, nebo jen některých svých písmen. Písmena, která chce vyměnit, položí bokem a vytáhne si odpovídající počet nových písmen. Odložená písmena vrátí do pytlíku. Tím jeho kolo končí.
8. Kterýkoliv hráč může vyzvat k ověření správnosti právě zahraného slova (či slov). V tom případě je slovo/a zkontrolováno podle předem dohodnutého slovníku. Pokud se všechna nově vytvořená slova ve slovníku nachází, hráč, který vyzýval, ztrácí své další kolo. V opačném případě hráč, který byl na řadě, si vezme zpět právě umístěná písmena a hraje další hráč.

9. Hra končí, když byla vytažena všechna písmena (pytlík je prázdný) a některý z hráčů umístil své poslední písmeno, nebo již není možné nic umístit.

Obrázek 2.1: Ukázka hry Scrabble.

3 Použité metody pro zpracování obrazu a klasifikaci

Tato kapitola se podrobněji zabývá metodami, které byly použity ve výsledné aplikaci. Je zde popsáno předzpracování obrazu, které je důležité pro jeho další digitální zpracování. Dále jak nalézt objekty pomocí detekce hran a jak mezi nimi rozpoznat ty, které nás zajímají.

Také si popíšeme klasifikační metody užité pro OCR (Optical Character Recognition), tj. rozpoznání jednotlivých znaků.

3.1 Převod z barevného prostoru

Každý pixel barevného obrazu je reprezentován třemi hodnotami, jednou pro každou barevnou složku. Každá složka nabývá hodnoty v rozmezí 0-255, což nám dává 24 bitů na jeden pixel. To je zbytečně mnoho dat. Jelikož většinou nepotřebujeme pracovat s barvami, můžeme data zredukovat, aniž bychom přišli o podstatnou informaci. To nám usnadní a hlavně urychlí práci s obrazem.

3.1.1 Grayscale

Tato funkce převádí barevný obraz na obraz ve stupních šedi. Šedá barva vznikne, když všechny tři barevné složky pixelu (R, G, B) mají stejnou intenzitu [1]. Díky tomu, dojde ke zjednodušení obrazu, jelikož každý pixel je reprezentován pouze jednou hodnotou intenzity, namísto tří hodnot pro barevný obraz.

Existuje více rovnic pro převod barevného obrazu do stupňů šedi, významné jsou tyto [2][3]:

$$Grey = 0.299R + 0.587G + 0.114B \quad (1)$$

$$Grey = 0.2126R + 0.7152G + 0.0722B \quad (2)$$

$Grey$ je výsledná intenzita pixelu a R, G, B jsou intenzity jednotlivých barevných složek pixelu v původním obrazu.

Lidské oko je různě citlivé na barevné složky. Nejcitlivější je na zelenou složku a nejméně citlivé na složku modrou. Z tohoto důvodu je dobré použít rovnici (2), jelikož dává největší váhu zelené složce.

3.2 Detekce hran

Při zpracování obrazu nás zajímají objekty, které jsou v něm zobrazeny. Detekce objektů v obraze spočívá ve zvýraznění jejich hran.

Hranové detektory hledají změny intenzity v obraze. Hrany jsou pixely, v jejichž okolí dochází k prudké změně intenzity [4].

3.2.1 Cannyho hranový detektor

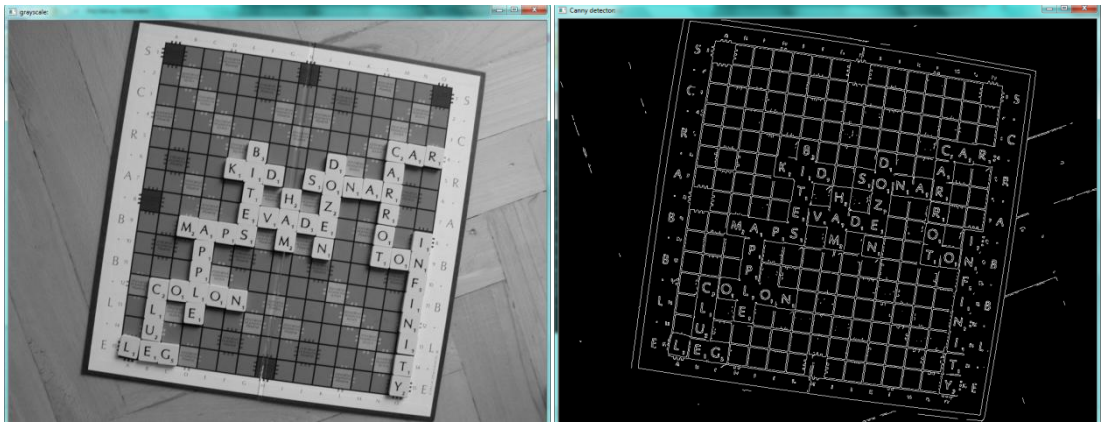
Cannyho hranový detektor byl navržen J. F. Cannym roku 1986. Jeho cílem bylo navrhnout optimální algoritmus, jehož optimálnost vychází z těchto tří kritérií [4][5]:

- Dobrá detekce. Musí být detekovány všechny hrany a nesmí být detekována místa, která hranami nejsou.
- Dobrá lokalizace. Body detekované hrany musí být co nejblíže středu skutečné hrany.
- Pouze jedna odezva na jednu hranu. Každá hrana musí být detekována pouze jednou.

Algoritmus pracuje v několika krocích [1]. Nejprve je obraz vyhlazen pomocí Gaussovy konvoluce. Poté je na vyhlazený obraz aplikován jednoduchý operátor (např. Sobel operator) pro základní nalezení hran.

Dále dochází ke ztenčení hran, tj. odstranění bodů, které nejsou maximem (non-maximal suppression). V tomto kroku procházíme osmi-okolí daného hranového pixelu a odstraňujeme pixely s nízkou intenzitou. Tím zajistíme, že hrana bude detekována v místě nejvýraznější změny intenzity.

Poté následuje proces prahování, který je řízen dvěma prahy $T1$ a $T2$ ($T1 > T2$). Začíná na pixelu s intenzitou vyšší než $T1$ a pokračuje v obou směrech od pixelu, dokud intenzita neklesne pod $T2$. Toto pomáhá zachovat celistvost nevýrazných hran.



Obrázek 3.1: Vlevo původní obrázek, vpravo obrázek po aplikaci Cannyho hranového detektoru.

3.3 Segmentace

Segmentace slouží k rozdělení obrazu na části, které mají souvislost s objekty nebo oblastmi reálného světa zachycenými v obraze [4].

3.3.1 Thresholding

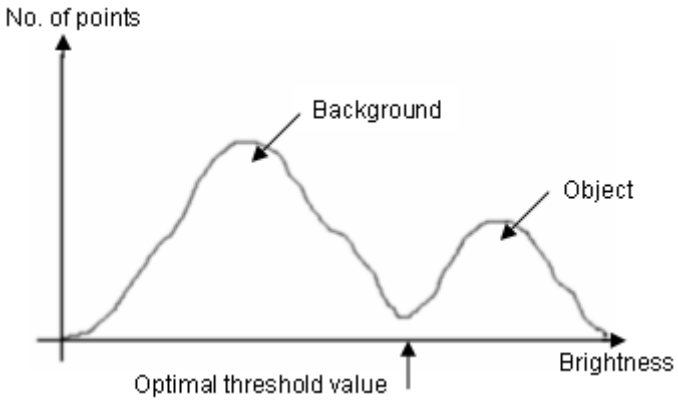
Thresholding (prahování) je funkce dělící obraz na objekty a pozadí, jelikož tyto prvky mají rozdílnou intenzitu.

Prahování je transformace vstupního obrazu f na výstupní binární obraz g následovně [4]:

$$\begin{aligned} g(i,j) &= 1\ pro\ f(i,j) \geq T, \\ &= 0\ pro\ f(i,j) < T, \end{aligned} \tag{3}$$

kde T je práh, $g(i,j) = 1$ pro objekty a $g(i,j) = 0$ pro pozadí.

Práh je zvolen na základě histogramu. Tam, kde lze jednoznačně oddělit objekty od pozadí je zvolen globální práh (na základě celého obrazu). V opačném případě je obraz rozdělen na části a v každé části je zvolen lokální práh. To je nazýváno adaptivní prahování.



Obrázek 3.2: Ukázka histogramu.
Převzato z: http://www.dandiggins.co.uk/arlib-3.html.

3.3.2 Houghova transformace

Cílem Houghovy transformace je nalézt instance objektů určité třídy tvarů. Princip Houghovy transformace je popsán v [4][6].

Metoda pracuje na základě hlasování, které se provádí v prostoru parametrů. Kandidáti objektů jsou vybráni jako lokální maxima v tzv. akumulátoru, který je zkonstruován při výpočtu Houghovy transformace.

Základní Houghova transformace byla určena k identifikaci přímek, později byla rozšířena k identifikaci tvarů, které lze analyticky popsat. Těmito tvary jsou nejčastěji kružnice, elipsy a čtverce. Nás bude zajímat způsob identifikace přímek.

Přímku lze vyjádřit parametrickou rovnicí:

$$y = mx + b \tag{4}$$

Úpravou této rovnice získáme rovnici přímky v m, b parametrickém prostoru:

$$b = -xm + y \tag{5}$$

Hlavním omezením této rovnice je, že parametry m, b jsou neohraničeny. Pokud bodem prochází přímka ve vertikálním směru, tak se oba parametry blíží nekonečnu. Řešením je použít jinou rovnici přímky:

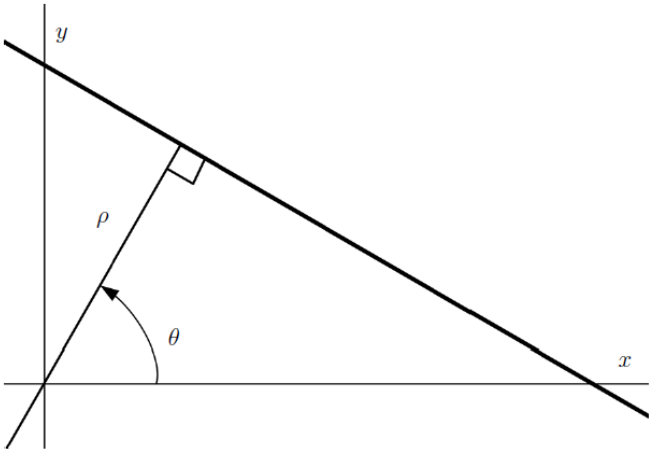
$$\rho = x * \cos\theta + y * \sin\theta \tag{6}$$

Zde je přímka definována pozicí k počátku souřadnic.

Při výpočtu je zkonstruována imaginární přímka spojující počátek souřadnic s nejbližším bodem na zkoumané přímce. Koeficient ρ je vzdálenost zkoumané přímky od počátku souřadnic a θ je úhel mezi imaginární přímkou a x-ovou osou.

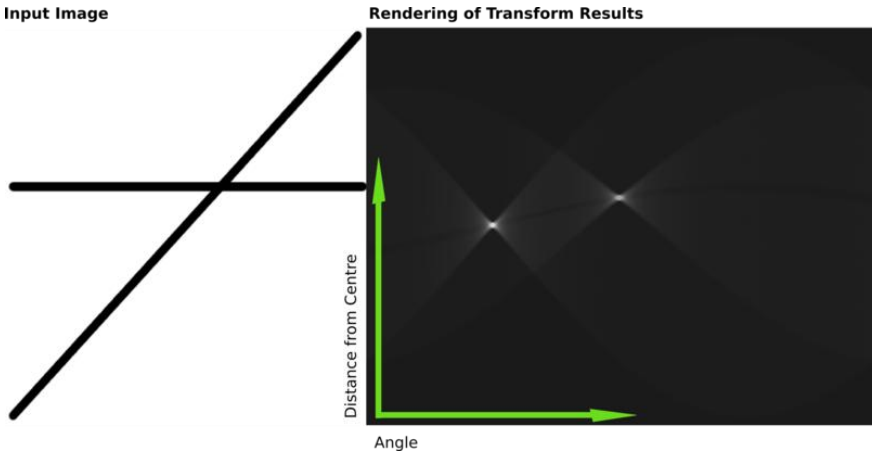
Algoritmus 1 Houghova transformace

for each edge point
  for $(\theta = \theta min;\ \theta \leq \theta max;\ \theta += \Delta\theta)$
    $\rho = x * cos\theta + y * sin\theta$
    Accumulator$[\rho][\theta] + +$
  end for
end for



Obrázek 3.3: θ a ρ parametry přímky.
Převzato z: [6]

Každým bodem může procházet nekonečně mnoho různých přímek, z nichž každá má rozdílnou dvojici parametrů ρ, θ. Všechny tyto přímky tvoří jednu křivku v ρ, θ parametrickém prostoru. Poté, co jsou všechny body převedeny do ρ, θ parametrického prostoru, je nutné najít hlavní průsečíky představující přímky. Tyto průsečíky jsou lokální maxima v již zmíněném akumulátoru.



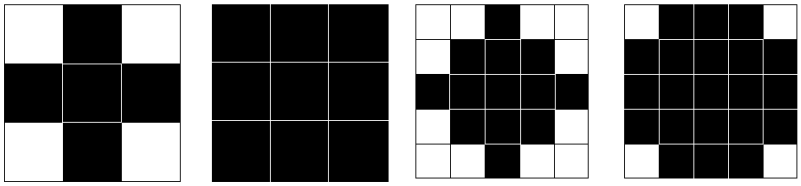
Obrázek 3.4: Ukázka akumulátoru pro dvě přímky.
Převzato z: http://en.wikipedia.org/wiki/Hough_transform

3.4 Matematická morfologie

Matematická morfologie zahrnuje širokou škálu operátorů pro zpracování obrazu. Tyto operátory jsou použity pro analýzu binárních obrazů, detekci hran, odstranění šumu, segmentaci,…

Vstupem všech těchto operátorů je binární obraz a strukturní element. Tyto dva prvky jsou zkombinovány na základě nastaveného operátoru (sjednocení, průnik, doplněk,…). Strukturní element se skládá ze vzoru, specifikovaného jako souřadnice určitého počtu bodů vztažené k počátku, což je u strukturních elementů prostřední bod.

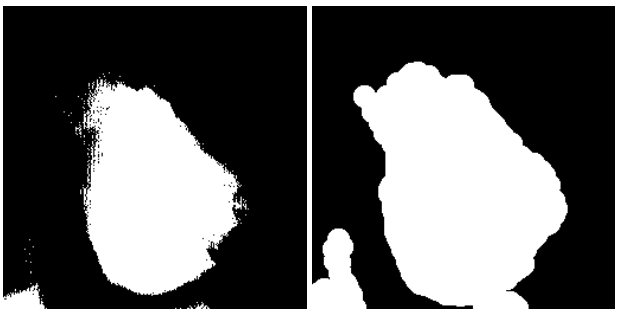
Při aplikaci operátoru se postupně prochází pixely obrazu, střed strukturního elementu je zarovnáván s těmito pixely a následně jsou porovnávány body strukturního elementu s odpovídajícími pixely v obraze. Výsledek tohoto porovnání záleží na daném operátoru [1].



Obrázek 3.5: Příklady strukturních elementů.

3.4.1 Dilatace

Dilatace pracuje na základě vektorového součtu, kdy binární obraz kombinuje s předdefinovaným strukturním elementem. Slouží ke zvýraznění objektů, k zaplnění malých děr a úzkých zálivů [4].



Obrázek 3.6: Vlevo originální obrázek, vpravo obrázek po aplikaci dilatace.
Převzato z: http://www.inf.ufsc.br/~visao/khoros/html-dip/c9/s4/front-page.html

3.4.2 Eroze

Eroze je duální operací k dilataci, nejedná se však o inverzní operace. Eroze pracuje s vektorovým rozdílem. Slouží ke zjednodušení struktury objektu, některé jeho části s malou šířkou zmizí. Může dojít k rozdělení objektu na několik jednodušších.

Díky erozi lze získat obrys objektu. Dosáhneme toho odečtením erodovaného obrázku od originálního [4].

Obrázek 3.7: Vlevo originální obrázek, vpravo obrázek po aplikaci eroze.
Převzato z: http://www.inf.ufsc.br/~visao/khoros/html-dip/c9/s4/front-page.html

3.5 Gaussovo rozmazání

Při popisu bylo vycházeno z [4].
Jedná se o rozmazání obrazu pomocí Gaussovy funkce. Používá se k odstranění šumu a redukci detailů.

Rovnice Gaussovy funkce ve 2-D:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{7}$$

kde x je vzdálenost od počátku na x-ové ose, y je vzdálenost od počátku na y-ové ose a σ je standardní odchylka Gaussovy distribuce.

Výsledkem rovnice jsou hodnoty, které jsou použity ke zkonstruování konvoluční matice, která je aplikována na originální obrázek. Hodnota každého pixelu je spočítána jako váhový průměr hodnot pixelů z jeho okolí. Originální hodnota pixelu získává nejvyšší váhu a sousedním pixelům se váha snižuje se vzrůstající vzdáleností od originálního pixelu. To pomáhá zachovat hranice objektů.

3.6 Klasifikace

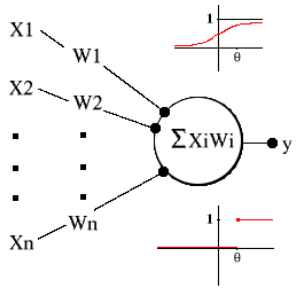
Klasifikace slouží k přiřazení objektů ke třídám, které je reprezentují. V našem případě budou těmito objekty znaky abecedy. Na základě klasifikace bude každý z těchto znaků přiřazen k jedné z předem známého počtu tříd. Tím zjistíme, o jaké písmeno se jedná. Dále budou stručně popsány dva druhy klasifikátorů.

3.6.1 Neuronové sítě

Problematika neuronových sítí je popsána v [7].
Neuronové sítě jsou složeny z elementárních prvků (neuronů). Tyto sítě mají tzv. učící schopnost, kdy na jednom konci sítě vkládáme příklady a na druhém řešení a tato síť je schopna znalosti vstřebat. Pokud síti zadáme příklad, který nebyl v množině učících příkladů, neuronová síť je přesto často schopna vygenerovat správnou odpověď, což nazýváme generalizací.

Každý neuron má určitý počet vstupů a generuje jeden výstup. Pokud součet vstupů vynásobených vahami přesáhne určitou mez (práh), neuron vrátí na výstupu hodnotu 1. V opačném

případě vrátí 0. Pokud vstupem neuronu budou souřadnice bodu ve dvourozměrném prostoru, je tento neuron schopen od sebe oddělit dvě množiny bodů přímkou (samozřejmě, jen pokud je to možné). Neurony se dělí na diskrétní a spojité.



Obrázek 3.8: Ukázka neuronu.
Převzato z: http://www.ibm.com/developerworks/library/l-neural/

Z neuronů lze tvořit vrstevnaté sítě, kdy všechny neurony z jedné vrstvy jsou spojeny jen s neurony vyšší vrstvy. Takto lze klasifikovat průniky libovolného počtu různých polorovin.

Učící schopnost sítě vychází z konceptu backpropagation, neboli algoritmu zpětného šíření. V případě jednoho neuronu se tento neuron na začátku nachází ve stavu, kdy má nějak nastavené váhy. Na vstup dostane učící vzor a na výstup odpověď. Úkolem učící fáze je minimalizovat chybu pro všechny učící vzory změnou nastavení svých vah. U vrstevnaté sítě algoritmus aplikujeme nejprve na výstupní neurony a chybu přenášíme do nižších vrstev až ke vstupním neuronům.

3.6.2 Support Vector Machines

Při popisu bylo vycházeno z [8].
Support Vector Machine (SVM) je druh klasifikátoru založený na vektorovém prostoru, jehož hlavním cílem je najít oddělující hranici mezi dvěma třídami, která je maximálně vzdálena od všech trénovacích datových bodů. Vzdálenost od hranice k nejbližšímu datovému bodu se nazývá okraj klasifikátoru. Tyto nejbližší body se nazývají podpůrné vektory, neboli support vectors (Obrázek 3.9). Metoda hledá hranici s maximálním okrajem, jelikož body ležící blízko hranice vedou k velmi nepřesnému rozhodování.

Oddělující hranice je definována parametrem b a normálovým vektorem $\vec{w}$, který je na ni kolmý (též váhový vektor). Klasifikační funkce vypadá následovně:

$$f\left(\vec{x}\right) = sign\left(\vec{w}\,T\vec{x} + b\right) \tag{8}$$

kde $\vec{x}$ je bod z trénovací sady.
Definujeme si funkční hranici bodu $\vec{x_i}$ a řekneme, že tato hranice musí být alespoň 1:

$$y_i\left(\vec{w}\,T\vec{x_i} + b\right) \geq 1 \tag{9}$$

Vzdálenost každého bodu od hranice je:

$$r_i = \frac{y_i\left(\vec{w}\,T\vec{x_i} + b\right)}{\left|\vec{w}\right|} \tag{10}$$

Poté hranice ρ je vyjádřena vztahem:

$$\rho = \frac{2}{\left|\vec{w}\right|} \tag{11}$$

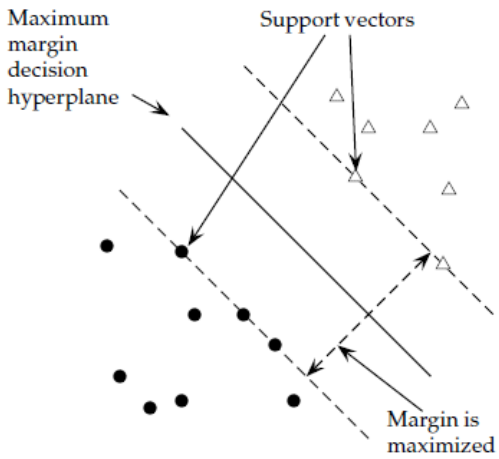
Jelikož hledáme maximální hranici, musíme tento vztah maximalizovat, což je to stejné, jako minimalizovat vztah:

$$\frac{\left|\vec{w}\right|}{2} \tag{12}$$

Díky tomu dostaneme standardní formulaci SVM jako minimalizačního problému, kdy hledáme $\vec{w}$ a b takové, že:

- Vztah (12) je minimalizován
- a pro všechny datové body $\left\{\left(\vec{x_i}, y_i\right)\right\}$ platí vztah (9).

Podrobnější popis SVM lze nalézt v [4][8].



Obrázek 3.9: Hranice a podpůrné vektory.
Převzato z: [8]

4 Aplikace rozpoznání stavu hry

4.1 Cíle a požadavky

Cílem práce bylo vytvořit již zmíněnou aplikaci pro rozpoznání stavu hry Scrabble. Aplikace musí pracovat v reálném čase.

Důležitým prvkem aplikace je detekce a klasifikace písmen, která je zároveň i prvkem kritickým. Ani komerční řešení nenabízí stoprocentní úspěšnost a proto ji nelze očekávat od výsledné aplikace. Proto zde bude důležitá snaha o co nejlepší úspěšnost při detekci a klasifikaci písmen.

4.1.1 Vstupní data

Vstupními daty je fotografie hrací desky. Hrací deska by na fotografii měla převládat (pozadí by nemělo být dominantní). Dnešní chytré mobilní telefony mají většinou kvalitní fotoaparáty s velkým rozlišením, proto budeme předpokládat, že na vstupu bude dostatečně kvalitní snímek, jehož kvalitu nebude třeba nijak dále upravovat. Hrací deska by měla být vyfocena pokud možno kolmo k fotoaparátu s tím, že aplikace by si měla poradit s mírným zkreslením (rybí oko, vyfoceno z úhlu), jaké ukazuje Obrázek 4.1.

Obrázek 4.1: Zkreslená hrací deska.

4.2 Návrh aplikace

Aplikaci můžeme rozdělit do několika kroků, které si dále popíšeme:

- Předzpracování obrazu
- Detekce matice s písmeny
- Klasifikace písmen a tvorba slov

4.2.1 Předzpracování obrazu

Vstupní obraz je třeba připravit tak, aby v něm bylo možné nalézt hrací desku. Musíme si uvědomit první krok, který je potřeba udělat. Hrací deska na vstupním obraze asi nikdy nebude přesně zarovnaná, to znamená, že může být různě natočená. Teoreticky by se dalo pracovat i s natočenou hrací deskou, ovšem to by byla zbytečná komplikace, obzvlášť když řešení tohoto problému není nijak složité.

Nejprve je potřeba nalézt hrany. K tomu je použit Cannyho hranový detektor. Ten pracuje s obrazem ve stupních šedi, proto musíme vstupní obraz převést pomocí funkce grayscale. Dále pomocí Houghovy transformace nalezneme nejvýznamnější směr, podle něhož obrázek natočíme. Díky tomu se nám bude lépe detekovat samotná hrací deska. Použitá Houghova transformace bude popsána dále.

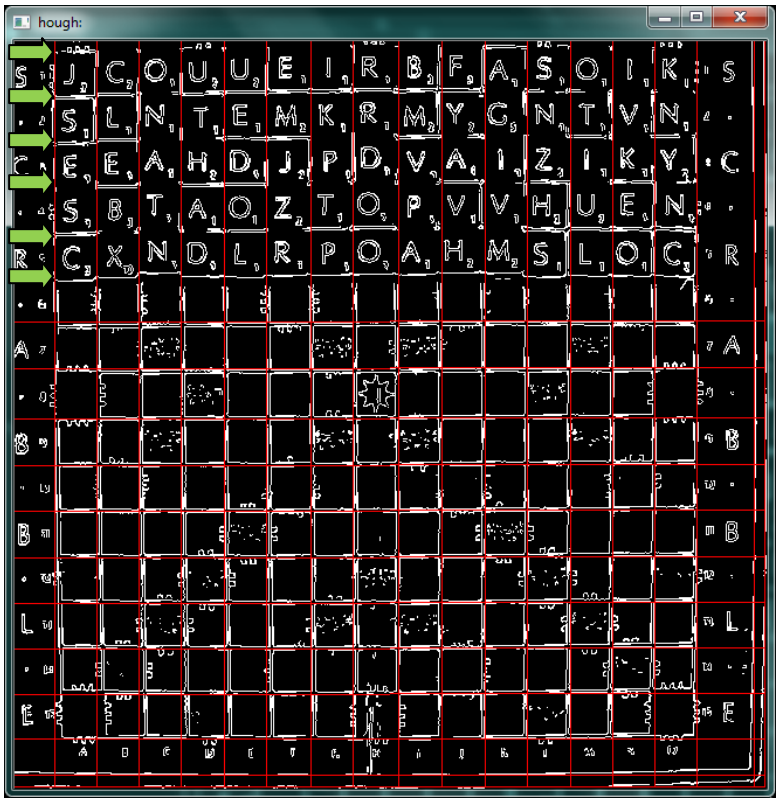
4.2.2 Detekce matice s písmeny

Dalším krokem, který je potřeba udělat před detekcí samotných písmen, je detekce matice s písmeny, tj. plochy na hrací desce, kam hráči umisťují svá písmena.

Vycházíme z toho, že máme zarovnaný obrázek. Nyní potřebujeme vyříznout samotnou hrací desku. K tomu nám poslouží Houghova transformace, kdy najdeme nejvýznamnější směry, díky nimž detekujeme rohy hrací desky a tu pak vyřízneme (podrobněji popsáno dále).

Dále potřebujeme z hrací desky získat matici s písmeny. To provedeme stejně jako detekci celé hrací desky s tím rozdílem, že zde nás bude zajímat, zda je matice na obrázku celá a není nějak oříznuta. Tato kontrola spočívá v tom, že budeme počítat jednotlivé čáry mezi čtverečky. Houghova transformace ovšem nemusela detekovat veškeré čáry (Obrázek 4.2), proto bude potřeba chybějící čáry doplnit. To provedeme tak, že zjistíme nejčastější vzdálenost mezi čárami, které máme detekovány. Poté procházíme jednotlivé čáry, které jsou od sebe ve vzdálenosti, kterou jsme zjistili jako nejčastější (s určitým rozmezím) a tam kde čára chybí, ji doplníme. Následně spočítáme všechny tyto čáry, a jelikož máme matici 15x15 čtverečků, tak čar musí být 16x16. Není-li tomu tak, prohlásíme vstup za neplatný. V opačném případě matici vyřízneme.

Matice je většinou nějak deformovaná, což je dáno deformací vstupního obrazu, proto provedeme interpolaci do čtverce.



Obrázek 4.2: Ukázka chybějících čar po Houghově transformaci.

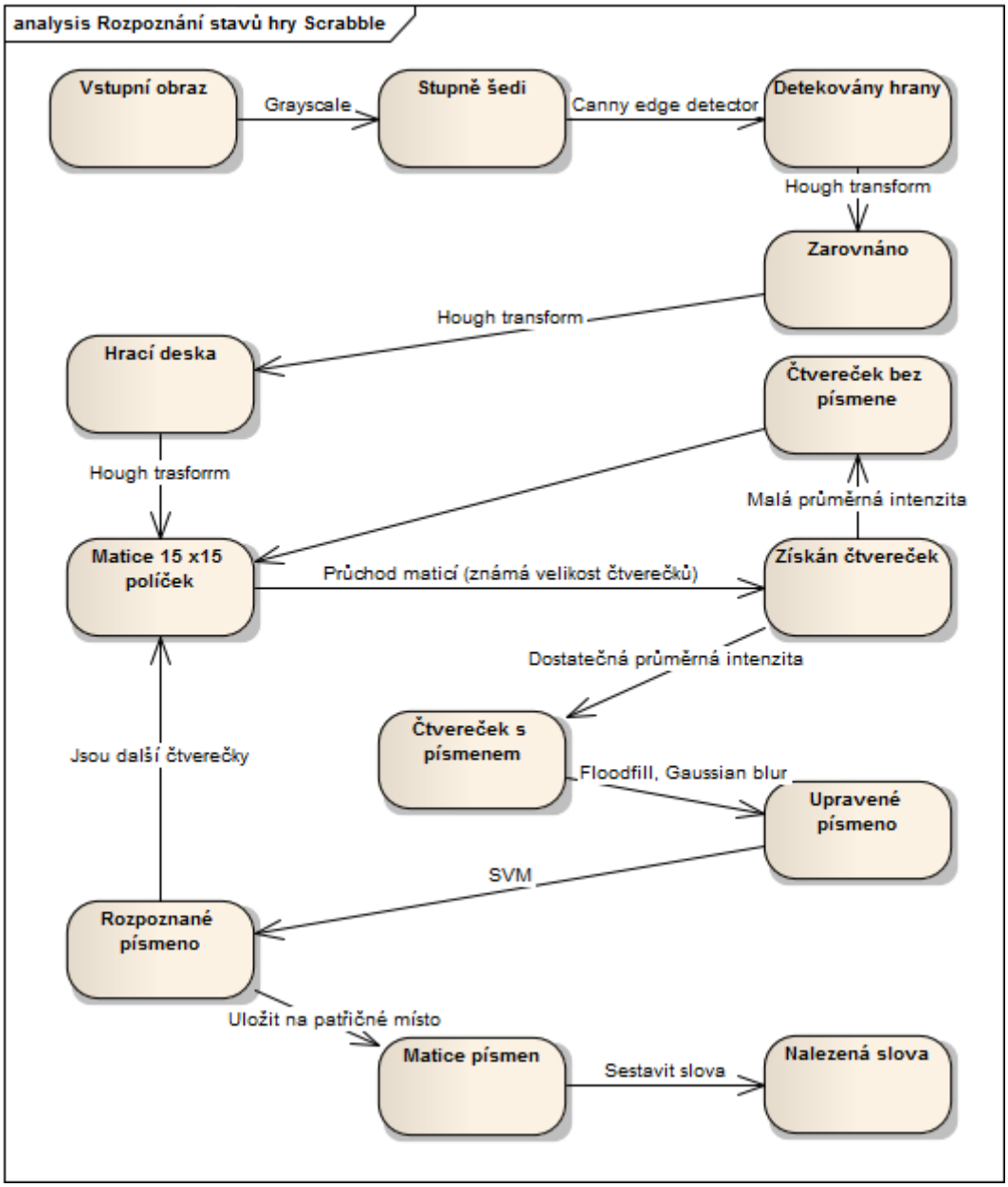
4.2.3 Klasifikace písmen a tvorba slov

V předchozím kroku jsme získaly matici s písmeny. Nyní musíme jednotlivá písmena nalézt a klasifikovat.

Matici máme interpolovanou do čtverce. Víme, že je na ní 15x15 políček. Z toho plyne, že rozměry matice můžeme podělit 15, z čehož získáme rozměry jednoho čtverečku (matici si musíme zvětšit či zmenšit tak, aby nám vyšla celá čísla). Nyní můžeme pohodlně procházet maticí s krokem rovnajícím se rozměru jednoho čtverečku a postupně tyto čtverečky vyřezávat a dále s nimi pracovat.

U každého čtverečku je třeba rozhodnout, zda obsahuje písmeno, či nikoliv. Budeme zjišťovat průměrnou intenzitu čtverečků, a pokud její hodnota přesáhne určitou mez, prohlásíme daný čtvereček za čtvereček s písmenem. Takovýto čtvereček je třeba před samotnou klasifikací upravit (bude popsáno dále).

Nakonec potřebujeme písmeno klasifikovat, tj. zjistit o jaké písmeno se jedná. K tomu použijeme knihovnu LIBSVM (popsáno dále). Jednotlivá písmena budeme ukládat do matice, ze které ke konci získáme jednotlivá slova (každé slovo má 2 a více znaků).



Obrázek 4.3: Stavový diagram aplikace.

4.3 Implementace

Tato část se soustředí na vlastní implementaci výsledného programu. Vychází z rozdělení použitého u návrhu aplikace. V každé části je popsána realizace stěžejních bodů.

Aplikace je implementována pouze na PC. Implementace na mobilní telefony je otázkou budoucí práce. Program je napsán v jazyce C++ za použití knihoven OpenCV a LIBSVM.

OpenCV je volně dostupná knihovna pro digitální zpracování obrazu. Obsahuje více než 2500 algoritmů zahrnujících počítačové vidění a strojové učení. Tato knihovna umožňuje například detekci a rozpoznávání tváří, identifikaci objektů, sledování pohybujících se objektů na videu atd. Má rozhraní pro jazyky C++, C, Python a Java a běží na operačních systémech Windows, Linux, Android a Mac OS [9].

LIBSVM je knihovna pro klasifikaci na základě podpůrných vektorů, distribučního odhadu (one-class SVM) a podporuje multi-třídní klasifikaci. Poskytuje jednoduché rozhraní pro použití v uživatelských programech. Knihovna zahrnuje různé formulace SVM, multi-třídní klasifikaci, cross validaci pro vytvoření modelu, odhad pravděpodobnosti atd. Obsahuje rozhraní pro jazyky C++, Java, Python, Matlab, Perl, Ruby a mnoho dalších [10].

4.3.1 Houghova transformace

Rozhodl jsem se pro vlastní implementaci Houghovy transformace. Tato transformace je stěžejní částí programu a proto pro mě bylo důležité přesně znát její chování a výstupy.

Běžná Houghova transformace hledá v obraze segmenty čar. Chceme-li z těchto segmentů získat přímky, musíme vzít pozici přímky v ρ, θ parametrickém prostoru (zajímá nás hodnota ρ a θ) a vypočítat kudy přímka prochází. Toto by se dalo eliminovat použitím m, c parametrického prostoru, z kterého by bylo možné získat přímo obecnou rovnici přímky. Jak jsme si ale řekli dříve, toto řešení selhává na vertikálních přímkách, kdy se oba parametry blíží nekonečnu. Proto je nutné použít ρ, θ parametrický prostor. Výpočet přímek z tohoto prostoru se mi zdál zbytečně složitý a chtěl jsem přímky získávat snáze. K čemu vlastně potřebujeme přímky? Při hledání hrací desky, či matice s písmeny musíme najít rohy dané oblasti. Tyto rohy jsou reprezentovány průsečíky přímek procházejících sousedními hranicemi oblasti. Proto nám nestačí pouhé segmenty těchto hranic, jelikož ty se většinou neprotnou.

Pro usnadnění hledání přímek jsem si vytvořil strukturu `Hough`, která reprezentuje jeden prvek v ρ, θ parametrickém prostoru. Mimo počet bodů procházejících daným segmentem do této struktury ukládám i souřadnice jeho krajních bodů. Tyto body nám určují přímky, jejichž průsečíky můžeme snadno vypočítat (bude popsáno dále).

Při výpočtu Houghovy transformace musíme určit dva parametry: `rhoMax`, což je maximální vzdálenost přímky od počátku a rozmezí θ udávající počet přímek, které daným bodem prochází (každá v jednom úhlu z rozmezí θ). Tyto parametry také určují rozměry akumulátoru reprezentovaného maticí struktur `Hough`.

Jako počátek jsem označil střed obrázku. Z toho plyne, že `rhoMax` má hodnotu poloviny délky úhlopříčky, což lze vypočítat pomocí Pythagorovy věty. Běžně se užívá rozmezí θ velikosti 180°, zde je zvoleno rozmezí θ 0°-360°, tudíž budeme prokládat přímky daným bodem ve všech směrech, což nám usnadní detekci hrací desky a matice s písmeny (sekce 4.3.3).

Samotný výpočet vychází z Algoritmus 1. Ve zmíněném algoritmu je počítáno s počátkem v rohu obrázku. My potřebujeme tento počátek posunout doprostřed, což je zajištěno následujícím výpočtem:

$$rho = (int)\left(\left(x - \frac{width}{2}\right) * \cos(theta)\right) + \left(\left(y - \frac{height}{2}\right) * \sin(theta)\right) \qquad (13)$$

Dále už jen zkontrolujeme, zda vypočítané rho spadá do vymezeného intervalu $\langle 0, rhoMax)$ a uložíme potřebné hodnoty do akumulátoru.

4.3.2 Předzpracování obrazu

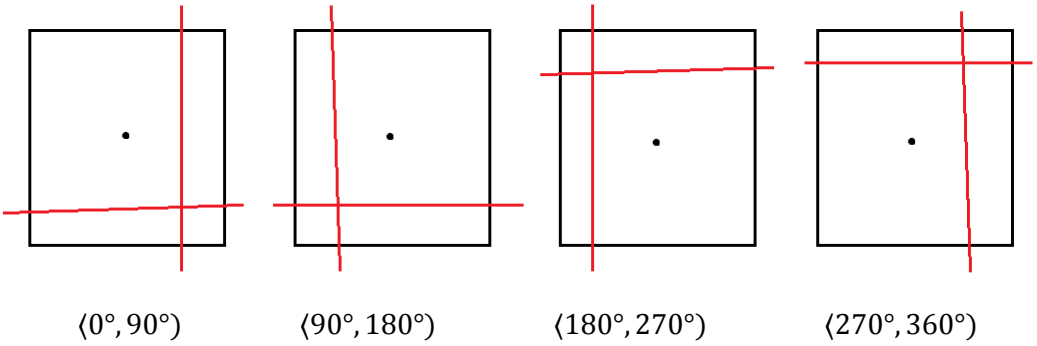
Předzpracování obrazu spočívá v nalezení hran a zarovnání.

Převod do stupňů šedi je implementován za pomoci rovnice (2), která přepočítá hodnotu každého pixelu. Pro nalezení hran je použita funkce z knihovny OpenCV `cvCanny`, která pracuje se dvěma prahy. Tyto prahy byly zvoleny na základě testování a jejich hodnota je 150 a 200.

Po nalezení hran potřebujeme obrázek zarovnat. K tomu je použita Houghova transformace. Po aplikaci této transformace je nutné najít směr, podle kterého bude obrázek zarovnán. K tomu slouží dvě funkce `findMax` a `findDirection`. První jmenovaná nalezne maximální hodnotu v akumulátoru v zadaném intervalu úhlu θ. Vezmeme-li v úvahu, že hrací deska má zhruba tvar čtverce, můžeme interval θ užitý při Houghově transformaci $\langle 0°, 360° \rangle$ rozdělit na čtyři stejné intervaly, přičemž v každém z nich bude jeden roh hrací desky. Poté nám stačí hledat maximální hodnotu akumulátoru pouze v jednom z těchto intervalů. Zvolen byl interval $\langle 0°, 90° \rangle$. V tomto intervalu nalezneme maximální hodnotu. Tato hodnota nám udává nejvýznamnější čáru, která ovšem nemusí mít s naší hrací deskou nic společného (chybná detekce, pozadí,…). My potřebujeme najít nejčastější směr. Vezmeme polovinu nalezené maximální hodnoty, čímž se zbavíme méně významných čar. Funkce `findDirection` hledá nejvýznamnější směr na základě počtu čar v daném úhlu. Pro každý úhel spočítá počet čar a vrátí ten úhel, pro nějž byl tento počet největší. Tím získáme směr, podle kterého zarovnáme obrázek. O zarovnání se stará funkce z knihovny OpenCV `warpAffine`.

4.3.3 Detekce matice s písmeny

Pro detekci hrací desky a poté i matice s písmeny je potřeba si uvědomit, co jsme získaly zvolenou Houghovou transformací.



Obrázek 4.4: Detekce čar Houghovou transformací v jednotlivých intervalech.

Obrázek 4.4 ukazuje, jaké čáry jsou detekovány v jednotlivých intervalech, s tím, že veškeré čáry jsou detekovány od kraje obrázku po střed obrázku. Jelikož interval θ má velikost 360°, jsou intervaly rozděleny po 90° stupních.

Po zarovnání obrazu jsou jednotlivé hrany hrací desky téměř vodorovné s odpovídajícími hranami obrázku. Z toho plyne, že v každém z těchto čtyř intervalů najdeme jednu hranu hrací desky. Budeme hledat nejvýznamnější směr v každém intervalu tak, jak bylo popsáno v předchozí sekci. Jelikož potřebujeme pouze okraje hrací desky, postupujeme od kraje obrázku k jeho středu a hledáme první významnou čáru v daném směru. Takto získáme čtyři čáry značící okraje hrací desky.

Pro vyříznutí hrací desky potřebujeme znát souřadnice jejích rohů. Jak bylo řečeno, do akumulátoru mimo jiné ukládáme i souřadnice krajních bodů nalezených segmentů čar. Díky těmto dvěma bodům můžeme snadno vypočítat rovnici přímky, která daným segmentem prochází. Máme body $A[x_1, y_1]$ a $B[x_2, y_2]$:

1. Vypočítáme souřadnice normálového vektoru pomocí následujících dvou rovnic.

$$\begin{aligned} n_1 &= y_2 - y_1 \\ n_2 &= x_1 - x_2 \end{aligned} \tag{14}$$

2. Tyto souřadnice dosadíme do obecné rovnice přímky:

$$ax + by + c = 0 \tag{15}$$

za parametry a, b:

$$n_1 x + n_2 y + c = 0 \tag{16}$$

3. Dosadíme souřadnice jednoho z bodů a vypočítáme parametr c.

Nyní vypočítáme průsečík dvou přímek procházejících dvěma sousedními hranicemi oblasti, kterou potřebujeme detekovat. Z výše uvedeného postupu jsme získaly parametry c obou přímek, takže celý problém můžeme zapsat jako soustavu dvou rovnic:

$$\begin{aligned} a_1 x + b_1 y + c_1 &= 0 \\ a_2 x + b_2 y + c_2 &= 0 \end{aligned} \tag{17}$$

kde a_1, b_1, a_2 a b_2 jsou souřadnice normálových vektorů přímek. Řešením této soustavy rovnic získáme souřadnice průsečíku.

Nakonec potřebujeme najít samotnou matici s písmeny. Postup je stejný jako u detekce hrací desky. Zde ovšem potřebujeme otestovat, zda se v obrázku nachází celá matice s písmeny (není nijak oříznutá). K tomu slouží funkce `editAccumulator` a `findPlayArea`.

První jmenovaná je pouze pomocná funkce, která upraví v akumulátoru hodnotu všech významných čar, které nás v tomto kroku budou zajímat, na konstantu. To proto, abychom v dalším kroku mohli pohodlně procházet jednotlivé čáry a nemuseli testovat, zda je daná čára významná, či nikoliv.

Hlavní funkcí v tomto kroku je funkce `findPlayArea`. Tato funkce prochází všechny čáry vždy ve dvou rovnoběžných směrech (snadnému procházení čar pomohla funkce `editAccumulator`). Při průchodu testuje vzdálenost sousedních čar. Všechny tyto vzdálenosti jsou ukládány do pole struktur `Dist` spolu s počtem jejich výskytu (tolerance vzdálenosti je +- 1). Ze získaných vzdáleností je zvolena ta nejčastější.

Nyní přichází na řadu detekce celistvosti hrací desky. Nejprve procházíme obraz od kraje do středu s krokem rovnajícím se nejčastější vzdálenosti a značíme jednotlivé čáry. Díky tomu doplníme chybějící čáry, které nebyly detekovány Houghovou transformací (Obrázek 4.2). Poté procházíme obraz od středu ke kraji a počítáme označené čáry. V každém směru by jich mělo být osm. Není-li tomu tak, je obraz prohlášen za neplatný vstup. V opačném případě poslední (osmá) čára značí okraj oblasti. Poté vypočítáme průsečíky okrajů a oblast vyřízneme. Získanou oblast interpolujeme do čtverce pomocí funkce `cvWarpPerspective`.

4.3.4 Klasifikace písmen a tvorba slov

V tomto stádiu máme matici s písmeny. Víme, že je na ní 15x15 čtverečků, čehož využijeme pro jejich detekci. Změníme velikost obrazu na 600x600 pixelů. Díky tomu pak jednotlivé čtverečky budou mít velikost 40x40 pixelů. Obraz procházíme s krokem 40 pixelů v obou směrech a

vyřezáváme jednotlivé čtverečky. U každého čtverečku musíme rozhodnout, zda je v něm písmeno. Toho dosáhneme pouhou kontrolou průměrné intenzity.

Obrázek 4.5: Detekce písmene uvnitř čtverečku.

Nemůžeme ovšem kontrolovat průměrnou intenzitu celého čtverečku jelikož zde mohou být okraje, které povedou k falešné detekci (Obrázek 4.5). Proto je potřeba čtvereček trochu oříznout. Zvolena byla oblast o rozměrech 16x16 pixelů uvnitř čtverečku. Je vypočítána průměrná intenzita této oblasti, a pokud její hodnota přesahuje hodnotu 30, ve čtverečku se vyskytuje písmeno. Tato hodnota byla zvolena na základě testování, kdy dávala nejlepší výsledky.

Jak je vidět (viz Obrázek 4.5), tak ve čtverečku je kromě písmene a okrajů také index udávající počet bodů. Pro klasifikaci ovšem potřebujeme samotné písmeno. Použijeme dilataci a erozi (3.4.1, 3.4.2), díky čemuž bude písmeno více spojité. Pomocí funkce `floodFill` nalezneme největší spojitý segment pixelů v ořízntutém čtverečku (opět by mohlo dojít k detekci okrajů). Tento segment značí naše písmeno. Písmeno připravené ke klasifikaci ukazuje Obrázek 4.6.

Obrázek 4.6: Příprava písmene ke klasifikaci. Vlevo čtvereček s písmenem, vpravo odstraněny rušivé části.

Pro klasifikaci písmen byla zvolena knihovna LIBSVM. Nejprve bylo potřeba natrénovat klasifikační model.

K tomuto účelu byl poskytnut interní dataset panem Lukášem Polokem[1]. Jedná se o sadu 240 obrázků ke každému z 26i písmen abecedy (anglická abeceda). Tyto obrázky mají velikost 32x32 pixelů. Ukázka pro písmeno A je na Obrázek 4.7. Nejprve bylo nutné každý obrázek převést na příznakový vektor (feature vector) a tyto vektory spojit do jednoho souboru, který slouží jako trénovací sada. Příznakový vektor je vektor struktur reprezentujících jednotlivé pixely obrazu. Struktura má tvar $a{:}\,b$, kde a je číslo pixelu a b je hodnota intenzity daného pixelu. Tuto hodnotu je potřeba převést do intervalu $\langle 0, 1\rangle$, což provedeme pouhým podělením intenzity pixelu hodnotou 255.

Obrázek 4.7: Ukázka datasetu pro písmeno A.

Máme-li trénovací sadu, můžeme začít s trénováním klasifikačního modelu. Byl zvolen jeden lineární multiklasifikátor (C-SVC). Trénování probíhá pomocí skriptu `grid.py`, který je umístěn v knihovně LIBSVM. Tento skript pracuje na principu *grid search*, kdy je vytvořena mřížka všech

[1] Ing. Lukáš Polok, Ústav počítačové grafiky a multimédií (UPGM), FIT VUT, ipolok@fit.vutbr.cz

kombinací parametrů (nás zajímá pouze parametr C), z nichž skript postupně vybírá jednotlivé kombinace parametrů a volá utilitu `svm-train` (opět přímo v knihovně LIBSVM) s těmito parametry. Během tohoto procesu probíhá cross validace.
Cross validace, jak je popsáno v [11], se používá při hledání parametrů modelu takových, aby klasifikátor mohl co nejpřesněji předpovědět neznámá data.

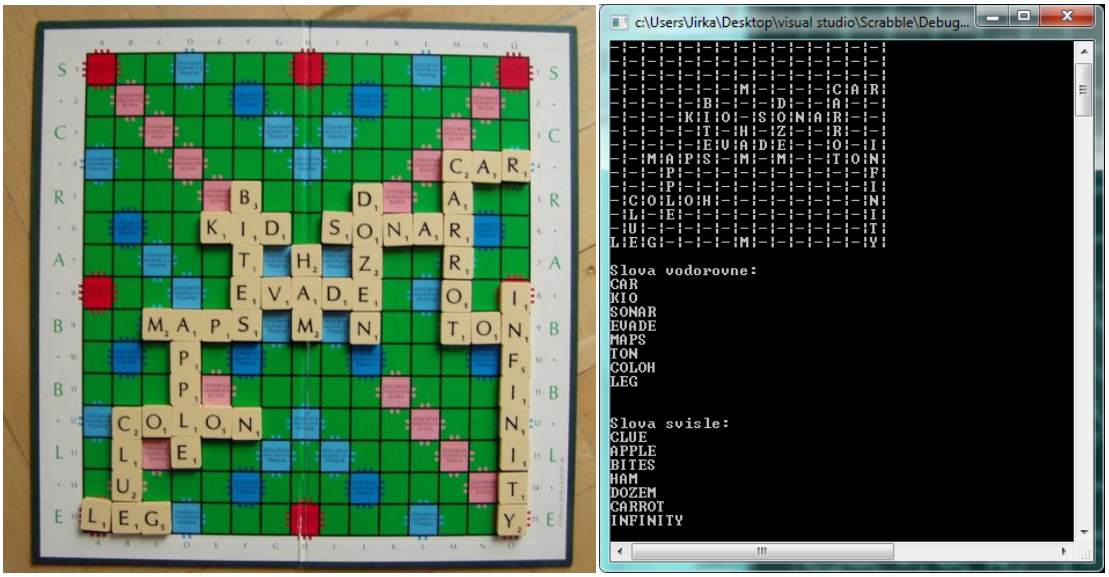
Základem je rozdělení souboru dat na dvě části, z nichž jedna je považována za neznámou (validační) a druhá za trénovací. Přesnost předpovědi získaná z validačního souboru dat udává schopnost klasifikovat nový neznámý soubor dat.

U v-fold cross validace je datová sada rozdělena na v stejně velkých podmnožin. Postupně je každá podmnožina (validační sada) testována klasifikátorem natrénovaným na ostatních podmnožinách (trénovací sada). Poté dochází k promíchání dat a novému rozdělení do podmnožin. Přesnost (accuracy) je pak procentní vyjádření množství správně klasifikovaných dat.

Na základě nejlepší přesnosti je vybrán odpovídající parametr C. S tímto parametrem je spuštěna utilita `svm-train` na celém datasetu, díky čemuž získáme finální klasifikační model.

V programu je model načten pomocí funkce `svm_load_model`. Dále je vytvořen příznakový vektor našeho detekovaného písmene a je zavolána funkce `svm_predict`, která vrací label třídy klasifikovaného písmene (v tomto případě je to pořadové číslo písmene v abecedě, číslováno od 0).

Toto číslo si uložíme do matice o velikosti 15x15 na místo, které odpovídá místu písmene na hrací desce. Nakonec tuto matici procházíme nejprve po řádcích, poté po sloupcích a hledáme jednotlivá slova. Výstup je proveden na příkazový řádek z důvodu chybějícího grafického uživatelského rozhraní (Obrázek 4.8).



Obrázek 4.8: Ukázka vstupu s odpovídajícím výstupem

5 Testování a vyhodnocení

V této kapitole bude popsáno testování výsledné aplikace. Popíšeme si výsledky a úspěšnost nejdůležitějších částí aplikace, jimiž jsou detekce písmen a jejich klasifikace.

5.1 Testovací sada

Byla nafocena řada obrázků hrací desky, na kterých byla aplikace testována. Byly zvoleny jak ideální obrázky (tj. kolmo k hrací desce, bez deformace), tak i obrázky, které měly aplikaci více otestovat (vyfoceno z úhlu, oříznutá hrací deska,…). Příklad vstupu ukazuje Obrázek 5.1.

Obrázek 5.1: Příklady vstupních obrázků. Vlevo ideální vstup, vpravo nevhodný vstup.

Dále byly zvoleny obrázky s různým počtem písmen, od prázdné desky až po umístění všech písmen. Všechny tyto obrázky byly vkládány na vstup aplikace a byly vyhodnoceny výsledky.

5.2 Výsledky

U obrázků, které byly vyfoceny z úhlu nebo na kterých byla hrací deska oříznuta, bohužel došlo k chybné detekci hrací desky a poté, při hledání matice s písmeny, byly obrázky prohlášeny za neplatný vstup. Nebyl prostor toto vyřešit a tak se dále budeme zabývat pouze ideálními vstupy.

Jak bylo řečeno, ideální vstup je takový, kdy je hrací deska téměř kolmo k fotoaparátu. Nevadí ani mírné natočení hrací desky. V takových případech došlo vždy ke klasifikaci písmen.

Důležitými prvky aplikace vhodnými k testování jsou správnost detekce písmene v jednotlivých čtverečcích a správnost klasifikace písmen.

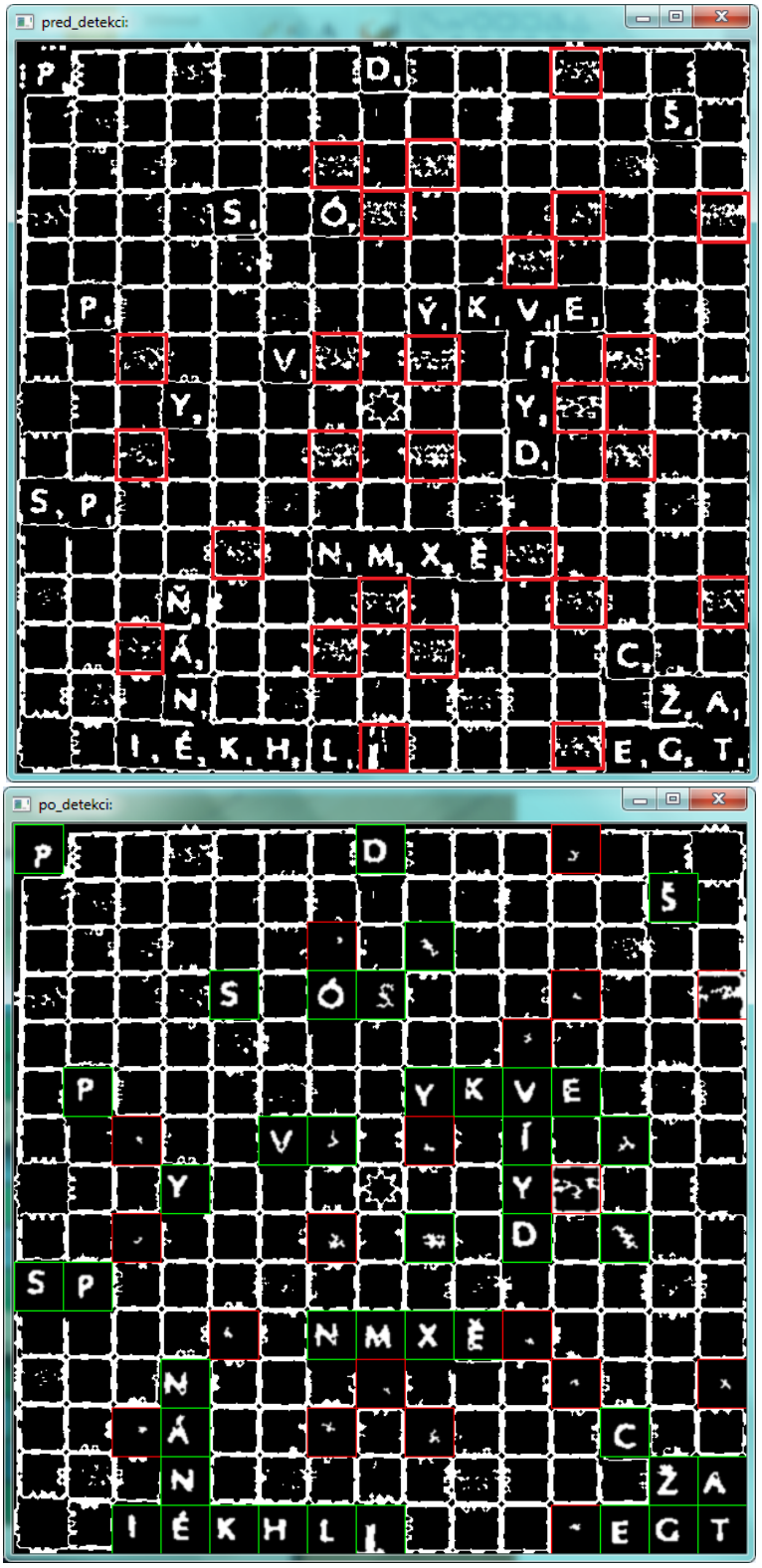
5.2.1 Detekce písmen

Podíváme-li se na hrací desku, vidíme, že se zde nacházejí různá bonusová pole, ve kterých je napsán nějaký text. Před detekcí písmen máme obraz upravený, viz Obrázek 5.2. Nahoře jsou označeny čtverečky, ve kterých bylo chybně detekováno písmeno (schválně je zvolen extrémní případ, běžně k tolika chybným detekcím nedochází).

Těsně před klasifikací probíhá další kontrola. V této fázi již máme nalezen ve čtverečku největší spojitý segment bílých pixelů, který by měl reprezentovat písmeno. Nyní, na základě rozměrů

segmentu, je většina čtverečků prohlášena za chybně detekované a klasifikace u nich neprobíhá. Tyto čtverečky jsou zobrazeny červeně, viz Obrázek 5.2 dole.

Úspěšnost odhalení chybné detekce zobrazuje Tabulka 1. Bohužel ke stoprocentnímu odhalení chybně detekovaných písmen dojde jen výjimečně.



Obrázek 5.2: Chybná detekce písmen.

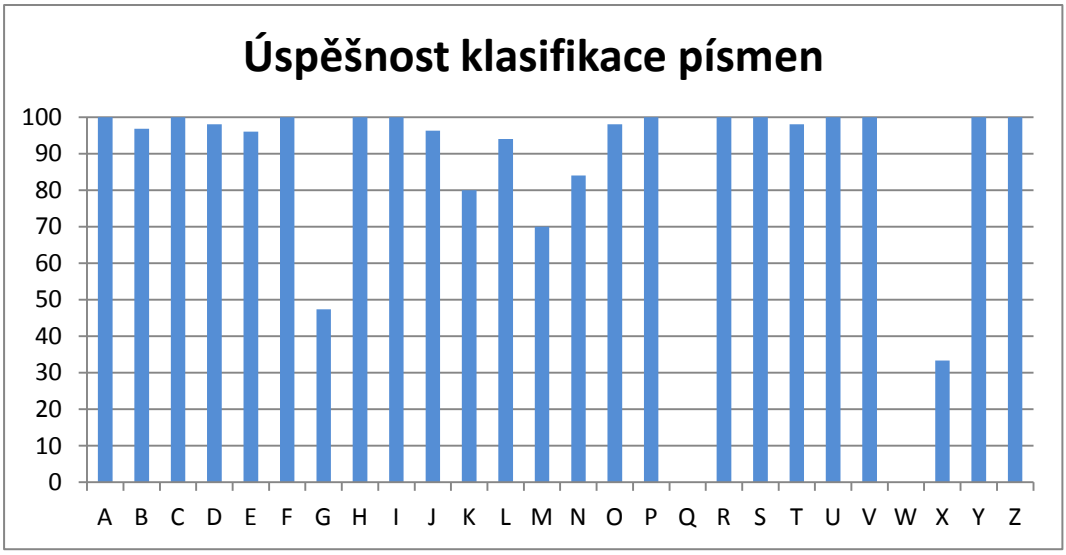
Úspěšnost odhalení chybné detekce písmen			
Obrázek	**Počet chybně detekovaných písmen**	**Počet odhalených chybných detekcí**	**Úspěšnost v %**
1	2	0	0
2	4	4	100
3	3	2	67
4	28	25	89
5	37	26	70
6	38	31	82
7	30	18	60
8	26	18	69
9	35	20	57
10	40	23	83
11	8	8	100
12	5	3	60
13	2	1	50
14	1	0	0
15	26	19	73
16	26	20	77
17	27	13	48
18	31	16	52
19	39	28	72
20	31	17	55
Průměrná úspěšnost			63

Tabulka 1: Úspěšnost odhalení chybné detekce písmen.

Pokud je počet chybně detekovaných písmen malý, dochází k extrémním hodnotám úspěšnosti (0%, 100%). K tomu dochází na, běžnou hrou zaplněných, deskách, kdy většina bonusových polí je obsazena nějakým písmenem.

5.2.2 Klasifikace písmen

Nyní se podívejme na úspěšnost klasifikace písmen. Jak bylo řečeno, nelze očekávat, že tato úspěšnost bude u všech písmen stoprocentní. Graf 1 zobrazuje procentní úspěšnost klasifikace u jednotlivých písmen. Klasifikační model byl trénován na písmena anglické abecedy, proto nebylo možné klasifikovat písmena s interpunkcí. Písmena Q a W v grafu nemají žádnou hodnotu, jelikož se v české verzi hry nevyskytují (pro mě jediná dostupná verze).



Graf 1: Úspěšnost klasifikace písmen v %. Písmena Q a W v české verzi hry nejsou.

Z grafu vyplývá, že klasifikace u většiny písmen dosahuje stoprocentní úspěšnosti. U dvojic písmen, která jsou si hodně podobná, jako jsou X a K, G a C, M a N, dochází často k záměně.

Jelikož data byla testována ručně, nebylo možné otestovat stovky až tisíce vzorků pro každé písmeno, proto byla většina písmen testována na 50i vzorcích. U písmen, kterých je ve hře málo byl počet vzorků ještě menší.

5.2.3 Skládání slov

Skládání slov probíhá při průchodu maticí s oklasifikovanými písmeny. Slova jsou značně ovlivněna úspěšností detekce a klasifikace jednotlivých písmen. Obrázek 4.8 zobrazuje ideální výstup, kdy jednotlivá písmena jsou správně oklasifikována a slova nejsou ovlivněna chybně detekovanými písmeny. Z toho plyne, že na výstupu jsou vypsána přesně ta slova, která jsou na vstupní hrací desce.

Ovšem jak bylo řečeno, úspěšnost odhalení chybné detekce písmen není stoprocentní a poté všechna chybně detekovaná písmena jsou oklasifikována. To vede k situaci, kdy je na výstupu víc písmen, než je na vstupní hrací desce a tato písmena jsou zapojena do tvorby slov, a proto většina slov neodpovídá těm na vstupu (viz Obrázek 5.3).

Obrázek 5.3: Ukázka slov ovlivněných chybnou detekcí.

6 Závěr

V rámci této práce byly nastudovány metody pro zpracování obrazu, detekci a rozpoznání objektů v obraze. Dále byly nastudovány metody pro OCR (Optical Character Recognition), které se běžně v bakalářském studiu podrobně nevyučují.

Byla navržena aplikace využívající některé z nastudovaných funkcí. Tyto funkce jsou popsány na začátku práce. Dále byla nafocena vlastní sada fotografií, užitá při implementaci a testování aplikace.

Implementace byla optimalizována na tuto sadu fotografií, přičemž byly nastaveny některé parametry tak, aby byly výsledky co nejlepší. Ukázalo se, že některé vstupy nebyly vhodně zvoleny, a proto byla pro další použití vybrána sada pouze ideálních vstupů (Obrázek 5.1 vlevo). Pro tuto sadu aplikace funguje bez problému. Bohužel aplikace je natolik složitá, že nebylo v mých silách ji zvládnout celou v daném časovém intervalu. Proto bylo upuštěno od kontroly jednotlivých slov podle slovníku a bodového vyhodnocení.

Testování aplikace spočívalo ve vyhodnocení úspěšnosti detekce a klasifikace písmen. Z testování vyplynulo, že kritickou částí aplikace je falešná detekce písmen a nedostatečná úspěšnost jejího odhalení. Naopak u klasifikace byla dosažena vysoká úspěšnost, u většiny písmen až stoprocentní. Tabulka 1 zobrazuje úspěšnost odhalení chybné detekce písmen, Graf 1 zobrazuje úspěšnost klasifikace jednotlivých písmen.

Pokračováním práce by mohlo být doladění stávajících chyb, především u detekce písmen a jejich úpravy před klasifikací. Poté naimplementovat kontrolu slov podle slovníku a bodové vyhodnocení slov.

Kontrola podle slovníku může být provedena online, kdy budeme posílat dotazy na existenci jednotlivých slov a kontrolovat odpovědi. Offline kontrola by spočívala v přidání slovníku přímo do aplikace.

U bodového ohodnocení slov stačí v aplikaci vytvořit databázi písmen s jejich bodovým ohodnocení a sečíst body jednotlivých písmen ve slově.

Pro využití aplikace se nabízí mobilní telefony, kdy uživatel pomocí fotoaparátu na tomto zařízení vyfotí hrací desku a na displeji uvidí vyhodnocení. Pro implementaci na mobilní telefony bude nutné vytvořit grafické uživatelské rozhraní.

Literatura

[1] Fisher, R. Perkins, S. Walker, A. Wolfart, E. *Image processing learning resources* [online]. Created in 2000. [vid. 2013-04-27]. Dostupné z: http://homepages.inf.ed.ac.uk/rbf/HIPR2/wksheets.htm.

[2] Summers, J. *Conversion to grayscale* [online]. Created in 2011. [vid. 2013-05-04]. Dostupné z: http://entropymine.com/imageworsener/grayscale/.

[3] Castleman, R. K. *Digital Image Processing.* 2nd ed. New Jersey: Prentice Hall. 1995. 667p. ISBN-10: 0132114674. ISBN-13: 978-0132114677.

[4] Sonka, M. Hlavac, V. Boyle, R. *Image Processing, Analysis, and Machine Vision*. 3rd ed. Toronto: Thomson. 2008. 829 p. ISBN-10 0-495-08252-X. ISBN-13 978-0-495-08252-1.

[5] Canny, J. F. *A computational approach to edge detection*. IEEE Transactions on Pattern Analysis and Machine Intelligence. 8(6):679-698. 1986.

[6] Kort, P. *The use of the Hough transform in shape-from-shading*. Regina, 2007. Diplomová práce. University of Regina, Faculty of Graduate Studies and Research. Dostupné z: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.138.7525.

[7] Klán, P. *Neural Networks for Chemists* [online]. [vid. 2013-05-04]. Dostupné z: http://www.cs.cas.cz/pklan/CHI7.pdf.

[8] Manning, Ch. D. Raghavan, P. Schütze, H. *Introduction to Information Retrieval*. 1st ed. Cambridge University Press. 2008. 496 p. ISBN-10 0521865719. ISBN-13 978-0521865715.

[9] OpenCV (Open source computer vision) [online]. rev. 2013. [vid. 2013-04-29]. Dostupné z: http://opencv.org/.

[10] Chang, C.-C. Lin, C.-J. *LIBSVM: a library for support vector machines* [online]. [vid. 2013-05-06]. ACM Transactions on Intelligent Systems and Technology, 2:27:1-27:27. 2011. Dostupné z: http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[11] Hsu, C.-W. Chang, C.-C. Lin , C.-J. *A practical guide to support vector classification* [online]. [vid. 2013-05-06]. Taipei: National Taiwan University, Department of Computer Science. 2010. Dostupné z: http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf.

Seznam příloh

Příloha 1. DVD

Příloha 1. DVD

Přiložené DVD obsahuje:

- dataset\ Dataset pro trénování klasifikátoru, zmíněný v sekci 4.3.4.
- doc\ Dokumentace vygenerovaná nástrojem Doxygen.
- fotky\ Testovací sada fotografií hrací desky.
- libsvm-3.16\ Knihovna LIBSVM.
- models\ Natrénovaný klasifikační model.
- Scrabble\ Projekt přímo spustitelný v Microsoft Visual Studiu.
- src\ Samostatné zdrojové kódy aplikace.
- zprava\ Text této zprávy ve formátu pdf a docx.
- manual.pdf Uživatelský manuál popisující postup při spouštění aplikace.
- plakat.pdf Plakát prezentující tuto práci.