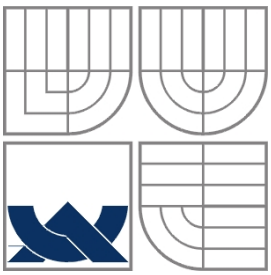
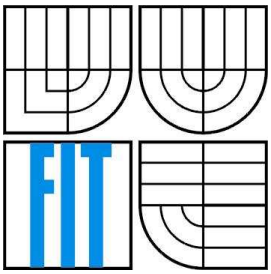


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ROZPOZNÁNÍ KONTROLNÍHO KÓDU Z OBRÁZKU

CODE DETECTION FROM CONTROL IMAGE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MILOSLAV RŮŽIČKA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VÍTĚZSLAV BERAN

BRNO 2009

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2008/2009

Zadání diplomové práce

Řešitel: **Růžička Miloslav, Bc.**
Obor: Počítačová grafika a multimédia
Téma: **Rozpoznání kódu z kontrolního obrázku**
Kategorie: Zpracování obrazu

Pokyny:

1. Prostudujte základy zpracování obrazu. Zaměřte se zejména na problematiku detekce hran a elementárních objektů v obraze. Zorientujte se v současných metodách rozpoznávání textu v obraze.
2. Navrhněte metodu, která bude z kontrolního obrazu detekovat a klasifikovat písmena a číslice kontrolního kódu.
3. Vytvořte anotovanou testovací sadu dat.
4. Navrženou metodu implementujte a vytvořte experimentální aplikaci, která bude sloužit i k demonstraci vlastností navržené metody.
5. Proveďte experimenty zkoumající přesnost a stabilitu navrženého systému. Experimenty pečlivě zdokumentujte a okomentujte. Diskutujte omezení metody a případná řešení.
6. Vytvořte stručný plakát prezentující vaši diplomovou práci, její cíle a výsledky.

Literatura:

- Hlaváč, V.: Zpracování signálů a obrazů, Praha, ČVUT, 2005.
- Rafael C. Gonzalez, Richard E. Woods: Digital image processing, Prentice-Hall, 2002.
- Galbiati, L. J.: Machine vision and digital image processing fundamental, Prentice-Hall, 1990.
- Dále dle pokynu vedoucího.

Při obhajobě semestrální části diplomového projektu je požadováno:

- Body 1, 2 a 3.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Beran Vítězslav, Ing.**, UPGM FIT VUT
Datum zadání: 22. září 2008
Datum odevzdání: 26. května 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Práce se zabývá problematikou rozpoznání kontrolního kódu v obrázku. Uvádí související oblasti ze zpracování obrazu, mezi které patří odstranění šumu, prahování, barevné modely, segmentace objektů a OCR. Práce dokumentuje výhody a nedostatky dvou vybraných metod segmentace objektů a navrhuje vlastní systém segmentace objektů. Dále je popsán navržený systém pro segmentaci a klasifikaci objektů.

Abstract

Work deals with code detection from control image. The document presents relevant image processing techniques dealing with a noise reduction, thresholding, color models, object segmentation and OCR. This project examines advantages and disadvantages of two selected methods for object segmentation and introduces developed system for object segmentation. The developed system for object segmentation and classification is realized, evaluated and results are discussed in details.

Klíčová slova

CAPTCHA, šum, prahování, HSV, RGB, segmentace, OCR, klasifikace, porovnání vzorů.

Keywords

CAPTCHA, noise, thresholding, HSV, RGB, segmentation, OCR, classification, template matching.

Citace

Růžička Miloslav: Rozpoznání kódu z kontrolního obrázku, diplomová práce, Brno, FIT VUT v Brně, 2009

Rozpoznání kódu z kontrolního obrázku

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Vítězslava Berana. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Miloslav Růžička
21.5.2009

Poděkování

Děkuji panu Ing. Vítězslavu Beranovi za odborné vedení, dále pak své rodině a přítelkyni za podporu v době studia.

Obsah

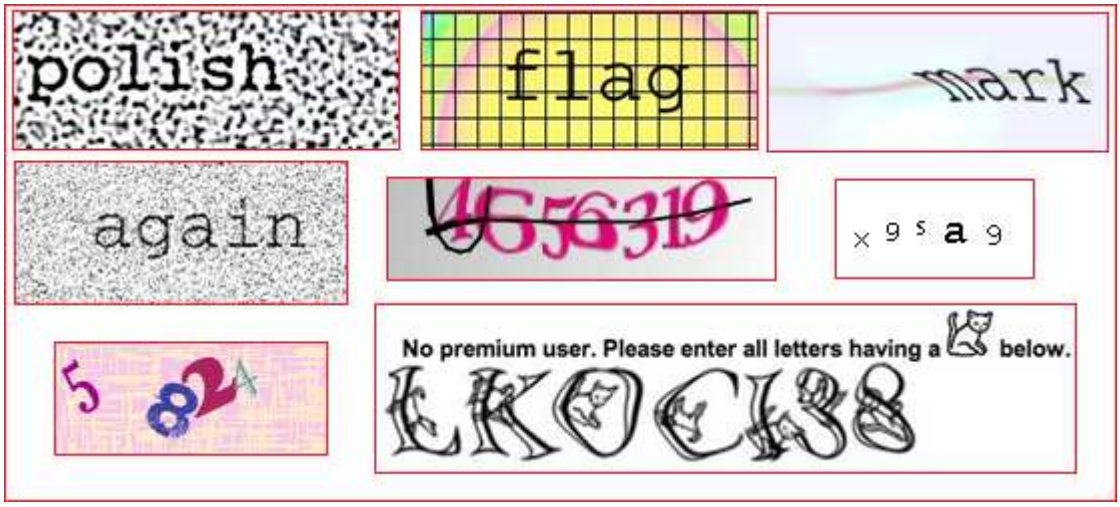
1 Úvod

S rozvojem moderních technologií došlo také k rozvoji zločinnosti v této oblasti. Zářným příkladem jsou služby nabízené zdarma. Ty bývají nejčastěji zneužívány, jelikož jejich použití je obvykle anonymní. Například možnost odeslání SMS z internetu zdarma či zřízení a používání emailové adresy bývá zneužito k odesílání spamu. Reakcí poskytovatelů těchto služeb je přidání do procesu odesílání mechanismus, který se snaží zjistit, zda službu používá člověk, kde lze předpokládat, že ji bude používat řádným způsobem, nebo počítačový program, který obvykle službu zneužije. Jednou z technik, jak odlišit počítačový program od člověka, je Turingův test. A právě jednou jeho variantou - CAPTCHA - se zabývá tato práce, ve které se budu snažit ukázat možný přístup k detekci a klasifikaci číslic v obrázku. Na základě výsledků této práce poté můžeme konstatovat, zda je tato ochrana dostatečná či nikoliv.

1.1 Turingův test - CAPTCHA

CAPTCHA je akronym pro „Completly Automated Public Turing test to tell Computers and Humans Apart“. Jedná se tedy o automatizovaný Turingův test. Jako Turingův test se dá označit jakákoliv zkouška, která má za cíl rozlišit mezi člověkem a strojem. Jeho historie sahá do padesátých let minulého století, kdy sloužil k jednoduchému testu, zdali náhodná osoba pozná, jestli mluví s člověkem nebo umělou inteligencí. Turingův test má dnes mnoho podob a kdo jej pokoří (zejména někteří chatovací roboti), může usilovat o tzv. Loebnerovu cenu, o kterou se v každoročním soutěžním klání bojuje už od začátku devadesátých let minulého století.

V případě, kterým se zabývá tato práce, má uživatel za úkol opsat deformovaný text z obrázku do vstupního políčka. Obrázek 1 ukazuje několik příkladů CAPTCHA.



Obrázek 1- Příklad CAPTCHA kódů

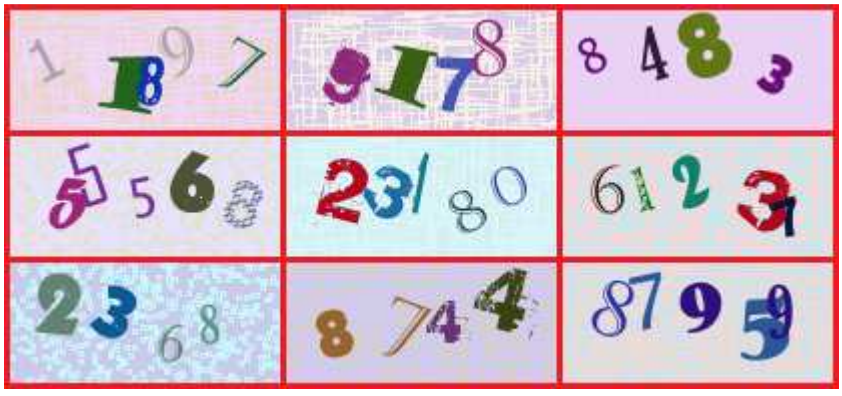
Z některých kontrolních obrázků je zřejmé, že je opravdu problém i pro člověka rozpoznat kód v obrázku. Bohužel to je osud některých ochran. Aby měly přijatelnou účinnost, musejí být natolik složité, že má s nimi problém i legální uživatel, proti kterému nejsou určeny.

Ochrana využívá předpokladu, že lidský mozek dokáže rozeznat i deformovaný text, ale internetový robot za použití technologie OCR nebude schopen text správně přečíst. CAPTCHA se tedy používá k eliminaci robotů, kteří by jinak zahltili diskusní fóra spamem, či by registrovali velká množství emailů na free-email-serverech, ze kterých by poté bylo odesíláno velké množství spamů. Nevýhodou je potom nepřístupnost pro zrakově postižené. Tento problém některé weby řeší pomocí zvukového přehrání kontrolního textu, není to ale zdaleka samozřejmost. Pokud by to však samozřejmost byla, mohli by se roboti zaměřit na rozpoznání z nahrávky a opět by se muselo řešit strojové rozpoznání, tentokráte zvuku. Nahrávka by tedy musela být poupravena, aby se stala pro robota „nečitelnou“.

Jak již bylo zmíněno, existuje veliké množství různých CAPTCHA založených na opsání kódu z obrázku, obecný systém rozpoznávající kódy ze všech druhů takovýchto obrázků by bylo velice složité (ne-li nemožné) vytvořit. Tato práce si tedy vybrala kontrolní kódy na stránce www.vodafonesms.cz, kde je nutné opsat řadu čísel pro možnost odeslání SMS z internetu zdarma.

Kontrolní kód v obrázku je tvořen různě barevnými, velkými, natočenými, tučnými i tenkými, někdy překrývajícími se číslicemi. Číslice můžeme klasifikovat do několika tříd:

- tučné, neporušené číslice
- tučné, porušené číslice
- tenké, neporušené číslice
- číslice vyplněné texturou
- stínované číslice



Obrázek 2 - Ukázka několika kontrolních kódů

Další způsoby rozpoznání člověka a stroje

Další metody více či méně založené na CAPTCHA by mohly být např. otázka uvedená v obrázku, na kterou by musel člověk odpovědět. Možná by už ani tato otázka nemusela být vložena do obrázku, jelikož stroj by stejně nepochopil, na co se ho ptáme. Tudíž by ani nevěděl, jak má odpovědět. Využila by se tak inteligence člověka.

Ve zdroji [1] autor nastiňuje možnost zobrazení 3D textu, jelikož částečně zdeformovaný text jsou někteří roboti schopni přečíst a při narůstající deformaci už začínají mít problémy lidé. 3D text je založen na schopnosti lidského mozku snadno rozeznat prostorové objekty ve dvojrozměrném průmětu. Zde by podle autora automatizované systémy měly mít problémy s automatickým rozpoznáním.

Zajímavosti

Každý den lidé (jak uvádí zdroj [2]) na internetu vyplní správně šedesát miliónů CAPTCHA testů, přičemž jeden text zabere návštěvníkovi průměrně 10 sekund. To znamená, že během pouhých 24 hodin to v globálním měřítku činí 150 tisíc hodin práce.

Celkem zajímavým způsobem tohoto opisování textu z obrázku využili tvůrci systému reCAPTCHA (http://recaptcha.net). Jejich CAPTCHA je velice složitá, pokud je ale rozluštěna, prospěje dobré věci, jelikož jako kód CAPTCHA je použita textová fráze z papírových knih, s nimiž měly problémy při digitalizaci současné OCR programy. Jak systém pozná, že odesílatel kód správně rozpoznal? Obrázek je použit několikrát, a pokud více osob napíše ty samé znaky, systém to bere jako dostatečný důkaz o správném rozpoznání. Více v [2].

1.2 Obsah práce

V první kapitole jsou rozebírány ty oblasti zpracování obrazu, které souvisí s tématem této práce a které budou použity při implementaci navrhovaného systému, jako např. šum v obraze a možnosti jeho odstranění, omezení barevného prostoru prahováním či převodem do barevného modelu HSV a selekcí jen jedné složky. Dále je popsána segmentace a barevné modely RGB a HSV. Posledním tématem této kapitoly je nalezení kostry číslice, kde je představena metoda Zhang-Suen.

V druhé kapitole se práce zabývá segmentací. Jsou podrobně vysvětleny dva použité přístupy k segmentaci, jsou zhodnoceny jejich klady a zápory a na jejich základě je navržena metoda nová, která kombinuje oba dva přístupy.

Třetí kapitola se zabývá klasifikací číslic. Na začátku se nacházejí informace o optickém rozpoznávání (OCR), jeho historii a současném stavu, kde je uveden přehled, co je úspěšně vyřešeno (rozpoznání strojového písma) a co teprve čeká na vyřešení (rozpoznání ručně psaného písma). Je

zdůvodněn výběr klasifikátoru využívající principu hledání nejbližšího souseda. Jsou popsány vzory jednotlivých tříd, kroky normalizace a je vysvětlen princip čtyř klasifikátorů, které byly použity.

Čtvrtá kapitola se věnuje vytvoření anotovaných dat, které slouží k automatickému ohodnocení použitých metod. Dále je zhodnocena úspěšnost segmentace, normalizace a klasifikace. Výsledky klasifikátorů jsou přeneseny do grafů a je vybrána metoda, která podává nejlepší výsledky.

Poslední kapitola se věnuje krátkému zhodnocení navrženého systému a jsou nastíněny možnosti dalšího vývoje.

Tato práce navazuje na semestrální projekt stejného názvu řešený v předchozím semestru. Ze semestrálního projektu byla s úpravami převzata úvodní kapitola. Z druhé kapitoly týkající se segmentace byly převzaty podkapitoly věnující se popisu základní a pokročilé metody, na základě jejichž výsledků byla navržena výsledná dokonalejší metoda.

2 Použité metody

Kapitola se věnuje některým základním pojmům a metodám souvisejícím se zpracováním obrazu, které byly použity nebo se týkají tématu této práce.

Obraz je pro člověka jedním ze zdrojů informací. Člověk se podívá na obraz a za okamžik dovede zobrazenou scénu popsat. Člověk vidí souvislosti mezi objekty, uvědomuje si, že se některé objekty v obraze překrývají a hned si domýšlí jejich zakryté části. Člověk nemá problémy s rozpoznáním objektů při jejich natočení, prostorové deformaci, zašumění obrazu atd.

Pro počítač je však obraz jen pouhými daty v paměti bez jakéhokoliv „vyššího“ významu. Obraz je složen z řádků, každý řádek z pixelů. Pixel jako element obrazu může být reprezentován jednou hodnotou, která představuje intenzitu jasu u šedotónového obrazu, či více hodnotami, kde jejich význam záleží na použitém barevném modelu. Důvodem, proč je tak těžké naučit počítač rozpoznávat objekty v obraze, je, že ani současná věda přesně neví, jak je obraz zpracováván v lidském mozku.

2.1 Šum v obraze

Jednou ze základních poruch v obraze je šum. U digitální fotografie za šum v obraze může více zdrojů, nikdy však není do obrazu vložen úmyslně, protože zhoršuje vlastnosti obrázku (neberu v úvahu různé metody zpracování obrazu, kde právě vložení šumu může vést k lepším výsledkům). V našem případě je však této vlastnosti využito, jelikož cílem je právě zabránit automatickému rozpoznání. Tedy přidání šumu do obrázku zhoršuje kvalitu obrazu, a to je autorovým záměrem.

Šum v obraze můžeme rozdělit na dva nejčastější typy šumu:

- Náhodný šum
- Gaussův šum

Náhodný šum, také známý pod názvem nezávislý šum, je způsoben např. vadnými CCD elementy. Příkladem tohoto šumu je šum typu „sůl a pepř“, který postihuje jen izolované prvky nebo malé skupiny extrémními hodnotami. U Gaussova šumu má amplituda normální (Gaussovo) rozdělení pravděpodobnosti a postihuje všechny pixely v obraze.

K odstranění šumu můžeme použít lineární filtraci (průměrování) nebo nelineární filtraci (medián, rotační masky). Pro odstranění šumu typu pepř a sůl se dobře hodí např. Mediánový filtr, který pracuje s okolím aktuálně zpracovávaného pixelu a hodnota pro tento pixel je určena jako medián zvoleného okolí.

2.2 Prahování obrazu

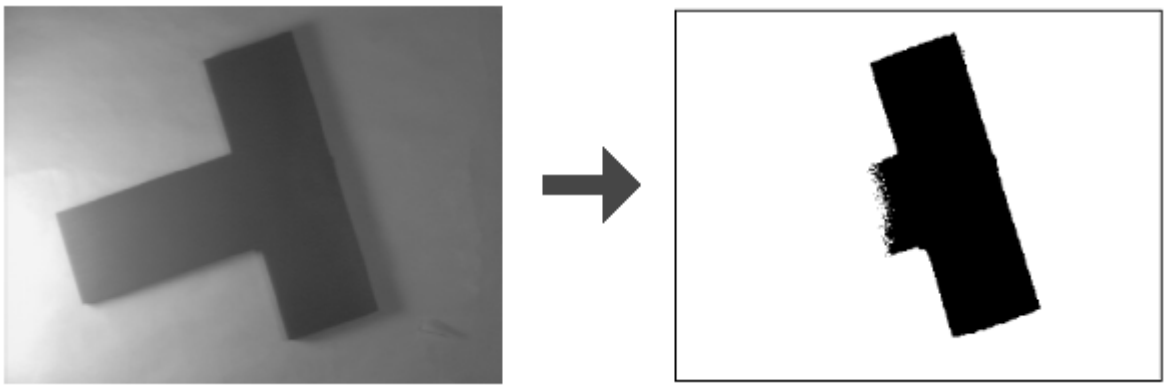
Jednou ze základních, velice jednoduchých segmentačních metod je prahování, které slouží k omezení barevného prostoru obrazu. Prahování je tedy funkce, která upravuje jasové či barevné složky pixelů v obraze podle předpisu:

$$f(c) = \begin{cases} A & pokud \quad c < práh \\ B & pokud \quad c >= práh \end{cases}$$

C je vstupní hodnota jasu nebo barvy, $f(c)$ je výsledná hodnota, práh je prahovací hodnota, *A* a *B* jsou nové hodnoty pro vstupní hodnotu *c* pod a nad prahem. Výsledek prahování tedy závisí na hodnotě prahu. Jako vstupní obrázek pro prahování se většinou předpokládá šedotónový obraz. Následující kapitoly popisují způsoby určování prahu.

Globální prahování

Tato metoda určování prahovací hodnoty se vyznačuje jedním prahem pro celý obrázek. O možnosti použití globálního prahování vypovídá histogram obrazu. Histogram je graf zobrazující počet bodů v obraze pro každou hodnotu intenzity. Např. pro 8-bitový obrázek v odstínech šedi je to 256 možných intenzit. Pokud se nacházejí v takovémto histogramu dvě významná maxima, intenzity jednotlivých bodů budou seskupeny okolo dvou dobře separovatelných hodnot. Pokud rozdělení hodnot nemá tuto charakteristiku, potom je pravděpodobné, že při prahování dojde ke ztrátě důležité informace v obraze.



Obrázek 3 - Ukázka nevhodného obrázku pro globální prahování (obrázek převzat z [3])

Lokální prahování

Metoda lokálního prahování hledá prahovou hodnotu pro každý pixel v obraze zvlášť vyšetřením intenzit jasu v jeho okolí. Velikost okolí musí být natolik velké, aby pokrylo dostatečné množství

pixelů pozadí a objektu. Na druhou stranu velikost okolí nesmí přesáhnout mez, kdy se již na pixelech projevuje různé osvětlení v obraze.

Práh z okolí pixelu můžeme vypočítat několika způsoby. Může to být např. medián jasu z pixelů okolí, střední hodnota jasu okolí nebo průměr minimální a maximální hodnoty jasu v okolí.

2.3 Segmentace

Segmentace obrazu patří k nejdůležitějším krokům zpracování obrazu. Segmentací rozumíme rozčlenění obrazu na segmenty – části obrazu, které mají určitý vztah k objektům v obraze.
Každý segment splňuje nějaké tvrzení, např:

- všechny pixely segmentu mají stejnou úroveň šedi
- všechny pixely segmentu se neliší v úrovni šedi o více než stanovenou hodnotu
- všechny pixely obrazu mají stejnou barvu v barevném obrazu

O kompletní segmentaci mluvíme, jestliže části jednoznačně korespondují s objekty v obraze. Pokud získané segmenty nesouhlasí přesně s objekty v obraze, mluvíme o částečné segmentaci.

Možné přístupy zmiňuje autor zde [4]. Jen připomenu asi nejjednodušší segmentaci, a tou je prahování, o kterém se zmiňuji v předcházející kapitole.

Barevná segmentace

Klasická segmentace si většinou všímá jasové složky v obraze a informaci skrytou v barvě vůbec nepoužije. Základním krokem tedy bývá převod obrázku na stupně šedi. Barevná segmentace jde však o krok dál a k segmentaci používá i informaci skrytou v barvě v obraze. Základní model, ve kterém je barevná informace zakódována (RGB), však není příliš vhodný pro zpracování, a proto se využívá převodu do jiných modelů, které mají pro další zpracování lepší vlastnosti.

Reprezentace barev

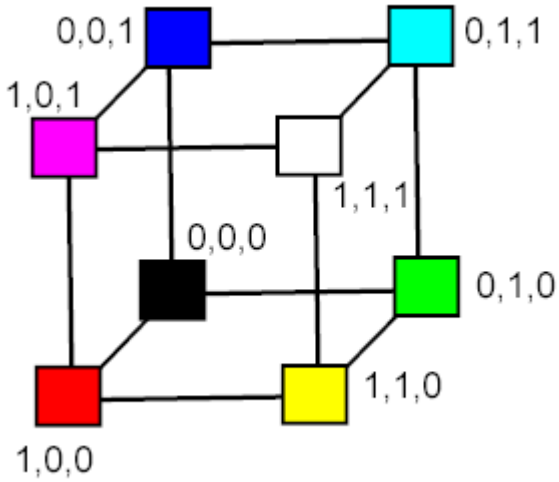
Jedním z nejdůležitějších atributů, používaných při zpracování obrazu, je barevná informace. Každá barva odpovídá určité frekvenci elektromagnetického vlnění v pásmu 10^8 MHz. Rozsah barev je od červené – 430 nm až po fialovou – 750 nm. Člověk je schopen rozlišit více než 4×10^5 barevných odstínů.

Existuje mnoho metod (barevných modelů), kterými se popisuje barevná informace, a na jejich základě je možný každý barevný odstín dekomponovat na jeho základní složky.

Barevný model RGB

Základní složky tohoto modelu vycházejí z faktu, že lidské oko obsahuje tři základní druhy buněk citlivých na barvu. Tyto buňky jsou citlivé na vlnové délky, které zhruba odpovídají červené (vlnová

délka 630 nm), zelené (530 nm) a modré (450 nm) barvě. Kombinací (aditivním způsobem) těchto barev lze získat téměř všechny barvy barevného spektra.



Obrázek 4 - Barevný model RGB (obrázek převzat z [5])

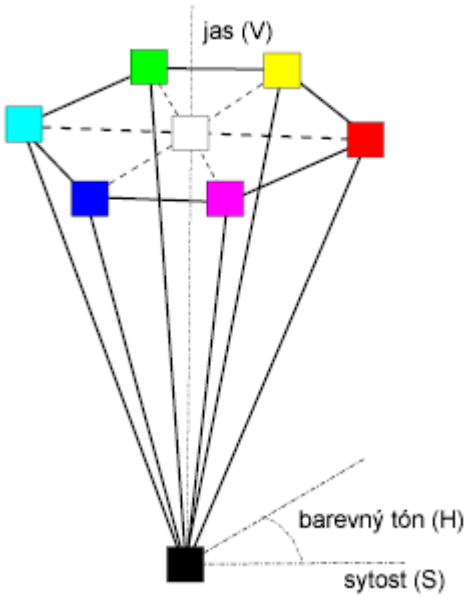
Barevný model RGB se nejčastěji zobrazuje jako krychle umístěná v osách RGB. Barvy ležící na úhlopříčce mezi vrcholem černé barvy [0,0,0] a bílé barvy [1,1,1] tvoří všechny šedotónové odstíny.

Barevný model HSV

Tento model odpovídá popisu barev, na který je člověk zvyklý. Model HSV má tři základní parametry:

- H – hue – barevný odstín. Hodnota, která reprezentuje jednu ze základních barev modelu RGB.
- S – saturation – saturace (sytost). Saturace popisuje např. rozdíly mezi červenou a růžovou. Tmavě červené odpovídá vysoká hodnota saturace a naopak růžové odpovídá nízká hodnota saturace.
- V – value – jas. Reprezentuje množství světla přijatého senzorem.

Barevný model HSV se nejčastěji zobrazuje jako šestiboký jehlan, jehož vrchol leží v počátku souřadného systému. Souřadnice *s* a *v* se mění od 0 do 1. Souřadnice *h* je úhlová. Vrchol jehlanu v bodě [0,0,0] představuje černou barvu. Bílá barva je ve středu podstavy. Jas klesá od podstavy k vrcholu. Sytost je dána vzdáleností od osy jehlanu. Základní barvy se nachází ve vrcholech šestiúhelníkové podstavy.



Obrázek 5 - Barevný model HSV (obrázek převzat z [5])

Informace o dalších barevných modelech lze nalézt zde [5].

2.4 Zužování

Zužování je metodou zpracování obrazu, ve které je oblast obrazu vyjádřeného v binární podobě zredukována do podoby úseček. Tyto úsečky svou polohou odpovídají takzvaným středovým úsečkám. Také je možno je nazvat kostrou původní oblasti.

Jedná se o algoritmus, který je možno využít v řadě metod sloužících pro rozpoznání textu, protože oblasti obrazu jsou redukovány na jejich základní informaci, která je vhodnou formou pro následnou analýzu za účelem rozpoznání. Na obrázku 7 je znázorněna tato redukce.

Důležitou vlastností metody je, že musí zachovávat propojenost oblastí. To znamená, že pokud jsou některé části obrazu před redukcí propojeny, i výsledný zredukovaný tvar by měl pro odpovídající části dodržet propojení.

Nejčastěji užívaný přístup iterativně testuje osmi-okolí každého pixelu a odstraňuje ty, které leží na okraji. Proces končí ve chvíli, kdy se během iterace neprovede žádná změna.

Zhang-Suen

Metoda zužování Zhang - Suen je zástupcem výše zmíněného přístupu. Pracuje s oblastí okolí pixelu o velikosti 3x3 pixely a iterativně odstraňuje pixely ležící na okrajích oblasti.

Postup zužování metodou Zhang - Suen:

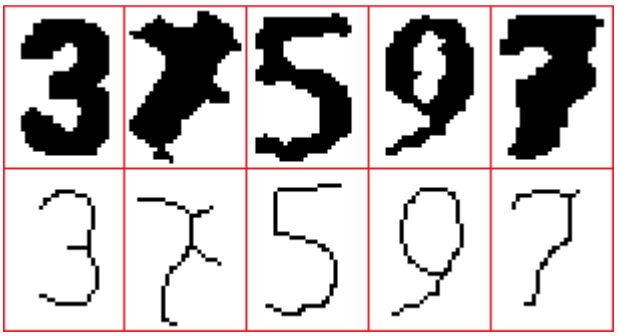
1. Procházíme všechny pixely obrazu. Pokud narazíme na pixel obsahující hodnotu popředí (černá barva), přejdeme na bod **2**. V případě, že byl vyšetřen poslední pixel obrazu, přejdeme na bod **5**.
2. Z hodnot osmi-okolí pixelu *b* určíme hodnotu *N*, která odpovídá počtu pixelů popředí v daném okolí.

	s	
z	**b**	v
	j	

Obrázek 6 - Vyšetřovaný bod (b) a jeho čtyř-okolí (s, j, v, z)

3. Určíme hodnotu P značící počet přechodů s hodnotou pozadí na pixel s hodnotou popředí při procházení okolí pixelu b po směru nebo proti směru hodinových ručiček.
4. Pokud platí $(N \in \langle 2,6 \rangle) \wedge (P = 1) \wedge (s * j * v = 0) \wedge (z * v * j = 0)$, označíme vyšetřovaný pixel a pokračujeme v provádění bodu **1** v místě, kde bylo přerušeno.
5. Nastavíme všechny označené pixely na hodnotu pozadí.
6. Znovu procházíme všechny pixely obrazu. Pokud narazíme na pixel obsahující hodnotu popředí, přejdeme na bod **7**. V případě, že byl vyšetřen poslední pixel obrazu, přejdeme na bod **10**.

7. Z hodnot osmi-okolí pixelu *b* určíme hodnotu *N*, která odpovídá počtu pixelů popředí v daném okolí.
8. Určíme hodnotu *P* značící počet přechodů z pixelu s hodnotou pozadí na pixel s hodnotou popředí při procházení okolí pixelu *b* po nebo proti směru hodinových ručiček.
9. Pokud platí $(N \in \langle 2,6 \rangle) \wedge (P = 1) \wedge (s * j * z = 0) \wedge (z * v * s = 0)$ (*s*, *j*, *z* a *v* jsou hodnoty intenzity sousedních pixelů (obrázek 6)), označíme vyšetřovaný pixel a pokračujeme v provádění bodu **6** v místě, kde bylo přerušeno.
10. Nastavíme všechny označené pixely na hodnotu pozadí.
11. Pokud byl alespoň jeden pixel změněn, znovu provedeme bod **1**, jinak končíme.



Obrázek 7 - Příklady nalezení kostry metodou Zhang-Suen
(v horní části jsou zobrazeny zdrojové objekty, v dolní pak nalezené kostry)

Tato metoda podává velice dobré výsledky, pokud jsou číslice zakulacené, neobsahují díry, či nejsou jiným způsobem poškozeny. Jak můžeme vidět na obrázku výše, pokud číslice nějaké takové poškození obsahuje (jako např. číslice jedna, která obsahuje výstupky), algoritmus neprodukuje příliš dobré výsledky. Bude tedy nutné nějakým způsobem číslice předzpracovat, aby byly odstraněny nežádoucí efekty, které způsobují vygenerování neodpovídající kostry.

3 Segmentace čísel v obrazu

Jak již bylo řečeno, práce se zabývá kontrolními obrázky na stránkách společnosti Vodafone. Pro segmentaci a rozpoznání číslic z takovýchto obrázků existuje jednoduchý přístup. Pro odstranění pozadí stačí obrázek procházet od levého horního bodu směrem doprava dolů a první dvě barvy z obrázku odstranit. Tento přístup je velice jednoduchý a funguje opravdu dobře. Slabinou je jeho neobecnost. Kdyby byl generátor takovýchto obrázků pozměněn, aby generoval pozadí z více barev nebo aby se počet barev na pozadí měnil s každým obrázkem, takovýto přístup nemá šanci uspět. Další postup zde nebudu uvádět, jelikož se opět jedná o nepříliš obecné metody, které ovšem pro tento konkrétní příklad fungují. Já se budu zabývat obecnějšími algoritmy, takže navržený systém detekce nebude úzce zaměřen na tento druh kontrolních obrázků. Nicméně při volbě jednotlivých kroků budu vycházet z kontrolních obrázků výše uvedených, a proto se dají předpokládat horší výsledky pro jinou třídu kontrolních obrázků.

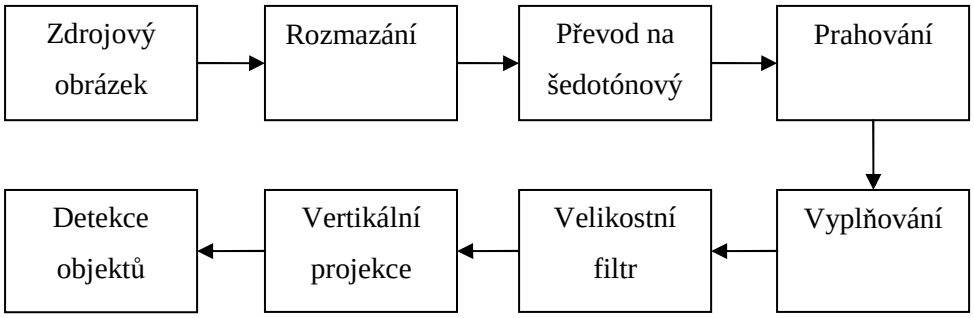
Pro segmentaci objektů (číslic) v obraze byly naimplementovány, otestovány a zhodnoceny dva různé přístupy.

První základní, kdy je vstupní obraz převeden do odstínů šedi a prahován. Tím získáme binární obraz, kde bílá barva určuje pozadí a černá barva objekty. Dále pomocí vertikální projekce získáme sloupce nenulových pixelů. Shluky nenulových pixelů jsou považovány za jednotlivé objekty (v našem případě číslice) v obraze.

Druhý pokročilý přístup (barevně založený) si všímá faktu, že číslice v obrázku mají různou barvu. Obraz je převeden do modelu HSV, jelikož zde je barva oproti modelu RGB reprezentována jednou složkou. Nad tímto obrazem je spočítán histogram. Z obrazu je odstraněno několik maxim, jelikož představují barvu na pozadí a barvu šumu (podrobnosti v následující kapitole). A ze zbylých maxim histogramu je z obrazu vyňata konkrétní barva. Pokud vertikální projekce tvoří shluk o „správné“ velikosti, je tento shluk považován za objekt resp. číslici.

3.1 Základní metoda

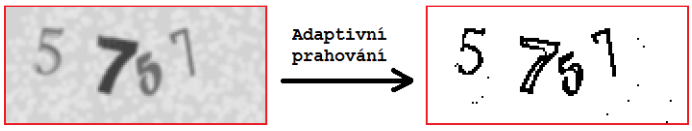
Jako první způsob detekce objektů v obraze jsem zvolil následující metodu, která k dosažení cíle kombinuje základní, jednoduché úpravy nad obrazem.



Obrázek 8 - Blokové schéma primitivní metody

Popis jednotlivých kroků

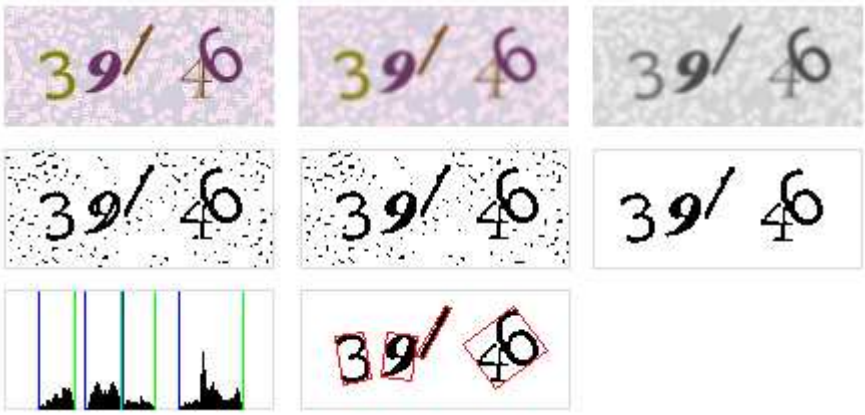
- **Rozmazání** – Je použito k potlačení šumu v obraze. Zdrojový obrázek je rozmazán Gaussiánem s konvolučním jádrem o velikosti 3.
- **Převod na šedotónový** – Kvůli omezení barevného prostoru je obrázek převeden na šedotónový.
- **Prahování** – K segmentaci obrázku je použito prahování. Globální prahování se pro tuto třídu obrázků nehodí, jelikož by došlo k odmazání písmen s vyšším jasem. Je tedy použito adaptivní prahování, které volí práh na základě vyšetření okolí pro každý pixel. Použitý algoritmus má však špatnou vlastnost (obrázek 9) označení vnitřních částí velmi tučných písmen za pozadí. K tomuto jevu dochází z toho důvodu, protože celé okolí pixelu, pro který aktuálně algoritmus počítá hodnotu prahu, obsahuje barvu popředí (objektu). Řešením by bylo zvýšit velikost okolí, ze kterého se pro každý pixel počítá hodnota prahu. V tom případě ale v obraze zůstává více šumu. Z pohledu dalších kroků je lepší varianta první případ, jelikož by vertikální projekce pak mohla detekovat větší počet číslic.



Obrázek 9 - Ukázka označení vnitřní části tučné číslice za pozadí

- **Vyplňování** – V praxi se ukázalo, že vyprahované číslice obsahují pixelové díry. Proto byl do řetězce přidán tento krok, který se je pokusí vyplnit. Pro každý pixel je vyšetřeno jeho 4-okolí, a pokud mají alespoň tři pixely nebo v horizontálním či vertikálním směru dva pixely černou barvu, je pixel obarven černou barvou, tedy označen za pixel objektu.

- **Velikostní filtr** – Odstraní z obrazu objekty, které jsou složeny z počtu pixelů pod stanovenou mezí. Tím dojde k odstranění šumu z obrazu, který nebyl odstraněn pomocí rozmazání. Je důležité, aby se tento krok nacházel v pořadí až za vyplňováním, jelikož by jinak mohlo dojít k odmazání částí objektů, které jsou vlivem rozmazání a prahování rozděleny do několika částí.
- **Vertikální projekce** – Předposlední částí v řetězci je vertikální projekce. Výhodou je rychlost algoritmu. Nevýhodou je předpoklad, že se objekty nepřekrývají a ani nedotýkají.
- **Detekce objektů** – Detekce vychází z předešlého kroku. Najde shluky pixelů v projekci a ty označí za samostatné objekty. Pro názornost je pro pixely každého objektu spočítán nejmenší ohraničující obdélník.



Obrázek 10 - Ukázka zpracování základní metodou. Na obrázek byly postupně aplikovány: Rozmazání, převod do stupňů šedi, adaptivní prahování, vyplňování, velikostní filtr, vertikální projekce, ohraničení objektů.

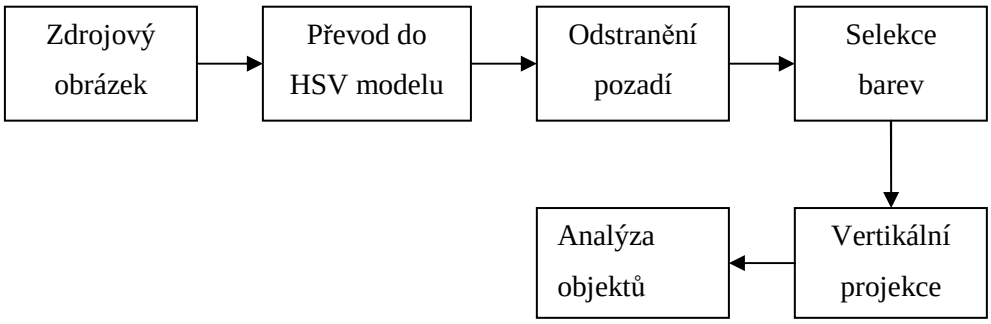
Hlavní nedostatek tohoto postupu spatřuji v kroku vertikální projekce. Jelikož jedním z úmyslných poškození je právě umístění objektů přes sebe, či alespoň tak, aby se dotýkaly. Pokud se tedy objekty dotýkají či překrývají, jsou takto spojené objekty detekovány jako jeden.

Výhodou tohoto postupu je jeho robustnost v otázce složení objektu z několika barev. Rozmazáním, převodem do šedotónového obrázku a prahováním dochází k potlačení této skutečnosti a je tak stejně dobře segmentován objekt složený z jedné barvy či objekt složený z více barev.

3.2 Pokročilá metoda

Nedostatky předcházející metody mě vedly k nalezení dokonalejší metody, která by byla schopná detekovat dotýkající či překrývající se objekty.

Tato metoda si všímá rozdílné barvy objektů v obrázku a snaží se je na základě této skutečnosti detekovat.



Obrázek 11 - Blokové schéma pokročilé metody

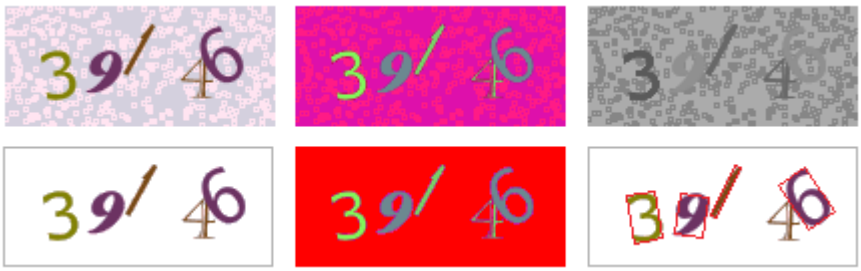
Popis jednotlivých kroků zpracování

- **Převod do HSV modelu** – Jak je uvedeno v teoretické části této práce, barva v modelu HSV je určena odstínem, sytostí a jasem. Samotný odstín barvy je určen jen složkou H. Převodem do tohoto modelu a pracováním dále jen se složkou H tedy pracujeme jen s odstínem barvy, který nás zajímá, jelikož překrývající se objekty jsou tvořeny jiným odstínem barvy.
- **Odstranění pozadí** – V primitivní metodě bylo pozadí odstraněno rozmazáním a adaptivním prahováním. Zde je použit histogram složky H obrazu. V cyklu přes maxima histogramu se počítá vertikální projekce selektované barvy. Pokud je více než půlka šířky obrázku nenulových pixelů, je tato barva označena jako pozadí či šum v obraze a je z obrázku odstraněna. Samozřejmě pokud objekt v popředí je jen světlejší odstín pozadí (složka H je stejná, jen se liší složka S nebo V), je bohužel touto metodou spolu s šumem či pozadím odstraněn také. Z testovaných obrázků však došlo k odstranění objektů jen v jednotkách procent, takže tuto metodu lze považovat za úspěšnou.
- **Selekce barev** – Pro obraz bez pozadí je opět spočítán histogram. A pro všechny zastoupené odstíny jsou provedeny následující kroky.
- **Vertikální projekce** – Opět jako v předešlé metodě je provedena vertikální projekce a podle shluků nenulových pixelů je vytvořen seznam objektů.
- **Analýza objektů** – Každý objekt ve vytvořeném seznamu je podroben zkoumání, zda se jedná o objekt či jen pár pixelů bez významu, a podle výsledku je tento objekt v seznamu ponechán nebo je ze seznamu odstraněn. Opět pro názornost je pro každý shluk pixelů tvořících jeden objekt vypočten nejmenší ohraničující obdélník.

Výhodou tohoto přístupu je nalezení objektů, které se mohou dotýkat nebo překrývat. Dále pak metoda oproti předchozímu postupu nespojuje objekty, které se nacházejí nad sebou.

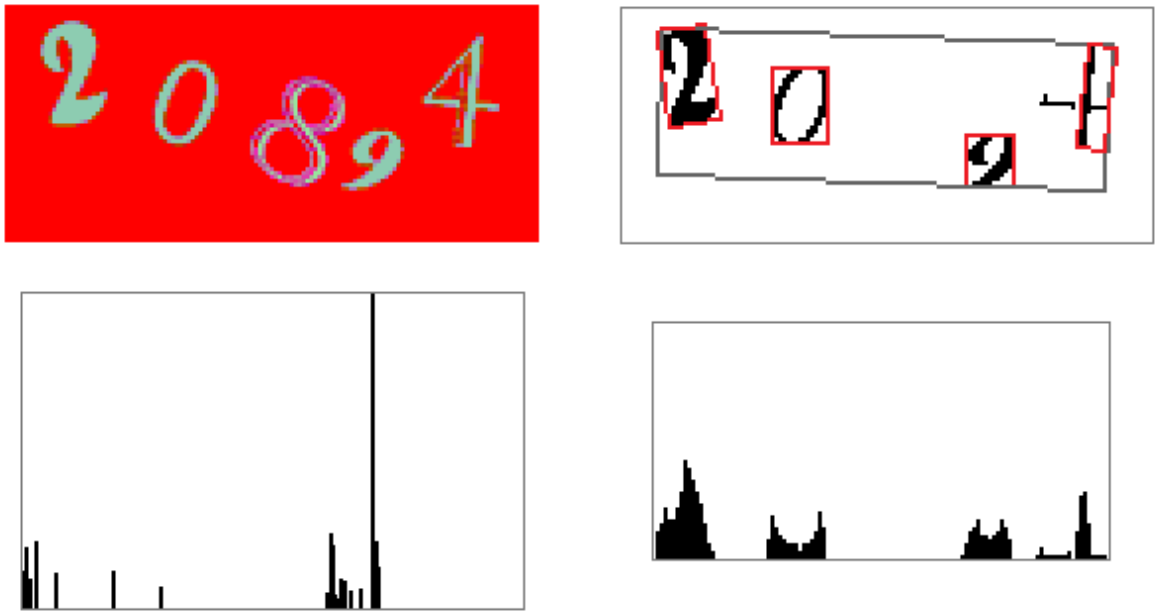
Nevýhodou je nalezení jen takových objektů, které jsou tvořeny jednou barvou v modelu HSV. Pokud je objekt složen z více barev (obrázek 12), algoritmus iterující přes histogram vyjme vždy jen jednu barvu, což pro objekt složený z více barev představuje jenom jeho část, která je buď natolik malá, že není ani označena jako objekt, nebo reprezentuje úplně jiný tvar (jiný objekt). Příkladem může být práce jen s částí číslice čtyři na obrázku 12.

Obrázek 12 – Chybná detekce objektů (8, 4) složených z více barev



Obrázek 13 - Ukázka segmentace podle barvy (zleva doprava: zdrojový obrázek, obrázek v HSV modelu, H složka, obrázek bez pozadí a šumu, obrázek v HSV modelu, obrázek s vyznačenými objekty)

Jak ukazuje obrázek 13, segmentace na základě barvy nemá problém s dotýkajícími či překrývajícími se objekty a šestka byla správně nalezena. Bohužel však čtyřka není složena z jedné barvy a detekována nebyla.

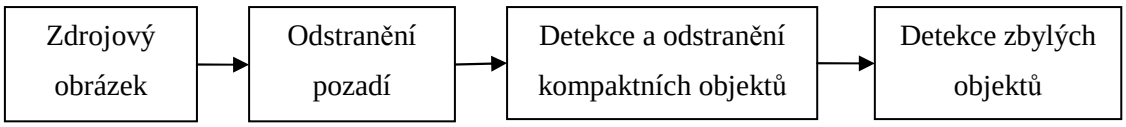


Obrázek 14 - Ukázka detekce objektů podle barvy

Obrázek 14 v jednotlivých svých částech popisuje jednu iteraci přes histogram. Vlevo nahoře se nachází obrázek v HSV modelu zbaven pozadí a šumu, vlevo dole pak histogram složky H. Vpravo nahoře je obrázek tvořený jen barvou odpovídající maximu v histogramu. Vpravo dole se pak nachází vertikální projekce, podle které jsou detekovány objekty. Je zde vidět, že z posledního objektu - čtyřky - se přenesla jenom část. Bohužel i vertikální projekce se rozpadla na dvě části a čtyřka tak nebyla správně detekována.

3.3 Návrh výsledné segmentační metody

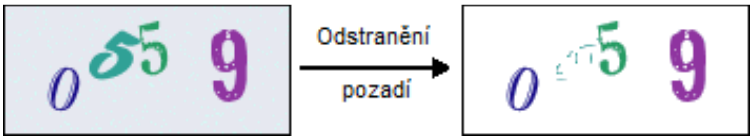
V předchozích kapitolách jsem se zabýval dvěma přístupy řešícími segmentaci. Základní metoda měla největší nedostatek v předpokladu nedotýkajících resp. nepřekrývajících se objektů. Na základě tohoto nedostatku základní metody byla vyzkoušena segmentace na základě barvy. Tato metoda se dokáže vypořádat s dotýkajícími resp. překrývajícími se objekty, jelikož takovéto objekty musejí mít odlišnou barvu, jinak by je nedokázal odlišit ani člověk. Bohužel i tato metoda má jedno zásadní omezení, a to takové, že při selekci jedné konkrétní barvy v obrazu dojde u objektů složených z více barev k vybrání jen části pixelů z objektu a objekt tak není správně detekován. V následujících kapitolách se tedy pokusím popsat segmentaci vstupního obrázku na jednotlivé kandidáty na číslice. Tato metoda byla navržena podle výsledků výše zmíněných metod. Segmentace probíhá v následujících krocích:



Obrázek 15 - Blokové schéma navržené metody

Odstranění pozadí

Odstranění pozadí z obrázku je provedeno v barevném modelu HSV. Jen pro složku H, která reprezentuje barevný odstín, je spočítán histogram. Poté se v cyklu přes maxima histogramu počítá vertikální projekce selektované barvy. Pokud je více než půlka šířky obrázku nenulových pixelů, je tato barva označena jako pozadí či šum v obraze a je z obrázku odstraněna. Samozřejmě pokud objekt v popředí je jen jiným odstínem pozadí (složka H je stejná, jen se liší složka S nebo V), je bohužel touto metodou spolu s šumem či pozadím odstraněn také objekt v popředí. Z testovaných obrázků však došlo k odstranění objektů jen v několika případech, takže tuto metodu lze považovat za použitelnou.



Obrázek 16 – Nepříjemný jev, kdy při odstraňování pozadí dojde k odmazání nějaké číslice

Detekce a odstranění kompaktních objektů

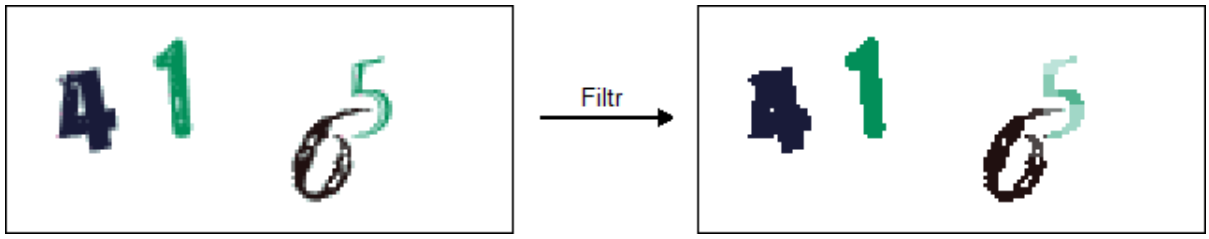
Původním cílem tohoto kroku bylo odstranění jednobarevných dobře detekovatelných objektů, které se klidně mohou dotýkat jiných číslic. Na obrázku 17 je to např. číslice dva, která je tvořena jednou barvou a jedná se o poměrně větší objekt.

Obrázek 17 - Číslice 2 je považována za jednobarevný dobře detekovatelný objekt

Jejím odstraněním z obrázku by následně došlo ke snadnému detekování číslic osm a čtyři. Lidskému oku se zdá, že by tento způsob mohl i fungovat, bohužel není tomu tak. Většinou po odstranění maximálně zastoupené barvy zůstane v obrázku „obálka“ číslice (obrázek 18) tvořená lehce jinou barvou (resp. barvami), a ta pak způsobí chyby v segmentování zbylých číslic.

Obrázek 18 - Nejvíce zastoupená barva číslice dva byla odstraněna (obarvena na bílo)

Právě byly nastíněny důvody, proč metoda nemůže dosáhnout požadovaných cílů. Záměru s rozpoznáním (a následným vymazáním) dobře detekovatelných objektů jsem se však nevzdal a vymyslel následující postup, jak odstranit část objektů z obrázků a zmenšit tak počet objektů postupujících do dalších částí zpracování. Cíle v odstranění jednoho z více spojených objektů však nebude dosaženo. Na obrázek s odstraněným pozadím aplikuji opakovaně filtr, který pro každý pixel a jeho okolí 3x3 nalezne nejvíce zastoupenou barvu popředí (barva pozadí se do výpočtu nezapočítává) a pixelu je tato barva nastavena, dokud dochází ke změně barvy alespoň jednoho pixelu (obrázek 19). Poté jsou v obrázku nalezeny souvislé objekty, které jsou tvořeny jednou barvou. Tyto objekty jsou označeny za rozpoznané a jejich pixely jsou obarveny barvou pozadí, tím tedy dojde k jejich vyřazení z dalšího zpracování. Bohužel při tomto postupu nikdy nedojde k odstranění alespoň jednoho z dvojice dotýkajících se objektů a tak původní záměr není v tomto smyslu vůbec naplněn. Tento krok tedy neusnadňuje rozpoznání dotýkajících se objektů.



Obrázek 19 - Aplikace filtru na obrázek

Na obrázku 19 můžeme pozorovat, že všechny pixely prvních dvou číslic (4, 1) mají stejnou barvu. Tyto objekty jsou tedy uloženy jako detekované a z obrázku jsou odstraněny, takže se neúčastní dalšího zpracování (samozřejmě až do kroku klasifikace). Tím dojde k mírnému urychlení dalších kroků segmentace, jelikož se bude pracovat s méně objekty.

Detekce zbylých objektů

Ani v této metodě se nepodařilo splnit původní záměr v použití základní metody. Po aplikování předchozí metody nedochází k odstranění jednoho z dvojice dotýkajících se objektů. Základní metoda spočívající ve vertikální projekci (viz. kapitola 3.1), která spojené objekty detekuje jako jeden, je tedy nepoužitelná a musí se použít sofistikovanějšího přístupu. Navržený algoritmus se skládá ze dvou částí:

1. Nalezení jednobarevných spojitých oblastí

2. Spojování jednobarevných spojitých objektů do větších celků

Nalezení jednobarevných spojitých oblastí

Pro všechny zastoupené barvy v obraze je postupně spočítána vertikální projekce. Ze shluků je vytvořen seznam objektů, které postupují do dalšího kroku zpracování.

Obrázek 20 – Jednobarevné spojité oblasti (pro názornost byly zobrazeny ohraničující obdélníky)

Spojování jednobarevných spojitých objektů do větších celků

Jako nejobecnější (s ohledem na neobecná pravidla, která byla vytvářena na trénovací množině, která na testovací množině občas nelogicky propojovala objekty z dvou různých číslic) se ukázala metoda spojování objektů propojená s klasifikátorem. Nejlepší metodou by bylo vyzkoušet všechny možnosti propojení objektů, ty všechny klasifikátorem ohodnotit a vybrat to propojení objektů, které dává celkově nejlepší výsledek. Stavový prostor všech možných kombinací propojení objektů je však velký (vzhledem k rychlosti klasifikátoru), a proto je vhodné zanést do algoritmu různé úpravy vedoucí ke snížení počtu vyšetřovaných kombinací. Objekty jsou tedy postupně spojovány až do vzniku jednoho objektu.

Algoritmus spojování objektů

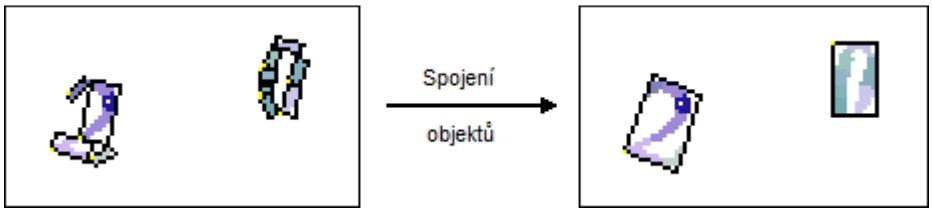
1. if (počet objektů > 0) then ulož ohodnocení (ohodnocení znamená klasifikace, čím menší hodnota, tím více se objekt podobá nějakému vzoru) všech objektů
2. while (počet objektů > 1) do begin
3. najdi a spoj dva nejlépe se hodící objekty pro spojení
4. ulož ohodnocení všech objektů
5. end
6. vyber z uložených ohodnocení to nejlepší, a to prohlaš za optimální propojení objektů

Z algoritmu jasně vyplývá, že výběr dvou objektů, které budou spojeny do jednoho, je zásadní. Pokud by hned v prvních iteracích algoritmu byly spojeny nevhodné objekty, bude se s touto chybou pracovat dále a algoritmus objekty špatně propojí. Funkce hledající dva nejvhodnější objekty pro spojení pracuje následovně:

1. Pro nejmenší objekt najdi:
    a. nejbližší dotýkající se objekt, pokud žádný neexistuje, tak

b. nejbližší objekt, který se nachází do určité vzdálenosti

2. Pokud nebyl k nejmenšímu objektu nalezen druhý objekt pro spojení, najdi dva nejbližší objekty (bez ohledu na jejich velikost)



Obrázek 21 - Ukázka výsledku algoritmu spojování objektů do větších celků

Z pseudokódu je patrné, že spojování souvisejících objektů, záleží čistě na prostorové metrice. Nejprve se hledá v objektech dotýkajících se, poté v objektech prostorově blízkých. Zkoušel jsem přidat pravidla související s barevnou informací jednotlivých objektů, např. „spoj objekty blízké s podobnou barvou“ atd. Bohužel tyto pravidla měla za následek zhoršení výsledků algoritmu, jelikož v několika případech docházelo k propojení dvou objektů, které k sobě nepatřily (byly z jiných číslic, ale blízko sebe a složeny z části stejných barev). V těch případech, kde pravidla využívající barvu objektů sloužila ke správnému propojení, stačilo k dosažení stejného výsledku pracovat čistě s prostorovou metrikou. Výsledný algoritmus propojování menších objektů tedy informaci o barevném složení objektů nevyužívá.

4 Rozpoznání číslic

Tato kapitola se věnuje popisu optického rozpoznání znaků. Dále vysvětluje výběr použitého typu klasifikátoru a popisuje princip několika navržených metod klasifikace.

4.1 Klasifikace s využitím OCR metod

Optické rozpoznávání znaků, obvykle zkracováno jako OCR, zahrnuje počítačový software navržený pro převod obrazu se strojově tištěným textem (obvykle pořízeným pomocí scanneru) do podoby textu editovatelného pomocí standardního textového editoru. OCR je dnes předmětem výzkumu v oboru umělé inteligence a počítačového vidění.

Historie

Již v roce 1929 si Rakušan Gustav Tauschek nechal patentovat metodu mechanického rozpoznávání textu na principu porovnávání rozpoznávaného znaku se šablonou pomocí světločivého receptoru. Originální text byl podsvícen a jestliže po přiložení tmavé šablony neprocházelo do receptoru žádné světlo, byl znak vyhodnocen jako znak na aktuálně použité šabloně. Patent byl později zakoupen společností IBM stejně jako mnoho dalších patentů vztahujících se k OCR v následujících letech.

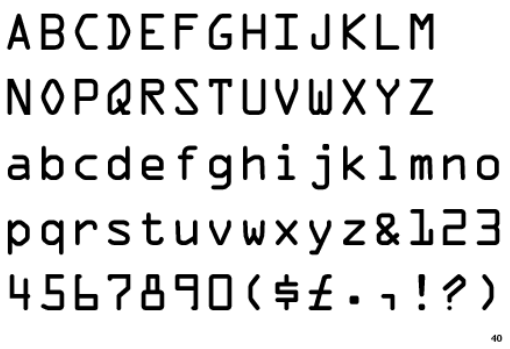
První systém pro převod textu do editovatelné digitální podoby byl patentován roku 1953 americkým kryptoanalytikem Davidem H. Shepardem pod prozaickým názvem „Apparatus for reading“. Již rok před patentováním založil Shepard společnost Intelligent Machines Research Corporation (IMR), která pomohla výsledky jeho práce prezentovat komerčnímu světu. Velký zájem o technologii automatického rozpoznávání textu měl díky nárůstu používanosti platebních šeků bankovní sektor. Jeho zájem se sice po čase přesunul k technologii magnetického inkoustu, avšak další zakázky na sebe nenechaly dlouho čekat. Široké uplatnění našla metoda OCR například v poštovních službách, kde se dodnes využívá při třídění zásilek.

První generace snímacích systémů využívala složité soustavy zrcadel, hranolů, štěrbin a světelných násobičů, které rozkládaly obraz snímaného znaku a směrovaly jej do matice světločivých prvků, kde bylo dopadající světlo převedeno do digitální podoby. Tyto přístroje byly jen málo flexibilní a vyžadovaly, aby byl rozpoznávaný text psán speciálním typem písma v přesně vymezených polích na stránce.

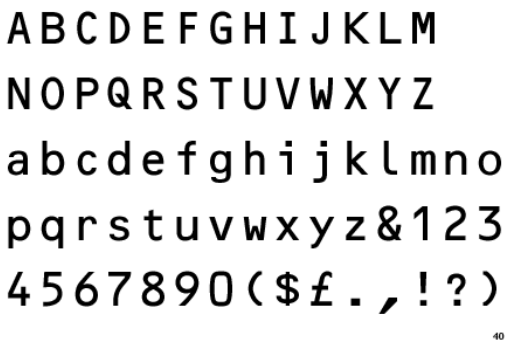
Přístroje druhé generace, představené v polovině 60. let, využívaly katodovou trubici a světelné pero pro metodu známou jako sledování křivek. To dalo podnět pro další výzkum v oblasti rozpoznávání ručně psaných textů.

Třetí generace přišla na počátku 70. let. Snímaná předloha byla osvícena a odražené světlo bylo vyhodnocováno polem fotocitlivých diod. Šlo tedy již prakticky o zařízení, které dnes nazýváme scanner.

Ke zlomu ve vývoji došlo v roce 1966, kdy se v USA standardizovalo tzv. písmo OCR-A, první písmo umožňující strojové učení. Písmo se používalo ve velkých bankách. Tvary tohoto písma byly zjednodušeny, aby strojové učení bylo co nejpřesnější, bohužel však na úkor čitelnosti pro člověka. To vedlo ke vzniku standardu OCR-B v roce 1968. Autorem byl Adrian Frutiger. Tento standard zlepšil čitelnost pro člověka, pro stroj je hůře čitelný než standard OCR-A.



Obrázek 22 - Standard OCR-A (obrázek převzat z [6])



Obrázek 23 - Standard OCR-B (obrázek převzat z [6])

Současnost

Oblasti automatického rozpoznávání textu můžeme rozlišit na několik oblastí podle vzniku digitálního obrazu s textem (ne ve smyslu digitalizace předlohy, ale ve smyslu vzniku předlohy). Strojový text a jeho rozpoznání se v dnešní době považuje za téměř vyřešené. Některé komerční systémy dosahují chybovosti v řádu jednotek znaků na stránku, a to se většinou týká speciálních znaků, jako jsou závorky, interpunkční znaménka atd. I programy dodávané např. s tiskárnou dosahují velké kvality rozpoznání. Upravení takového textu do výsledného tvaru představuje úpravu několika znaků na stránku.

Ručně psaný text a jeho rozpoznání je předmětem dnešního vývoje. Znaky v tomto textu jsou spojeny do slov, takže segmentace řádků na jednotlivá písmena je velice obtížná, přitom je to důležitý

předpoklad pro následné správné rozpoznání. Existují algoritmy pro „rozpoznávání za běhu“, které těží ze znalosti pořadí, směru a rychlosti jednotlivých znaků.

Kurzíva také čeká na vyřešení. Zde se dosahuje jako u ručně psaného textu malé úspěšnosti rozpoznání. Jelikož segmentace jednotlivých písmen je v tomto stylu písma velice obtížná, používá se i metoda, která se snaží rozpoznat celá slova.

Složitost rozpoznání počítačem vygenerovaného textu závisí na způsobu, jakým takovýto text vznikl. Autor ve své práci [8] ukazuje na vznik obrazového spamu. Autor takového spamu využívá zaměření anti-spamových filtrů na text a ne na přílohy. Text spamu je tedy umístěn do obrazové přílohy. Takovýto text tedy nebývá žádným způsobem poškozen a v podstatě se jedná o stejný případ, jako když naskenujeme strojový text bez jakéhokoliv šumu a dalších artefaktů, které mohou vzniknout při digitalizaci dokumentu. Druhým typem takovýchto „textových obrázků“ jsou (viz. kapitola 1.1) kontrolní obrázky s kódem, kdy je text v obrázku záměrně poškozen takovým způsobem, aby automatické metody rozpoznání selhaly, ale přitom rozpoznání textu pro člověka bylo bezproblémové.

4.2 Vyhledání nejbližšího souseda

Posledním krokem je klasifikace segmentovaných objektů do jednotlivých tříd. Jedná se v podstatě o nalezení zobrazení $f: P \rightarrow C$, kde P je příznakový vektor a $C = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. Jedním ze způsobů, jak klasifikovat vzor do odpovídající třídy, je pravidlo nejbližšího souseda. Rozhodl jsem se použít tohoto způsobu rozpoznání než rozpoznávače založeného např. na neuronových sítích, protože třída obrázků, které se budu snažit rozpoznat, má tendence ke změnám (autor této ochrany může např. přidat další typ prostorově deformovaných číslic apod.). Dostatečná reakce na to může být v mém případě jen přidání několik reprezentantů do tříd se vzory. U neuronové sítě by nejspíše došlo na zdlouhavé přeučování na změněné trénovací sadě. Tímto krokem se tedy moje řešení stane flexibilnější vůči případným změnám.

Pro každou třídu existuje jeden až několik reprezentantů, které vybíráme tak, aby třídu co nejlépe charakterizovali. Existuje více různých přístupů pro výběr reprezentantů. Nejčastěji vybíráme reprezentanty z trénovací množiny, ale nemusí to být pravidlem. Nejčastěji volíme počet reprezentantů pro každou třídu stejný, ale opět to nemusí být pravidlem. Množinu reprezentantů pro třídu *i* si označme Y_i, potom $Y = \bigcup Y_i$.

Vyzkoušel jsem následující dva způsoby, jak sestrojit vzory pro jednotlivé třídy:

1. Ručně jsem nakreslil několik nejlépe odpovídajících reprezentantů pro každou třídu.

Obrázek 24 - Ukázka tří ručně nakreslených reprezentantů nuly

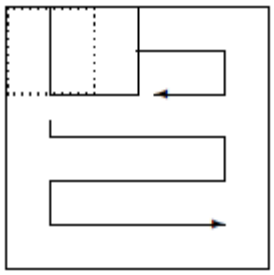
2. Vstupní množinu obrázků jsem rozdělil na trénovací a testovací část. Z trénovací množiny jsem vybral dobře segmentované objekty a ty jsem prohlásil za reprezentanty dané třídy.

Obrázek 25 - Čtyři reprezentanti programem vysegmentovaných nul

Další krok představuje výběr měřitelných veličin - příznaků, které reprezentují binární obraz. Příznaky zapsané do posloupnosti tvoří tzv. příznakový vektor. Důležité je vybrat takovou množinu příznaků, které daný objekt co nejlépe popisují, ale přitom ho odlišují od ostatních. Preferují se příznaky odolné vůči šumu či očekávaným prostorovým transformacím. Výhodou také je, pokud se dají rychle a jednoduše spočítat.

Výběr příznaků

Autor ve své práci [7] zmiňuje více typů příznaků, podle dostupných výsledků jsem se rozhodl použít tzv. příznaky typu Win, které jsou odolné vůči šumu. Obrázek je postupně odshora procházen oknem o dané velikosti střídavě zleva doprava a zprava doleva (obrázek 26).



Obrázek 26 - Postup při získávání příznakového vektoru typu Win (tečkovaně je vyznačena první pozice okna, plnou čarou následující pozice okna)

Sousední pozice okna se z 50% jak v horizontálním, tak ve vertikálním směru překrývají. V každé pozici okna se spočítá poměr nenulových pixelů (tedy pixelů objektu) k ploše okna. Různých pozic (a tedy i hodnot v příznakovém vektoru) je celkem:

$$počet\ hodnot = \left(\left\lfloor \frac{šířka\ obrázku}{H\_překrytí * šířka\ okna} \right\rfloor - 1\right) * \left(\left\lfloor \frac{výška\ obrázku}{V\_překrytí * výška\ okna} \right\rfloor - 1\right)$$

H_překrytí znamená horizontální překrytí v intervalu $\langle 0, 1)$ a *V_překrytí* znamená vertikální překrytí opět v intervalu $\langle 0, 1)$. V mém konkrétním případě má obrázek rozměry 40*30 pixelů. Okno má rozměry 6*6 a používá se 50% překryv pro oba směry. Po dosazení vychází 117 příznaků. Střídavé procházení zleva doprava a naopak má význam pro zachování „spojitosti“ měřených hodnot.

Vlastní rozpoznání pak může probíhat pomocí metriky euklidovské vzdálenosti. Pro dva n-dimenzionální příznakové vektory $x = (x_1, \dots, x_n)$ a $y = (y_1, \dots y_n)$ je euklidovská vzdálenost definována:

$$\|x - y\| = \sqrt{(x_1 - y_1)^2 + \cdots + (x_n - y_n)^2}$$

Na základě této metriky je nalezen nejbližší reprezentant $y_i \in Y_i$, pro kterého platí:

$$\|x - y_i\| \leq \|x - y\| \; pro \; všechna \; y \in Y$$

a neznámý vzor je klasifikován do třídy i.

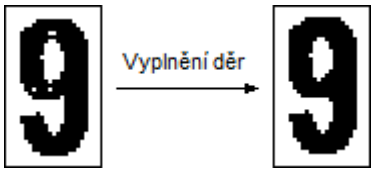
Jelikož je odmocnina rostoucí funkce, můžeme ji z výpočtu vynechat, protože nemá vliv na porovnání dvou příznakových vektorů.

Normalizace

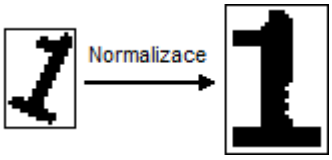
Ve zdrojovém obrázku se nacházejí číslice různých velikostí a natočení, proto musí být výpočet příznaků prováděn na tzv. normalizovaných obrázcích. Rozměry každého objektu vstupujícího do klasifikace jsou upraveny na požadovanou velikost (v našem případě na velikost 30x40). Další problém představuje rotace číslic v obrázku. Po úpravě velikosti následuje úprava natočení číslice do vzpřímené polohy. Úhel natočení jednotlivých číslic byl zjištěn pomocí knihovny OpenCV. Ta obsahuje funkci, která pro zadaný seznam bodů vypočítá nejmenší ohraničující obdélník. O tomto obdélníku jsou k dispozici střed, šířka, výška a úhel natočení. Po úpravě úhlu, která vychází ze zjištění, zda je vypočítaný ohraničující obdélník naležato či nastojato, následuje krok narovnání číslice do vzpřímeného stavu.

Pokud nebude číslice natočena správně do vzpřímené polohy, nemůžeme při klasifikaci takovýchto objektů očekávat dobré výsledky. Možnosti řešení máme v podstatě dvě. Buď zdokonalíme systém normalizace, který bude správně natáčet i takovéto objekty, nebo s touto chybou v normalizaci budeme počítat a při ručním vytváření vzorů tuto nepřesnost zaneseme i do vzorů jednotlivých tříd.

Dalším zařazeným krokem v normalizaci jsou morfologické operace. Po segmentaci se totiž v obrázcích nacházejí díry (pixely pozadí), zubaté okraje atd., které je potřeba vyplnit. Tyto vlastnosti (vyplňování děr, vyhlazení zubatých okrajů) splňuje operace binární uzavření, a proto byla použita.



Obrázek 27 - Aplikování binárního uzavření má za následek vyplnění děr



Obrázek 28 - Normalizace číslice jedna (rotace a zvětšení na požadovaný rozměr)

Pravidlo *k* nejbližších sousedů

Pro klasifikaci neznámého vzoru *x* je nutné spočítat jeho vzdálenost od všech reprezentantů v příznakovém prostoru. Asi nejčastěji se k výpočtu vzdálenosti uplatňuje již zmíněná euklidovská metrika.

Existují dva různé postupy, jak klasifikovat vzor x do některé z *c* tříd:

1. Najdeme *k* nejbližších reprezentantů a zjistíme, do které třídy patří. Neznámý vzor *x* je klasifikován do třídy, která je nejvíce zastoupena mezi *k* nejbližšími reprezentanty.
2. Procházíme postupně reprezentanty od nejbližšího k nejvzdálenějšímu, dokud jsme nenalezli požadovaný počet (*k*) reprezentantů z jedné třídy. Do této třídy poté klasifikujeme i neznámý vzor *x*.

V prvním případě se může stát, že algoritmus nedokáže rozhodnout mezi dvěma stejně zastoupenými třídami.

Parametr *k* je nutné volit s ohledem na počet reprezentantů v každé třídě. Pro druhý postup klasifikace nesmí *k* přesáhnout nejmenší počet reprezentantů v některé ze tříd. Obvykle volíme hodnotu parametru *k* dosti malou. Nejlepší je hodnotu parametru *k* volit experimentálně vyzkoušením více hodnot a zvolit tu, která dává nejlepší výsledky.

4.3 Extrakce příznaků z obrazu

K nalezení nejbližšího souseda jsem použil několik metod, jejich popis se nachází v této kapitole. Nejprve jsem implementoval template matching velice primitivně prostým porovnáním vzoru a neznámého objektu. Výsledky však nebyly dobré (viz. kapitola 5.3). Vyzkoušel jsem proto další způsob s použitím příznaků typu Win, který podával o hodně lepší výsledky. V posledních dvou metodách jsem se pokusil využít koster objektů, které mají předpoklady podávat dobré výsledky.

Jednoduchý přístup

Klasifikátor má na vstupu normalizovaný objekt a ručně nakreslené reprezentanty pro každou třídu. K porovnání vzoru a neznámého objektu jsem použil metodu, kdy je neznámý objekt normalizován na velikost vzoru, binární obrázky vzoru a neznámého objektu jsou na sebe virtuálně přiloženy a je sečten počet pixelů, ve kterých se vzor i neznámý objekt shodují (tzn. oba obsahují barvu objektu nebo oba obsahují barvu pozadí) a počet pixelů, ve kterých se neshodují (tzn. jeden z nich obsahuje barvu objektu a druhý barvu pozadí). Vzor, který má s neznámým objektem největší počet shodných pixelů a zároveň nejmenší počet neshodných pixelů, je vybrán jako nejbližší danému neznámému objektu a neznámý objekt je klasifikován do třídy, ze které pocházel nejbližší vzor. Zde byl tedy parametr *k* zvolen 1.

Příznaky typu Win

Jednoduchý přístup popsaný v kapitole výše nedával příliš dobré výsledky (viz kapitola 5.3), proto jsem se rozhodl použít odlišný přístup, který použil autor ve své práci [7].

Předchozí metoda v podstatě používala příznakový vektor o velikosti dva. Tato hodnota je příliš malá a je jasné, že větší velikost příznakového vektoru může podávat lepší výsledky. Proto byly v tomto případě použity příznaky typu Win. Při klasifikaci je pak pro neznámý objekt (stejně jako pro všechny reprezentanty všech tříd) spočítán příznakový vektor typu Win, je spočtena jeho vzdálenost od všech vzorů ze všech tříd a neznámý objekt je klasifikován do třídy, ze které pocházel příznakový vektor vzoru, který byl nejméně vzdálen příznakovému vektoru neznámého objektu. Opět jako v předešlé metodě byla hodnota parametru *k* zvolena 1.

Druhou odlišností oproti první metodě je rozdělení obrázků na testovací a trénovací sadu (výraz trénovací přesně neodpovídá účelu, jelikož tato sada není použita k žádnému učení/trénování, ale jsou z ní vytvořeny reprezentanti tříd). Z trénovací sady byly programem vysegmentovány neznámé objekty, ty byly ručně oklasifikovány a zařazeny do tříd vzorů. Nebylo tedy ručně nakresleno několik reprezentantů pro každou třídu.

Kostra objektů

Poslední metoda, kterou jsem vyzkoušel, si všímá faktu různé tloušťky objektů ze stejné třídy, proto musí existovat více reprezentantů, aby byly pokryty všechny varianty každé číslice. Nabízí se tedy možnost nalézt kostru objektu (samozřejmě i reprezentanti jednotlivých tříd musejí být kostrami skutečných číslic) a klasifikaci provést na porovnání resp. hledání nejbližší kostry. Pro nalezení kostry objektu jsem použil metodu Zhang-Suen, popsanou v teoretickém rozboru v první kapitole. Nejdůležitější částí této metody je nalezení nejbližší kostry. Vyzkoušel jsem dva přístupy.

První přístup počítá blízkost koster způsobem, kdy je pro každý pixel kostry vzoru neznámého objektu nalezen nejbližší pixel v kostře vzoru. Celkově je blízkost kostry neznámého objektu a kostry

vzoru ohodnocena jako suma všech vzdáleností pro všechny pixely kostry neznámého objektu. Vzdálenost dvou pixelů jsem spočítal jako prostý součet absolutních hodnot rozdílů x-ové a y-ové souřadnice. Algoritmus musí pracovat způsobem, kdy pro každý pixel neznámého objektu hledá pixel kostry vzoru a ne naopak. Kostry jednotlivých vzorů nemají stejný počet pixelů, takže pokud by algoritmus pracoval způsobem, kdy by pro každý pixel vzoru hledal nejbližší pixel v neznámém objektu, suma všech takovýchto vzdáleností by mohla být menší pro kostru vzoru méně odpovídající s málo pixely, než pro kostru lépe odpovídající, ale s více pixely. Tuto skutečnost potvrdily i testy, které jsem provedl. Výsledky klasifikace pro případ, kdy se pro každý pixel kostry vzoru hledá nejbližší pixel v neznámém objektu, byly výrazným způsobem horší a celková klasifikace obrázků se blížila k 0%. Bohužel ani „správný“ způsob hledání nejbližších pixelů nepodává příliš dobré výsledky (viz kapitola 5.3).

Druhý přístup kopíruje metodu v předchozí kapitole (příznaky typu Win) s tím rozdílem, že příznaky typu Win jsou počítány z koster objektů.

5 Výsledky

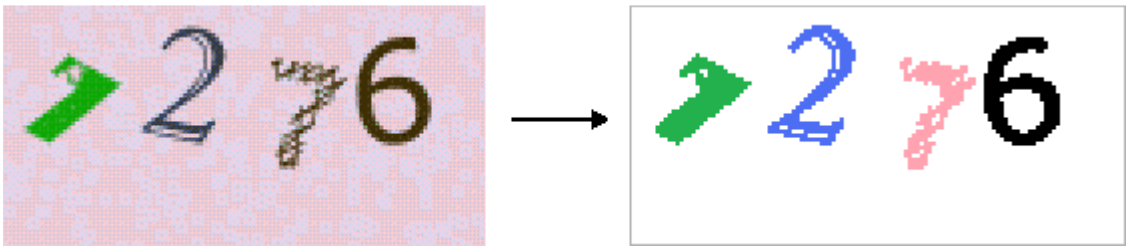
V této kapitole zhodnotím dosažené výsledky segmentace a výsledky jednotlivých metod klasifikace. Poslední podkapitola obsahuje celkové vyhodnocení úspěšnosti rozpoznání kódu z kontrolního obrázku.

5.1 Anotovaná data

Jako u každé práce zabývající se řešením nějakého problému, i zde je potřeba zhodnotit dosažené výsledky, případně porovnat několik použitých přístupů. Metoda navržená v této práci si klade za cíl detekovat číslice v kontrolních obrázcích a jejich klasifikaci do tříd. Ke zhodnocení navržené metody je tedy potřeba testovací sada, která bude obsahovat správně detekované a rozpoznané objekty.

Data resp. kontrolní obrázky použité při vývoji aplikace byly rozděleny na trénovací a testovací sadu. Z celkových 300 stažených obrázků připadlo 100 na trénovací sadu a 200 na testovací sadu. Obrázky v trénovací i testovací sadě byly anotovány způsobem, kdy jsem ručně rozpoznal jednotlivé číslice na obrázku a zapsal je do souboru se správnými výsledky. Číslice jsou v souboru zapsány popořadě, jak se nacházejí v obrázcích (jsou tedy seřazeny podle osy x).

Jelikož segmentace podávala velice dobré výsledky (až na zmíněné problémy s odmazáváním číslic při mazání pozadí) nepovažoval jsem za nutné vytvoření anotovaných dat i pro segmentaci, tedy takových anotovaných dat, ve kterých by bylo vyznačeno, které pixely patří kterému objektu resp. číslici. Vytvoření těchto dat by bylo velice časově náročné a jelikož segmentace podává velice dobré výsledky, rozhodl jsem se je nevytvářet.



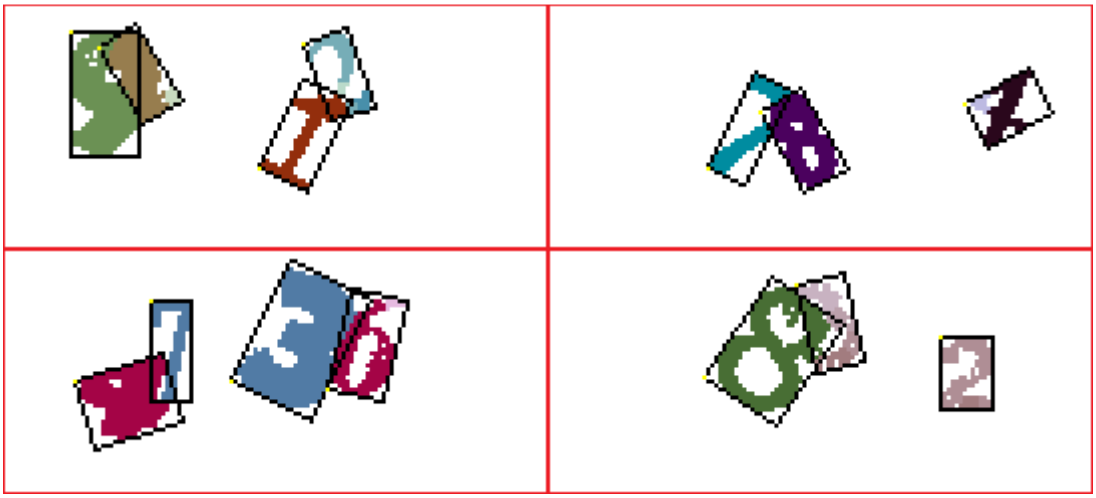
Obrázek 29 - Anotování dat obrázku pro segmentaci

Obrázek 29 znázorňuje možný přístup, jak anotovat objekty (jejich pixely) v obrázcích. Pro každou číslici v obrázku by bylo potřeba pixely ručně obarvit jednou barvou. Spolu s výše uvedenou anotací pro klasifikátor by pak byly dostupné údaje jak o číslicích v obrázku, tak o pixelech, který každé číslici náleží.

5.2 Segmentace

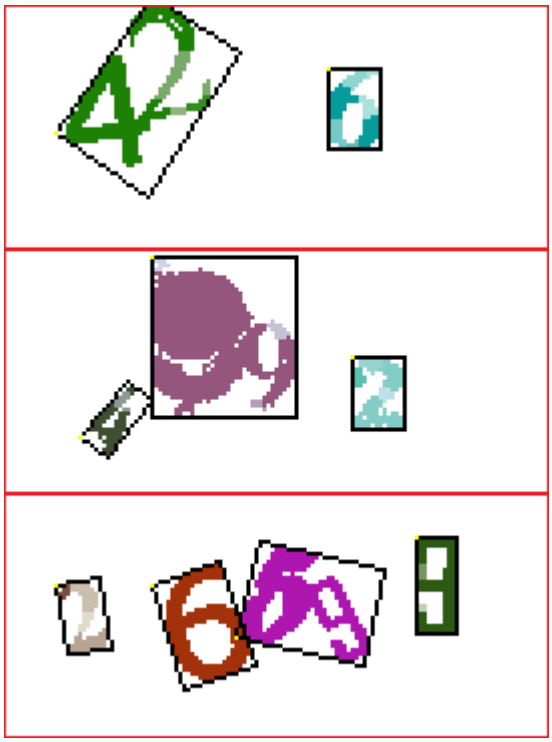
Výsledky segmentačního algoritmu jsou velice dobré. Segmentace skončila špatně v 6% případů (výsledkem segmentace byl jiný počet číslic, než se ve skutečnosti v obrázku nachází). Na těchto 6% se podílí krok odstranění pozadí, který někdy odmaže číslici z obrázku a také kroky vlastní segmentace, tedy nalezení jednobarevných spojitých oblastí a spojování jednobarevných spojitých objektů do větších celků.

Na obrázku 30 jsou zobrazeny čtyři příklady obrázků, které byly dobře segmentovány i přesto, že obsahují dotýkající se nebo významným způsobem překrývající se číslice. Podařilo se tedy vytvořit algoritmus, který dokáže segmentovat neosamocené číslice.



Obrázek 30 - Obrázky dobře segmentovaných číslic

Na obrázku 31 jsou naopak zobrazeny číslice, u kterých dochází ke špatné segmentaci. Ve všech případech se jedná o dvě propojené číslice několika pixely. Jak už víme, s tímto by se algoritmus dokázal vypořádat. Číslice jsou však tvořeny stejnou barvou a to je ten hlavní důvod, který ve spojení s tím, že se číslice dotýkají, způsobí segmentaci na jednu číslici. Číslice jsou tedy již propojeny z kroku „nalezení jednobarevných spojitých oblastí“. Možné řešení se nabízí snad jen v použití algoritmu, který by nějakým způsobem zjistil, že se jedná o propojené číslice a pokusil by se je od sebe oddělit. Takovýto algoritmus jsem však do práce nezahrnul, a tak se jeho vyvinutí nabízí jako možné rozšíření této práce.



Obrázek 31 - Obrázky, u kterých segmentace nedopadla správně

5.3 Klasifikace

Normalizace

V případě normalizace byl největší problém s některými variantami číslice čtyři (obrázek 32). Na níže uvedeném obrázku si můžeme všimnout, že vzory čtyřek, u kterých dochází ke špatné detekci natočení, mají jednu společnou vlastnost. V levém horním rohu tvoří část číslice zkosení, na kterém je přichycena jedna strana obalujícího obdélníku, ze kterého je počítána rotace číslice. Jelikož jsou však u klasifikační metody využívající příznaků typu Win vzory jednotlivých tříd programem automaticky segmentovány a uloženy, jsou teda i vzory špatně natočeny. Porovnávají se tedy dva stejně špatně natočené vzory, takže nám špatné natočení nevadí.

Obrázek 32 - Nalezené ohraničující boxy, které neodpovídají realitě natočení číslice čtyři (červeně jsou znázorněny automaticky nalezené boxy, zeleně pak boxy, které bychom očekávali)

Jednoduchý přístup

U tohoto přístupu jsem vyzkoušel zvýšit hodnotu parametru *k* (tedy nalézt více nejbližších vzorů) a z nich vybrat nejvíce zastoupenou třídu. Výsledky však byly výrazně horší. Zatímco třída nejbližšího vzoru většinou odpovídala, na dalších místech se umisťovaly vzory z jiných tříd. Toto chování je dáno velkou rozdílností reprezentantů jednotlivých tříd. Dobré výsledky bychom s větší hodnotou parametru *k* mohli dosáhnout při rozpoznávání vytisknutého textu, kde jsou jednotlivá písmena po segmentaci skoro stejná a tedy i vzory v jednotlivých třídách jsou velice podobné. V našem případě však existuje velké množství velice odlišných vzorů reprezentujících stejnou třídu (velká vnitřní variabilita), a proto pro větší hodnotu parametru *k* vycházejí horší výsledky.

Tabulka 1 - Výsledky rozpoznání jednotlivých číslovek, hodnota parametru *k* je rovna 1

číslice (úspěšnost detekce)	**počet detekcí pro jednotlivé číslovky**									
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0 *(74%)*	53	1	0	1	0	0	9	1	1	5
1 *(92%)*	0	58	0	0	0	1	0	4	0	0
2 *(69%)*	0	12	70	4	0	1	0	7	2	5
3 *(67%)*	0	6	9	44	0	1	0	2	2	1
4 *(56%)*	0	16	1	0	39	0	5	5	1	2
5 *(63%)*	0	1	0	12	0	54	5	0	4	9
6 *(70%)*	2	3	0	3	1	4	49	0	7	1
7 *(75%)*	0	25	0	0	1	0	1	84	0	0
8 *(79%)*	1	5	0	5	0	0	2	0	50	0
9 *(71%)*	6	2	0	4	0	3	1	0	5	52
celkem *(71%)*										

Pro přehlednost nebyly do tabulky zahrnuty jednotlivé vzory, pro které program počítá statistiku při špatné klasifikaci. Podrobné výsledky jsem umístil do kapitoly s přílohami A.5. Z těch se můžeme dozvědět, že „nejproblémovějším“ vzorem je vzor číslice jedna s názvem *4.png* .

Tabulka 2 - Počty chybných klasifikací číslovek

číslice	počet chybných klasifikací na vzor 4.png číslice jedna
2	11x
4	11x
7	17x

Zkusil jsem tedy tento vzor odstranit a znovu nechat oklasifikovat sadu číslic. Celková úspěšnost stoupla však jen o 1%. Sice už nebylo tolik chybných klasifikací na vzor *4.png* číslice jedna z ostatních tříd, úspěšnost klasifikace číslice jedna však klesla přibližně o 20%. Bohužel se jedná o vzor dobře reprezentující číslici jedna a jeho smazáním dojde ke snížení úspěšnosti klasifikace číslice jedna. Odstraněním vzoru tedy nedošlo k výraznému zlepšení výsledků klasifikace.

Úkolem uživatele služby chráněné CAPTCHA je opsání všech číslic na obrázku. Rozpoznání můžeme považovat za úspěšné jen v tom případě, kdy došlo k rozpoznání všech číslic na obrázku. Bohužel i jenom jedna špatně rozpoznaná číslice vede k neúspěchu.

Tabulka 3 - Výsledky rozpoznání celých obrázků (tedy všech číslic v obrázku)

Rozpoznaných obrázků	Nerozpoznaných obrázků
20%	80%

I když z výsledků rozpoznaných číslovek bychom na první pohled očekávali poněkud lepší výsledky celkového rozpoznání, projevil se zde fakt, že k úspěšnému rozpoznání kódu z obrázku je potřeba rozpoznat všechny číslovky. Takže i když rozpoznáme čtyři číslovky a pátou ne, nebyl kód z obrázku správně rozpoznán.

Výsledku 20% odpovídá i teorie o náhodných pokusech nezávislých jevů. Rozpoznání každé číslice je nezávislý jev (nefigurují tu pravidla, že se např. číslice neopakují v jednom obrázku atd.) a tedy výslednou pravděpodobnost získáme prostým vynásobením pravděpodobností každého rozpoznání. Počet číslic v obrázcích je čtyři nebo pět, výsledná pravděpodobnost rozpoznání by se tedy měla pohybovat mezi pravděpodobností pro čtyři a pro pět objektů. Jelikož jsou číslice na všech pozicích klasifikovány stejným algoritmem, můžeme tedy spodní mez intervalu vypočítat podle pravděpodobnosti rozpoznání pro pět číslic:

$$0{,}71^5 \cong 0{,}18$$

a horní mez jako pravděpodobnost rozpoznání pro čtyři číslice:

$$0{,}71^4 \cong 0{,}25$$

Výsledná pravděpodobnost se nachází v intervalu $\langle 0.18, 0.25 \rangle$.

Příznaky typu Win

Tabulka 4 - Výsledky rozpoznání jednotlivých číslovek z testovací sady obrázků (příznaky typu Win)

číslice (úspěšnost detekce)	**počet detekcí pro jednotlivé číslovky**									
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0 *(95%)*	68	1	0	1	0	0	1	0	0	0
1 *(93%)*	0	59	1	0	1	0	0	1	1	0
2 *(91%)*	0	2	92	2	1	0	0	3	1	0
3 *(90%)*	0	0	2	59	0	3	0	0	1	0
4 *(82%)*	1	2	1	0	57	0	0	0	2	7
5 *(96%)*	1	0	0	0	0	82	1	0	0	1
6 *(85%)*	2	0	0	0	0	4	60	0	3	1
7 *(85%)*	0	13	2	0	1	0	0	95	0	0
8 *(96%)*	0	0	0	0	0	0	2	0	61	0
9 *(89%)*	2	1	2	0	0	1	0	0	2	65
celkem *(90%)*										

Jak vidíme z tabulky výsledků pro jednotlivé číslice, vytvoření reprezentantů z trénovací množiny a použití příznaků typu Win významným způsobem zlepšují klasifikaci. Celkem bylo správně klasifikováno 90% číslic.

Analýzou podrobných výsledků v příloze A.5 můžeme zjistit, že se ve vzorech jednotlivých číslic nenachází výrazně špatný vzor, který by zhoršoval výsledky klasifikace jednotlivých číslic. Zlepšení výsledků bychom tedy mohli dosáhnout spíše než výměnou vzorů zvolením takových příznaků, které by byly zaměřeny na tvarové vlastnosti objektů a dokázaly by jednotlivé třídy lépe od sebe oddělit.

Tabulka 5 - Výsledky rozpoznání celých obrázků (tedy všech číslic v obrázku)

Rozpoznaných obrázků	**Nerozpoznaných obrázků**
64%	36%

I v tomto případě bychom opět z výsledků klasifikace jednotlivých číslovek mohli očekávat lepší výsledky. I tak je ale 64% správně rozpoznaných kódů z obrázků značné zlepšení oproti první metodě.

Opět výsledek odpovídá teoretickým předpokladům, jelikož se nachází v intervalu pro čtyři a pro pět rozpoznaných objektů.

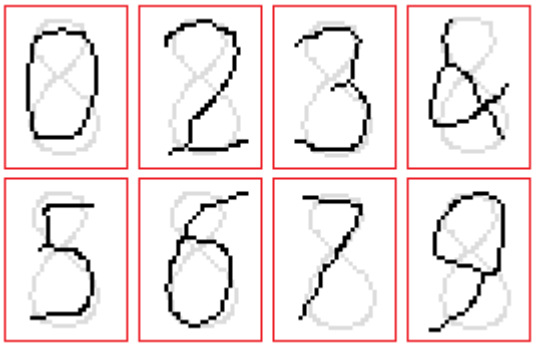
Pokud bychom požadovali např. 90% úspěšnost rozpoznání celých obrázků, jednoduchým výpočtem zjistíme, že klasifikátor by musel pracovat s přesností 98%, a to už se nacházíme v kategorii velice přesných klasifikátorů, kde by možná pomohly neuronové sítě, které obecně v klasifikátorech podávají velice dobré výsledky.

Kostra objektů

Tabulka 6 - Výsledky rozpoznání jednotlivých číslovek z testovací sady obrázků (kostra objektů)

číslice (úspěšnost detekce)	**počet detekcí pro jednotlivé číslovky**									
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0 *(70%)*	50	1	0	0	0	0	2	0	14	4
1 *(84%)*	0	53	6	2	1	0	0	0	1	0
2 *(71%)*	1	7	72	2	3	0	0	0	12	4
3 *(55%)*	0	1	3	36	2	2	0	0	21	0
4 *(68%)*	0	2	2	0	47	1	2	0	13	2
5 *(41%)*	0	3	0	6	2	35	19	0	18	2
6 *(45%)*	2	2	0	0	2	0	38	0	26	0
7 *(31%)*	5	20	26	2	4	0	0	35	11	8
8 *(85%)*	1	3	0	0	3	0	2	0	54	0
9 *(21%)*	3	1	1	4	14	0	4	3	27	16
celkem *(56%)*										

Z hodnot v tabulce si můžeme všimnout, že každá číslice s výjimkou jediného případu u číslice jedna (u číslice osm to naopak samozřejmě očekáváme) byla v několika případech klasifikována na číslici osm. Obrázek 33 názorně ukazuje, proč k tomuto jevu dochází. U koster zobrazených číslic si můžeme všimnout, že se většina jejich pixelů překrývá (či jsou velice blízko) s částí pixelů kostry číslice osm. Proto jsou tedy číslice z jiných tříd klasifikovány na třídu osm. Ke stejnému jevu samozřejmě dochází i mezi jinými třídami, např. při klasifikaci číslice sedm, kdy ji klasifikátor ve velkém množství případů zařazuje do třídy jedna či dva a při klasifikaci číslice devět, která je zase klasifikována do třídy čtyři.



Obrázek 33 - Přiložení koster všech tříd (mimo třídu jedna a osm) na kostru číslice osm (šedá barva)

Rozpoznání jednotlivých číslic dává horší výsledky než předchozí metoda. Tedy ani celkové rozpoznání výsledků nebude lepší.

Tabulka 7 - Výsledky rozpoznání celých obrázků (tedy všech číslic v obrázku)

Rozpoznaných obrázků	Nerozpoznaných obrázků
12%	88%

Druhý přístup, který používá příznaky typu Win, podává o 9% lepší výsledky. I přes toto zlepšení nejsou výsledky dobré.

Tabulka 8 - Výsledky rozpoznání jednotlivých číslovek z testovací sady obrázků (kostra objektů)

číslice (úspěšnost detekce)	**počet detekcí pro jednotlivé číslovky**									
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0 *(87%)*	62	2	0	0	0	0	2	2	0	3
1 *(71%)*	0	45	0	7	0	0	1	10	0	0
2 *(59%)*	0	17	60	8	0	0	2	8	0	6
3 *(80%)*	3	6	1	52	0	0	0	2	0	1
4 *(52%)*	0	10	0	2	36	0	7	8	1	5
5 *(57%)*	0	0	0	12	1	49	18	0	4	1
6 *(68%)*	6	4	0	1	0	1	48	4	2	4
7 *(73%)*	0	22	0	6	0	0	0	82	0	1
8 *(66%)*	6	1	0	4	0	1	4	3	42	2
9 *(36%)*	12	7	1	9	1	2	5	4	5	27
celkem *(65%)*										

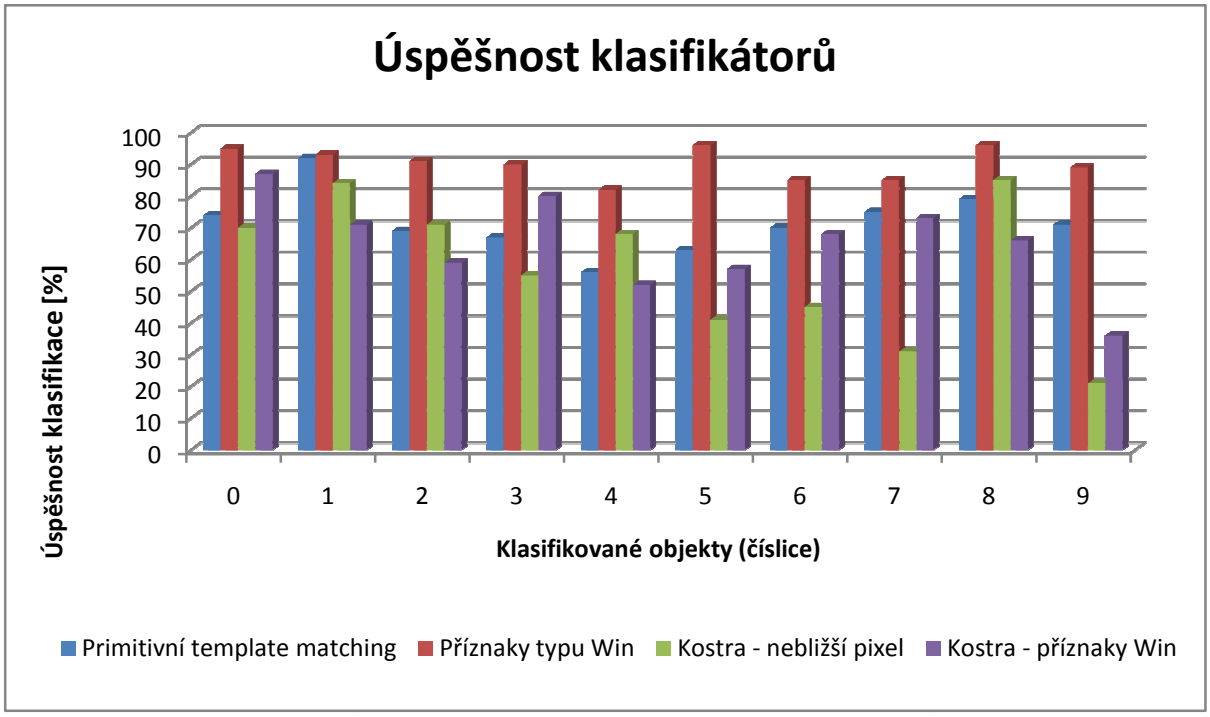
Tabulka 9 - Výsledky rozpoznání celých obrázků (tedy všech číslic v obrázku)

Rozpoznaných obrázků	Nerozpoznaných obrázků
16%	84%

Celkové výsledky rozpoznání obrázků jsou samozřejmě stále velice špatné a je tedy jasné, že navržené způsoby hledání nejbližší kostry nejsou příliš dobré a k dosažení lepších výsledků by bylo potřeba použít lepší metody. Tato oblast se tedy nabízí jako jedno z možných rozšíření této práce.

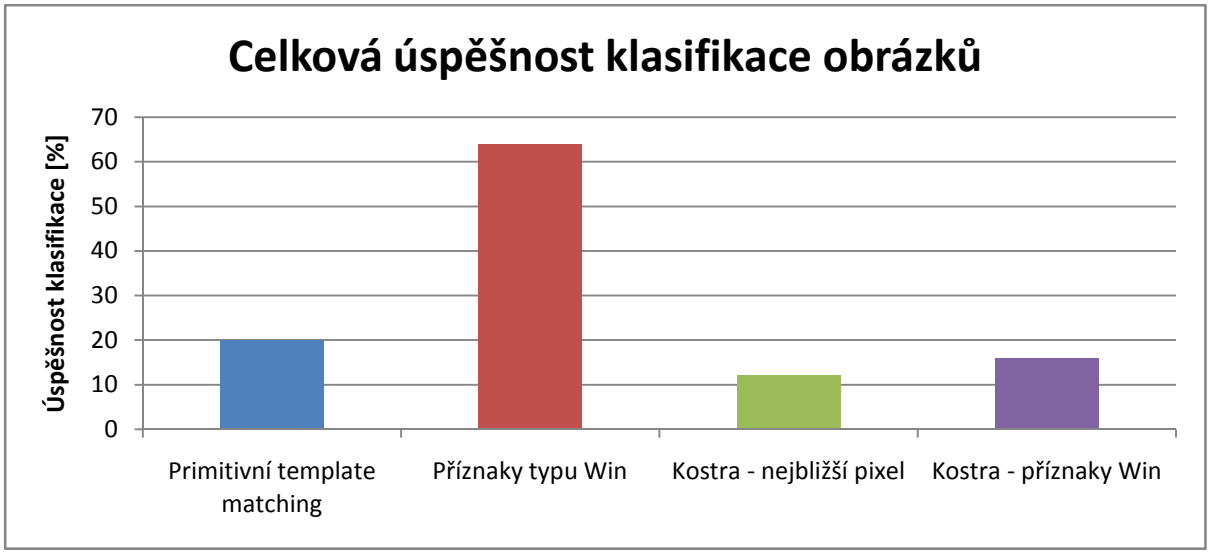
Zhodnocení klasifikátorů

V předešlých kapitolách byly prezentovány výsledky metod založených na jednoduchém porovnání pixelů, příznacích typu Win a koster objektů.



Obrázek 34 - Srovnání výsledků použitých klasifikátorů

Z grafu na obrázku 34 můžeme odečíst, že nejlepších výsledků dosahuje klasifikátor používající vzory z trénovací sady a příznaky typu Win. Tato metoda dosahuje na všech třídách přes 80% rozpoznání jednotlivých číslic. Na několika třídách se rozpoznání dostává za hranici 90%. Jako nejhorší se ukázal přístup klasifikace využívající koster objektů. Špatné výsledky bych spíše přisuzoval použitým metodám na porovnávání blízkosti dvou koster objektů. Osobně si myslím, že spojení klasifikátoru používajícího kostry objektů spolu s metodami, které jsou vhodné na porovnávání koster, má předpoklady podávat velmi dobré výsledky.

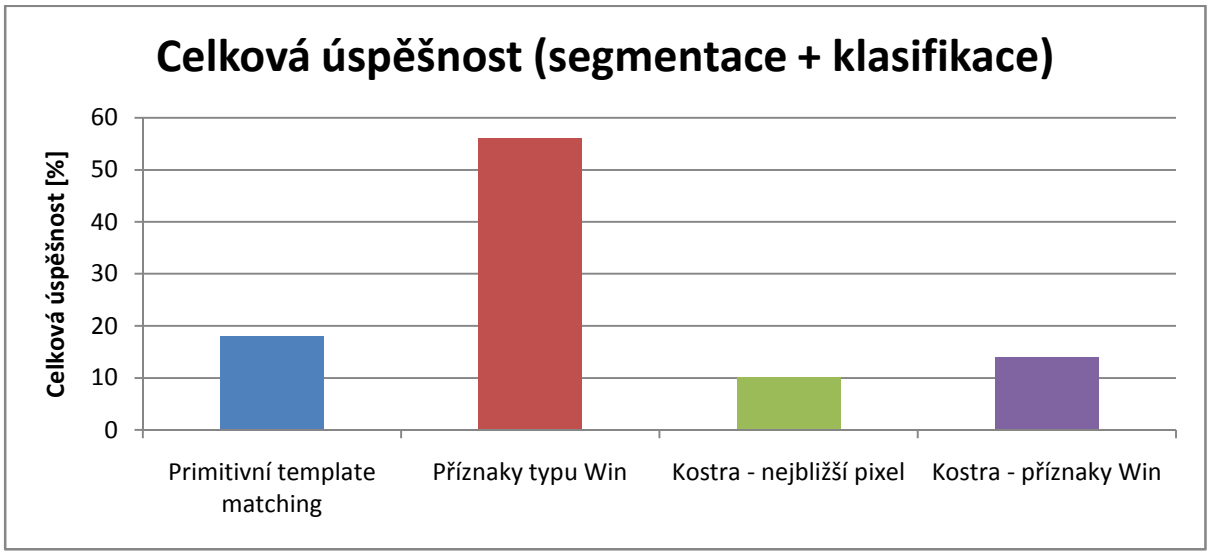


Obrázek 35 - Srovnání celkových výsledků klasifikátorů. Jedná se o výsledky rozpoznání celých obrázků.

Již z předchozího grafu jsme mohli odvodit klasifikátor s nejlepšími výsledky. Graf výše toto potvrzuje a klasifikátor se vzory vytvořenými z trénovací sady a příznaky typu Win podává nejlepší výsledky.

5.4 Celková úspěšnost rozpoznání

Spojením výsledků segmentace a klasifikace dostaneme celkové výsledky. Při segmentaci dochází někdy k detekci špatného počtu objektů. V kapitole 3.3 se zmiňuji, že při kroku odstranění pozadí dochází k odstranění objektů z obrázku, což znemožní úspěšné rozpoznání obrázku již v tomto počátečním kroku.



Obrázek 36 - Celkové srovnání výsledků rozpoznání. Do výsledků jsou zahrnuty výsledky segmentace a klasifikace.

Celková úspěšnost je tedy pro nejlepší metodu 56%. Krok segmentace odstranění pozadí, který bohužel průměrně v 6% případů odmaže i nějakou číslici z obrázku a detekce objektů, kterému se nepovede v obrázku správně detekovat objekty, zhoršily celkové výsledky s použitím nejlepší klasifikační metody o 8%. To se dá již považovat za nezanedbatelné zhoršení. Další úpravy by se tedy měly týkat jak klasifikátoru, kde prostor ke zlepšení dozajista je (ať už je to výběr jiných příznaků, které lépe popisují objekty, výběr jiných metod pro porovnávání příznakových vektorů nebo spojení více přístupů ke klasifikaci do jednoho klasifikátoru), tak úpravou kroků segmentace (odstranění pozadí, detekce objektů), kde se na začátku mohlo zdát, že chyba je malá, ale do celkového rozpoznání se projevila 8%. Další kroky segmentace už podávají velice dobré výsledky, a proto si myslím, že snaha o zlepšení by měla být zacílena do výše popsaných částí programu.

6 Závěr

Tato práce navazuje na semestrální projekt, který rozpracoval teoretické možnosti přístupu k problematice detekce objektů v obrázku, experimentoval se dvěma metodami, ukázal na jejich nedostatky a klady a navrhl metodu, která kombinuje oba přístupy za cílem dosažení lepších výsledků.

Dalším krokem v rozpoznání je klasifikace a tedy i tvorbou klasifikátoru se tato práce zabývá. Bylo navrženo a otestováno několik druhů klasifikátorů, z nichž nejlepší klasifikuje číslice s úspěšností 90% a kontrolní kód z obrázku je schopný rozpoznat v 56%.

Z dosažených výsledků můžeme konstatovat, že ochrana použitá na stránkách společnosti Vodafone je nedostatečná. U nejlepší metody dochází k rozpoznání více než každého druhého obrázku. Celkově vzato je těžké říct, kde je hranice, kterou bychom považovali za dostatečnou. I desetiprocentní rozpoznání by potenciálnímu útočníkovi mohlo stačit, jelikož počítače jsou dnes již velice rychlé a pokud by se nepodařilo rozpoznat nějaký obrázek, stačí jednoduše stáhnout obrázek další a tak pořád dokola, než se obrázek povede rozpoznat. Můžeme tedy vyvodit závěr, že čistá ochrana pomocí rozpoznání kódu z obrázku je nedostatečná a je potřeba ji kombinovat s dalšími metodami, jako je limit stažených obrázků apod. Toto však cílem práce není, cílem práce bylo ukázat na slabiny tohoto způsobu zabezpečení, a to bylo prakticky dokázáno vytvořením programu, který je schopen polovinu obrázků rozpoznat.

Posledním nemalým přínosem této práce považuji moje seznámení s knihovnou OpenCV. Jedná se o velice kvalitní knihovnu. Podporuje velké množství obrazových formátů a obsahuje spoustu funkcí pro zpracování obrazu. Knihovna je naimplementována velice kvalitně, což se projevuje na rychlosti zpracování funkcí.

Jako možné rozšíření jsem již zmínil úpravu klasifikátoru využívajícího kostry objektů, který podává špatné výsledky, ale má myslím velký potenciál. Také by určitě bylo zajímavé vyzkoušet kombinace různých klasifikátorů. Dalším slabším místem je krok v segmentaci, při kterém dochází k odstranění pozadí. Při tomto kroku dochází v 6% k znehodnocení obrázku (odstranění číslice spolu s pozadím). Tento krok by chtělo podle mého názoru zlepšit nebo úplně odstranit, kdy by se detekovaly objekty přímo ze zdrojového obrázku. Jako poslední vylepšení se dá v každém případě navrhnout zobecnění rozpoznávače na více druhů kontrolních obrázků a ne jenom na jednu konkrétní třídu společnosti Vodafone. Je otázkou, zda zobecnění na více tříd obrázků lze považovat za vylepšení nebo spíše kompletní předělání, jelikož by dozajista musela být upravena celá logika aplikace.

Literatura

[1] Hozík, Martin.: *Textová 3D CAPTCHA*., 7.6.2008, [online]. [cit. 3.1.2009]. Dostupný na WWW: <http://doublethink.cleverweb.cz/22-textova-3d-captcha>

[2] Čížek, Jakub.: *CAPTCHA: Jak se stát otrokem podivného obrázku*., 18.11.2008, [online]. [cit. 3.1.2009]. Dostupný na WWW: <http://www.zive.cz/Clanky/CAPTCHA-Jak-se-stat-otrokem-podivneho-obrazku/sc-3-a-144482/default.aspx>

[3] Genčúr, Martin.: *Nástroj na zpracování fotografovaného textu*., 2007, [bakalářská práce], FIT VUT Brno

[4] Španěl, Michal, Beran, Vítězslav.: *Obrazové segmentační techniky*, 19.1.2006, [online]. [cit. 3.1.2009]. Dostupný na WWW: <https://www.fit.vutbr.cz/study/courses/POV/private/lectures/pov_04_segmentace_obrazu.pdf>

[5] Čerba, Otakar.: *Barevné modely*. [přednáška], [online]. [cit. 3.1.2009]. Dostupný na WWW: <http://gis.zcu.cz/studium/pok/Materialy/Book/ar03s01.html>

[6] *OCR, optické rozpoznávání písma*. *Quido magazín*, [cit. 3.1.2009]. Dostupný na WWW: <http://www.quido.cz/objevy/ocr.htm>

[7] Vala, Tomáš.: *Rozpoznání SPZ z jednoho snímku*., 2006, [diplomová práce], Univerzita Karlova v Praze, Matematicko-fyzikální fakulta.

[8] Bílek, Jan.: *Rozpoznání textu v obraze*., 2008, [bakalářská práce], FIT VUT Brno

[9] *OpenCV knihovna*, [online]. [cit. 11.5.2009], dostupný na WWW: <http://sourceforge.net/projects/opencvlibrary/>

[10] Kasík, Pavel.: *Jak se počítač naučil číst milion knížek ročně*., 24.11.2007, [online]. [cit. 20.5.2009]. Dostupný na WWW: <http://technet.idnes.cz/jak-se-pocitac-naucil-cist-milion-knizek-rocne-fo8-/tec_technika.asp?c=A071123_182221_tec_technika_pka>

Seznam použitých zkratek a symbolů

RGB	Red, Green, Blue
HSV	Hue, Saturation, Value
OCR	Optical Charakter Recognition
CAPTCHA	Completely Automated Public Turing test to tell Computers and Humans Apart
CCD	Charged Coupled Device
SMS	Short Message Systems

Seznam příloh

A Přílohy

A.1 Obsah CD

Stromová struktura adresářů na přiloženém CD je následující:

- adresář **obr** – obsahuje obrázky s kontrolním kódem
- adresář **src –** zdrojový kód programu (složka s projektem z Microsoft Visual C++ 2005)
- adresář **test** – zde je připraven spustitelný soubor se všemi potřebnými adresáři a soubory
- adresář **vzory**
    - adresář **aut –** vygenerované vzory číslic pro klasifikátor z trénovací sady
    - adresář **rucne –** ručně nakreslené vzory číslic pro klasifikátor
- adresář **stahovac**
    - soubor **stahovac.exe** – program pro stažení obrázků ze stránek společnosti Vodafone
    - adresář **vodafone** – do tohoto adresáře ukládá program stahovac.exe stáhnuté obrázky
- adresář **instal**
    - soubor **dotnetfx.exe** – instalační balík .NET Framework 2.0
    - soubor **OpenCV_1.0.exe** – instalační balík knihovny OpenCV
- soubor **dp.pdf** – tato práce v elektronické podobě
- soubor **plakát.pdf** – vytvořený plakát reprezentující tuto práci

A.2 Návod na použití aplikace

Spustitelný soubor *kody.exe* se všemi potřebnými knihovnami a adresáři se vzory se nachází v adresáři *test*. Pro vyzkoušení aplikace tedy stačí spustit tento soubor s příslušnými parametry (viz. dále).

Aplikace ke své práci potřebuje adresář *vzory* se vzory pro klasifikátor a v módu rozpoznání více obrázků (viz. dále) adresář *obr*. Program lze spustit z příkazového řádku operačního systému Windows (testováno na systému Windows Vista Bussines SP1 a Windows XP Professional SP2). Aplikace ke svému chodu potřebuje nainstalovaný balík .NET Framework Version 2.0, který jsem umístil do adresáře *instal*. V adresáři *instal* se ještě nachází instalační balík knihovny OpenCV, ale ten není potřeba instalovat, jelikož v již zmíněném adresáři *test* jsem přiložil všechny potřebné knihovny.

Režimy aplikace

Program lze pomocí parametrů spustit ve dvou režimech. První režim slouží k rozpoznání více obrázků, kdy jsou také na monitoru zobrazeny očíslované jednotlivé kroky rozpoznání. Druhý slouží k rozpoznání jediného souboru, kdy je na monitoru zobrazen jen výsledek výsledného rozpoznání.

V případě prvního režimu je nutné zadat parametry příkazového řádku následovně:

kody.exe –more start stop –k typ_klasifikátoru

Význam jednotlivých parametrů je uveden v tabulce 10.

Tabulka 10 - Popis parametrů v módu rozpoznání více obrázků

pořadí parametru	**možné hodnoty**	**význam parametru**
1	-more	Klíčové slovo znamenající aktivování prvního módu, tedy že se bude rozpoznávat více obrázků.
2	1 až 300	Jméno souboru bez koncovky, od kterého začne hromadné rozpoznávání.
3	1 až 300	Jméno souboru bez koncovky, kterým skončí hromadné rozpoznávání.
4	-k	Klíčové slovo, které říká, že následuje volba klasifikátoru
5	W, P, K	W – klasifikátor využívající příznaky typu Win a automaticky vygenerované vzory (adresář vzory/aut) P – klasifikátor využívající tzv. primitivní template matching, pracuje s ručně nakreslenými vzory (adresář vzory/rucne) K – klasifikátor využívající příznaky typu Win počítané z koster z ručně nakreslených vzorů (adresář vzory/rucne)

V tomto režimu je pro každý obrázek zobrazeno několik oken s postupem zpracování. Okna jsou očíslována, jejich význam je uveden dále. K přechodu na další obrázek je potřeba stisknout jakoukoliv klávesu při aktivním okně aplikace s obrázkem (ne okně s příkazovým řádkem). Aplikace využívá soubor *./obr/spravne.txt*, ve kterém se nacházejí správné výsledky klasifikace pro soubory *1.png* až *300.png*. Po rozpoznání posledního obrázku aplikace vypíše na standardní výstup statistiku rozpoznání jednotlivých obrázků, jednotlivých číslic a počet špatně segmentovaných obrázků. Statistika obsahuje také informace, na který vzor byla číslice klasifikována při špatné klasifikaci. Lze tak jednoduše identifikovat vzor z jiné třídy, který způsobuje špatné výsledky.

Příklad spuštění aplikace v módu rozpoznání více obrázků pro rozpoznání obrázků *10.png* až *20.png* ze složky s názvem *obr*, klasifikátor bude využívat příznaky typu Win:

kody.exe –more 10 20 –k W

Popis obsahu zobrazených oken

- 1-zdroj – obrázek s kódem, který je aktuálně zpracováván
- 2-obr_bezPozadi – z obrázku bylo odstraněno pozadí
- 3-obr_median – výsledek aplikace filtru, po kterém jsou samostatné objekty složeny z jedné barvy
- 4-bloby – každá spojitá oblast je obarvena jednou barvou
- 5-obr_median_bloby – v červených rámečcích se nacházejí objekty, které je nutné dále zpracovat, v zelených rámečcích se nacházejí nalezené číslice
- 6-obr_median_bez_blobu – zobrazení jen těch objektů, které je nutno dále zpracovat segmentačním algoritmem
- 7-predBestRozdeleni – zobrazení všech jednobarevným shluků pixelů, ze kterých se algoritmus pokusí detekovat číslice
- 8-BestRozdeleni – výsledek detekce číslic, pro každou číslici byl nalezen ohraničující rámeček
- 9-rozpoznano – do spodní části zdrojového obrázku jsou vepsány číslice, které byly v obrázku rozpoznány

V případě druhého režimu je nutné zadat parametry příkazového řádku následovně:
kody.exe cesta_k_souboru –k typ_klasifikátoru
Význam jednotlivých parametrů je uveden v tabulce 11.

Tabulka 11 - Popis parametrů v módu rozpoznání jednoho obrázků

pořadí parametru	**možné hodnoty**	**význam parametru**
1	např. D:\obr.png	Cesta k souboru s obrázkem, který má být rozpoznán. Cesta může být zadána absolutně i relativně.
2	-k	Klíčové slovo, které říká, že následuje volba klasifikátoru
3	W, P, K	W – klasifikátor využívající příznaky typu Win a automaticky vygenerované vzory (adresář vzory/aut) P – klasifikátor využívající tzv. primitivní template matching, pracuje s ručně nakreslenými vzory (adresář vzory/rucne) K – klasifikátor využívající příznaky typu Win počítané z kostry z ručně nakreslených vzorů (adresář vzory/rucne)

Příklad spuštění aplikace v módu rozpoznání jednoho obrázků *./obr/8.png*, klasifikátor bude využívat příznaky typu Win:

kody.exe ./obr/8.png –k W

A.3 Statistika rozpoznání

Jak již bylo uvedeno, v módu rozpoznání více obrázků aplikace na konci svého běhu vypíše podrobnou statistiku o klasifikaci. Následuje ukázka výpisu statistiky pro klasifikátor využívající „primitivní“ template matching:

Rozpoznano spravne/celkem :36/200---18%
Segmentovano spravne/celkem :173/200
Rozpoznano obrazku (bez spatne segmentovanych): 20%
Statistika jen pro dobre segmentovane obrazky
celkova statistika pro kazdou cislici (spravne/celkem = % OK)

cislice 0: 53/71 - 74%
1: 1x (4.png: 1x,)
3: 1x (2.png: 1x,)
6: 9x (3.png: 7x, 4.png: 1x, 5.png: 1x)
7: 1x (6.png: 1x,)
8: 1x (1.png: 1x,)
9: 5x (5.png: 5x,)

cislice 1: 58/63 - 92%
5: 1x (1.png: 1x,)
7: 4x (5.png: 2x, 6.png: 2x,)

cislice 2: 70/101 - 69%
1: 12x (3.png: 1x, 4.png: 11x,)
3:4x (2.png: 1x, 3.png: 1x, 4.png: 2x)
5: 1x (2.png: 1x,)
7: 7x (3.png: 1x, 4.png: 1x,
5.png: 2x, 6.png: 3x)
8: 2x (1.png: 2x,)
9: 5x (5.png: 5x,)

cislice 3: 44/65 - 67%
1: 6x (2.png: 1x, 4.png: 5x,)
2: 9x (4.png: 9x,)
5: 1x (1.png: 1x,)
7: 2x (5.png: 1x, 6.png: 1x,)
8: 2x (1.png: 2x,)
9: 1x (3.png: 1x,)

cislice 4: 39/69 - 56%
1:16x(3.png:2x, 4.png:11x, 5.png: 3x)
2: 1x (5.png: 1x,)
6: 5x (4.png: 5x,)
7: 5x (2.png: 1x, 5.png: 2x, 6.png: 2x)
8: 1x (1.png: 1x,)
9: 2x (5.png: 2x,)

cislice 5: 54/85 - 63%
1: 1x (2.png: 1x,)
3: 12x (1.png: 9x, 3.png: 3x,)
6: 5x (3.png: 4x, 5.png: 1x,)
8: 4x (1.png: 4x,)
9: 9x (5.png: 9x,)

cislice 6: 49/70 - 70%
0: 2x (3.png: 2x,)
1: 3x (4.png: 3x,)
3: 3x (1.png: 1x, 2.png: 2x,)
4: 1x (1.png: 1x,)
5: 4x (1.png: 3x, 2.png: 1x,)
8: 7x (1.png: 7x,)
9: 1x (5.png: 1x,)

cislice 7: 84/111 - 75%
1: 25x (2.png: 8x, 4.png: 17x,)
4: 1x (4.png: 1x,)
6: 1x (4.png: 1x,)

cislice 8: 50/63 - 79%
0: 1x (3.png: 1x,)

1: 5x (4.png: 5x,)
3: 5x (1.png: 4x, 2.png: 1x,)
6: 2x (5.png: 2x,)
cislice 9: 52/73 - 71%
0: 6x (1.png: 1x, 3.png: 5x,)
1: 2x (2.png: 1x, 4.png: 1x,)
3: 4x (1.png: 2x, 2.png: 2x,)
5: 3x (1.png: 3x,)
6: 1x (4.png: 1x,)
8: 5x (1.png: 2x, 4.png: 3x,)
Uspesnost pro cislice: 553/771 --- (71%)

Ze statistiky můžeme např. vyčíst, že číslice sedm byla dobře klasifikována v 84 ze 111 případů. Dále vidíme, že nejvíce byla špatně klasifikována na číslici jedna, a to ve 25 případech. Můžeme i identifikovat vzor (pro každou třídu je jich více). Je to vzor *4.png*, na který byla číslice sedm klasifikována v 17 případech. Tento vzor se tedy nachází na cestě *vzory/rucne/1/4.png*.

Zobrazená statistika je tedy výborným pomocníkem při vyvíjení aplikace, jelikož se pak autor může zaměřit čistě na úpravu „problémových" vzorů ve třídách a dosáhnout tak lepších výsledků.

A.4 Použité nástroje

Při testování a implementaci popsaných metod jsem použil tyto nástroje a knihovny:

- Microsoft Visual Studio 2005
- OpenCV [9] – Jedná se o volně dostupnou grafickou knihovnu původně vyvíjenou společností Intel. Knihovna obsahuje mnoho funkcí ohledně zpracování obrazu. Podporuje velké množství grafických formátů. Použitím této knihovny jsem se mohl soustředit čistě na práci na logice aplikace a nemusel jsem trávit čas vylaďováním algoritmů pro zpracování obrazu.

Samotná práce je pak naprogramována v jazyce C.

A.5 Podrobné výpisy programu

V této příloze jsou vloženy podrobné výpisy programu pro testované klasifikátory. Ve všech případech se jednalo o hromadné rozpoznání obrázků *101.png* až *300.png* ze složky *./obr* (viz. příloha A.1). Výpis byl naformátován do dvou sloupců, aby nezabíral zbytečně mnoho místa.

Klasifikátor využívající jednoduchého přístupu (tzv. primitivní template matching)

Rozpoznano spravne/celkem :36/200---18%
Segmentovano spravne/celkem :173/200
Rozpoznano obrazku (bez spatne segmentovanych): 20%
Statistika jen pro dobre segmentovane obrazky
celkova statistika pro kazdou cislici (spravne/celkem = % OK)
cislice 0: 53/71 - 74%
1: 1x (4.png: 1x,)

3: 1x (2.png: 1x,)
6: 9x (3.png: 7x, 4.png: 1x, 5.png: 1x,)
7: 1x (6.png: 1x,)
8: 1x (1.png: 1x,)
9: 5x (5.png: 5x,)
cislice 1: 58/63 - 92%
5: 1x (1.png: 1x,)
7: 4x (5.png: 2x, 6.png: 2x,)
cislice 2: 70/101 - 69%
1: 12x (3.png: 1x, 4.png: 11x,)
3: 4x (2.png: 1x, 3.png: 1x, 4.png: 2x,)
5: 1x (2.png: 1x,)
7: 7x (3.png: 1x, 4.png: 1x, 5.png: 2x,
6.png: 3x,)
8: 2x (1.png: 2x,)
9: 5x (5.png: 5x,)
cislice 3: 44/65 - 67%
1: 6x (2.png: 1x, 4.png: 5x,)
2: 9x (4.png: 9x,)
5: 1x (1.png: 1x,)
7: 2x (5.png: 1x, 6.png: 1x,)
8: 2x (1.png: 2x,)
9: 1x (3.png: 1x,)
cislice 4: 39/69 - 56%
1: 16x (3.png: 2x, 4.png: 11x, 5.png: 3x,)
2: 1x (5.png: 1x,)
6: 5x (4.png: 5x,)
7: 5x (2.png: 1x, 5.png: 2x, 6.png: 2x,)
8: 1x (1.png: 1x,)
9: 2x (5.png: 2x,)
cislice 5: 54/85 - 63%
1: 1x (2.png: 1x,)
3: 12x (1.png: 9x, 3.png: 3x,)
6: 5x (3.png: 4x, 5.png: 1x,)
8: 4x (1.png: 4x,)
9: 9x (5.png: 9x,)
cislice 6: 49/70 - 70%
0: 2x (3.png: 2x,)
1: 3x (4.png: 3x,)
3: 3x (1.png: 1x, 2.png: 2x,)
4: 1x (1.png: 1x,)
5: 4x (1.png: 3x, 2.png: 1x,)
8: 7x (1.png: 7x,)
9: 1x (5.png: 1x,)
cislice 7: 84/111 - 75%
1: 25x (2.png: 8x, 4.png: 17x,)
4: 1x (4.png: 1x,)
6: 1x (4.png: 1x,)
cislice 8: 50/63 - 79%
0: 1x (3.png: 1x,)
1: 5x (4.png: 5x,)
3: 5x (1.png: 4x, 2.png: 1x,)
6: 2x (5.png: 2x,)
cislice 9: 52/73 - 71%
0: 6x (1.png: 1x, 3.png: 5x,)
1: 2x (2.png: 1x, 4.png: 1x,)
3: 4x (1.png: 2x, 2.png: 2x,)
5: 3x (1.png: 3x,)
6: 1x (4.png: 1x,)
8: 5x (1.png: 2x, 4.png: 3x,)
Uspesnost pro cislice: 553/771 --- (71%)

Klasifikátor využívající příznaky typu Win

Rozpoznano spravne/celkem :112/200---56%
Segmentovano spravne/celkem :173/200
Rozpoznano obrazku (bez spatne segmentovanych): 64%
Statistika jen pro dobre segmentovane obrazky
celkova statistika pro kazdou cislici (spravne/celkem = % OK)
cislice 0: 68/71 - 95%
1: 1x (42.png: 1x,)
3: 1x (30.png: 1x,)
6: 1x (1.png: 1x,)

cislice 1: 59/63 - 93%
2: 1x (22.png: 1x,)
4: 1x (37.png: 1x,)
7: 1x (2.png: 1x,)
8: 1x (3.png: 1x,)
cislice 2: 92/101 - 91%
1: 2x (16.png: 1x, 42.png: 1x,)
3: 2x (5.png: 1x, 22.png: 1x,)
4: 1x (37.png: 1x,)
7: 3x (19.png: 1x, 26.png: 1x, 29.png: 1x,)
8: 1x (29.png: 1x,)
cislice 3: 59/65 - 90%
2: 2x (15.png: 1x, 33.png: 1x,)
5: 3x (7.png: 1x, 15.png: 2x,)
8: 1x (29.png: 1x,)
cislice 4: 57/69 - 82%
0: 1x (39.png: 1x,)
1: 2x (7.png: 1x, 42.png: 1x,)
8: 2x (17.png: 1x, 29.png: 1x,)
9: 7x (16.png: 1x, 22.png: 2x, 40.png: 4x,)
cislice 5: 82/85 - 96%
0: 1x (10.png: 1x,)
6: 1x (10.png: 1x,)
Uspesnost pro cislice: 698/771 --- (90%)

9: 1x (16.png: 1x,)
cislice 6: 60/70 - 85%
0: 2x (32.png: 1x, 41.png: 1x,)
5: 4x (5.png: 1x, 29.png: 1x, 30.png: 1x, 31.png: 1x,)
8: 3x (12.png: 1x, 13.png: 1x, 23.png: 1x,)
9: 1x (5.png: 1x,)
cislice 7: 95/111 - 85%
1: 13x (1.png: 1x, 3.png: 1x, 5.png: 1x, 8.png: 1x, 11.png: 2x,
19.png: 1x, 24.png: 1x, 32.png: 1x, 33.png: 1x, 38.png: 1x, 41.png: 1x, 42
.png: 1x,)
2: 2x (31.png: 1x, 37.png: 1x,)
4: 1x (9.png: 1x,)
cislice 8: 61/63 - 96%
6: 2x (26.png: 2x,)
cislice 9: 65/73 - 89%
0: 2x (10.png: 1x, 22.png: 1x,)
1: 1x (11.png: 1x,)
2: 2x (18.png: 1x, 37.png: 1x,)
5: 1x (8.png: 1x,)
8: 2x (3.png: 1x, 21.png: 1x,)

Klasifikátor využívající koster objektů (nejbližší pixel)

Rozpoznano spravne/celkem :21/200---10%
Segmentovano spravne/celkem :173/200
Rozpoznano obrazku (bez spatne segmentovanych): 12%
Statistika jen pro dobre segmentovane obrazky
celkova statistika pro kazdou cislici (spravne/celkem = % OK)
cislice 0: 50/71 - 70%
1: 1x (5.png: 1x,)
6: 2x (3.png: 2x,)
8: 14x (1.png: 13x, 4.png: 1x,)
9: 4x (4.png: 1x, 5.png: 3x,)
cislice 1: 53/63 - 84%
2: 6x (4.png: 4x, 5.png: 2x,)
3: 2x (1.png: 1x, 2.png: 1x,)
4: 1x (5.png: 1x,)
8: 1x (2.png: 1x,)
cislice 2: 72/101 - 71%
0: 1x (1.png: 1x,)
1: 7x (3.png: 4x, 4.png: 2x, 5.png: 1x,)
3: 2x (3.png: 2x,)

4: 3x (1.png: 1x, 3.png: 2x,)
8: 12x (1.png: 11x, 4.png: 1x,)
9: 4x (5.png: 4x,)
cislice 3: 36/65 - 55%
1: 1x (4.png: 1x,)
2: 3x (4.png: 3x,)
4: 2x (3.png: 2x,)
5: 2x (2.png: 2x,)
8: 21x (1.png: 18x, 4.png: 3x,)
cislice 4: 47/69 - 68%
1: 2x (5.png: 2x,)
2: 2x (4.png: 2x,)
5: 1x (1.png: 1x,)
6: 2x (3.png: 1x, 4.png: 1x,)
8: 13x (1.png: 9x, 2.png: 2x, 3.png: 2x,)
9: 2x (5.png: 2x,)
cislice 5: 35/85 - 41%
1: 3x (3.png: 1x, 5.png: 2x,)
3: 6x (1.png: 4x, 3.png: 2x,)
4: 2x (3.png: 1x, 5.png: 1x,)
6: 19x (2.png: 3x, 3.png: 6x, 4.png: 3x, 5.png: 7x,)
8: 18x (1.png: 18x,)
9: 2x (3.png: 1x, 5.png: 1x,)
cislice 6: 38/70 - 54%
0: 2x (3.png: 2x,)
1: 2x (3.png: 1x, 5.png: 1x,)
4: 2x (3.png: 1x, 4.png: 1x,)
Uspesnost pro cislice: 436/771 --- (56%)

Klasifikátor využívající koster objektů (příznaky typu Win)

Rozpoznano spravne/celkem :28/200---14%
Segmentovano spravne/celkem :173/200
Rozpoznano obrazku (bez spatne segmentovanych): 16%
Statistika jen pro dobre segmentovane obrazky
celkova statistika pro kazdou cislici (spravne/celkem = % OK)
cislice 0: 62/71 - 87%
1: 2x (2.png: 2x,)
6: 2x (3.png: 2x,)
7: 2x (3.png: 1x, 6.png: 1x,)
8: 26x (1.png: 23x, 2.png: 1x, 4.png: 2x,)
cislice 7: 35/111 - 31%
0: 5x (1.png: 1x, 3.png: 4x,)
1: 20x (1.png: 7x, 2.png: 4x, 3.png: 1x, 4.png: 2x, 5.png: 6x)
2: 26x (1.png: 1x, 3.png: 1x, 4.png: 15x, 5.png: 9x)
3: 2x (3.png: 2x,)
4: 4x (2.png: 1x, 3.png: 3x,)
8: 11x (1.png: 7x, 2.png: 4x,)
9: 8x (5.png: 8x,)
cislice 8: 54/63 - 85%
0: 1x (3.png: 1x,)
1: 3x (3.png: 2x, 5.png: 1x,)
4: 3x (4.png: 2x, 5.png: 1x,)
6: 2x (2.png: 1x, 4.png: 1x,)
cislice 9: 16/73 - 21%
0: 3x (1.png: 2x, 3.png: 1x,)
1: 1x (3.png: 1x,)
2: 1x (4.png: 1x,)
3: 4x (1.png: 2x, 2.png: 1x, 3.png: 1x,)
4: 14x (1.png: 1x, 2.png: 1x, 3.png: 9x, 4.png: 3x,)
6: 4x (2.png: 2x, 4.png: 2x,)
7: 3x (1.png: 1x, 2.png: 1x, 5.png: 1x,)
8: 27x (1.png: 18x, 2.png: 3x, 3.png: 1x, 4.png: 5x,)
9: 3x (3.png: 1x, 5.png: 2x,)
cislice 1: 45/63 - 71%
3: 7x (1.png: 1x, 3.png: 6x,)
6: 1x (4.png: 1x,)

7: 10x (3.png: 9x, 5.png: 1x,)

cislice 2: 60/101 - 59%

1: 17x (2.png: 1x, 3.png: 11x, 4.png: 5x,)

3: 8x (3.png: 8x,)

6: 2x (4.png: 2x,)

7: 8x (3.png: 6x, 5.png: 1x, 6.png: 1x,)

9: 6x (3.png: 2x, 5.png: 4x,)

cislice 3: 52/65 - 80%

0: 3x (3.png: 3x,)

1: 6x (2.png: 4x, 4.png: 2x,)

2: 1x (4.png: 1x,)

7: 2x (3.png: 1x, 6.png: 1x,)

9: 1x (3.png: 1x,)

cislice 4: 36/69 - 52%

1: 10x (2.png: 6x, 3.png: 3x, 4.png: 1x,)

3: 2x (3.png: 2x,)

6: 7x (4.png: 7x,)

7: 8x (3.png: 8x,)

8: 1x (1.png: 1x,)

9: 5x (3.png: 1x, 5.png: 4x,)

cislice 5: 49/85 - 57%

3: 12x (3.png: 12x,)

4: 1x (1.png: 1x,)

6: 18x (3.png: 7x, 4.png: 11x,)

8: 4x (1.png: 4x,)

9: 1x (3.png: 1x,)

cislice 6: 48/70 - 68%

0: 6x (3.png: 6x,)

Uspesnost pro cislice: 503/771 --- (65%)

1: 4x (2.png: 4x,)

3: 1x (3.png: 1x,)

5: 1x (1.png: 1x,)

7: 4x (3.png: 3x, 6.png: 1x,)

8: 2x (1.png: 2x,)

9: 4x (3.png: 3x, 5.png: 1x,)

cislice 7: 82/111 - 73%

1: 22x (2.png: 16x, 3.png: 4x, 4.png: 2x,)

3: 6x (3.png: 6x,)

9: 1x (3.png: 1x,)

cislice 8: 42/63 - 66%

0: 6x (3.png: 6x,)

1: 1x (3.png: 1x,)

3: 4x (1.png: 1x, 3.png: 3x,)

5: 1x (2.png: 1x,)

6: 4x (4.png: 4x,)

7: 3x (3.png: 3x,)

9: 2x (3.png: 2x,)

cislice 9: 27/73 - 36%

0: 12x (3.png: 12x,)

1: 7x (2.png: 3x, 3.png: 3x, 4.png: 1x,)

2: 1x (5.png: 1x,)

3: 9x (3.png: 3x, 5.png: 6x,)

4: 1x (4.png: 1x,)

5: 2x (2.png: 2x,)

6: 5x (4.png: 5x,)

7: 4x (3.png: 3x, 6.png: 1x,)

8: 5x (1.png: 5x,)