VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MATEMATIKY

INSTITUTE OF MATHEMATICS

OBRAZOVÉ DESKRIPTORY

IMAGE DESCRIPTORS

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. Marek Dula

AUTHOR

VEDOUCÍ PRÁCE — Mgr. Jana Procházková, Ph.D.

SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav matematiky
Student:	**Bc. Marek Dula**
Studijní program:	Matematické inženýrství
Studijní obor:	bez specializace
Vedoucí práce:	**Mgr. Jana Procházková, Ph.D.**
Akademický rok:	2022/23

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Obrazové deskriptory

Stručná charakteristika problematiky úkolu:

V oblasti počítačového vidění jsou deskriptory nezbytnou součástí pro algoritmy rozpoznávání a detekce objektů. Obrazové deskriptory jsou algoritmy a metodiky schopné globálního popisu objektu a vrací vektor, který abstraktně popisuje obsah obrazu. Existují různé druhy založené na barvě, tvaru nebo struktuře rozpoznávaného objektu.
V rámci diplomové práce se student nejdříve seznámí s různými druhy a jejich matematickým popisem a charakteristikami. Následně se bude věnovat vybraným druhům:
1. Fourierovy deskriptory
2. Deskriptory vzniklé pomocí momentové metody
3. Hu momenty, Zernike momenty

Cíle diplomové práce:

1. Nastudovat problematiku týkající se různých druhů deskriptorů.
2. Matematický základ vybraných druhů deskriptorů.
3. Implementace a srovnání na reálných snímcích.

Seznam doporučené literatury:

KUMAR, R. M. a K. SREEKUMAR. A Survey on Image Feature Descriptors. International Journal of Computer Science and Information Technologies. 2014, 5(6), 7668-7673.

GONZALEZ, R. C. a R. E. WOODS. Digital image processing. 3rd ed. Upper Saddle River, N.J.: Prentice Hall, c2008. ISBN 978-0-13-168728-8.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2022/23

V Brně, dne

L. S.

doc. Mgr. Petr Vašík, Ph.D. ředitel ústavu	doc. Ing. Jiří Hlinka, Ph.D. děkan fakulty

Abstrakt
Zpracovávání digitálních fotografií je rozrůstající se odvětví, při kterém se ve velké míře využívají obrazové deskriptory pro analýzu objektů na fotografii. V dnešní době prakticky všechny fotografie prochází jejich analýzou, a to například pro naše pohodlí při následném procházení digitálních fotografií či naši bezpečnost, ale i pro marketingové účely.

Summary
Image processing is a growing industry that makes extensive use of image descriptors to analyze objects in a photo. Nowadays, practically all photos go through Image processing, for example for our convenience when browsing digital photos or for our safety, but also for marketing purposes.

Klíčová slova
Obrazové deskriptory, Fourierovy deskriptory, Hu momenty, Zernike momenty, Klasifikátory, Rocchio klasifikátor, Bayesovského klasifikátor, kNN

Keywords
Image descriptors, Fourier descriptors, Hu moments, Zernike moments, Classifiers, Rocchio classifier, Bayesian classifier, kNN

DULA, M. *Obrazové deskriptory.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2023. 48 s. Vedoucí diplomové práce Mgr. Jana Procházková, Ph.D..

Prohlašuji, že jsem tuto práci napsal samostatně z uvedených zdrojů.

Bc. Marek Dula

Děkuji všem, kteří mně pomohli při psaní této práce. Předně bych chtěl poděkovat mé vedoucí Mgr. Janě Procházkové, Ph.D. za veškerou její pomoc. Dále pak i Bc. Martinu Buriánkovi za pomoc při korektuře.

Bc. Marek Dula

Obsah

1 Úvod **3**

2 Základní pojmy **4**

3 Nalezení hranice objektu ve fotografii **10**
3.1 Nalezení hranice v binární fotografii . . . 10
3.1.1 Úprava binární fotografie na množinu pozadí a objektu . . . 10
3.1.2 Vlastní metoda nalezení hranice objektu . . . 10
3.1.3 Nalezení hranice pomocí eroze . . . 12
3.2 Nalezení hranice v černobílé fotografii . . . 12
3.2.1 Robertsův operátor . . . 13
3.2.2 Operátor Prewittové . . . 13
3.3 Nalezení hranice v barevné fotografii . . . 14

4 Obrazové deskriptory **15**
4.1 Obrazové deskriptory chápající objekt jako podmnožinu. . . . 15
4.1.1 Momentová metoda . . . 15
4.1.2 HU momenty . . . 16
4.1.3 Zúplnění HU momentů . . . 17
4.1.4 Zernike momenty . . . 17
4.2 Obrazové deskriptory založené na hranici . . . 19
4.2.1 Fourierův deskriptor hranice . . . 19
4.3 Otestování invariancí uvedených metod . . . 20

5 Klasifikace výsledků **23**
5.1 Metody založené na vzdálenosti . . . 23
5.1.1 Metoda nejbližšího prvku . . . 24
5.1.2 Metoda nejbližších sousedů . . . 24
5.1.3 Rocchio klasifikátor . . . 25
5.2 Metody založené na rozložení etalonů . . . 25
5.2.1 Metoda podpůrných vektorů . . . 26
5.2.2 Bayesovského klasifikátor . . . 26

6 Programové zpracování **28**
6.1 Nalezení hranice . . . 28
6.1.1 Moje metoda nalezení hranice . . . 28
6.2 Obrazové deskriptory . . . 29
6.2.1 Fourierův deskriptor . . . 29
6.2.2 Zernike momenty . . . 29
6.3 Klasifikátory . . . 31
6.3.1 KNN klasifikátor . . . 31
6.3.2 Rocchio klasifikátor . . . 32
6.3.3 Bayesovského klasifikátor . . . 32

7 Testování obrazových deskriptorů **34**
7.1 Vhodné nastavení klasifikátorů 34
7.1.1 Vhodná metrika 34
7.1.2 Vhodná volba k pro klasifikátor KNN 34
7.2 Optimální nastavení obrazových deskriptorů 35
7.2.1 Optimální množství zachovaných frekvencí Fourierova deskriptoru . 35
7.2.2 Optimální množství Zernike momentů 36
7.3 Porovnání výsledků jednotlivých metod při optimálním nastavení 37

8 Ukázka na reálných snímcích **38**
8.1 Ukázka fotografií 38
8.2 Úprava fotografií 38
8.2.1 Převod na černobílou digitální fotografii 38
8.2.2 Nalezení objektu ve fotografii 39
8.3 Použití obrazových deskriptorů na fotografie 41
8.3.1 Hu momenty 41
8.3.2 Zernike momenty 41
8.3.3 Fourierovy deskriptory 42
8.4 Klasifikace fotografií knihovnou MPEG-7 42
8.5 Klasifikace fotografií na etalonech z reálných fotografií 43
8.5.1 Porovnání uvažovaných knihoven etalonů 43
8.6 Možné spojení deskriptorů pro klasifikaci 44
8.6.1 Možná další charakteristika 44
8.6.2 Klasifikace při využití více deskriptorů 45

9 Závěr **46**

10 Seznam příloh **48**

1. Úvod

V dnešní době, kdy vzniká velké množství fotografií a videí, je stále více žádanější určit, co se na dané fotografií či videu nachází. Ať již se jedná o práci bezpečnostních složek, či usnadnění hledání v pořízených fotografiích, jako to dělají například společnosti Google LLC či Apple Inc. S obrazovými deskriptory se již každý zajisté v nějaké podobě potkal.

Přestože lidé při pohledu na fotografii dokážou ihned rozpoznat objekt na ní, a případně i více objektů, které se na ní nacházejí, pro počítačový algoritmus se již o tak jednoduchý problém nejedná. Zároveň požadujeme, aby toto rozpoznání proběhlo shodně bez ohledu na úhlu pohledu, velikosti objektu, jeho natočení či poloze v obraze. Přestože se z lidského pohledu jedná o samozřejmost, v případě obrazových deskriptorů to nemusí být vždy tak jednoduché. Při rozpoznávání objektu lze tento úkol rozdělit na tři hlavní podproblémy, které je potřeba vyřešit.

Prvním krokem je podle zvolené metody nelezení objektu v obrazu, nebo jeho hranice. Případně u pokročilejších metod pak nalezení všech objektů v obraze a jejich separace.

Druhý krok při rozpoznávání objektů je popsání charakteristik nalezeného objektu. Budeme popisovat tvar hranice objektu, případně tvar objektu. Tuto informaci je však nutné nějak kvantifikovat, k tomu nám boudou sloužit obrazové deskriptory.

Poslední krok, který je nutné učinit, je klasifikace objektu na základě získaných charakteristik. K tomu je zapotřebí mít informace o charakteristikách pro různé objekty. Ty se nejčastěji získají za pomoci fotografií známých objektů a vyhodnocení charakteristik pro ně. Mezi častý způsob klasifikace patří vyhledání známých objektů, jejichž charakteristiky se nejvíce blíží posuzovanému objektu.

Klasifikaci objektu v obraze lze rozdělit dle těchto kroků na tři samostatné algoritmy, které si blíže přiblížíme v této diplomové práci. Nejprve si ujasníme terminologii, kterou budeme v práci využívat. Blíže se podíváme na nalezení objektu a jeho hranice v obrazu. Dále se zaměříme na popis obrazových deskriptorů. Jmenovitě Fourierův deskriptor, HU momenty a Zernike momenty. Přiblížíme si různé algoritmy pro klasifikaci a pokusíme se shrnout rozdíly mezi nimi. Podíváme se na možná nastavení těchto metod a praktické použití na zvolených fotografiích.

2. Základní pojmy

V této kapitole definujeme pojmy, se kterými budeme dále pracovat. Část definic pojmů pochází z mé bakalářské práce [4], na kterou tato práce navazuje. Dále jsme při definování pojmů vycházeli z knihy [2] a znalostí získaných během studia.

Definice 2.1 (Digitální fotografie)**.** *Digitální fotografie je omezená diskrétní dvourozměrná funkce, definována na omezené množině. Digitální fotografii budeme nazývat případně i jako obrazová funkce, abychom zdůraznili že se jedná o funkci. Pro reprezentaci této funkce nejčastěji používáme matici.*

Tato funkce udává množství fotonů, které dopadly na snímač fotoaparátu za měřený čas. Množství fotonů převádíme na intenzitu, neboli jas. Každá hodnota diskrétní dvourozměrné funkce img(x,y) následně odpovídá intenzitě získané převedením počtu dopadených fotonů na příslušnou část snímače.

Definice 2.2 (Definiční obor)**.** *Oblast, pro kterou je funkce definována, se nazývá definiční obor. Definiční obor obrazové funkce budeme označovat jako rozměry fotografie.*

Definice 2.3 (Obor hodnot funkce)**.** *Množina všech hodnot, kterých může funkce nabývat, se nazývá obor hodnot funkce.*

Definice 2.4 (Barevná digitální fotografie)**.** *Digitální fotografie, která je zobrazením do dvou, či vícerozměrného diskrétního prostoru, se označuje jako barevná digitální fotografie. Můžeme ji zapsat jako zobrazení*

$$img : R \to Z^k, k \in \boldsymbol{N}^+, k > 1$$
$$R = \{0, 1, ..., m-1\} \times \{0, 1, ..., n-1\}, m, n \in \boldsymbol{N} \tag{2.1}$$

Budeme uvažovat trojrozměrný prostor RGB, kde dimenze tohoto prostoru budeme chápat jako intenzitu jednotlivých barev. Dimenze v prostoru RGB odpovídají intenzitě barev modrá, zelená a červená. Jedná se o nejrozšířenější a nejpoužívanější zápis barevné digitální fotografie. Jiný, nežli trojrozměrný prostor je například barevný prostor CMYK, který popisuje intenzitu čtyř barev, a to azurové (Cyan), purpurové (Magenta), žluté (Yellow) a černé (blacK). Příklad barevné digitální fotografie můžeme vidět na obrazu 2.1, část a).

Definice 2.5 (Černobílá digitální fotografie)**.** *Digitální fotografie, kde oborem hodnot obrazové funkce $img(x, y)$ je konečná podmnožina $\boldsymbol{N}^+$, se nazývá černobílá fotografie. Můžeme tedy tuto funkci chápat jako zobrazení*

$$img : R \to W$$
$$R = \{0, 1, ..., m-1\} \times \{0, 1, ..., n-1\}, m, n \in \boldsymbol{N}$$
$$W = \{0, 1, ..., w-1\}, w \in \boldsymbol{N} \tag{2.2}$$

Podmínku, že se jedná o konečnou podmnožinu $\mathbf{N}^+$, lze zjemnit na podmínku, že se jedná o podmnožinu reálných čísel. Obor hodnot img je W a většinou se množství jeho prvků volí jako n^2, kde n udává počet bitů, na kolik budeme chtít tuto hodnotu zapsat. Čím větší je obor hodnot digitální fotografie, tím více detailů je fotografie schopna zachytit. Pro porovnávaní a zobrazování fotografií s různým oborem hodnot můžeme tyto hodnoty převést na interval <0,1> podělením původních hodnot funkce velikostí oboru hodnot funkce. Černobílou digitální fotografii můžeme vidět na obrazu 2.1, část b).

Barevnou digitální fotografii je možné převést na černobílou digitální fotografii. Nejjednodušším způsobem může být zachování hodnot pouze jedné dimenze (jedné barvy), avšak může tak docházet ke ztrátě informací. Jako lepší způsob lze uvažovat vážený průměr jednotlivých složek. V případě trojrozměrného prostoru RGB, kde budeme jeho jednotlivé rozměry značit jako r (červená), g (zelená) a b (modrá), můžeme tento vážený průměr definovat jako

$$i = c_r r + c_g g + c_b b. \tag{2.3}$$

Konstanty c_r, c_g, c_b se nejčastěji volí s větší váhou na zelenou, neboť lidské oko je na tuto barvu nejvíce citlivé, a nejméně je citlivé na barvu modrou. Hodnoty těchto koeficientů se často volí jako $c_r = 0.299, c_g = 0.587, c_b = 0.114$.

Definice 2.6 (Binární digitální fotografie)**.** *Digitální fotografii, jejíž obor hodnot nabývá pouze 2 hodnot, nazýváme binární digitální fotografie. Jedná se o zobrazení*

$$img : R \to 0, 1$$
$$R = \{0, 1, ..., m-1\} \times \{0, 1, ..., n-1\}, m, n \in \boldsymbol{N}. \tag{2.4}$$

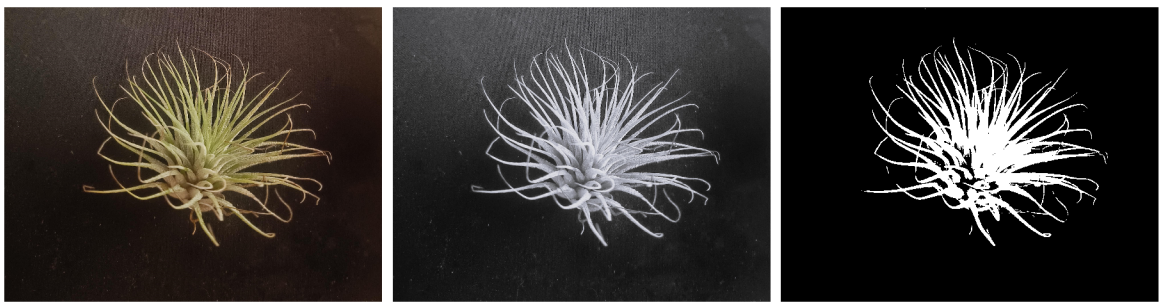
Tato fotografie se nejčastěji využívá, pokud chceme zachovat pouze určitou informaci. Například nachází-li se v daném bodě objekt. Příklad binární digitální fotografie můžeme vidět na obrazu 2.1, část c).

Jakákoliv jiná digitální fotografie může být převedena na binární digitální fotografii. Nejčastěji se převod na binární obraz z černobílé digitální fotografie definuje jako

$$bin(x, y) = 1, \text{pokud } \ img(x, y) \geq h$$

$$bin(x, y) = 0, \text{pokud } \ img(x, y) < h,$$

kde h nám značí mezní hodnotu. Tento postup bývá označován jako prahování, neboli thresholding. V případě barevné digitální fotografie ji lze první převést na černobílou fotografii.



(a) Barevný obraz (b) Černobílý obraz (c) Binární obraz

Obrázek 2.1: Ukázka obrazů

Definice 2.7 (Pixel). *Prvek obrazové funkce se nazývá pixel. Jeho pozice je pevně určena hodnotami x a y udávajícími jeho polohu. Hodnoty x a y budeme nazývat souřadnice pixelu.*

Definice 2.8 (Objekt). *Množinu všech pixelů zachycujících posuzovaný tvar budeme nazývat objekt.*

Definice 2.9 (Pozadí). *Množina všech pixelů, které nenáleží do množiny objektu, budeme označovat jako pozadí. Pozadí můžeme chápat jako doplněk objektu.*

Definice 2.10 (Sousední pixely). *Pro pixel p na souřadnicích (x, y) definujeme osm sousedních pixelů. Čtyři přímé sousedy, které leží na souřadnicích*

$$(x+1, y), (x-1, y), (x, y+1), (x, y-1) \tag{2.5}$$

a dále čtyři diagonální sousedy ležící na souřadnicích

$$(x+1, y+1), (x-1, y-1), (x-1, y+1), (x+1, y-1). \tag{2.6}$$

Pokud neuvádíme při odkazování na sousední pixely jinak, uvažujeme přímé i diagonální sousedy.

Definice 2.11 (Hraniční pixel). *Je to pixel, v jehož sousedních pixelech se nachází pixel náležející do množiny objetu, a zároveň se v jeho okolních pixelech nachází i pixel, který do množiny objektu nenáleží.*

Definice 2.12 (Hranice objektu). *Množina všech hraničních pixelů se nazývá hranice objektu.*

Definice 2.13 (Hranový detektor). *Výpočetní postup určený k nalezení hraničních pixelů se označuje jako hranový detektor.*

Definice 2.14 (Klasifikátor). *Výpočetní postup, který slouží ke klasifikaci, neboli ohodnocení, dat na základě dat s již známou klasifikací, se nazývá klasifikátor.*

Definice 2.15 (Prahování). *Funkce, která na základě definované mezní hranice přiřadí posuzované hodnotě jednu ze dvou hodnot, v našem případě 1 nebo 0, podle toho zdali je daná hodnota nižší nebo vyšší nežli mezní hodnota, se nazývá prahování. Pro mezní hodnotu se používá označení práh. Předpis této funkce:*

$$f(x) = \begin{cases} A \text{ pokud } x < \text{práh} \\ B \text{ jinak} \end{cases}$$

Definice 2.16 (Etalon). *Digitální fotografie, na které se nachází známý objekt, se nazývá etalon.*

Etalony následně slouží k posouzení neznámých objektů za pomoci porovnání s již známými objekty.

Definice 2.17 (Metrika)**.** *Nechť P je libovolná množina. Metrika je zobrazení $\rho : P \times P \rightarrow < 0, \infty)$, které navíc splňuje podmínky:*

$$\rho(a,b) = \rho(b,a)$$
$$\rho(a,c) \leq \rho(a,b) + \rho(b,c)$$
$$\rho(a,b) = 0 \text{ právě tehdy, když } a = b$$

Metrika společně s množinou P tvoří metrický prostor.

Definice 2.18 (Konvoluce)**.** *Konvoluce je operace dvou spojitých jednorozměrných funkcí $f(t)$ a $h(t)$, která se značí $*$, a definujeme ji jako*

$$f(t) * h(t) = \int_{-\infty}^{\infty} f(\tau)h(t-\tau)d\tau. \tag{2.7}$$

Funkci $h(t)$ nazýváme maskou konvoluce. Rozšíříme konvoluci na dvoudimenzionální spojité funkce $f(x,y)$ a $h(x,y)$

$$f(x,y) * h(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(\tau_1,\tau_2)h(x-\tau_1, y-\tau_2)d\tau_1 d\tau_2. \tag{2.8}$$

Jelikož však uvažujeme digitální fotografii jako dvoudimenzionální diskrétní funkci, budeme využívat dvoudimenzionální diskrétní konvoluci, kterou definujeme jako:

$$f(x,y) * h(x,y) = \sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty} f(i,j)h(x-i, y-j) \tag{2.9}$$

kde $f(x,y)$ a $h(x,y)$ jsou diskrétní dvoudimenzionální funkce.

Definice 2.19 (Derivace obrazové funkce)**.** *Velikost změny intenzity pixelů vůči sousedním pixelům nazýváme derivací obrazové funkce. Jeden z možných způsobů výpočtu velikosti derivace $f_D(x,y)$ obrazové funkce $f(x,y)$ je*

$$f_D(x,y) = 4 * f(x,y) - f(x-1,y) - f(x+1,y) - f(x,y-1) - f(x,y+1). \tag{2.10}$$

Tato definice je ekvivalentní s konvolucí obrazové funkce $f(x,y)$ s konvoluční maskou:

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Při tomto výpočtu jsme se omezili na přímé sousedy pixelu, můžeme však uvažovat i diagonální sousedy, a případně i sousedy sousedních pixelů.

Definice 2.20 (Směrová derivace obrazové funkce)**.** *Směrovou derivací obrazové funkce rozumíme změnu intenzity jasu ve zvoleném směru, při zanedbání změny intenzity v ostatních směrech. Nejjednodušší způsob definice směrové derivace ve směru osy x v pixelu (z,y) je:*

$$G_x(z,y) = f(z,y) - f(z+1,y) \tag{2.11}$$

Definice 2.21 (Gradient). *Gradient je vektor směrových derivací obrazové funkce v daném pixelu. V případě dvoudimenzionálního prostoru jej definujeme jako:*

$$G[f(x,y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} \tag{2.12}$$

Definice 2.22 (Podmnožina množiny). *Řekneme, že množina A je podmnožinou množiny B, pokud platí, že všechny prvky množiny B jsou i prvky množiny A. Značíme $B \subseteq A$.*

$$B \subseteq A \Leftrightarrow \forall x \in B \Rightarrow x \in A \tag{2.13}$$

Definice 2.23 (Posun množiny). *Posun množiny A bodem $z = (z_1, z_2)$ značíme jako B_z a je definován jako*

$$B_z = \{c | c = a + z, a \in A\}. \tag{2.14}$$

Definice 2.24 (Reflexe množiny). *Reflexi množiny B značíme jako $\hat{B}$ a je definována jako*

$$\hat{B} = \{c | c = -b, b \in B\}. \tag{2.15}$$

Definice 2.25 (Eroze). *Erozi množiny pixelů objektu A pomocí strukturního elementu B značíme $A \ominus B$ a definujeme ji jako:*

$$A \ominus B = \{z | B_z \subseteq A\} \tag{2.16}$$

Kde $A, B \subset Z^2$.

Tuto definici lze slovně interpretovat jako množinu všech bodů z, pro které platí, že posun strukturního elementu B bodem z je podmnožinou A. Tuto definici lze ekvivalentně zapsat pomocí množiny pozadí A^C jako

$$A \ominus B = \left\{z | B_z \cap A^C = \emptyset\right\}. \tag{2.17}$$

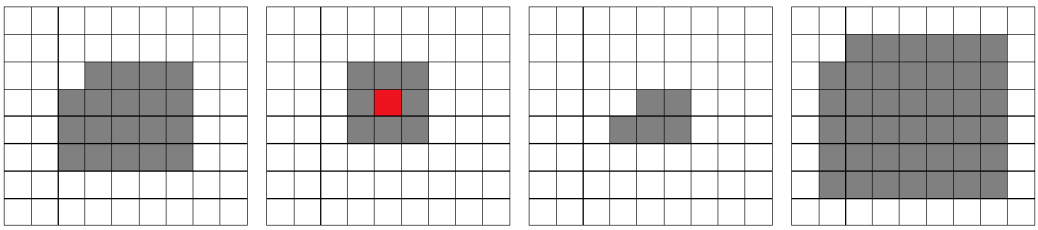
Definice 2.26 (Dilatace). *Dilataci množiny pixelů objektu A pomocí strukturního elementu B značíme $A \oplus B$ a definujeme ji jako:*

$$A \oplus B = \left\{z | \hat{B}_z \cap A \neq \emptyset\right\} \tag{2.18}$$

Kde $A, B \subset Z^2$. Dilataci lze chápat jako opačnou operaci k erozi.

V případě binární fotografie *img* budeme množinou objektu A rozumět množinu pixelů, pro které platí $img(x, y) = 1$.

Příklad dilatace a eroze můžeme vidět na následujícím obrazu 2.2, kde bod $(0, 0)$ strukturního elementu je vyznačen červeně.



(a) Množina A (b) strukturní element B (c) $A \ominus B$ (d) $A \oplus B$

Obrázek 2.2: Ukázka eroze a dilatace množiny.

Definice 2.27 (Otevření množiny)**.** *Otevření (anglicky opening) množiny A pomocí strukturního elementu B značíme $A \circ B$ a definujeme jej jako:*

$$A \circ B = (A \ominus B) \oplus B. \tag{2.19}$$

Otevření množiny A lze chápat jako množinu všech bodů, které jsme schopni pokrýt pomocí strukturního elementu B za podmínky, že celý strukturální element B leží v množině A.

Definice 2.28 (Uzavření množiny)**.** *Uzavření (anglicky closing) množiny A pomocí strukturního elementu B značíme $A¥B$ a definujeme jej jako:*

$$A¥B = (A \oplus B) \ominus B. \tag{2.20}$$

Uzavření množiny A lze chápat jako množinu všech bodů, které nejsme schopni pokrýt pomocí strukturního elementu B za podmínky, že celý strukturální element B leží v doplňku množiny A.

Poslední čtyři uvedené operace (eroze, dilatace, otevření a uzavření) patří do skupiny morfologických operací. Pro libovolnou množinu A a libovolný strukturní element B, $(A, B \subset Z^2)$ platí:

$$A \ominus B \subseteq A \subseteq A \oplus B.$$

$$A \circ B \subseteq A \subseteq A¥B. \tag{2.21}$$

3. Nalezení hranice objektu ve fotografii

Při řešení tohoto problému se především zaměřujeme na úplnost a správnost nalezené hranice. Zohledníme i časovou náročnost jednotlivých metod. Nutnou podmínkou pro správné fungování Fourierova deskriptoru, který bude vysvětlen v části 4.2, je seřazení hraničních pixelů proti směru hodinových ručiček. Proto schopnost seřezaní hraničních pixelů bude jedním z klíčových požadavků.

3.1. Nalezení hranice v binární fotografii

Obecně je binární fotografie funkce nabývající pouze hodnot 0 a 1. Uvedené metody však uvažujeme již pro binární fotografii, kde došlo k rozdělení na množinu pozadí, kde obrazová funkce $f(x, y) = 0$, a množinu objektu, kde budeme uvažovat hodnotu obrazové funkce $f(x, y) = 1$. Ukážeme si tedy postup pro nalezení takovéto množiny. Pro nalezení hranice objektu v takovémto obraze lze využít jednodušší deskriptory. Pro nalezení hranice v binární fotografii lze využít i metody uvedené pro černobílou fotografii.

3.1.1. Úprava binární fotografie na množinu pozadí a objektu

Před použitím uvedených metod nalezení hranice je nejprve nutné rozdělit binární fotografii img na množinu objektu a pozadí. Množinu všech pixelů pro které platí $img(x, y) = 1$ označíme A.

V případě, že množina A obsahuje pixely zachycující objekt a navíc i některé pixely zachycující pozadí, bude našim cílem odstranění pixelů zachycujících pozadí. Jedná se o nežádoucí body v množině A. To provedeme otevřením množiny $A \circ B$, kde B je vhodný strukturální element. Velikost B by měla být dostatečně velká, aby došlo k odstranění nežádoucích pixelů, ale zároveň ne příliš velká, aby nedošlo k odstranění velkého množství pixelů objektu z A.

V případě, že množina A neobsahuje všechny pixely zachycující objekt, a neobsahuje pixely pozadí, je naším cílem doplnit množinu A o chybějící pixely. Za tímto účelem provedeme otevření $A¥B$, kde B je vhodný strukturální prvek. Použití příliš velkého strukturálního prvku B má za následek vyhlazování hranice objetu.

Případ, že množina A neobsahuje všechny pixely zachycující objekt, a zároveň obsahuje i některé pixely zachycující pozadí, je nejsložitější. V tomto případě provádíme oba výše uvedené kroky.

Množinu A následně chápeme jako množinu objektu. Doplněk této množiny jako pozadí. V případě že většina bodů $img(x, y) = 1$ náleží před úpravou binární fotografie pozadí, uvažujeme množinu A jako body pro které platí $img(x, y) = 0$.

3.1.2. Vlastní metoda nalezení hranice objektu

Za účelem nalezení celé hrany, s uspořádáním hraničních pixelů proti směru hodinových ručiček, jsem si navrhl vlastní hranový detektor. Tento hranový detektor je uzpůsobený na binární obraz, ve kterém se nachází pouze jeden celistvý objekt. V případě, že se v

obraze nachází více objektů, bude nalezena hranice pouze jednoho z nich. Tento problém lze vyřešit rozdělením obrazu na více částí. Tento postup se nazývá segmentace. Pokud se však v objektu nachází uzavřená oblast, kde není objekt, nedojde k nalezení hranice této oblasti. I tento problém by se dal vyřešit další úpravou.

Postup detektoru:

1. Rozšíříme definiční obor binární digitální fotografie ve všech směrech o jeden pixel. Hodnota $f(x, y)$ v těchto pixelech se bude rovnat nule.

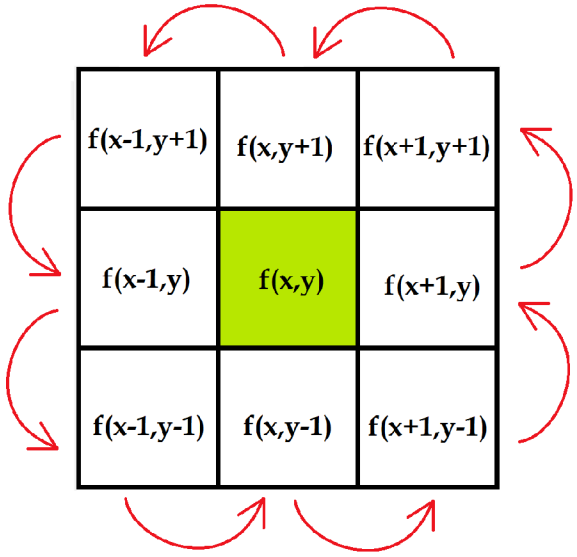
2. Z množiny pixelů splňujících rovnici $f(x, y) = 1$ vybereme ten s nejmenší hodnotou souřadnice x. V případě, že existuje více pixelů na této souřadnici x, volíme z nich pixel s nejmenší hodnotou souřadnice y. Tento pixel zapíšeme jako první hraniční pixel.

3. V sousedních pixelech nalezeného hraničního pixelu hledáme první pixel splňující podmínku $f(x, y) = 1$. Začneme sousedním pixelem na souřadnicích $(x - 1, y + 1)$ a dále pokračujeme po sousedních pixelech nalezeného hraničního pixelu proti směru hodinových ručiček do nalezení prvního takového pixelu. V případě, že nenalezneme žádný, je objekt pouze izolovaný pixel a dále nepokračujeme.

4. Nalezený pixel zapíšeme jako hraniční pixel. V sousedních pixelech nově nalezeného hraničního pixelu hledáme první pixel, splňující podmínku $f(x, y) = 1$. Sousední pixely začneme prohledávat od následujícího pixelu proti směru hodinových ručiček, od předposledně nalezeného hraničního pixelu.

5. Zkontrolujeme, zdali nově nalezený pixel není shodný s prvním hraničním pixelem. Pokud ano, nalezli jsme všechny hranové pixely a končíme s hledáním. Pokud ne, opakujeme předchozí krok, dokud nalezený pixel není shodný s prvním nalezeným hraničním pixelem.

První krok tohoto postupu nám slouží pro případ, že by se objekt nacházel na hranici definičního oboru, neboť by pak pro pixel na této hranici nebyly všechny okolní pixely definovány. Tímto krokem předcházíme této komplikaci. Procházení sousedních pixelů můžeme vidět na 3.1.



Obrázek 3.1: Postup procházení okolí pixelu

3.1.3. Nalezení hranice pomocí eroze

Pro nalezení hranice β_A množiny objektu A v binární fotografii odečteme od množiny objektu A množinu po provedení eroze zvoleným strukturním elementem B.

$$\beta_A = A - (A \ominus B) \tag{3.1}$$

Volbou strukturního elementu B můžeme ovlivnit šířku nalezené hranice. Jako základní volba B se nabízí S_{eroze}, kdy touto volbou dochází k nalezení celé hranice, nalezená hranice bude šířky jednoho pixelu. V případě volby strukturního elementu velikosti 5x5, bude nalezená hranice šířky dvou až tří pixelů. Uvedenou erozi $A \ominus B$ lze spočítat konvolucí s konvoluční maskou K_{eroze}, a následným prahováním získané funkce, kdy za práh zvolíme hodnotu 9. Tato hodnota odpovídá součtu prvků množiny B.

Nalezení hranice erozí pak lze přímo spočítat pomocí konvoluce, kdy použijeme konvoluční masku K_{hrana}, a následné prahování provedeme pro práh 0.

$$S_{eroze} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad K_{hrana} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

3.2. Nalezení hranice v černobílé fotografii

V případě, že chceme naleznout hranici objektu v černobílé fotografii, lze využít některou z níže uvedených metod, případně převést černobílou fotografii na binární a využít některou z metod pro binární fotografii. Jako další z metod pro nalezení hranice v černobílé fotografii, kterou zde ovšem neuvádíme, je například Cannyho hranový detektor. Lze o něm zjistit více například v bakalářské práci [4].

3.2.1. Robertsův operátor

Robertsův operátor definuje novou diskrétní dvourozměrnou funkci M definovanou následovně:

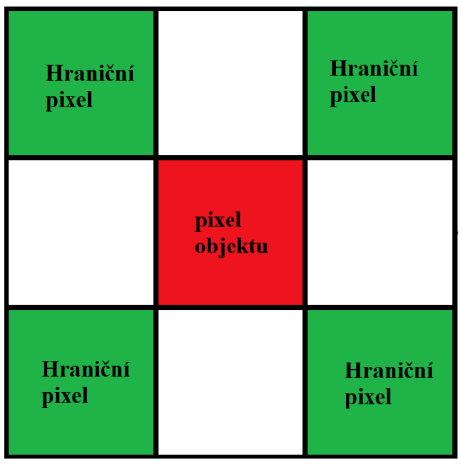
$$M(x,y) = \sqrt{(F(x,y) - F(x+1,y+1))^2 + (F(x+1,y) - F(x,y+1))^2} \tag{3.2}$$

Můžeme ji definovat i jako konvoluci pomocí dvou konvolučních masek s binárním obrazem. Tyto dvě konvoluční masky můžeme chápat jako směrové derivace obrazu a určit pomoci nich i směr hrany, avšak to pro naše účely není potřeba. Jsou definovány jako:

$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \tag{3.3}$$

Pokud je absolutní hodnota $M(x,y)$ nad zvolenou mez, pak prvek obrazové funkce $f(x,y)$ je hraničním pixelem.

Tato metoda je velmi rychlá, avšak může nastat problém s nespojitostí nalezené hranice. Nemusí dojít k nalezení celé hranice. Příklad tohoto problému můžeme pozorovat na obrázku 3.2, kde vidíme nalezení hranových pixelů v okolí jednoho pixelu objektu. Zároveň je problematické následné seřazení hraničních pixelů pro Fourierův deskriptor.



Obrázek 3.2: Nalezení hraničních pixelů okolo pixelu objektu

3.2.2. Operátor Prewittové

Jedná se o obdobu Robertsova operátoru, avšak při výpočtu využíváme všechny sousední pixely. Používáme tedy při výpočtu větší masky konvoluce.

Definujeme dvě směrové derivace G_x a G_y.

$$\begin{aligned} G_x &= (f(x+1,y-1)+f(x+1,y)+f(x+1,y+1))-(f(x-1,y-1)+f(x-1,y)+f(x-1,y+1)) \\ G_y &= (f(x+1,y-1)+f(x,y-1)+f(x-1,y-1))-(f(x+1,y+1)+f(x,y+1)+f(x-1,y+1)) \end{aligned} \tag{3.4}$$

Tyto směrové derivace můžeme taktéž definovat jako konvoluci s konvoluční maskou příslušné směrové derivace:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad G_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \tag{3.5}$$

Následně můžeme definovat funkci M jako velikost gradientu podle metriky. Za hraniční pixely následně uvažujeme body, pro něž hodnota M je nad zvolenou mezí. Využitím větší konvoluční masky dojde k odstranění předchozího problému necelistvosti hranice. Mělo by dojít k nalezení celé hranice, až na speciální případy.

3.3. Nalezení hranice v barevné fotografii

Pro nalezení hranice v barevné fotografii převedeme fotografii na černobílou fotografii a následně postupujeme jako při detekci hran v černobílé fotografii. Ačkoliv lze některé z uvedených metod zobecnit i na případ barevné fotografie, jednalo by se o stejný výsledek jako při převodu fotografie na černobílou, proto zde tyto rozšíření neuvádíme.

4. Obrazové deskriptory

Účelem obrazových deskriptorů je rozpoznání určeného objektu v obraze. Určení nemusí být přesné, můžeme požadovat i vrácení pravděpodobnosti, že na posuzované fotografii je daný objekt. Obrazové deskriptory lze rozdělit podle způsobu, jakým se dívají na objekt. V dalších kapitolách popíšeme i výhody a nevýhody jednotlivých obrazových deskriptorů. Dvě základní třídy obrazových deskriptorů, kterými se budeme dále blíže zabývat, jsou obrazové deskriptory založené na hranici a obrazové deskriptory chápající objekt jako podmnožinu v $\mathbf{R}^2$.

Požadavky na obrazové deskriptory:

1. Invariance vůči posunutí – nezávisí na pozici objektu v obraze
2. Invariance vůči rotaci – nezávisí na natočení objektu v obraze
3. Invariance vůči změně měřítka – nezávisí na velikosti objektu
4. Invariance vůči stranovému otočení

Požadavek na invarianci vůči stranovému otočení objetu může být i nežádoucí. Můžeme někdy požadovat i opak posledního požadavku, například v případě rozpoznávání písma je nežádoucí invariance vůči stranovému otočení, neboť jinak bude docházet k záměně některých písmen.

4.1. Obrazové deskriptory chápající objekt jako podmnožinu.

Tato skupina obrazových deskriptorů chápe objekt jako podmnožinu. Pro rozpoznání objektu je nutné znát, na kterých pixelech v obraze se daný objekt nachází. Příkladem takového deskriptoru jsou například Hu momenty.

4.1.1. Momentová metoda

Momentová metoda chápe objekt jako podmnožinu R^2. V našem případě uvažujeme podmnožinu v množině všech pixelů obrazu. Existuje více deskriptorů, které využívají k posuzování objektu momenty. Všechny ovšem vychází ze základních obecných momentů. Obecně pro dvoudimenzionální funkci $f(x, y)$ definujeme momenty jako:

$$M_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q f(x, y) dx dy \tag{4.1}$$

V obraze, jakožto diskrétním prostoru, budeme definovat tyto momenty pro obrazovou funkci $img(x, y)$ následovně:

$$m_{pq} = \Sigma_x \Sigma_y x^p y^q img(x, y) \tag{4.2}$$

Abychom zaručili požadované invariance, budeme dále provádět následující kroky:

Budeme požadovat, aby se počátek souřadnicové soustavy nacházel ve středu objektu. Souřadnice středu získáme následovně:

$$\overline{x} = \frac{m_{10}}{m_{00}},\ \overline{y} = \frac{m_{01}}{m_{00}} \tag{4.3}$$

Tím jsme získali centrální momenty:

$$\mu_{pq} = \Sigma_x \Sigma_y (x - \overline{x})^p (y - \overline{y})^q img(x, y) \tag{4.4}$$

Všechny momenty je následně nutné normalizovat, abychom zaručili invarianci vůči změně měřítka. Budeme tedy požadovat, aby $m_{00} = 1$. Získáme tak centrální normalizované momenty. Abychom zaručili invarianci vůči rotaci, budeme volit natočení os splňující následující podmínky:

$$m_{10} = 0,\ m_{01} = 0,\ m_{11} = 0 \tag{4.5}$$

Tyto podmínky definují natočení os. K určení osy x a y slouží podmínky:

$$m_{20} >= m_{02} \tag{4.6}$$

K určení směru os slouží podmínky:

$$m_{03} >= 0,\ m_{30} >= 0 \tag{4.7}$$

Tímto získáme i invarianci vůči stranovému převrácení. V případě, že je to nežádoucí, nahrazujeme v podmínce 4.7 požadavek na m_{30} podmínkou, že se jedná o pravotočivou soustavu.

Souřadnicová soustava, která splňuje tyto podmínky, se nazývá hlavní souřadnicová soustava objektu. Momenty k hlavní souřadnicové soustavě nazýváme hlavní momenty. Problém této metody je její náročnější výpočet v porovnání s Hu momenty, které neřeší natočení os. Jedná se však o metodu, která je dobrá pro představení fungování momentových metod. Zároveň z této metody vychází spousta dalších metod včetně Hu momentů, které si představíme nyní.

4.1.2. HU momenty

Jedná se o variaci momentové metody. Vychází z centrálních momentů μ_{pq}, definovaných v momentové metodě rovnicí (4.4). Definujeme si normalizaci těchto momentů jako:

$$Q = 1 + \frac{i + j}{2}$$

$$\eta_{ij} = \frac{\mu_{ij}}{\mu_{00}^{Q}},\ \ i + j \geq 2 \tag{4.8}$$

Pomocí těchto normalizovaných centrálních momentů definujeme Hu momenty následovně:

$$I_1 = \eta_{20} + \eta_{02} \tag{4.9}$$
$$I_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \tag{4.10}$$
$$I_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \tag{4.11}$$
$$I_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \tag{4.12}$$
$$I_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \tag{4.13}$$
$$I_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \tag{4.14}$$
$$I_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30}12+)^2 - 3(\eta_{21} + \eta_{03})^2] - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \tag{4.15}$$

Invarianci vůči posunutí nám zaručí použití centrálních momentů. Invarianci vůči změně měřítka nám zaručují normalizované momenty, kdy všechny momenty převedeme na poměrnou velikost vůči momentu m_{00}. Invarianci vůči rotaci nám zaručují Hu momenty, jak dokázal Hu v [5]. Stranové převrácení obrazu se projeví změnou měřítka momentu I_7. Ostatní momenty zůstávají totožné. Lze tedy u této metody zvolit, zda chceme uvažovat invarianci vůči stranovému převrácení obrazu.

4.1.3. Zúplnění HU momentů

Toto zúplnění představil pan Jan Flusser ve své publikaci [7]. Hu momenty, jak je představil pan Hu a jak jsme si je definovali, netvoří úplnou bázi a zároveň nejsou ani nezávislé. Tuto skutečnost lze jednoduše ověřit. Obecně se jako závislý uvažuje třetí HU moment I_3, který lze vyjádřit jako:

$$I_3 = \frac{I_5^2 + I_7^2}{I_4^3} \tag{4.16}$$

Ověřit správnost lze dosazením rovnic pro Hu momenty, případě jej lze nalézt v uvedené publikaci. Moment I_3 lze tedy z Hu momentů odstranit.

Aby vznikla úplná báze, je nutné nahradit závislý moment I_3 novým momentem I_8.

$$I_8 = \eta_{11}[(\eta_{30} + \eta_{12})^2 - (\eta_{03} + \eta_{21})^2] - (\eta_{20} - \eta_{02})(\eta_{30} + \eta_{12})(\eta_{03} + \eta_{21}) \tag{4.17}$$

Moment I_8, stejně jako moment I_7 , není invariantní vůči zrcadlení a ozrcadlení fotografie se projeví změnou znaménka momentu, avšak velikost zůstává stejná.

4.1.4. Zernike momenty

Zernikovy momenty jsou ortogonální momenty, založené na popisu obrazu pomocí Zernikových polynomů. Tyto polynomy jsou úplnou ortogonální bází na jednotkovém kruhu. Velikost této báze je libovolná konečná. Místo kartézských souřadnic (x, y) se zde uvažují polární souřadnice (ρ, θ), které definujeme následovně

$$\rho = \sqrt{(x^2 + y^2)},\ \theta = arctg(\frac{y}{x}) \tag{4.18}$$

kde ρ udává vzdálenost od počátku souřadnicové soustavy a θ je úhel spojnice vedoucí k danému bodu od osy x.

Abychom zaručili invarianci vůči posunutí této metody, posouváme před převodem do sférických souřadnic počátek souřadnic do středu (těžiště) objektu. Protože je metoda definována pro jednotkový kruh, normalizujeme vzdálenost ρ, abychom objekt převedli na jednotkový kruh. Normalizaci lze ve stručnosti zapsat jako $\rho_i^* = \frac{\rho_i}{max(\rho_i)}$, kde ρ^* je normovaná vzdálenost a dále ji budeme využívat místo ρ.

Zernikovy polynomy na jednotkovém kruhu jsou definovány

$$V_{nm}(x,y) = V_{nm}(\rho,\theta) = R_{nm}(\rho)exp(im\theta) \tag{4.19}$$

Kde R je radiální polynom definovaný jako

$$R_{nm}(\rho) = \sum_{s=0}^{(n-|m|)/2} \frac{(-1)^s(n-s)!}{s!(\frac{n+|m|}{2}-s)!(\frac{n-|m|}{2}-s)!}\rho^{n-2s}. \tag{4.20}$$

Hodnoty n a m udávají řád Zernike polynomu. Obě tyto hodnoty uvažujeme jako celá čísla a hodnotu n uvažujeme navíc jako nezápornou. Při volbě n, m navíc platí podmínky:

$$n - |m| = \text{sudé číslo},$$

$$n \geq |m|.$$

Zernike moment n, m pro funkci $f(x,y)$ je definován jako

$$Z_{nm} = \frac{n+1}{\pi} \iint_{x^2+y^2\leq 1} f(x,y)V_{nm}(\rho,\theta)dxdy. \tag{4.21}$$

Digitální fotografii uvažujeme jako diskrétní funkci, nelze tedy uvažovat integrál. Musíme jej nahradit sumou, a Zernike momenty na digitální fotografii tedy definujeme jako

$$Z_{nm} = \frac{n+1}{\pi}\Sigma_x\Sigma_y f(x,y)V_{nm}(\rho,\theta),\ x^2+y^2 \leq 1. \tag{4.22}$$

Abychom zajistili invarianci vůči změně měřítka, je potřeba znormovat tyto momenty. To provedeme následovně:

$$Z_{nm}{}' = \frac{Z_{nm}}{m_{00}} \tag{4.23}$$

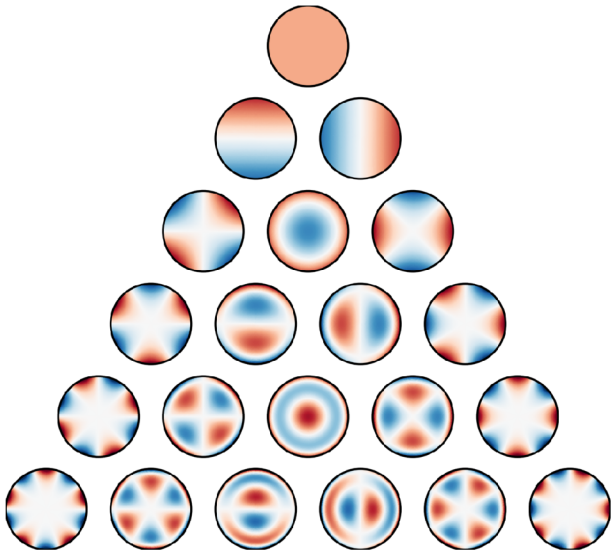
kde m_{00} je moment definovaný jako:

$$m_{00} = \sum_x \sum_y f(x,y) \tag{4.24}$$

Jako poslední nám zbývá vyřešit problém invariance vůči otočení. V případě pootočení o úhel α se nám změní hodnota Zernike momentu na $Z_{nm}{}' = Z_{nm}e^{-im\alpha}$. Budeme tedy uvažovat pouze velikost Zernike momentu, tj. $|Z_{nm}|$. Tím získáme invarianci i vůči otočení.

Na základě zvolené hodnoty n můžeme určit řadu hodnot Zernike momentů pro všechny nezáporné hodnoty m, kdy pro jednoznačnost budeme postupovat od nejnižší hodnoty m po nejvyšší. V případě sudé hodnoty n budeme volit $m = 0, 2, 4, ..., n$ a pro

případ liché hodnoty n budeme volit $m = 1, 3, 5, ..., n$. Zernike moment pro konkrétní volbu m a n budeme značit Z_n^m.



Obrázek 4.1: Ukázka Zernike polynomů, zdroj:[8]

4.2. Obrazové deskriptory založené na hranici

Tato skupina obrazových deskriptorů chápe objekt pouze jako jeho hranici. Pro tuto metodu stačí znát pouze hranici objektu, kdy tuto hranici máme definovanou na diskrétních bodech. Není nutné znát hodnotu obrazové funkce v žádném bodě. Jako příklad si zde uvedeme Fourierův deskriptor hranice.

4.2.1. Fourierův deskriptor hranice

Fourierův deskriptor hranice je založen na rozpoznávání objektů dle uspořádané hranice. Uspořádanou hranici definujeme jako řadu všech nalezených hraničních pixelů, seřazenou tak, jak se nacházejí za sebou na hranici objektu.

Převedeme tuto řadu pixelů na řadu komplexních čísel. Pro každý hraniční pixel budeme uvažovat souřadnice x a y tohoto pixelu. K převodu na řadu komplexních čísel c využijeme následující rovnici:

$$c(u) = x(u) + iy(u), \text{ pro } u = 0, 1, ..., n-1 \tag{4.25}$$

kde hodnota n udává počet hraničních pixelů objektu. Na tuto řadu komplexních čísel $c(u)$ následně použijeme diskrétní Fourierovu transformaci, která je definována jako

$$d(u) = \sum_{t=0}^{n-1} c(t) e^{-i2\pi tu/n}, u = 0, 1, ..., n-1 \tag{4.26}$$

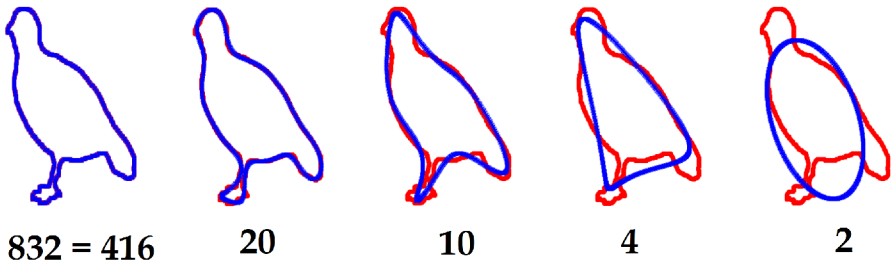
Komplexní koeficienty $d(u)$ nazýváme Fourierovy deskriptory hranice. Z těchto koeficientů můžeme následně získat zpět řadu komplexních čísel $c(t)$ pomocí inverzní Fourierovy transformace, kterou definujeme následovně:

$$c(t) = \frac{1}{n}\sum_{t=0}^{n-1} d(u)e^{i2\pi tu/n}, u = 0, 1, ..., n-1 \tag{4.27}$$

Při zachování nižšího počtu hodnot Fourierových deskriptorů hranice dochází k aproximaci této hranice. Zachováním pouze prvních P deskriptorů, tj. $d(u) = 0$ pro $u > P-1$, získáme následující aproximaci řady komplexních čísel $c(t)$, kterou označíme $\widehat{c}$.

$$\widehat{c}(t) = \frac{1}{P}\sum_{t=0}^{P-1} d(u)e^{i2\pi tu/P}, u = 0, 1, ..., n-1 \tag{4.28}$$

Počet získaných hraničních bodů bude tedy stejný, avšak k jejich získání budeme používat nižší počet Fourierových deskriptorů. Hodnota u udává frekvenci Fourierových deskriptorů $d(u)$. Větší hodnota u znamená vyšší frekvenci, a tím popisuje menší detaily hranice. Můžeme ponechat pouze několik deskriptorů pro nejnižší frekvence a bude se již jednat o dostatečný popis objektu pro jeho rozpoznání. Rekonstrukci obrazu při zachování různého počtu Fourierových deskriptorů lze vidět na obrázku 4.2.



Obrázek 4.2: Rekonstrukce obrazu za pomoci Fourierových deskriptorů

Pro každý objekt, který rozpoznáváme, je potřeba vzorová databáze objektů, takzvané etalony třídy. Pro tyto etalony spočítáme Fourierovy deskriptory a vytvoříme tak hodnoty pro danou třídu. Získat tyto hodnoty nám stačí pouze jednou, následně si je můžeme uložit a pracovat pak již pouze s Fourierovými deskriptory. Abychom zjistili, zdali je posuzovaný objekt některým ze známých objektů, porovnáme hodnoty spočítaných Fourierových deskriptorů posuzovaného objektu s Fourierovými deskriptory etalonů. Vzdálenost Fourierových deskriptorů nám následně prozradí, jak moc posuzovaný objekt náleží do dané třídy.

Jak tato metoda řeší invariance: Posunutí objektu v obraze změní hodnotu $d(0)$. Tato hodnota však udává pouze posun v obraze, proto uvažujeme $d(0) = 0$. Tímto dojde k přesunutí objetu do počátku soustavy souřadnic. Změna měřítka ovlivní stejným způsobem všechny Fourierovy deskriptory. Abychom vyřešili problém změny měřítka, vynásobíme řadu Fourierových deskriptorů takovou hodnotou, aby $d(1) = 1$. Jedná se tedy o úpravu $d(u) = d(u)/d(1)$. Rotaci obrazu v komplexní rovině lze chápat jako násobení e^i.

4.3. Otestování invariancí uvedených metod

Na závěr si ukážeme výsledek pro uvedené metody na množině fotografií, která obsahuje fotografii písmene R a její invariance. Dále pak fotografii písmen B a H. Fotografie této množiny můžeme vidět na obrázku 4.3.

(a) R (b) posun R (c) rotace R (d) zrcadlení R (e) změna měřítka R (f) B (g) H

Obrázek 4.3: Množina fotografií na otestování invariancí

První 4 číslice výsledků jednotlivých obrazových deskriptorů pro tuto množinu můžeme vidět v následujících tabulkách. V případě, že se vypočtené hodnoty pro invarianci R rovnají hodnotám pro R, není tato invariance v tabulce uvedena.

Hu momenty pro testované fotografie

Moment:	I_1	I_2	I_3	I_4	I_5	I_6	I_7
R	0.2831	0.0051	3.281e-5	7.24e-5	-1.206e-9	3.915e-6	3.316e-09
zrcadlení R	0.2831	0.0051	3.281e-5	7.24e-5	-1.206e-9	3.915e-6	-3.316e-09
malé R	0.2894	0.0062	4.949e-5	6.215e-5	-1.082e-09	4.264e-6	3.273e-09
H	0.3366	4.6e-5	5.296e-9	8.016e-7	5.224e-14	-5.456e-9	1.019e-27
B	0.2789	0.0069	8.023e-5	1.036e-6	-8.868e-12	7.490e-8	-3.285e-12

Hu momenty pro rotaci R a posun R jsou stejné jako Hu momenty pro R.

Můžeme vidět, že Hu momenty jsou skutečně invariantní vůči rotaci, posunu i změně měřítka. Zrcadlení se projeví pouze změnou znaménka pro poslední moment I_7. Odlišnost hodnot Hu momentů při změně měřítka R je způsobena tím, že fotografie je diskrétní funkce a při změně měřítka tedy dochází k interpolaci bodů na diskrétní body funkce a tím dochází k odlišnostem. Stejný problém nastane i v případě rotace, kdy diskrétní body funkce nemusí odpovídat diskrétním bodům nové funkce. I zde bude docházet k odlišnostem.

Můžeme si na tomto příkladu ještě navíc ukázat skutečnost, že moment I_3 je závislý, jako jsme uvedli v části 4.1.3 dosazením do rovnice (4.16). Vezmeme nezaokrouhlené hodnoty Hu momentů pro fotografii R a dosadíme:

$$3.28192523018399e-5 = \frac{-1.2062488188581059e-09^2 + (3.3166553466133474e-9)^2}{(7.240042681978764e-5)^3}$$

Můžeme vidět, že tato rovnost skutečně platí.

Zernike momenty Z_5 pro testované fotografie

Moment:	Z_5^1	Z_5^3	Z_5^5
R	0.04273	0.01316	0.02919
malé R	0.04686	0.01008	0.02921
H	0.00293	0.00051	0.00097
B	0.02371	0.00349	0.01156

Zernike momenty pro rotaci R, posun R a zrcadlení R jsou stejné jako Zernike momenty pro R.

Zernike momenty jsou invariantní vůči rotaci, posunu, změně měřítka i zrcadlení. Invarianci vůči zrcadlení nemůžeme zanedbat.

Fourierovy deskriptory pro testované fotografie

frekvence:	0.0625	0.125	-0.125	0.0625
R	0.64378	0.18717	0.10884	0.06018
malé R	0.63663	0.18665	0.11303	0.06366
H	0.91165	0.01234	0.00794	0.06805
B	0.83348	0.02089	0.02864	0.11697

Fourierovy deskriptory pro rotaci R, posun R a zrcadlení R jsou stejné jako Fourierovy deskriptory pro R.

Fourierovy deskriptory jsou invariantní vůči rotaci, posunu, změně měřítka i zrcadlení. Invarianci vůči zrcadlení nemůžeme zanedbat.

5. Klasifikace výsledků

Většina metod, které jsme si zde ukázali, nám jako výstup vypočítá vektor, případně lze výstup chápat jako řadu čísel popisujících daný objekt na obraze. Tyto hodnoty udávají v případě Fourierova deskriptoru amplitudy vybraných frekvencí popisujících hranici, v případě momentových metod nám udávají hodnotu příslušného momentu. Vyhodnocení těchto hodnot však již deskriptor přenechává na dalším zpracování. Na vyhodnocení těchto hodnot a určení popisovaného objektu se zaměříme v této části. Použitá metoda vyhodnocení může ovlivnit přesnost algoritmu, je tedy podstatné uvažovat při našem posuzování přesnosti algoritmů i použitou metodu vyhodnocení. Nejdříve se zaměříme na klasifikátory založené na vzdálenosti posuzovaného objektu od známých etalonů a následně na klasifikátory založené na pozici naměřených hodnot posuzovaného objektu.

5.1. Metody založené na vzdálenosti

Jako první se zaměříme na klasifikátory, které na vyhodnocení využívají vzdálenost posuzovaného objektu od etalonů. Jedná se o intuitivní způsob vyhodnocení, který určitě každého z nás v nějaké podobě napadne. Řadu n čísel popisujících objekt budeme chápat jako bod v n-dimenzionálním prostoru, kde čísla z řady bereme jako souřadnice. Společným problémem pro všechny z těchto metod je zvolení vhodné metriky.

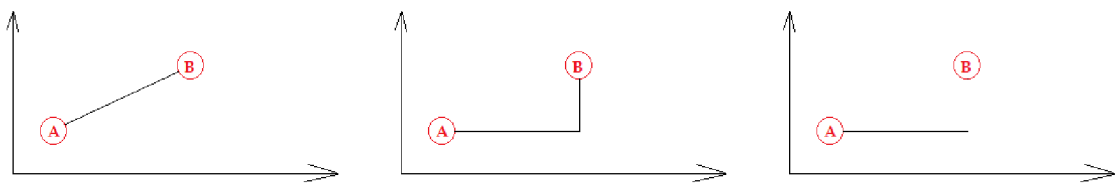
Tato volba může záviset na specifiku porovnávaných objektů, či použitém deskriptoru. U Fourierových deskriptorů například můžeme přikládat vyšší váhu nižším frekvencím, neboť tyto frekvence přenáší více informací o tvaru objektu.

Uvažujme posuzovaný objekt Y, jemuž odpovídají hodnoty $\{y_1, y_2, ...y_n\}$ a etalon X, kterému odpovídají hodnoty $\{x_1, x_2, ...x_n\}$.

Nejznámějšími metrikami jsou:

- Euklidovská metrika $\rho_2 = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2}$
- Manhattanská metrika $\rho_1 = \sum_{i=1}^{n}|y_i - x_i|$
- Kosinová metrika $\rho_{cos} = \dfrac{\sum_{i=1}^{n} y_i x_i}{\sqrt{\sum_{i=1}^{n}(y_i^2)\sum_{i=1}^{n}(x_i^2)}}$
- Maximová metrika $\rho_\infty = max\{|y_1 - x_1|, |y_2 - x_2|, ..., |y_n - x_n|\}$
- ρ_3 metrika $\rho_3 = \sqrt[3]{\sum_{i=1}^{n}(|y_i - x_i|)^3}$

Pro některé z uvedených metrik si způsob, jakým měří vzdálenost, ukážeme na následujícím obrázku 5.1.

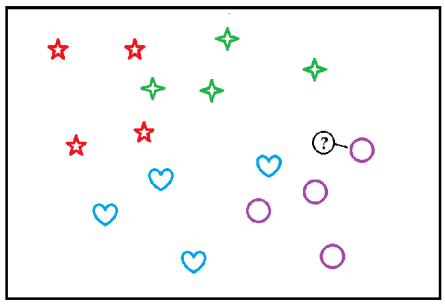


(a) Euklidovská metrika (b) Manhattanská metrika (c) Maximová metrika

Obrázek 5.1: Základní metriky

5.1.1. Metoda nejbližšího prvku

Jako první intuitivní myšlenka vyhodnocení nás napadne nalezení nejbližšího prvku z etalonů. Posuzovaný objekt pak uvažujeme za stejný objekt jako nejbližší etalon. Za pomoci zvolené metriky určíme vzdálenost objektu od všech etalonů a nalezneme ten s nejmenší vzdáleností. Příklad této metody pro dvoudimenzionální prostor můžeme vidět na obrázku 5.2.



Obrázek 5.2: Nejbližší etalon dle ρ_2 v prostoru dimenze 2

5.1.2. Metoda nejbližších sousedů

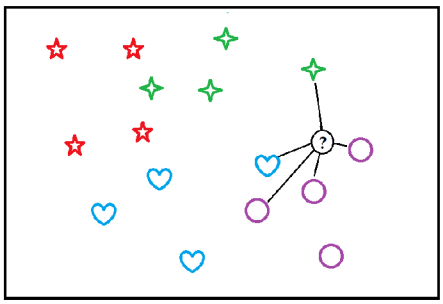
Označuje se zkratkou kNN z anglického názvu *k-nearest neighborhood*. V případě, že již mluvíme o konkrétním počtu nejbližších prvků, nahrazujeme ve zkratce kNN písmeno k tímto počtem. Stejně jako u klasifikátoru 5.1.1 záleží i u této metody na volbě konkrétní metriky, při posuzování vzdálenosti etalonů od posuzovaného prvku.

Postup:

- K posuzovanému objektu nalezneme k nejbližších etalonů, jinak řečeno k nejbližších sousedů.
- Určíme, kolik etalonů z k nejbližších etalonů náleží kterému objetu.
- Pokud má některý z objektů v tomto okolí nejvíc etalonů, je posuzovaný objekt určen jako tento objekt.
- V případě, že neexistuje jeden objekt s nejvíce etalony v tomto okolí, je nutné rozhodnout mezi objekty, které mají v tomto okolí nejvíce etalonů. To provedeme tak, že určíme průměrnou vzdálenost pro etalony stejného objektu, které se nachází v definovaném okolí. Posuzovaný objekt následně klasifikujeme jako ten, jehož průměrná vzdálenost je nejmenší.

Volba k závisí nejvíce na datech a množství etalonů a jedná se o přirozené číslo. Volba příliš malé hodnoty k bude mít za následek velkou citlivost na chyby v datech. Příliš velká hodnota k pak má za následek, že může do okolí patřit příliš mnoho etalonů jiného objektu. Za vhodné hodnoty k uvažujeme hodnoty splňující nerovnici $k < \sqrt{n}$, kde n je počet etalonů. Zároveň není vhodné zvolit k vyšší než je počet etalonů pro jeden objekt. Volba $k = 1$ je speciální případ, kdy se bude jednat o metodu nejbližšího prvku. Příklad kNN pro $k = 5$ můžeme vidět na obrázku 5.3.

Tuto metodu vyhodnocení lze využít i pro případ, že chceme určit pravděpodobnost, že se jedná o daný objekt. A to například určením, jaká část z k nejbližších sousedů přísluší danému objektu.



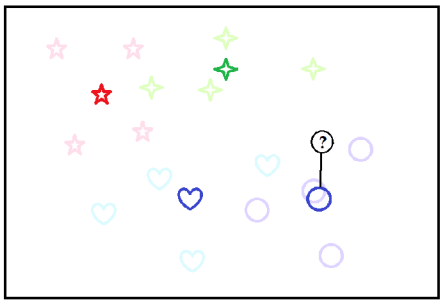
Obrázek 5.3: Nalezení 5NN pro klasifikaci ve 2D

5.1.3. Rocchio klasifikátor

V češtině můžeme narazit i na označení *Centroidová metoda*, v angličtině bývá tento klasifikátor označován i jako *Nearest centroid classifier*. Tento klasifikátor spočívá v nahrazení množiny etalonů pro jeden objekt jediným etalonem, který spočítáme jako centroid jeho etalonů. Uvažujme množinu etalonů X^k, kde $k \in \{1, 2, ..., p\}$ a každý etalon má příslušné hodnoty $\{x_1, x_2, ...x_n\}$. Centroid této množiny potom definujeme jako:

$$X_i^{cent} = \frac{1}{p} \sum_{j=1}^{p} x_i^j \tag{5.1}$$

Tímto krokem dochází ke zjednodušení následné výpočetní náročnosti při určování objektu. Avšak nemusí to být vhodné pro objekty, které mají velkou variabilitu pro některé z hodnot. Následná detekce pak probíhá Metodou nejbližšího prvku 5.1.1. Příklad, jak by tento klasifikátor postupoval, můžeme pozorovat na obrázku 5.4.



Obrázek 5.4: Rocchio klasifikátor. Světle vyznačeny původní etalony.

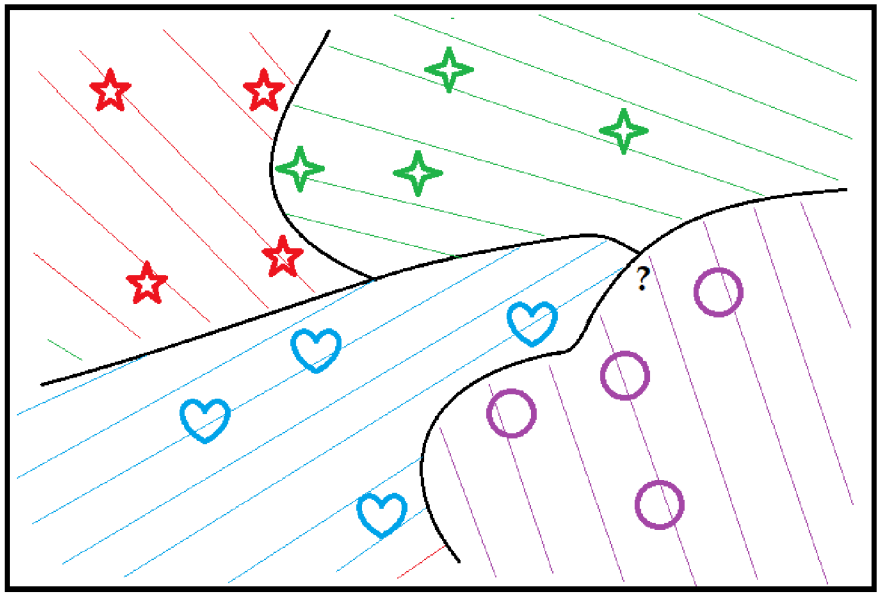
5.2. Metody založené na rozložení etalonů

V této části se podíváme na klasifikátory, které k rozpoznávání objektu využívají rozdělení bodů etalonů v prostoru, a na základě toho rozdělí prostor do podoblastí příslušících jednotlivým objektům.

5.2.1. Metoda podpůrných vektorů

Značí se zkratkou SVM z anglického názvu *Support vector machines*. Základem této metody je nalezení dělící hranice, která nejlépe rozděluje jednotlivé množiny etalonů. V našem případě množiny pro jednotlivé objekty. Ačkoliv je tato metoda primárně určena pro určení příslušnosti mezi dvěma třídami, lze ji rozšířit i pro více tříd. Dva hlavní způsoby tohoto rozšíření jsou přístupy „one to one“ a „one to rest“. V případě „one to one“ přístupu se snažíme naleznout dělící hranici mezi každými dvěma objekty. Jako jednu třídu uvažujeme etalony jednoho objektu a jako druhou množinu etalony druhého objektu. V případě přístupu „one to rest“ hledáme dělící hranici jednoho objetu od všech ostatních objektů. Jako jednu třídu uvažujeme etalony jednoho objetu a jako druhou třídu zbytek etalonů. Vzhledem k časové náročnosti se častěji využívá metoda „one to rest“. Tato metoda hledá primárně lineární hranici. V případě bodů v rovině hledá přímky. Avšak jelikož jsou naše data pravděpodobně lineárně neseparovatelná, budeme provádět transformaci dat. Transformaci dat provádíme nejčastěji do prostoru vyšší dimenze a následně v tomto novém prostoru hledáme lineární hranici. Jelikož není podmínka na to, aby byla transformace lineární, může být nalezená hranice i nelineární v původním prostoru.

Výhodou tohoto přístupu je, že stále hledáme lineární hranici, problém je však v nalezení optimální transformace. Za účelem nalezení této transformace se vynechává při tvorbě modelu část dat, následně se vytvoří model pro různé transformace a za pomoci otestování vynechaných dat se zvolí ta nejvhodnější.



Obrázek 5.5: SVM - ukázka možného rozdělení

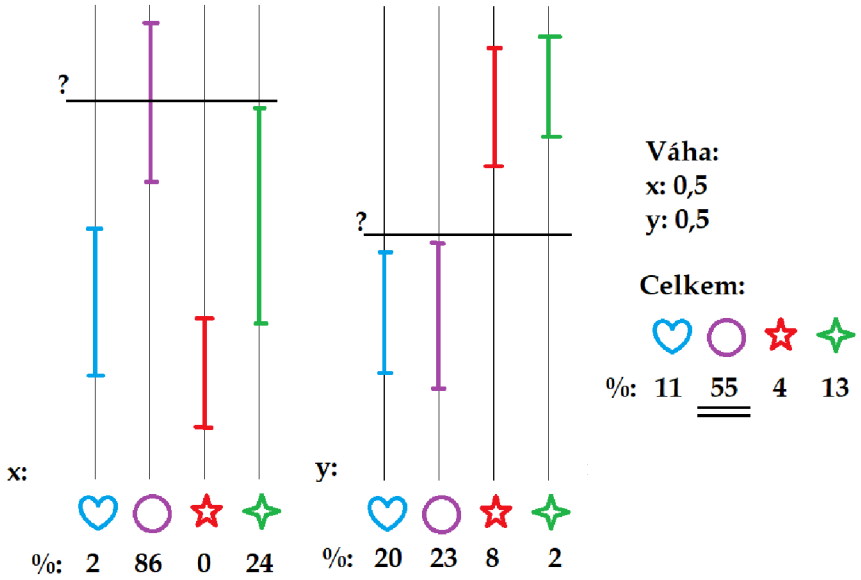
5.2.2. Bayesovského klasifikátor

Bayesovského klasifikátor je založen na myšlence, že rozdělení odpovídajících hodnot vektoru pro stejné objekty má normální rozdělení. V našem případě budeme uvažovat naivní Bayesovského klasifikátor, neboť se jako jediný hodí pro naše potřeby. Pro každý objekt budeme mít n rozdělení, pro každé pořadí v této řadě jeden. Při porovnávání neznámého objektu pak určujeme pravděpodobnosti, že hodnoty neznámého objektu náleží do rozdě-

lení známých objektů. Dále si určíme rozdělení vah mezi těmito n rozděleními, tak aby v součtu tyto váhy daly 1. Následně určíme pravděpodobnost, že posuzovaný objekt náleží do těchto rozdělení, s tím že každá složka přispívá dle stanovené váhy. Posuzovaný objekt pak určíme jako ten objekt, pro který nám vyšla největší pravděpodobnost, že v jeho rozdělení se nachází hodnoty posuzovaného objektu. Obecně nemusíme požadovat normální rozdělení, jedná se pouze o zjednodušení. Pro výpočet parametrů normálního rozdělení $N(\mu, \sigma^2)$ na stejné pozici ve vektoru definujeme

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i, \;\; \sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2, \tag{5.2}$$

kde n udává počet etalonů stejného objektu a x_i je přísluší hodnota i-tého etalonu na stejné pozici ve vektoru. Hodnotu μ nazýváme střední hodnota a σ^2 rozptyl normálního rozdělení.



Obrázek 5.6: Ukázka pro 2D případ, vyznačeny intervaly spolehlivosti

6. Programové zpracování

V této části diplomové práce se zaměříme na programové zpracování vybraných deskriptorů a některých metod pro nalezení hranice a klasifikaci. Budeme k tomu využívat programovací jazyk Python. V tomto zpracování je kladen důraz na jednoduchost a pochopitelnost kódu, spíše nežli na jeho časovou optimalizaci, neboť mnoho těchto algoritmů je možné najít zpracovaných optimálně v rychlejších programovacích jazycích. Některé kódy zde nejsou uvedené celé, avšak pouze jejich podstatné části pro pochopení metody. Celé programové zpracování těchto i dalších algoritmů, včetně komentářů, je uvedeno v příloze diplomové práce.

6.1. Nalezení hranice

Jako první se podíváme na metodu, jak naleznout hranici v posuzovaném obraze. Uvedu zde pouze část kódu z mé metody pro nalezení hranice. Ostatní metody pro nalezení hranice lze naleznout například v bakalářské práci [4].

6.1.1. Moje metoda nalezení hranice

Při programování této metody jsme použili knihovnu *numpy*. Jedná se o programové zpracování metody uvedené v části 3.1.2. Rozšíření obrazové matice a nalezení počátečního hraničního pixelu můžeme vidět v první části kódu na řádcích 2 až 10. Ověření, jestli se nejedná o samostatný pixel, provádíme na řádku 12 až 14. Třetí až pátý bod postupu detektoru můžeme vidět naprogramovaný na řádku 16 až 32. Procházení sousedních pixelů provádíme za pomoci série funkcí `if`.

Vstup: *img* je binární digitální obraz rozdělený na objekt a pozadí. Hodnota „True“ udává, že se na daném pixelu nachází objekt. Vstupní proměnnou uvažujeme jako pole polí.

Výstup: 2 pole udávající souřadnice x a y hraničních pixelů

```
1  def Hranice(img):
2      a= len(img); b= len(img[0])
3    Image=np.full((a+2, b+2), False)
4    Image[1:a+1,1:b+1]=img
5
6    [x,y] = np.where(Image==True)
7    x=x[0]; y=y[0]
8    xlist= np.array(x)
9    ylist= np.array(y)
10   x0=x; y0=y
11
12   [Test,Tests] = np.where(Image[0:x+2,y:y+2]==True)
13     if Test.size==1:
14     return
15
16   Posx = 1; Posy = -1
17     while True:
18       if Image[x+Posx,y+Posy] == True:
19           x=x+Posx
20           y=y+Posy
21         if x==x0 and y == y0:
```

```
            break
    xlist = np.append(xlist,x)
    ylist = np.append(ylist,y)
    Posx = -Posx
    Posy = -Posy
    if Posx == 1 and Posy == -1:
        Posy=0
    elif  Posx == 1 and Posy == 0:
        Posy=1
    ...
  return xlist,ylist
```

6.2. Obrazové deskriptory

V této části programového zpracování se podíváme na obrazové deskriptory. Výstupem těchto metod není určení objektu, ale popis obrazu pomocí vybraných charakteristik. Blíže se zde podíváme na zpracování Fourierova deskriptoru a Zernike momentů. Kód pro Hu momenty lze naleznout v příloze.

6.2.1. Fourierův deskriptor

Tento deskriptor jsme si uvedli v části 4.2.1 Využíváme knihovnu `numpy`. Z té pak používáme funkci `numpy.fft`. Tato funkce počítá jednodimenzionální diskrétní Fourierovu transformaci z vložené řady komplexních čísel.

Vstup: `x` a `y` jsou pole souřadnic hraničních pixelů.

Výstup: `Amp` je pole zachovaných Fourierových deskriptorů. Nejdříve obsahuje hodnotu klasifikátorů pro kladné frekvence od nejnižší, a následně hodnoty pro záporné frekvence od nejnižší frekvence.

```
def Fourier(x,y,Nechat):
  spojeni = x+y*1j
  fourier = np.fft.fft(spojeni)
  zero = 0*fourier
  zero[:Nechat+1] = fourier[:Nechat+1]
  zero[-Nechat:] = fourier[-Nechat:]
  amplitudes = 2 / len(x) * np.abs(zero)
  Amp =  np.append(amplitudes[1:Nechat+1],amplitudes[-Nechat:])
  Amp = Amp/sum(Amp)
  return Amp
```

6.2.2. Zernike momenty

Jedná se o programové zpracování metody uvedené v části 4.1.4, při uvažování binární fotografie. Budeme využívat knihovny `numpy` a `math`. Tuto metodu si rozdělíme na 3 funkce. První se podíváme na hlavní funkci `Zernike`. Tato funkce nejprve převádí objekt do počátku souřadnicové soustavy, (řádky tři až šest) a převod na sférické souřadnice dle rovnice (4.18) (řádky sedm až čtrnáct). Na závěr spočítáme pomocí funkce `ZernikeMoment` jednotlivé Zernike momenty.

Vstup: `img` je binární digitální obraz, rozdělený na objekt a pozadí, kde hodnota „True“ odpovídá množině objektu. Proměnná `n` udává řád Zernike polynomu.

Výstup: pole Zernike momentů. Tyto hodnoty jsou seřazeny od Zernike momentu nejvyššího řádu po nejnižší.

```
def Zernike(img,n):
  [x,y] = np.where(img==True)
  xstred=sum(x) / len(x)
  ystred=sum(y) / len(y)
  x = x-xstred
  y = y-ystred
  uhel = []
  rameno = []
  for i in range(len(x)):
    uhel.append(math.atan2(y[i],x[i]))
    rameno.append(math. sqrt(x[i]*x[i]+y[i]*y[i]))
  Polomer = 2+max(rameno)
  for i in range(len(x)):
    rameno[i] = rameno[i]/Polomer
    momenty = []
  m = n
  while m >= 0:
    momenty.append(ZernikeMoment(m,n,rameno,uhel))
    m -= 2
  return momenty
```

Funkce `ZernikeMoment` slouží pro výpočet Zernike momentu dle rovnice (4.23) vycházející z normalizace (4.22).

Vstup: `q,p` udává řád Zernike polynomu, kde `q` odpovídá hodnotě m a hodnota `p` hodnotě n z definice Zernike polynomu. `rameno` obsahuje pole vzdáleností pixelů objektu po převodu do sférických souřadnic. `uhel` obsahuje pole úhlů pixelů objektu po převodu do sférických souřadnic. Pro výpočet radiálního polynomu využíváme funkci `Zpol`.

Výstup: Tato funkce nám vrací velikost Zernike momentu Z_{pq}.

```
def ZernikeMoment(q,p,rameno,uhel):
  Zr = 0
  Zi = 0
  for i in range(len(rameno)):
    r = rameno[i]
    Zr = Zr+ZPol(r,p,q)*math.cos(q*uhel[i])/len(rameno)
    Zi = Zi+ZPol(r,p,q)*math.sin(q*uhel[i])/len(rameno)
  return math.sqrt(Zr*Zr + Zi*Zi)
```

Funkce `ZPol` slouží pro výpočet radiálního polynomu R definovaného rovnicí (4.20).

Vstup: `r` udává vzdálenost, pro kterou chceme vypočítat hodnotu radiálního polynomu. `n,m` udává řád radiálního polynomu R_{nm}.

Výstup: Hodnota radiálního polynomu $R_{nm}(r)$

```
def ZPol(r,n,m):
  radial = 0;
  u = int((n-abs(m))/2)
  for s in range(u+1):
    c = (-1)**s* math.factorial(n-s)/(math.factorial(s)*math.factorial(int
   ((n+abs(m))/2-s))*math.factorial(int((n-abs(m))/2-s)))
    radial = radial + c*r**(n-2*s)
  return radial
```

6.3. Klasifikátory

V poslední části se podíváme na kódy klasifikátorů, které na základě charakteristik získaných obrazovými deskriptory určují, o jaký objekt se jedná.

6.3.1. KNN klasifikátor

Zde uvedeme kód pro výpočet metody nejbližších sousedů definované v části 5.1.2. Budeme využívat knihovny `numpy` a `collections`. V uvedeném kódu uvažujeme euklidovskou metriku, v případě, že bychom chtěli uvažovat jinou, zaměníme v kódu výpočet proměnné `Vzdalenost` za požadovanou metriku. Programové zpracování všech metrik uvedených v části 5.1 lze naleznout v příloze diplomové práce. Samostatný kód si rozdělíme na dvě části. První se podíváme na nalezení k nejbližších prvků a následně na klasifikaci na základě nalezených k prvků.

Vstup: `k` udává počet požadovaných nejbližších sousedů při klasifikaci. `posuzovany` je pole hodnot posuzovaného objektu. `Etalon` je pole polí obsahující hodnoty etalonů. `nazvyEtal` je pole klasifikací etalonů.

Výstup: Klasifikace posuzovaného objektu.

```
def KNN(K,posuzovany,Etalon,nazvyEtal):
  Vzdalenosti = []
    Vzdalenost = sum(np.power(np.array(Test)-np.array(posuzovany),2))
    Vzdalenosti.append(Vzdalenost)
  Vzdalenosti = np.array(Vzdalenosti)
  Knej = []
  Vzdal = []
  for i in range(K):
    Nejblizsi = np.where(Vzdalenosti == Vzdalenosti.min())
    S = np.where(Vzdalenosti == Vzdalenosti.min())[0][0]
    Knej.append(nazvyEtal[S])
    Vzdal.append(Vzdalenosti[S])
    Vzdalenosti[S] = 1000
```

V první části kódu došlo k nalezení k nejbližších hodnot od posuzovaného objektu. Volba hodnoty `1000` v posledním řádku je čistě náhodná, tato hodnota slouží pouze k odsunutí již nalezeného etalonu, aby nebyl nalezen vícekrát jako nejbližší. V následující části určíme, kolikrát se daný objekt vyskytl v nejbližších sousedech. Využíváme k tomu funkci `Counter` z knihovny `collections`.

```
  a = dict(Counter(Knej))
  Nejpocetnejsi =[key for key, value in a.items() if value == max(a.values
    ())]
  if len(Nejpocetnejsi) == 1:
    return Nejpocetnejsi[0]
  else:
    CelkVzdal = [0] * (len(Nejpocetnejsi))
    for i in range(len(Nejpocetnejsi)):
      for p in range(K):
        if  Knej[p] == Nejpocetnejsi[i]:
          CelkVzdal[i] = CelkVzdal[i] + Vzdal[p]
    return Nejpocetnejsi[np.where(CelkVzdal ==np.array(CelkVzdal).min())
    [0][0]]
```

6.3.2. Rocchio klasifikátor

Uvádíme programové zpracování pro výpočet nové množiny etalonů uvedené v části 5.1.3. Uvedený kód nám vrací pouze novou sadu etalonů získanou použitím rovnice (5.1) pro etalony stejného objektu, následnou klasifikaci lze provést pomocí klasifikátoru 1NN. Výhoda tohoto přístupu je že v případě klasifikace více objektů, pro stejné etalony není nutné počítat vždy novou sadu etalonů. Využíváme knihovny `numpy`, knihovnu `collections`, ze které je funkce `Counter` a knihovnu `scipy.stats`.

Vstup: `Etalon` vkládáme pole polí hodnot etalonů. `collections` je pole klasifikací etalonů.

Výstup: `newetal` pole polí hodnot nových etalonů. `newetalnazvy` je pole klasifikací nových etalonů.

```
def Rocchio(Etalon,nazvyEtal):
  tridy = dict(Counter(nazvyEtal))
  newetal = []
  newetalnazvy = []
  for key in tridy.keys():
    pomocna = [0] * len(Etalon[0])
    for i in range(len(nazvyEtal)):
      if nazvyEtal[i] == key:
        pomocna = np.array(pomocna) + np.array(Etalon[i])
    newetal.append(np.array(pomocna)/(tridy.get(key)))
    newetalnazvy.append(key)
  return newetal, newetalnazvy
```

6.3.3. Bayesovského klasifikátor

Programové zpracování bayesovského klasifikátoru uvedeného v části 5.2.2 zpracováváme jako dvě samostatné funkce. První funkcí *Baymodel* vypočítáme model pro klasifikaci na základě etalonů. A druhá funkce *Bayclassify* slouží pro následnou klasifikaci objektu tímto modelem. Výhodou tohoto rozdělení je, že po vypočítání modelu pro danou množinu etalonů již není potřeba tento model znovu počítat při klasifikaci dalšího objektu. Zároveň pro klasifikaci stačí pouze model, není potřeba uchovávat etalony. Využíváme knihovny `numpy` a `collections`.

Jako první si představíme funkci `Baymodel` pro vytvoření modelu. Jedná se o výpočet parametrů normálního rozdělení pro každou z hodnot objektů na základě všech etalonů.

Vstup: `Etalon` je pole polí hodnot etalonů. `nazvyEtal` je pole klasifikací etalonů.

Výstup: `etalval` - pole polí polí, obsahující parametry normálního rozdělení pro každou z hodnot objektu pro každý objekt. `etalname` - pole názvů objektů. Dvojici výstupních parametrů můžeme chápat jako model pro klasifikaci.

```
def Baymodel(Etalon,nazvyEtal):
  tridy = dict(Counter(nazvyEtal))
  dimenze = len(Etalon[0])
  etalname = []
  etalval = []
  for key in tridy.keys():
    data =[]
    mean = np.zeros(dimenze)
    variance = np.zeros(dimenze)
    for i in range(len(nazvyEtal)):
```

```
      if nazvyEtal[i] == key:
        data.append(Etalon[i])
    data = np.array(data)
    for i in range(dimenze):
      mean[i] = sum(data[:,i])/tridy.get(key)
      variance[i] =np.sqrt(sum((data[:,i]-mean[i])**2)/(tridy.get(key)-1))
    etalname.append(key)
    etalval.append([mean, variance])
  etalval = np.nan_to_num(etalval)
  return etalval,etalname
```

Druhá funkce `Baymodel` slouží ke klasifikaci neznámého objektu na základě modelu získaného funkcí `Baymodel`.

Vstup: `etalval`, `etalname` - dle výstupu z první funkce. `unknow` - pole hodnot neznámého objektu. `vahy` - pole vah pro jednotlivou hodnotu.

Výstup: Klasifikace neznámého objektu

```
def Bayclassify(etalval,etalname,unknow,vahy):
  vahy = np.full(dimenze,1/dimenze)
  prob = np.zeros(len(etalname))
  for m in range(len(etalname)):
    for i in range(dimenze):
      if unknow[i] < etalval[m][0][i]:
        prob[m] = prob[m] + vahy[i]*norm(loc = etalval[m][0][i] ,...
         scale = etalval[m][1][i]).cdf(unknow[i])
      else:
        prob[m] = prob[m] + vahy[i]*(1-norm(loc = etalval[m][0][i] ,...
         scale = etalval[m][1][i]).cdf(unknow[i]))
  prob = np.nan_to_num(prob)
  return etalname[(np.where(prob == prob.max())[0][0])]
```

7. Testování obrazových deskriptorů

V této kapitole se zaměříme na posouzení výsledků obrazových deskriptorů a klasifikátorů. Při testování budeme využívat obrazovou databázi *MPEG-7 CE Shape-1*, dostupnou na adrese *https://www.imageprocessingplace.com/*. Tato knihovna obsahuje 1400 binárních digitálních fotografií. Jedná se konkrétně o 70 objektů, kdy každý objekt se nachází na 20 fotografiích. Za účelem testování správnosti vyloučíme část z fotografií této knihovny při vytváření množiny etalonů. Takto vyloučíme 10% fotografií pro každý objekt této databáze. Vyloučené fotografie klasifikujeme za pomocí určeného deskriptoru a klasifikátoru. Tuto klasifikaci následně porovnáme se známou klasifikací pro danou fotografii. Přesností dané metody budeme rozumět počet správně klasifikovaných objektů.

7.1. Vhodné nastavení klasifikátorů

V první části se zaměříme na vhodnou volbu metriky pro klasifikátory založené na vzdálenosti a volbu optimálního k pro metodu nejbližších sousedů. Tyto volby mohou záviset i na zvoleném deskriptoru. Při našem testování budeme uvažovat Hu momenty a Fourierův deskriptor hranice, abychom mohli pozorovat případnou závislost na volbě deskriptoru.

7.1.1. Vhodná metrika

Jako první se zaměříme na vhodnou volbu metriky pro výpočet vzdálenosti objektů. Při našem testování se zaměříme na metriky uvedené v části 5.1. Jako klasifikátory při našem testování využijeme metodu nejbližšího prvku, Rocchio klasifikátor a KNN pro 3 a 5 nejbližších sousedů. Za nejvhodnější metriku budeme následně uvažovat tu s nejvyšším počtem správně klasifikovaných objektů.

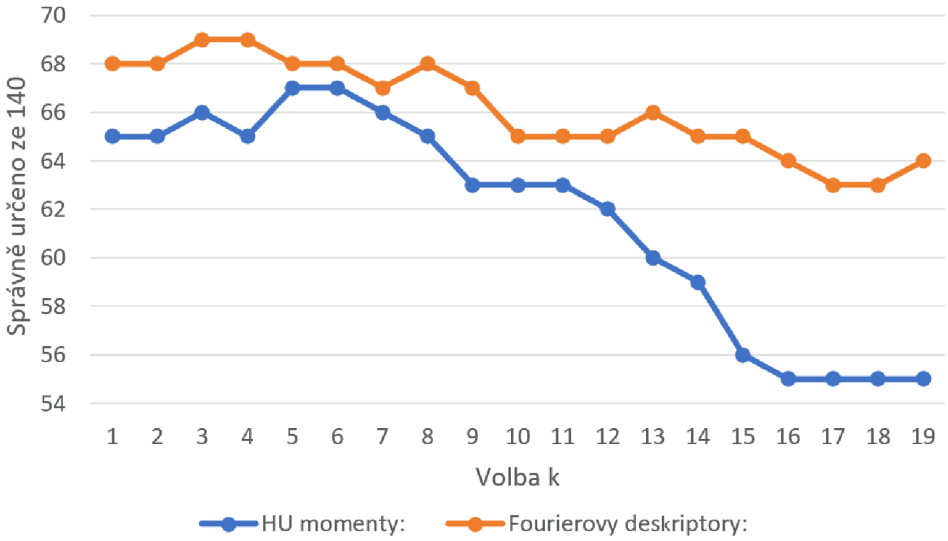
	Fourierův deskriptor				Hu momenty				Průměrně		
	Rocch	1NN	3NN	5NN	Rocch	1NN	3NN	5NN	Fourier	HU	Oba
P1	66	69	67	67	51	65	63	62	67,25	60,3	63,8
P2	65	68	68	67	53	66	65	65	67	62,3	64,6
P3	65	68	69	68	50	65	66	67	67,5	62	64,8
Pinf	61	70	69	68	51	65	65	66	67	61,8	64,4
Pcos	65	69	68	67	56	62	60	59	67,25	59,3	63,3

Obrázek 7.1: Počet nalezených správně ze 140

Můžeme vidět, že mezi posuzovanými metrikami není velký rozdíl a nelze z dat prokazatelně zvolit jednu metriku jako nejvhodnější. Můžeme vidět, že v závislosti na použitém detektoru a klasifikátoru mohou jiné metriky vycházet lépe. Přesto si pro další posuzování dovolíme zvolit metriku, která nám při testování vyšla nejlépe, a to ρ_3.

7.1.2. Vhodná volba k pro klasifikátor KNN

V této části se zaměříme na volbu optimálního počtu nejbližších sousedů k, ze kterých budeme klasifikovat objekt metodou nejbližších sousedů uvedenou v části 5.1.2. Při testování budeme uvažovat metriku ρ_3 a jako obrazové deskriptory Hu momenty a Fourierův deskriptor se zachováním 20 frekvencí.



Obrázek 7.2: Graf úspěšnosti v závislosti na volbě k

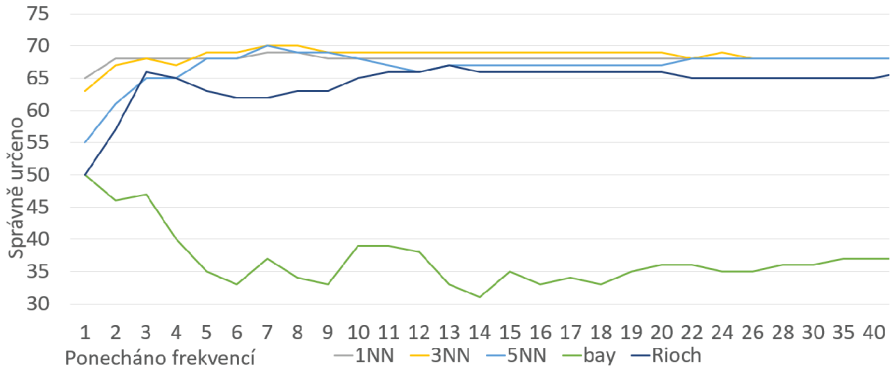
Můžeme vidět, že v našem případě se jako optimální volba k jeví volba v rozmezí od 3 po 6. Při zvyšování volby k nad 9 již dochází k poklesu správnosti, která se pro zvyšující volbu k již jenom zhoršuje. V našem testování máme pro každý objekt 18 etalonů. Optimální volba k se dá uvažovat v rozmezí 16-36% počtu etalonů pro objekt. Mezní hodnota k nám pak vyšla jako 50% počtu etalonů pro objekt. Nemusí se jednat o optimální pro všechny typy množiny etalonů, avšak se může jednat o výchozí nastavení před provedením vlastní optimalizace pro jiné data, případně pokud optimalizaci nelze provést.

7.2. Optimální nastavení obrazových deskriptorů

Když jsme již vyřešili vhodnou volbu klasifikátorů, je potřeba určit optimální nastavení obrazových deskriptorů. Konkrétně se zaměříme na vhodnou volbu počtu ponechaných frekvencích u Fourierova deskriptoru a vhodnou volbu zernikových polynomů. Pro nalezení optimálního nastavení budeme opět porovnávat počet správně určených objektů ze 140 testovaných.

7.2.1. Optimální množství zachovaných frekvencí Fourierova deskriptoru

Zachování většího množství frekvencí bude mít za následek vyšší citlivost na jemnější změny v tvaru objektu. Avšak velké množství může mít za následek velkou citlivost na malé změny. Naopak zachování malého počtu frekvencí může mít za následek vznik nedostatečného rozlišení některých objektů. Vliv množství zachovaných frekvencí na obraz můžeme vidět na obrázku 4.2, vliv na přesnost pak na následujícím grafu 7.3.

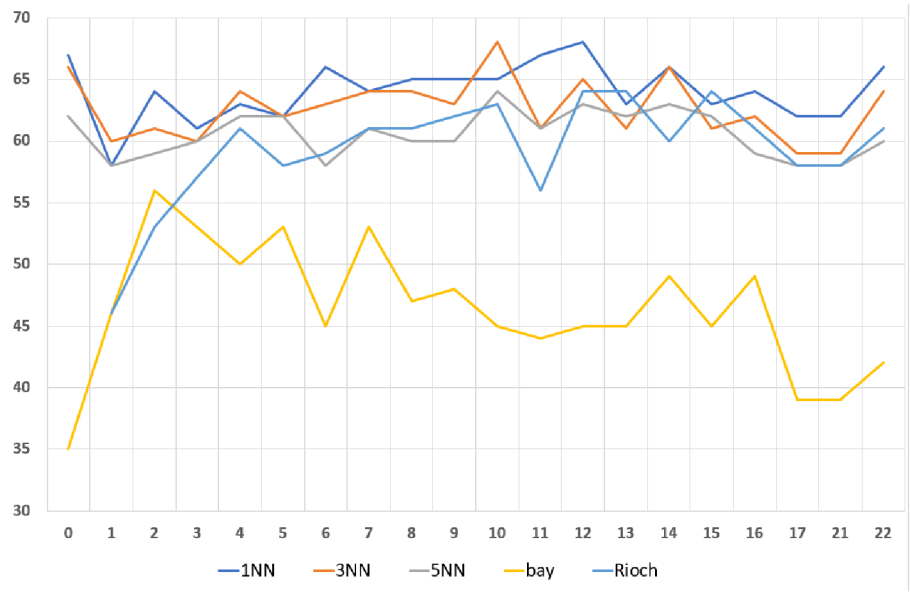


Obrázek 7.3: Graf úspěšnosti v závislosti na množství zachovaných frekvencí

Jako první bych se zaměřil na výsledky pro Bayesovského klasifikátor. Můžeme vidět, že se jedná o nejhorší výsledky a zároveň značný pokles při zvyšování počtu frekvencí. Nejpravděpodobněji to je způsobeno tím, že uvažujeme rovnoměrné váhy pro všechny frekvence, ačkoliv nižší frekvence udávají důležitější informaci. Při použití tohoto klasifikátoru je vhodnější zvolit jinou váhovou funkci. Jinak můžeme vidět, že úspěšnost deskriptoru pro ostatní klasifikátory roste do ponechání 4 frekvencí, a následně se již výrazně nemění. Za zmínku možná stojí lepší výsledky kNN oproti Rocchio klasifikátoru pro pět až deset ponechaných frekvencí. Nejlepšího výsledku, ačkoliv jenom nepatrně, jsme dosáhli ponecháním 8 frekvencí a použitím klasifikátoru 3NN nebo 5NN.

7.2.2. Optimální množství Zernike momentů

Jedná se o volbu Zernikových polynomů, konkrétně na jak velký počet budeme chtít obraz rozložit. Se zvyšujícím počtem roste přesnost, avšak i časová náročnost. Proto je vhodné najít optimální počet. Opět se podíváme na výslednou přesnost pro všechny klasifikátory. Graf 7.4 udává počet správně určených objektů ze 140.



Obrázek 7.4: Graf správně určených objektů ze 140 v závislosti na volbě řádu polynomu n ortogonální báze.

Jako optimální volby báze Zernikových polynomů nám vyšla dvě nastavení. Konkrétně báze $n = 10$ s příslušnou volbou $m = 0, 2, 4, 6, 8, 10$ při použití klasifikátoru 3NN. A dále pak báze $n = 12$ s příslušnou volbou $m = 0, 2, 4, 6, 8, 10, 12$ při použití klasifikátoru 1NN. Za zmínku stojí ovšem i překvapivě dobré výsledky při využití báze obsahující jediný polynom získaný volbou $n = m = 0$, kdy získáváme použitím 1NN dosti podobné výsledky, ovšem s daleko nižší časovou náročností. Volba $n = 0$ v našem případě vycházela přibližně dvacetkrát rychlejší. Touto volbou získáváme jedinou hodnotu popisující objekt, i následná klasifikace Zernike momentu je tedy rychlejší.

7.3. Porovnání výsledků jednotlivých metod při optimálním nastavení

V této části se podíváme na nejlepší dosažené výsledky popsaných obrazových deskriptorů, a stejně tak i na nejlepší výsledky popisovaných klasifikátorů. Jako úspěšnost zde uvádím počet správně určených objektů ze 140 testovaných.

Fourierův deskriptor dosáhl nejlepších výsledků při zachování 7 frekvencí a použití 3NN klasifikátoru. Dokázali jsme tak určit správně 70 ze 140 objektů.

V případě optimálního nastavení Zernike momentů s volbou $n = 10$ a použitím klasifikátoru 3NN jsme dosáhli nejlepší přesnosti. Dokázali jsme tak určit správně 68 ze 140 objektů.

Pro Hu momenty s použitím 5NN klasifikátoru jsme dosáhli nejlepší přesnosti. Dokázali jsme tak určit správně 67 ze 140 objektů

Z klasifikátorů pak lze doporučit jednoznačně klasifikátor 3NN a 5NN v případě, že nerozhoduje rychlost klasifikace. Pokud požadujeme menší časovou náročnost klasifikace, lze doporučit Rocchio klasifikátor.

Naopak nejhorší výsledky v testování měl Bayesovského klasifikátor a nelze jej tedy pro klasifikace našich výsledků doporučit. Tato skutečnost může být způsobena předpokladem normálního rozdělení hodnot deskriptorů, neboť tento předpoklad nemusí být obecně pro data pravdivý.

8. Ukázka na reálných snímcích

V této kapitole se podíváme na pár digitálních fotografií a provedeme klasifikaci objektů, které se na nich nachází. Ukážeme si úpravu fotografií v jednotlivých krocích, které je potřeba udělat, a na závěr i výsledné ohodnocení jednotlivými deskriptory a klasifikátory.

8.1. Ukázka fotografií

Jako první se podíváme na fotografie, které jsme zvolili ke klasifikaci. Je nutné podotknout, že se musí jednat o fotografii některého z objektů, který se nachází v množině etalonů, kterou máme. Posuzované reálné fotografie, které budeme klasifikovat, můžete pozorovat na obrázku 8.1. Pro testování na reálných snímcích jsme si zvolili fotografii tužky, klíče a koně. Je vhodné mít objekty na jednolitém pozadí pro jednodušší separaci na množinu objektu a pozadí. Na fotografii koně si ukážeme i případ, kdy není jednolité pozadí, ale existuje významný rozdíl mezi objektem a okolím. Obecně však zde uvedenou metodu pro separaci nelze použít pro fotografie s nejednolitým pozadím a je potřeba pro separaci využít jinou metodu.

(a) klíč (b) Kůň (c) tužka

Obrázek 8.1: Testované digitální fotografie

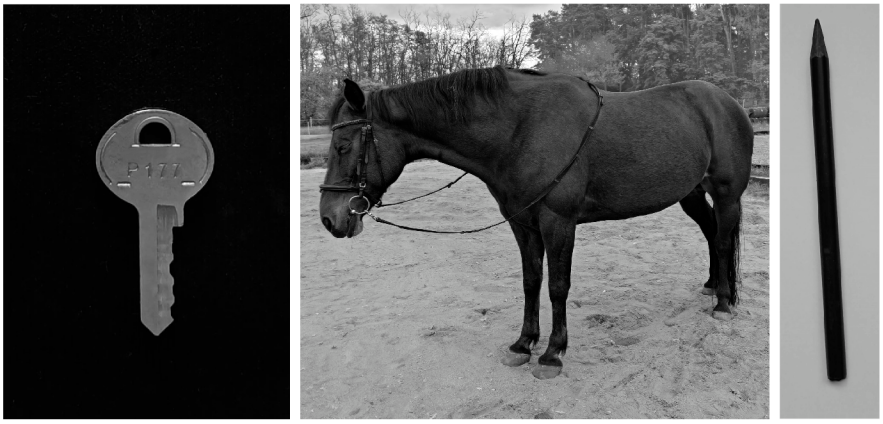
8.2. Úprava fotografií

Před samotným vyhodnocením je potřeba zpracovat fotografie. V ideálním případě je převést na binární digitální fotografie, a přesně to nyní uděláme následujícími kroky.

8.2.1. Převod na černobílou digitální fotografii

První krok, který uděláme, bude převod digitálních barevných fotografií na černobílé. Využijeme k tomu vztah definovaný rovnicí (2.3), který pro připomenutí zní $i = c_r r + c_g g + c_b b$. Při převodu na černobílou fotografii využíváme možnost volby jednotlivých konstant c_r, c_g, c_b pro dosažení největšího kontrastu intenzity pozadí vůči objektu. Pokud očekáváme světlý objekt na tmavém pozadí, volíme při převodu na černobílou fotografii předně barvu,

která se v objektu nachází nejvíce. Tento přístup můžeme vidět v případě fotografie klíče. Naopak v případě, že očekáváme objekt na světlém pozadí, dáváme největší váhu barvám, které se v objektu vyskytují nejméně. Jedná se o případ fotografie tužky. Stejný přístup jsme však využili i v případě fotografie koně. Na fotografii můžeme vidět hnědého koně a jako pozadí nám zde vystupuje žlutý písek a zelené stromy. V tomto případě jsme tedy zvolili zachování pouze informace o intenzitě zelené barvy. Zelená ani modrá barva nenabývá velké intenzity pro náš objekt. Avšak zelená barva nabývá velkou intenzitu v pozadí, a tak jsme využili pouze této barvy při převodu na černobílou fotografii. Fotografie po převodu na černobílou fotografii lze vidět na obrázcích 8.2.

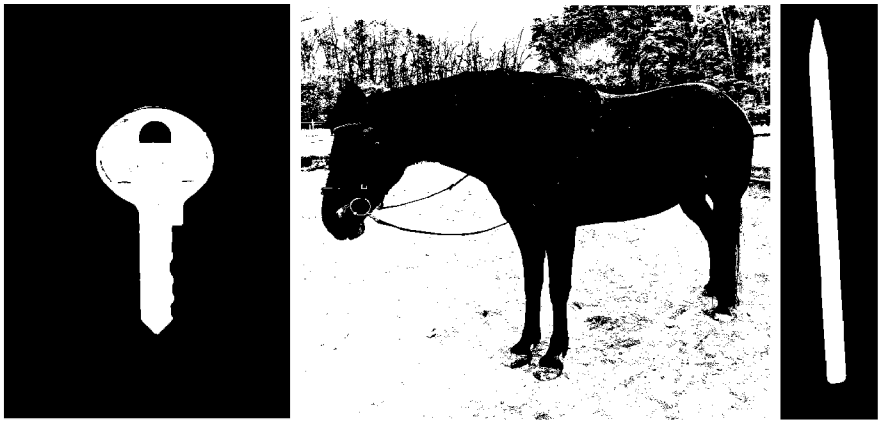


(a) klíč (b) Kůň (c) tužka

Obrázek 8.2: Převod na černobílé digitální fotografie

8.2.2. Nalezení objektu ve fotografii

Další krok, který potřebujeme vykonat, je určení množiny pixelů objektu. Jedná se vlastně o převod na binární digitální fotografii. Budeme k tomu využívat prahování, kde práh získáme pomocí Otsu metody. Otsu metoda je algoritmus pro nalezení prahu v obraze, který vychází z histogramu obrazu, a je založena na statistice. Snaží se naleznout optimální práh, který rozdělí pixely fotografie do dvou množin tak, aby byl maximální rozptyl mezi těmito množinami. Výhoda tohoto přístupu je jeho jednoduchost a rychlost. Avšak prahování lze využít pouze pro případy, kdy je intenzita objektu a pozadí jiná. Další nevýhodou je skutečnost, že takto nalezená množina objektu nemusí být spojitá, to můžeme vidět i na následujících obrazcích 8.3.



(a) Klíč (b) Kůň (c) Tužka

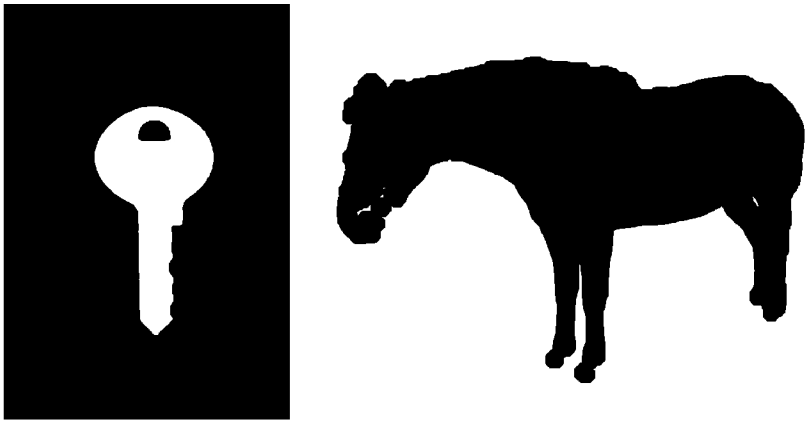
Obrázek 8.3: Fotografie rozdělené na objekt a pozadí prahováním

Abychom vyřešili nedokonalosti v rozdělení na množiny pozadí a popředí, využijeme morfologické operace pro úpravu binárních fotografií. Jejich použitím však dochází ke ztrátě jemných detailů na hranici, proto není vhodná volba příliš velkého strukturálního prvku.

Otevření budeme využívat pro odstranění nežádoucích pixelů, které se nám dostaly do množiny objektu. Při použití otevření si musíme dávat pozor na mezery v množině objektu.

Uzavření nám bude sloužit pro vyplnění mezer a nespojitostí v množině objektu. Při použití uzavření si musíme dávat pozor na případné nežádoucí pixely z pozadí.

Když se nyní vrátíme k binárním fotografiím 8.3, můžeme vidět, že v případě tužky se jedná o dokonalé rozdělení na množinu objektu a pozadí. V případě binární fotografie klíče můžeme vidět, že nedošlo k dokonalému nalezení všech pixelů, na kterých se nachází objekt, a v množině objektu se tedy nachází „díry“. Pro dosažení lepšího výsledku jsme tedy provedli uzavření binární fotografie pomocí vhodně zvoleného strukturálního elementu. V případě fotografie koně můžeme vidět, že velká část pozadí se nám dostala do množiny objektu. Zároveň se v množině objektu nacházejí i „díry“ jako v případě fotografie klíče. Nejprve jsme odstranili nežádoucí pixely pozadí z množiny objektu pomocí otevření a následně pro zaplnění „děr“ v množině objektu jsme provedli uzavření množiny, kterou jsme získali po provedení otevření. V případě fotografie koně došlo k nezanedbatelné změně binární fotografie. Je to způsobeno volbou velkého strukturního elementu, z důvodu množství pixelů z pozadí v množině objektu.



(a) klíč (b) Kůň

Obrázek 8.4: Fotografie po úpravě morfologickými operacemi

8.3. Použití obrazových deskriptorů na fotografie

Binární fotografie nyní převedeme pomocí obrazových deskriptorů na řadu čísel popisující daný objekt.

8.3.1. Hu momenty

Jako první si spočítáme Hu momenty pro naše fotografie. Zaokrouhlené hodnoty Hu momentů pro zvolené reálné fotografie můžeme vidět v tabulce níže.

Hu momenty reálných snímků

Moment:	I_1	I_2	I_3	I_4	I_5	I_6	I_7
klíč	0.3038	0.0409	0.0182	0.0065	6.99e-05	0.0013	2.35e-06
kůň	0.2797	0.0273	0.0010	3.3e-04	-1.43e-07	1.16e-06	-1.18e-06
tužka	1.523	2.2927	0.0042	0.0041	1.686e-05	0.0061	-6.154e-09

8.3.2. Zernike momenty

Jako další určíme pro reálné snímky Zernike momenty. Konkrétně budeme určovat Zernike momenty Z_{10}. Zaokrouhlené hodnoty Zernike momentů pro testované reálné fotografie můžeme vidět v následující tabulce.

Zernike momenty Z_{10} reálných snímků

Moment:	Z_{10}^0	Z_{10}^2	Z_{10}^4	Z_{10}^6	Z_{10}^8	Z_{10}^{10}
klíč	0.014	0.0087	0.0065	0.019	0.0288	0.0075
kůň	0.0232	0.0075	0.0062	0.0149	0.0051	0.0232
tužka	0.8619	0.1041	0.0863	0.1005	0.1028	0.0392

8.3.3. Fourierovy deskriptory

Na závěr určíme pro reálné snímky i Fourierovy deskriptory. Budeme ponechávat 8 frekvencí od nulové frekvence. Z důvodu přehlednosti jsou v tabulce níže ukázány pouze 3 frekvence od nulové frekvence. První řádek udává frekvenci.

Fourierovy deskriptory reálných snímků

frekvence:	0.0625	0.125	0.1875	-0.1875	-0.125	-0.0625
klíč	0.4903	0.0576	0.0799	0.0049	0.0821	0.2080
kůň	0.3855	0.1334	0.0371	0.0534	0.0915	0.0795
tužka	0.4569	0.0007	0.0571	0.0363	0.0013	0.3900

8.4. Klasifikace fotografií knihovnou MPEG-7

Posledním krokem je klasifikace za pomoci některého z klasifikátoru s použitím množiny známých etalonů. Opět zde využijeme knihovnu etalonů *MPEG-7 CE Shape-1*. Je nutné podotknout, že tato knihovna nevychází z reálných fotografií, a není proto příliš vhodná pro tuto klasifikaci. Pro dosažení lepších výsledků je lepší vytvoření vlastní knihovny etalonů na základě objektů, mezi kterými budeme chtít rozlišovat. Provedeme klasifikaci na základě všech uvedených klasifikátorů. Výsledek pro jednotlivé deskriptory a klasifikátory můžeme vidět v následujících tabulkách.

Klasifikace fotografie klíče

klasifikátor	Bayesovského	Rocchio	1NN	3NN	5NN
Hu momenty	strom	klíč	klíč	klíč	pták
Zernike	Hrnek	dítě	auto	auto	auto
Fourierovy	zařízení 9	klíč	klíč	klíč	klíč

Klasifikace fotografie koně

klasifikátor	Bayesovského	Rocchio	1NN	3NN	5NN
Hu momenty	Pták	brouk	kráva	kráva	kráva
Zernike	jablko	telefon	čepice	čepice	čepice
Fourierovy	krysa	krysa	velbloud	velbloud	kůň

Klasifikace fotografie tužky

klasifikátor	Bayesovského	Rocchio	1NN	3NN	5NN
Hu momenty	pružina	pružina	ryba	tužka	tužka
Zernike	stef	tužka	tužka	tužka	tužka
Fourierovy	zařízení 9	tužka	tužka	tužka	tužka

Na těchto výsledcích můžeme vidět, že v případě fotografie koně došlo použitím eroze a dilatace k velké změně množiny objektu a dochází tak k záměně objektu za podobné objekty, jako je kráva a velbloud. Zároveň můžeme vidět opět menší přesnost Bayesovského a Rocchio klasifikátoru. V případě fotografie koně můžeme poukázat i na další potenciální problém v případě klasifikace živočichů, na který je potřeba v případě klasifikace myslet. Všechny etalony koně obsahují koně s hlavou vzhůru. Díky tomu následně případ koně s hlavou dolů nemusí být správně rozpoznán.

8.5. Klasifikace fotografií na etalonech z reálných fotografií

Pro srovnání jsme si vytvořili vlastní knihovnu etalonů, při jejíž tvorbě jsme vycházeli z reálných fotografií, které jsme převedli na binární fotografie. Tato knihovna obsahuje etalony pro objekty: tužka, jablko, banán, vidlička, lžička, klíče, kůň. Pro každý objekt jsme vytvořili 7 etalonů. Zároveň je nutné zmínit, že testované fotografie nebyly využity při tvorbě této knihovny. Ačkoliv se jedná o menší knihovnu a pro každý objekt obsahuje méně etalonů, uvidíme, že díky tomu, že vychází z reálných snímků, bude dosahovat pro reálné fotografie mnohem lepších výsledků. Opět si zde ukážeme tabulku klasifikace pro fotografii klíče, koně a tužky.

Klasifikace fotografie klíče na reálných etalonech

klasifikátor	Bayesovského	Rocchio	1NN	3NN	5NN
Hu momenty	kůň	klíč	kůň	klíč	kůň
Zernike	kůň	klíč	klíč	banán	klíč
Fourierovy	jablko	klíč	klíč	klíč	klíč

Klasifikace fotografie koně na reálných etalonech

klasifikátor	Bayesovského	Rocchio	1NN	3NN	5NN
Hu momenty	kůň	kůň	kůň	kůň	kůň
Zernike	kůň	kůň	kůň	kůň	kůň
Fourierovy	kůň	kůň	kůň	kůň	kůň

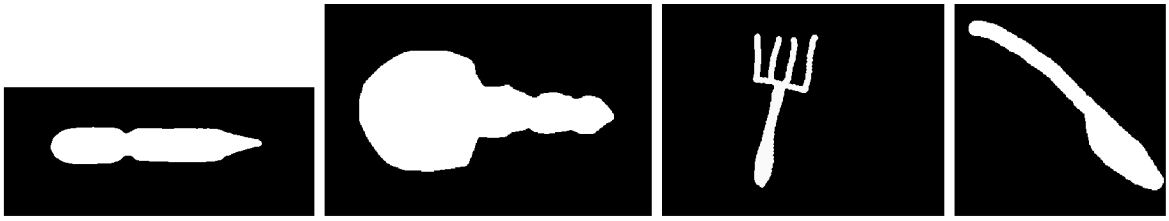
Klasifikace fotografie tužky na reálných etalonech

klasifikátor	Bayesovského	Rocchio	1NN	3NN	5NN
Hu momenty	tužka	tužka	tužka	tužka	tužka
Zernike	jablko	tužka	tužka	tužka	tužka
Fourierovy	tužka	tužka	tužka	tužka	tužka

Můžeme vidět, že použitím reálných fotografií pro tvorbu etalonů objektů jsme dosáhli daleko lepších výsledků, oproti použití knihovny *MPEG-7 CE Shape-1*. Je tedy nutné myslet na povahu fotografií, které budeme posuzovat. V případě, že bychom chtěli klasifikovat kreslené fotografie, například od dětí, je lepší vycházet při tvorbě knihovny také z kreslených fotografií.

8.5.1. Porovnání uvažovaných knihoven etalonů

Uvedeme si zde etalony pro stejné objekty z obou použitých knihoven. Na tomto srovnání půjde předně vidět rozdílný původ etalonů z těchto knihoven.



(a) Tužka (b) Klíč (c) Vidlička (d) Lžička

Obrázek 8.5: Etalony z knihovny *MPEG-7 CE Shape-1*

Knihovna *MPEG-7 CE Shape-1* je založena spíše na ručních kresbách daných objektů. Je tedy vhodná pro klasifikaci kreseb, avšak pro klasifikaci objektů na reálných fotografiích již tak vhodná není. Proto při jejím použití na reálné fotografie nebyla tato metoda příliš úspěšná.

(a) Tužka (b) Klíč (c) Vidlička (d) Lžička

Obrázek 8.6: vlastní knihovna

Vlastní knihovna vychází z fotografií reálných objektů. Etalony této knihovny pochází z fotografií reálných objektů, které jsme převedli na binární fotografie. Tato knihovna je vhodná pro klasifikaci fotografií reálných objektů. Avšak v případě klasifikace kreslených objektů již nemusí být tak vhodná.

8.6. Možné spojení deskriptorů pro klasifikaci

V této části se zamyslíme nad možností využít pro klasifikaci více deskriptorů, případně i některou další charakteristiku popisující daný objekt. Podmínkou na takovou charakteristiku je, že jsme schopni ji reprezentovat pomocí číselné hodnoty.

8.6.1. Možná další charakteristika

Příklad takové charakteristiky může být barva objektu, kterou můžeme využít například při rozpoznání ptáků. V případě černobílé digitální fotografie získáme touto charakteristikou jednu hodnotu udávající průměrnou intenzitu. V případě barevné digitální fotografie spočítáme hodnotu pro každou barvu barevného prostoru zvlášť. Barvu objektu označíme jako Img_{color} a definujeme ji jako:

$$Img_{color} = \frac{\sum_x \sum_y f(x,y)}{N} \tag{8.1}$$

kde hodnota N udává počet pixelů množiny objektu.

Barva objektu je invariantní vůči všem uvažovaným změnám, to jest změně měřítka, posunu, otočení i zrcadlení. Avšak tato charakteristika je závislá na světelných podmínkách při pořizování fotografie.

8.6.2. Klasifikace při využití více deskriptorů

Všechny hodnoty, ze kterých chceme provádět klasifikaci, budeme uvažovat jako jeden vektor. Jelikož pro uvažované deskriptory a charakteristiky mohou hodnoty nabývat různého rozsahu, je vhodné před samostatnou klasifikací z těchto hodnot znormovat možný rozsah hodnot. Toho lze například dosáhnou tím, že sečteme hodnoty na stejné pozici pro všechny etalony a testovaný objekt. Následně tímto součtem podělíme všechny hodnoty na příslušné pozici vektorů. Tento postup opakujeme pro všechny pozice vektoru. Pokud chceme předejít počítání s hodnotami blízkými nule, lze dělení provádět libovolným, avšak pro všechny hodnoty shodným násobkem součtu. Jako taková hodnota se nabízí například: 1/ *počet etalonů*.

V případě, že chceme, aby byl příspěvek od jednotlivých deskriptorů stejný, je nutné hodnoty příslušejících stejnému deskriptoru vydělit počtem hodnot daného deskriptoru. V případě, že chceme přiložit jednotlivým hodnotám různou váhu, stačí vynásobit dané hodnoty váhou pro danou hodnotu.

9. Závěr

V této práci jsme si přiblížili základní způsoby úpravy fotografie za účelem jejího zpracování. Konkrétně jsme si zde popsali převod na binární fotografii a úpravu binární fotografie morfologickými operacemi za účelem odstranit z ní případné vady. Tyto postupy jsme využili při klasifikaci objektů na reálných fotografiích. Dále jsme si přiblížili obrazové deskriptory, včetně jejich matematického základu. Tyto metody jsme použili na třídu etalonů, a stejně tak i na neznámé fotografie. Abychom mohli vyhodnotit výsledky získané obrazovými deskriptory, přiblížili jsme si v dalším kroku i několik klasifikátorů, které k tomu můžeme použít. Uvedené metody jsme i programově zpracovali.

Pro uvedené metody jsme nalezli optimální nastavení a doporučili vhodné kombinace obrazového deskriptoru a klasifikátoru. Dovolíme si zde optimální volby ještě jednou zopakovat. Jako nejvhodnější při klasifikaci je zvolit metriku ρ_3, případně euklidovskou metriku ρ_2. Pro klasifikaci deskriptory lze doporučit:

- Fourierův deskriptor s ponecháním osmi frekvencí od nulové frekvence a klasifikátor 3NN, či případně 5NN.
- Zernike momenty Z_{10} a klasifikátor 3NN.
- Hu momenty a klasifikátor 5NN.

Pro klasifikaci jsme si zároveň vytvářeli i vlastní knihovnu pro lepší výsledky. Můžeme doporučit využít při tvorbě knihovny fotografie stejného charakteru, jakého budou klasifikované objekty. Ukázku špatně zvolené knihovny etalonů pro dané fotografie, stejně jako správně zvolenou knihovnu jsme si v této práci taky ukázali.

Ačkoliv v práci neřešíme časovou náročnost, za zmínku rozhodně stojí i poznatky z testování, které jsme při tvorbě této práce pozorovali. Zernike momenty v uvedené formě jsou časově nejnáročnější, a výpočet Zernike momentů knihovny etalonů na běžném počítači může trvat i několik hodin, zatímco výpočet Hu momentů či Fourierových deskriptorů trvá řadově pouze desítky minut. Časová závislost závisí i na množství počítaných momentů, avšak již v případě výpočtu čtyř Zernike momentů je znatelná.

Literatura

[1] KUMAR, R. M. a K. SREEKUMAR. A Survey on Image Feature Descriptors. International Journal of Computer Science and Information Technologies. 2014, 5(6), 7668-7673.

[2] GONZALEZ, R. C. a R. E. WOODS. Digital image processing. 3rd ed. Upper Saddle River, N.J.: Prentice Hall, c2008. ISBN 978-0-13-168728-8.

[3] HWANG, Sun-Kyoo, Whoi-Yul KIM, Michał NIEDŹWIECKI, et al. A novel approach to the fast computation of Zernike moments. Pattern Recognition. 2006, 39(11), 2065-2076. ISSN 00313203. Dostupné z: doi:10.1016/j.patcog.2006.03.004

[4] DULA, Marek. Srovnání hranových detektorů. Brno, 2021. Dostupné také z: https://www.vutbr.cz/studenti/zav-prace/detail/132355. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav matematiky. Vedoucí práce Jana Procházková.

[5] M. K. Hu, "Visual Pattern Recognition by Moment Invariants", IRE Trans. Info. Theory, vol. IT-8, pp.179–187, 1962

[6] Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, Introduction to Information Retrieval, Cambridge University Press. 2008. ISBN 0521865719.

[7] J. Flusser: "On the Independence of Rotation Moment Invariants", Pattern Recognition, vol. 33, pp. 1405–1410, 2000.

[8] Zernike polynomials. Wikipedia [online]. [cit. 2023-05-04]. Dostupné z: https://en.wikipedia.org/wiki/Zernike_polynomials

[9] HUANG, Min & Ma, Y. & GONG, Qiuping. (2014). Image Recognition Using Modified Zernike Moments. Sensors and Transducers. 166. 219-223.

[10] M. K. Hu, "Visual Pattern Recognition by Moment Invariants", IRE Trans. Info. Theory, vol. IT-8, pp.179–187, 1962

[11] Palanikumar, Radhakrishnan. (2009). Kernel-based Object Detection. Seminar on Spatial Information Retrieval ISI-DRTC.

10. Seznam příloh

Funkce.ipynb - soubor programového jazyku Python. Obsahuje programové zpracování vybraných metod popsaných v této práci.