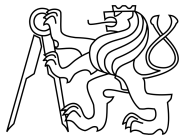
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Detekce počtu mláďat v ptačích hnízdech za pomoci nástrojů rozpoznání obrazu
Student:	Pavel Šuma
Vedoucí:	Ing. Josef Pavlíček, Ph.D.
Studijní program:	Informatika
Studijní obor:	Softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

Navhněte a implementujte knihovnu umožňující zpracovávat data z kamery umístěné v hnízdě (ptačí budce) s cílem rozpoznat počet mláďat v hnízdě. Pro rozpoznání použijte
existující algoritmy strojového vidění (např. OpenCV) a využijte data získaná ze serveru PtaciOnline.cz a další sw. nástroje projektu BirdObserver (athena.pef.czu.cz).

Postupujte v těchto krocích:
1. Proveďte detailní specifickaci požadavků.
2. Seznamte se s projektem BirdObserver a strukturou dat na serveru PtáciOnline.cz.
3. Proveďte analýzu a návrh knihovny.
4. Návrh implementujte, zdokumentujte a vhodným způsobem otestujte.
5. Knihovnu navrhněte a implementujte v jazyce Java tak, aby bylo možné ji formou závislostí (dependency) integrovat s ostatními nástroji projektu BirdObserver.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 23. ledna 2017

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra softwarového inženýrství

Bakalářská práce

Detekce počtu mláďat v ptačích hnízdech za pomoci nástrojů rozpoznání obrazu

Pavel Šuma

Vedoucí práce: Ing. Josef Pavlíček, Ph.D.

15. května 2017

Poděkování

Rád bych poděkoval Ing. Josefovi Pavlíčkovi, Ph.D. za odborné vedení bakalářské práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 15. května 2017 .....

Odkaz na tuto práci

Šuma, Pavel. *Detekce počtu mláďat v ptačích hnízdech za pomoci nástrojů rozpoznání obrazu*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Práce se zaměřuje na rozpoznávání objektů v obraze, konkrétně ptačích mláďat. První část popisuje potřebné metody počítačového vidění ke předzpracování obrazu, jeho segmentaci a rozpoznávání objektů na základě jejich tvarových vlastností. Druhá část tyto a další metody aplikuje na problematiku nalezení počtu ptačích mláďat v obrazovém záznamu ptačích budek a popisuje jejich úspěšnost. Celý program je implementován v jazyce Java bez dalších externích knihoven.

Klíčová slova počítačové vidění, rozpoznávání objektů, ptáci, segmentace prahováním, analýza tvarů

Abstract

The thesis focuses on the recognition of objects in pictures, namely birds. The first part describes the necessary computer vision methods for image preprocessing, segmentation, and object recognition based on their shape properties. The second part applies these and other methods to the issue of finding the number of little birds in the video recordings of bird nests and describes their success. The entire program is implemented in Java without other external libraries.

Keywords computer vision, object recognition, birds, thresholding, shape analysis

Obsah

Úvod **1**

1 Barevné modely a předzpracování obrazu **3**
1.1 RGB . . . 3
1.2 HSV/HSB . . . 3
1.3 Transformace do šedotónového modelu . . . 5
1.4 Histogram . . . 5
1.5 Ekvalizace histogramu . . . 5

2 Segmentace obrazu **7**
2.1 Prahování . . . 7
2.2 Shluková analýza . . . 7
2.3 Detekce hran . . . 8
2.4 Sousednost . . . 11

3 Detekce tvarů **13**
3.1 Řetězový kód . . . 13
3.2 Statistické momenty . . . 14
3.3 Prokrustova analýza . . . 15

4 Realizace **17**
4.1 Ptaci Online . . . 17
4.2 Představení vstupních dat . . . 18
4.3 Použité technologie . . . 18
4.4 Předzpracování . . . 19
4.5 Segmentace a nalezení shluků . . . 21
4.6 Tvarové vlastnosti . . . 23
4.7 Výsledky . . . 28

Závěr **29**

Literatura **31**

A Seznam použitých zkratek **33**

B Obsah přiloženého CD **35**

Seznam obrázků

1.1 RGB model 4
1.2 HSV/HSB model 4
1.3 Histogram 6
1.4 Ekvalizovaný histogram 6

2.1 K-means algoritmus 8
2.2 Detekce hran v obraze 10
2.3 Druhy sousednosti 11

3.1 Směry řetězových kódů 14
3.2 Trojúhelníky pro ukázku řetězového kódu 14

4.1 Vstupní snímek 18
4.2 Ukázka předzpracování vstupního snímku 19
4.3 Snímek po segmentaci 23
4.4 Hledání kruhových tvarů 24
4.5 Ukázka hledání kružnice na jednom tvaru 25
4.6 Obarvení shluků a jejich vnitřků 27
4.7 Nechtěný shluk 28

Seznam tabulek

4.1 Test základních geometrický tvarů 25
4.2 Test Huových momentů 26
4.3 Test metody hledání barev vnitřku zobáku 28

Úvod

Computer vision neboli počítačové vidění je odvětví zabývající se získáváním informací ze zachyceného obrazu. Hlavní motivací je zautomatizování a zefektivnění práce s obrazem, kterou by v opačném případě muselo provádět lidské oko. Díky zvyšování výkonu počítačů zažívá rozmach i tato disciplína a s tím se zvětšuje množina možných využití v reálném životě. Nicméně naučit počítač „vidět“ není vůbec lehká úloha, neboť obraz je koneckonců pouze posloupnost jedniček a nul, které jsou systematicky uspořádané. Proto je většina metod postavena na znalosti pokročilejších matematických teorií. Počítačové vidění se dále dělí na několik podkategorií. Pro toto téma je důležité předzpracování obrazu, jeho segmentace a dále rozpoznávání objektů nebo tvarů. Nejčastěji je známé rozpoznávání obličejů, využívané například při automatickém zaostření fotografie, identifikování osob ze záznamů kamer nebo na sociálních sítích. Dále se využívá například při rozpoznávání vozidel a jejich SPZ. V souvislosti s ornitologií se nejčastěji objevuje rozpoznávání druhu ptactva podle fotky ptáka či jeho vejce.

Ptáci Online je projekt zabývající se ochranou a monitorováním ptačích hnízd v blízkosti městských sídel. Z důvodu stále narůstajícího počtu sledovaných hnízd je žádoucí získávat co nejvíce informací o stavu a chování ptáků automaticky pomocí výpočetní techniky. Z pořízených záznamů a snímků lze pomocí algoritmů umělé inteligence zjišťovat mnoho informací. Úkolem této práce je naučit počítač rozpoznat počet mláďat žijících v hnízdě. Analýza proběhne na základě rozpoznání typických tvarových vlastností zkoumaných objektů, např. chovu ptáčat v době krmení (otevřené zobáčky).

Kapitola 1

Barevné modely a předzpracování obrazu

V této kapitole představím základní používané modely barev v počítačovém vidění a metody jejich transformací pro ulehčení a zlepšení následné práce s obrazem. Při psaní této kapitoly jsem čerpal zejména z [1, 2].

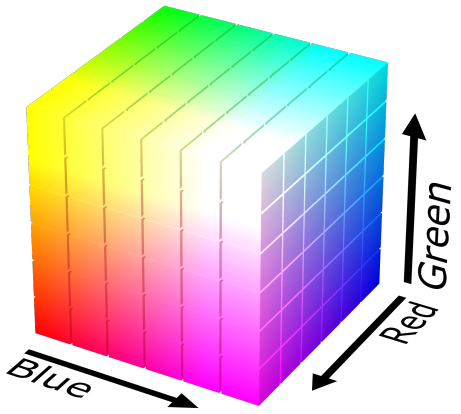
Barevné modely si kladou za cíl digitálně zobrazit co největší množství odstínů viditelného spektra. Těchto prostorů existuje celá řada a každý z nich má výhody i nevýhody, kvůli kterým se liší oblasti jejich využití.

1.1 RGB

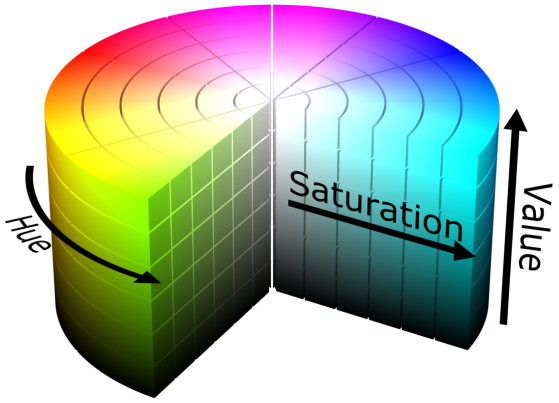
Barevný prostor se skládá ze tří primárních barev – červená (Red), zelená (Green), modrá (Blue). Každá složka je zakódovaná jedním bytem, kde hodnota 0 znamená, že se složka ve výsledné barvě nevyskytuje a hodnota 255 odpovídá nejvyšší intenzitě. Další odstíny se posléze vytvoří složením těchto jednotlivých složek, proto se tento prostor někdy nazývá aditivní. Trojice složek tedy celkem reprezentuje 16 777 216 odstínů (256^3). V některých případech je vhodné využít rozšířený model o další složku alfa, která nese informaci o průhlednosti (model RGBA). V prostoru lze vyjádřit rozložení barev jako jednotkovou krychli umístěnou v osách R, G a B.

1.2 HSV/HSB

Stejně jako předchozí model, se i tento skládá ze tří složek. Barevného tónu (Hue), sytosti (Saturation) a jasu (Value/Brightness). Prostor HSV lze geometricky vyjádřit jako kužel nebo jehlan. Barevný tón nabývá hodnot od 0°do 360°, kde primární červenou barvu reprezentuje 0°a 360°, zelenou 120°a modrou 240°. Sytost nabývá hodnot od 0 do 1, kde nejnižší hodnota je bílá barva a nejvyšší je primární barva. Jas také nabývá hodnot od 0 do 1 a podle



Obrázek 1.1: RGB model (zdroj: www.wikipedia.org)



Obrázek 1.2: HSV/HSB model (zdroj: www.wikipedia.org)

očekávání představuje nejnižší hodnotu barva černá. Tento model se využívá zejména kvůli výhodnému oddělení viditelné barvy a jejích odstínů, čímž se více přibližuje lidskému vnímání barev (červená, světle-červená).

Přechod mezi modely RGB a HSV:

$$
\begin{aligned}
V &= max(R,G,B) \\
S &= \begin{cases} \dfrac{V - min(R,G,B)}{V}, & V \neq 0 \\ 0, & V = 0 \end{cases} \\
H &= \begin{cases} \dfrac{60(G-B)}{V - min(R,G,B)}, & V = R \\ \dfrac{120 + 60(B-R)}{V - min(R,G,B)}, & V = G \\ \dfrac{240 + 60(R-G)}{V - min(R,G,B)}, & V = B \end{cases}
\end{aligned} \tag{1.1}
$$

1.3 Transformace do šedotónového modelu

Ve spoustě případech není nutné znát informace o barevném odstínu a je výhodné pracovat pouze s jednou hodnotou intenzity pro každý bod obrazu namísto tří. Proto se využívá konverze do odstínů šedi. Využívá se například při hledání hran nebo obrysů. Postupů pro konverzi existuje celá řada. Z modelu RGB lze použít například zprůměrování každé složky, nebo nejčastěji využívané vážené zprůměrování podle rovnice (1.2), která zohledňuje různou citlivost oka na barevné odstíny. Z modelu HSV je možné dostat odstíny šedi desaturací každého bodu, čili nastavení složky sytosti na 0.

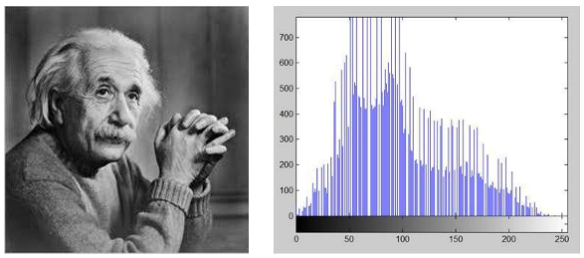
$$Gray = (Red * 0.3 + Green * 0.59 + Blue * 0.11) \tag{1.2}$$

1.4 Histogram

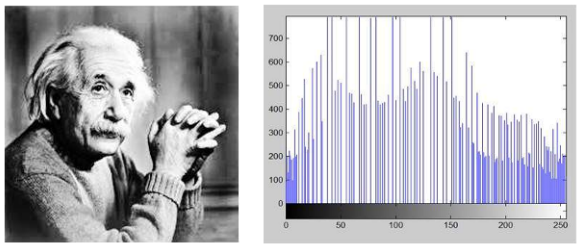
Histogram je dvourozměrný sloupcový graf zobrazující distribuci jednotlivých hodnot barvy/jasu v obraze. Každý sloupec má stejnou, předem zvolenou, šířku intervalu a výška sloupce vyjadřuje četnost veličiny v daném intervalu.

1.5 Ekvalizace histogramu

Metoda upravující rozložení jednotlivých intenzit histogramu za účelem zvýšení kontrastu obrazu. Nejužitečnější je tento postup pro obrazy, které jsou příliš světlé nebo příliš tmavé. Histogram barevných obrazů je možné ekvalizovat pro každou primární barvu zvlášť, tedy upravujeme tři nezávislé histogramy. Tento postup ale není optimální, neboť jednotlivé složky jsou na sobě pro daný bod závislé a tímto způsobem bychom měnili barevnou informaci. Proto se spíše postup aplikuje na šedotónový histogram nebo pouze na některou ze složek po převedení do modelu HSV (například jas). Mějme



Obrázek 1.3: Histogram černobílého obrazu (upraveno z: www.tutorialspoint.com)



Obrázek 1.4: Ekvalizovaný histogram černobílého obrazu (upraveno z: www.tutorialspoint.com)

obraz f definovaný jako matici pixelů se souřadnicemi i,j a množinu všech intenzit vyskytujících se v obraze s rozsahem od 0 do $L-1$. Nechť p vyjadřuje normalizovaný histogram f:

$$p_n = \frac{\text{celkový počet pixelů s intenzitou } n}{\text{celkový počet pixelů}} \quad n = 0, 1, ..., L-1 \tag{1.3}$$

Samotný ekvalizovaný obraz g bude definovaný jako

$$g_{i,j} = (L-1) \sum_{n=0}^{f_{i,j}} p_n \tag{1.4}$$

Kapitola 2

Segmentace obrazu

Segmentací obrazu se rozumí rozdělení obrazu do množiny částí, které dohromady tvoří celý prostor. Je to proces, který každému pixelu přiřadí určité označení. Cílem těchto částí je reprezentovat důležité oblasti obrazu. Motivací pro segmentaci je jednodušší analýza jednotlivých částí oproti zpracovávání celého obrazu. Po segmentaci se také ulehčí hledání objektů či tvarů. Rozdělení pixelů většinou probíhá na základě podobných vlastností jako je například barva, jas, textura nebo sousednost. Při psaní této kapitoly jsem vycházel především z [3, 4, 5, 6].

2.1 Prahování

Implementačně jednoduchá, výkonostně rychlá a přesto velmi účinná metoda segmentace, která na základě intenzity barvy či jasu rozdělí obraz na popředí a pozadí. Vznikají tedy dvě části, přičemž cílem bývá oddělit pro nás důležité body do reprezentující množiny popředí a zbytek do množiny pozadí. Pro úspěšnou segmentaci je klíčové stanovit správnou hodnotu prahu. Ta se dá nastavit různými způsoby, například některou z metod automatického prahování, která sama určí nejvhodnější rozdělení. Další možností je analýzou histogramu najít mezeru mezi dvěma vrcholy nepočetněji zastoupených intenzit. Dále se dá prahování rozdělit na globální a lokální. Při globálním prahování rozdělujeme celý obraz bez ohledu na jednotlivé regiony. Při lokálním bereme v potaz okolí daného bodu, můžeme tedy prahovat zprůměrované okolí pixelu.

2.2 Shluková analýza

Shluková analýza je takový proces, který rozděluje daný prostor do libovolně velkých skupin tak, aby si dva objekty v jedné skupině byly navzájem podobnější než dva objekty mezi skupinami. Čím větší jsou vzájemné podobnosti v jednom shluku a zároveň rozdíly mezi jednotlivými shluky, tím lepší a přes-

Obrázek 2.1: Ukázka K-means algoritmu, vlevo původní obrázek, uprostřed K=2, vpravo K=3 (zdroj: [5])

nější nám vyjde výsledná analýza. Pro „clustering“, jak se tato metoda nazývá anglicky je klíčová zvolená metrika, která definuje vzdálenost mezi vlastními objekty. Při zpracování obrazu se využívá nejčastěji euklidovská nebo manhattanská vzdálenost (2.1) barevných složek pixelu. Kromě využití v počítačovém vidění se shluková analýza hojně využívá například při vytěžování znalostí z dat.

Algoritmy shlukové analýzy se dají rozdělit na několik kategorií. Hierarchické neboli aglomerativní shlukování postupně spojuje nejbližší shluky (samotný pixel je také shluk) do větších. Tím vzniká dendrogram, který lze pomyslně utnout pro libovolný počet výsledných shluků.

Naopak algoritmy dělící postupně rozdělují celek do segmentů jejichž počet předem musí znát. Dělící algoritmus je například K-means, ten náhodně inicializuje středy shluků a pixely do nich rozřadí podle vzdálenosti. Středy aktualizují svoji polohu, aby odpovídaly průměru všech pixelů ve svém shluku. Tyto dva kroky se opakují dokud nastávají nějaké změny v rozřazení pixelů. Výsledek pro různá k (počet středů) je znázorněn na obrázku 2.1. V případě segmentace znamená k počet nejvýznamějších barev.

$$d(p,q) = \sum_{i=1}^{n} |p_i - q_i|, \quad p = (p_1, p_2, ..., p_n), q = (q_1, q_2, ..., q_n) \tag{2.1}$$

2.3 Detekce hran

Segmentovat obraz je možné nejen podle jednotlivých intenzit pixelů, ale i na základě vztahů mezi nimi. Hrana v obraze je pouze méně či více strmý přechod mezi intenzitou bodů. Detekce hran tedy rozdělí body obrazu do dvou množin – množina bodů náležících hran a množina všech ostatních bodů. Metody detekce hran jsou založeny na užití první či druhé derivace obrazové funkce.

Jednou z nejznámějších je Cannyho hranový detektor [7], jehož ukázka je na obrázku 2.2 a který lze shrnout do následujících bodů:

1. Všechny detektory hran jsou velmi ovlivněné i drobnými rozdíly v intenzitě, které ve skutečnosti nepředstavují hranu. Z toho důvodu je vhodné nejdříve odstranit šum provedením konvoluce například pomocí Gaussovské konvoluční masky, jejíž hodnoty dostaneme z Gaussovské funkce pro dva rozměry:

$$g(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.2}$$

2. Nalezení gradientu obrazu, neboli vektoru parciálních derivací:

$$\nabla f(x, y) = [G_x, G_y] = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right] \tag{2.3}$$

   Dále získáme jeho velikost:

$$|\nabla f(x, y)| = \sqrt{G_x^2 + G_y^2} \tag{2.4}$$

   A směr:

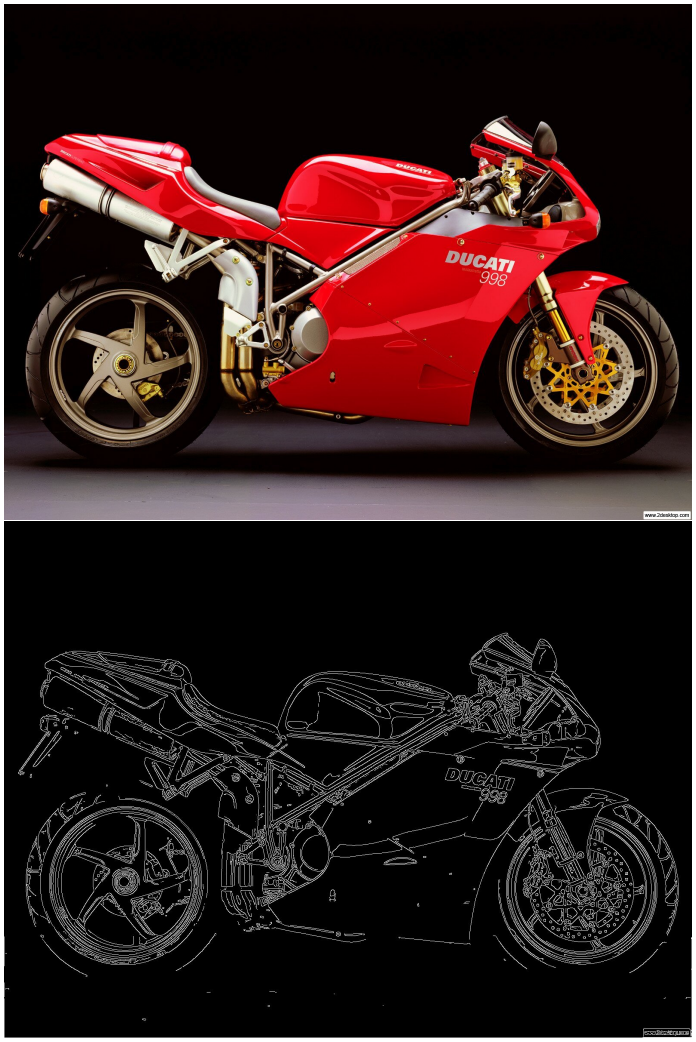
$$\theta = \arctan \frac{G_x}{G_y} \tag{2.5}$$

   V obraze lze aproximaci parciálních derivací získat například pomocí Sobelova operátoru, který konvolvuje s původním obrazem. Tedy na každý bod obrazu jsou aplikovány dvě konvoluční masky pro vertikální (2.6) a horizontální směr (rotací vertikální masky o 90°).

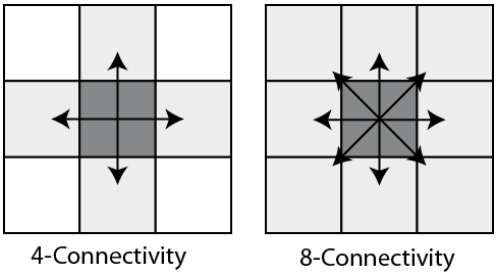
$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \tag{2.6}$$

3. Hrana není většinou reprezentována jediným skokem v intenzitě mezi dvěma pixely, ale mezi několika sousedními. Proto se pro každý bod ve směru vypočteného gradientu (který je kolmý na hledanou hranu) nalezne maximum velikosti gradientu a všechny ostatní body se zahodí. Tím zůstane hrana o šířce 1.

4. Provedení prahování s hysterezí. Pro tuto operaci je nutné vhodně zvolit dvě prahové hodnoty $T_1 > T_2$. Pokud je hodnota gradientu bodu hrany vyšší než T_1, pak je tento bod označený jako silně-hranový pixel. Pokud je jeho hodnota nižší než T_1, ale vyšší než T_2, pak je označen jako slabě-hranový pixel. Všechny ostatní body ignorujeme. Hrany neobsahující žádný silně-hranový pixel jsou z výsledné množiny hran odstraněny.



Obrázek 2.2: Nahoře původní obraz, dole stejný obraz po aplikaci Cannyho detektoru hran (zdroj: https://cyroforge.wordpress.com/canny-edge-detection)



Obrázek 2.3: Vlevo 4-konektivita, vpravo 8-konektivita (zdroj: https://sites.ualberta.ca/čcwj/teaching/image/morph)

2.4 Sousednost

Sousednost pixelu vyjadřuje jeho vztah s okolím. Díky spojitosti pixelů na základě sousednosti a některé další vlastnosti, například intenzity, lze obraz rozdělit do více komponent, které nemusí tvořit celek. Ve 2D prostoru definujeme 4-konektivitu, kde za přímé sousedy pixelu považujeme všechny, které se v dvojrozměrné mřížce dotýkají jeho hran. Čili právě jedna ze souřadnic se liší o jednotkovou vzdálenost. 8-konektivita rozšiřuje předešlou o další 4 pixely, které se v mřížce dotýkají i rohů. V této sousednosti se mohou obě souřadnice současně lišit maximálně o jednotkovou vzdálenost.

Kapitola 3

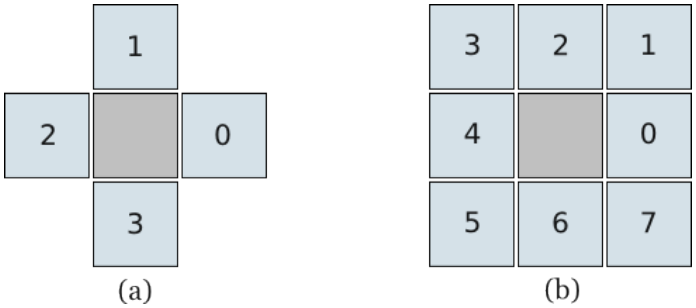
Detekce tvarů

Tato kapitola se bude zaobírat hledáním tvarů v předloze. Pro lidské oko je naprostou samozřejmostí identifikovat všechny objekty v zorném poli prakticky okamžitě. Na druhou stranu ani mozek by nedokázal poznat například stůl, pokud by ho viděl poprvé. Dokázal by ho oddělit od ostatních objektů pomocí detekce hran a prahování, ale už by ho nedokázal oštítkovat. Stejně tak v počítačovém vidění je v podstatě každá metoda této disciplíny založena na porovnávání kandidáta, nalezeného například pomocí segmentace, s šablonou hledaného tvaru. Porovnávat lze na základě mnoha vlastností. Typicky chceme používat algoritmy, které nekontrolují přesnou shodu, ale pouze podobnost do určité míry. Zároveň by měli rozpoznaz i takový tvar, který je pootočený, zmenšený nebo zvětšený a různými způsoby zakřivený. Při psaní této kapitoly jsem vycházel z literatury [8, 9, 10].

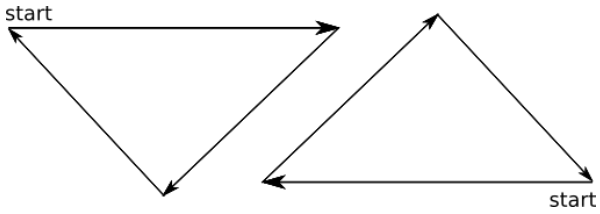
3.1 Řetězový kód

Mnohdy pro ověření tvaru není potřeba popisovat vztahy mezi všemi body. Řetězové kódy popisují tvar pouze na základě jeho hranice. Tato metoda od zvoleného startovního bodu nalezne jeho souseda, taktéž na hraně tvaru, a do vektoru zaznamená směr kterým se nachází. Stejný krok aplikuje pro zbylé body nacházející se na hranici tvaru, dokud se nevrátí do startovního bodu nebo nenalezne konec hranice. Směry je možné zakódovat do oboru číslic například jako na obrázku 3.1.

Existují dva druhy řetězových kódů, absolutní a relativní. Absolutní řetězový kód určuje výsledný směr vztažený k dvourozměrné souřadnicové mřížce obrazu (sever, jih, ...). Relativní řetězový kód určuje souřadnice ve vztahu k perspektivě jednotlivých bodů, nezávisle na okolí. Tím vznikne kód invariantní k otočení. Pro trojúhelník vlevo na obrázku 3.2 vychází absolutní řetězový kód $\{0, 5, 3\}$ a relativní $\{2, 5, 0\}$. Trojúhelník vpravo je pouze otočený o $180°$, a proto pro něj vychází relativní kód stejně jako u prvního trojúhelníku, zato absolutní kód vychází $\{4, 1, 7\}$.



Obrázek 3.1: Ukázka možného zakódování směrů (zdroj: http://www.nongnu.org/rapp/doc/rapp/chaincode.png)



Obrázek 3.2: Oba trojúhelníky budou mít stejný relativní řetězový kód

Pro určení podobnosti mezi dvěma tvary lze spočítat vzdálenost jejich řetězových kódů jako sumu absolutních rozdílů hodnot na stejných pozicích. Takovou vzdálenost je možné spočítat ale pouze pro stejné dlouhé řetězové kódy. Proto je vhodné složitější tvary, které jsou příliš rozsáhlé, aproximovat polygonem o daném počtu vrcholů. Jednodušší, ale méně přesnou, možností, je porovnávat histogramy řetězových kódů.

3.2 Statistické momenty

Tvar je ve své podstatě pouze množina dvourozměrných bodů rozmístěných podle určitých pravidel. Analýzou jeho geometrických vlastností pomocí statistických metod jsme schopni zmenšit zkoumanou množinu bodů na několik hodnot, které dostatečně definují jejich rozmístění.

Základní ukazatele nějakého rozdělení se nazývají momenty. Tyto hodnoty se vypočítají podle vzorce 3.1, kde x a y představují souřadnice bodů. Funkce $f(x, y)$ se v kontextu počítačového vidění uvádí jako intenzita daného bodu. V případě rozlišování tvarů nezávisle na intenzitě se $f(x, y)$ bere jako 1.

$$M_{pq} = \sum_{x} \sum_{y} x^p y^q f(x, y) \tag{3.1}$$

Z momentů je možné získat základní vlastnosti obrazce, jako je například

celková oblast M_{00} nebo centroid z rovnice 3.2.

$$\begin{aligned} \bar{x} &= \frac{M_{10}}{M_{00}} \\ \bar{y} &= \frac{M_{01}}{M_{00}} \end{aligned} \tag{3.2}$$

Centrální moment je veličina charakterizující chování rozdělení vůči jeho průměru. Tím je zajišťena invariantnost posunu obrazce, neboť nezáleží na umístění, ale na relativním rozmístění vzhledem ke středu. Mezi centrální momenty patří například rozptyl (druhého řádu), šikmost (třetího řádu) nebo špičatost (čtvrtého řádu). Lze je vypočítat pomocí rovnice:

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q f(x, y) \tag{3.3}$$

Pro porovnávání tvarů nezávisle na jejich velikosti je potřeba centrální moment normalizovat, tedy vydělit příslušně přeškálovaným nultým momentem (velikost oblasti).

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{(1+\frac{p+q}{2})}} \tag{3.4}$$

Na základě normalizovaných centrálních momentů představil M. K. Hu v roce 1962 7 invariantů [11]. Hlavní myšlenkou je, že kombinací různých normalizovaných centrálních momentů je možné vytvořit invariantní funkce dostatečně popisující různé aspekty obrazce nezávisle na velikosti, rotaci a dokonce osové souměrnosti. Pro kompletnost je zde uvádím:

$$\begin{aligned} \phi_1 &= \eta_{20} + \eta_{02} \\ \phi_2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\ \phi_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\ \phi_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\ \phi_5 &= (\eta_{30} - 3\eta_{12})^2(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\ &\quad + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\ \phi_6 &= (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\ \phi_7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] \\ &\quad - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \end{aligned} \tag{3.5}$$

3.3 Prokrustova analýza

Metoda pojmenovaná podle řecké mytologie, kde Procrustes byl obávaný lupič, který své oběti natahoval, nebo jim naopak uřezával končetiny, aby se přesně

vešli na jeho postel. Pro porovnání tvarů nepotřebujeme informace, které ho nijak neovlivňují, jako je pozice, rotace a velikost. Cílem je tedy transformovat jeden nebo oba tvary tak, aby měly stejnou pozici, otočení i velikost.

Oba tvary je možné zapsat jako matice A, B rozměrů $p \times k$, kde p je počet bodů a k je dimenze. V matici jsou k-tice souřadnic p bodů. Zarovnání obou tvarů lze vyjádřit jako hledání matice W takové, že $\|AW - B\|_F^2$ je minimální. Zápis $\|\cdot\|_F$ vyjadřuje Frobeniovu normu. Tento problém vyřešil Peter Schönemann ve své tezi v roce 1964 [12]. Nechť UDV^T je rozkladem matice A^TB na singulární hodnoty, pak $W = UV^T$ je ortogonální matice, která optimálně superimponuje matici A na matici B.

Po transformaci stačí vypočítat vzdálenost těchto tvarů jako sumu kvadratických vzdáleností jejich dvojic bodů a z ní určit míru podobnosti:

$$P_d^2 = \sum_{j=1}^{p} [(x_{j1} - x_{j2})^2 + (y_{j1} - y_{j2})^2] \tag{3.6}$$

Kapitola 4

Realizace

V této kapitole přiblížím vstupní data poskytovaná projektem PTÁCI ONLINE, jejich předzpracování a následně podrobný postup pro detekci počtu ptačích mláďat v obraze. Nakonec zhodnotím úspěšnost zvolených algoritmů.

4.1 Ptaci Online

„Projekt PTÁCI ONLINE je realizován Fakultou životního prostředí České zemědělské univerzity v Praze od roku 2014.“[13] „Cílem projektu je popularizovat ochranu ptáků v blízkosti lidských sídel, jejich hnízdění, včetně jeho monitoringu, s využitím speciálního technického zařízení tzv. chytré ptačí budky. Projekt s mottem věda lidem – lidé pro vědu se zaměřuje jak na osvětu pro širokou veřejnost, tak na získání cenných informací využitelných pro vědecké účely a praktickou ochranu ptáků v České republice.“[14]

Tyto budky, sloužící pro monitorování běžně hnízdícího ptactva, obsahují 1-2 kamery s nočním přísvitem, pohybový senzor ve vletovém otvoru, vestavěný počítač, teplotní senzor, senzor světelné intenzity a mikrofon. Přenos dat i napájení elektroniky probíhá přes ethernetový - tzv. PoE kabel (průměr do 1 cm). Tento kabel se na jednom konci zapojuje do řídící jednotky v budce a na druhém konci do PoE adaptéru, který je třeba dále připojit k ethernetové zásuvce a zdroji elektřiny. Po nastavení adresování zařízení v síti může uživatel veřejně sdílet videopřenos hnízdění. Aktuální kamerový systém chytré ptačí budky umožňuje sledovat přímý přenos hnízdění běžných druhů ptáků z pohodlí domova a zároveň umožňuje ukládání biologických informací v počítači vestavěném přímo v budce. Na vývoji budky se společně podílely Fakulta životního prostředí ČZU v Praze, Centrum Informatiky, Robotiky a Kybernetiky ČVUT v Praze a firma ELNICO, s.r.o.

Obrázek 4.1: Ukázka neupraveného samostatného snímku záznamu ptačí budky

4.2 Představení vstupních dat

Výstupem vestavěných kamer z ptačích budek je video záznam z vrchní perspektivy. Z takového záznamu lze jednoduše dostat množinu statických PNG souborů. Vzhledem k úspornosti kamer a přenášených dat má záznam poměrně malý počet snímků za vteřinu, není tedy taková množina příliš velká. Na obrázku 4.1 je ukázka jednoho takového snímku. Každý snímek se stane vstupem do mého programu, který vyhodnotí počet mláďat ve statickém obraze. Celkový výsledek počtu mláďat v záznamu bude kombinace jednotlivých výsledků, prezentovaný na konci kapitoly.

4.3 Použité technologie

Celý program je napsaný v programovacím jazyce Java z důvodu dobrých základních knihoven pro práci s obrazem. Dalším důvodem je skutečnost, že celý projekt sloužící k analýze ptačích hnízd je soubor Java knihoven, do kterého bude ta moje přidána. Z knihoven Javy jsem využil především AWT (Abstract Window Toolkit) pro načtení obrazu a práci s barevnými modely. Jedná se o původní nástroj pro tvorbu uživatelského rozhraní, který se od roku 1997 již nevyvíjí. Přesto je plně dostačující a snadno použitelný. Pro odladění a testování jsem pomocí knihovny Swing (nástupce AWT) vytvořil jednoduché okno se vstupním obrazem, do kterého vyznačuji pixely po segmentaci, ohraničující obdelníky kandidátů nebo změnu kontrastu po úpravě histogramu. Ukázky zpracování v této práci budou pocházet právě z této testovací aplikace.

Obrázek 4.2: Vlevo vstupní snímek před předzpracováním, vpravo snímek s upraveným jasem a kontrastem

4.4 Předzpracování

Vzhledem k různým konstrukcím ptačích budek a rozdílnou intenzitou světla při pořizování záznamů je výhodné data předzpracovat, aby se odstranily rozdíly v jasu nebo barvách. Proto jsem u každého snímku normalizoval histogram. Nejprve jsem ekvalizoval histogram všech tří složek barevného modelu RGB. Jas i kontrast se díky tomu u tmavých snímků razantně zlepšil, ale kvůli ekvalizaci každé barevné složky zvlášť se mírně změnily původní barevné odstíny. Z toho důvodu jsem se rozhodl provádět ekvalizaci histogramu pouze u složky jasu z modelu HSV. Na další straně uvádím algoritmus, na kterém je vidět práce s třídou `BufferedImage`, která zaštiťuje vstupní obraz a nabízí funkce pro získání a nastavení pixelu a třídou `Color`, která obsahuje funkce na převod RGB modelu do HSB a zpět. Pro optimální časovou složitost jsem zvolil datovou strukturu vyhledávací tabulka (Lookup Table) reprezentovanou jednorozměrným polem, ve kterém index představuje původní hodnotu jasu a položka v poli na tomto indexu představuje novou hodnotu jasu po transformaci, vypočítanou podle vzorce v kapitole 1.5. Na obrázku 4.2 je vidět výsledek této operace.

Listing 4.1: Funkce pro ekvalizaci histogramu

```
public static void adjustHistogram(BufferedImage bi) {
    int width = bi.getWidth();
    int height = bi.getHeight();
    int[] histogram = new int[1001];
    int i = 0;

    for (int x = 1; x < width; x++) {
        for (int y = 1; y < height; y++) {

         float[] hsbvals = getHSB(bi, x, y);
            int valueBefore = (int) (1000 * hsbvals[2]);
            histogram[valueBefore]++;
        }
    }

    int sum = 0;
    float[] lut = new float[1001];
    for ( i = 0; i < 1001; ++i )
    {
        sum += histogram[i];
        lut[i] = sum * 1000 / width / height;
    }

    for (int x = 1; x < width; x++) {
        for (int y = 1; y < height; y++) {

         float[]hsbvals = getHSB(bi, x, y);
         int valueBefore = (int) (1000 * hsbvals[2]);
            int valueAfter = (int) lut[valueBefore];
            hsbvals[2] = (float) valueAfter/1000;
            int rgb = Color.HSBtoRGB(hsbvals[0], hsbvals
                [1], hsbvals[2]);
            bi.setRGB(x, y, rgb);
        }
    }
}
```

4.5 Segmentace a nalezení shluků

Základním předpokladem pro moji úlohu, který lze pozorovat i na obrázku 4.2, je fakt, že ptačí mláďata mívají otevřené zobáky v době krmení. Ptáci mají zpravidla nevýrazné venkovní zabarvení kvůli ochraně proti predátorům. Na druhou stranu barvy uvnitř zobáků musí být dostatečně výrazné, aby je matka při krmení dokázala rychle najít i potmě a vložila do nich potravu. Úlohu hledání počtu mláďat tedy redukuji na hledání počtu zobáků, protože si tyto hodnoty odpovídají 1:1 a díky zmíněným vlastnostem se zobáky snáze hledají.

Protože tvary zobáků po detekci hran nemusí být vždy úplně uzavřené a barva zobáku je silnější předpoklad, tak jsem pro segmentaci vhodných kandidátů zvolil metodu prahování. Pro každý pixel tedy kontroluji, zda hodnoty odstínu a jasu odpovídají hodnotám žluté barvy podle rovnice 4.1. Tyto hodnoty rozmezí odstínu žluté jsem získal empirickou analýzou zobáčků z náhodně vybrané množiny vstupních snímků. Pro příliš nízké hodnoty jasu není barva rozpoznatelná, a proto jsem ho také omezil.

$$\begin{aligned} &H_{x,y} < 0.3 \wedge H_{x,y} > 0.02 \\ &B_{x,y} > 0.6 \end{aligned} \tag{4.1}$$

Z této segmentace dostaneme množinu bodů odpovídajících žluté barvě zobáků, ale nemáme žádnou informaci o tom, který bod patří ke kterému. Zobáky jsou souvislé oblasti, je tedy vhodné tuto množinu rozdělit do shluků podle sousednosti pixelů. Ve svém programu jsem použil 8-sousednost, protože chci odhalit i zaoblené tvary a zobáky nebývají tak extrémně blízko u sebe, aby mi tato metoda spojila dva do jednoho shluku. Prahování i shlukování provádím v jednom průchodu všech pixelů. Pro každý pixel zavolám funkci 4.2, ve které zkontroluji odstín. Pokud odpovídá barvě popředí, vytvořím nový shluk a pro každého souseda tohoto bodu opakuji tu samou činnost. Jakmile naleznu bod, jehož žádný soused neodpovídá popředí, shluk je kompletní. Abych zbytečně nekontroloval některý pixel vícekrát, udržuji si hashovací mapu, do které po navštívení pixelu vkládám hodnotu 1 pro klíč, který definují jeho souřadnice. Celá tato operace se jeví jako stvořená pro rekurzi, nicméně pixelů je velké množství a zanoření by tak bylo příliš hluboké. Proto jsem rekurzi přepsal jako iteraci pomocí zásobníku. Výsledkem je množina shluků jako například na obrázku 4.3.

Listing 4.2: Základní funkce pro segmentaci bodů

```
public static void processPoint(BufferedImage src, int x
    , int y, HashMap<Pixel, Integer> visited) {

    Pixel px = new Pixel(x,y);
    if (visited.containsKey(px)) {
        return;
    }
    visited.put(px, 1);
    if(isForeground(src, px, intensityRanges)) {
        Blob blob = new Blob();
        Deque<Pixel> q = new ArrayDeque<Pixel>();
        q.add(px);

        while(!q.isEmpty()) {
            Pixel tmp = q.pop();

            for (int k = 1; k < 9; k++) {
                Pixel nbr = neighbours(tmp, k, windowSize);
                if (!visited.containsKey(nbr)) {
                    if (isForeground(src, nbr)) {
                        q.add(nbr);
                    }
                }
                visited.put(nbr, 1);
            }
            blob.addPoint(tmp);

            if (blob.getSize() > 10000) {
                q.clear();
                break;
            }
        }
    }
}
```

Obrázek 4.3: Jednotlivé shluky jsou obarveny rozdílnými barvami, na obrázku lze rozeznat 3 zobáky

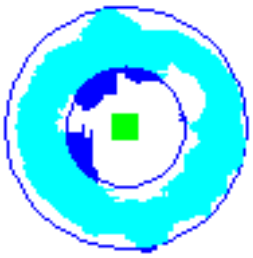
4.6 Tvarové vlastnosti

Na obrázku 4.3 lze také vidět, že kromě zobáků se našly i další oblasti, které nechceme počítat. Je nutné rozhodnout zda shluk odpovídá zobáku. Na takovou úlohu se dá najít spousty různých postupů, kde některé budou mít lepší výsledky. První omezující podmínku jsem nastavil na minimální a maximální počet pixelů shluku. Zobáky mají zpravidla podobnou velikost a navíc příliš malé tvary nelze spolehlivě identifikovat. Další podmínka pokrývá výšku a šířku shluku. Protože zobáky bývají souměrné, omezil jsem i poměr výšky a šířky. Díky těmto podmínkám se odfiltrovala zhruba polovina špatně označených oblastí.

4.6.1 Testovací množina

Rozdělení množiny tvarů do množiny odpovídající zobákům a množiny odpovídající ostatním tvarům lze interpretovat jako binární klasifikaci. Bohužel nemám k dispozici žádná apriorně oklasifikovaná data. Z toho důvodu jsem si vytvořil testovací množinu 12 snímků, každý z jiného záznamu. Do výběru jsem zvolil snímky s rozdílnými vlastnostmi. Obsahuje tedy například tmavé snímky, snímky s malými mláďaty, velkými mláďaty, snímky se zavřenými zobáky nebo rozmazaný snímek. Taková množina není samozřejmě dostatečně velká pro rozsáhlé testy a spolehlivé ohodnocení použitých metod, ale poslouží jako pomocný ukazatel při výběru té nejslibnější.

Pro vyhodnocení úspěšnosti používám statistickou analýzu senzitivity a spe-



Obrázek 4.4: Většina bodů otevřeného zobáku spadá do označené oblasti

cificity. Ty se vyhodnocují pomocí rovnic 4.2, kde TP značí správně ohodnocené shluky jako zobáky (True Positive), FP nesprávně ohodnocené shluky jako zobáky (False Positive), TN správně neohodnocené shluky jako zobáky a FN jako nesprávně neohodnocené shluky jako zobáky. Senzitivita je tedy ukazatel správně ohodnocených jevů ku celkovému počtu pozitivních jevů. Specificita naopak udává podíl správně neohodnocených jevů vůči celkovému počtu negativních jevů. Vysoká specificita znamená, že klasifikátor neoznačuje negativní jevy za pozitivní.

$$\begin{aligned} senzitivita &= \frac{TP}{TP+FN} \\ specificita &= \frac{TN}{TN+FP} \end{aligned} \tag{4.2}$$

4.6.2 Porovnání se základními geometrickými tvary

Jako první kritérium se nabízí tvar, neboť většina zobáků připomíná srdce nebo kruh s prostorem uvnitř. Rozhodl jsem se tedy shluku opsat mezikruží a porovnat poměr pixelů do něj spadající ku celkovému počtu pixelů shluku. Středem kružnic jsem zvolil průměr všech bodů spočtený podle rovnice 3.2. Jako poloměr jsem zvolil polovinu delší strany opisujícího obdélníku vstupního shluku. Neboli polovina nejdelší x-ové nebo y-ové vzdálenosti mezi dvěma pixely shluku. Pro vnitřní ohraničující kružnici jsem zvolil poloviční hodnotu poloměru první kružnice.

Na obrázku 4.5 je znázorněn výsledek této metody. V tabulce 4.1 jsou vidět statistické výsledky po použití na testovací množině. Specificita vychází 1, protože tato metoda neoznačila žádný nesprávný shluk jako zobák. Senzitivita je rovna 0,71, tedy metoda odchytila 71% zobáků.

4.6.3 Řetězový kód

Při segmentaci obrazu a následném rozřazení pixelů do shluků jsem si ty pixely, které mají nějakého souseda neodpovídajícího barvě, a tedy nepatřícího

Obrázek 4.5: Aplikace metody porovnávání tvaru s mezikružím

Tabulka 4.1: Test základních geometrický tvarů

	test pozitivní	test negativní
pozitivní	27	11
negativní	0	33

do shluku, uložil do speciální množiny. Tím jsem naplnil tuhle množinu body z hranice shluku, pro které jsem spočetl absolutní řetězový kód. Už při zběžném testu podobnosti histogramů těchto řetězových kódů pro zdánlivě velmi podobné tvary, mi vyšly dosti odlišené kódy. Tento fakt si vysvětluji drobnými, ale zároveň velmi častými výběžky a podobnými nesrovnalostmi. Řetězové kódy je vhodnější využít na méně komplikované a snáze rozeznatelné stejné tvary, jako například číslice a písmena. Proto jsem tuto metodu zavrhl.

4.6.4 Huovi momenty

Další metodu jsem zvolil analýzu statistických momentů, konkrétně Huovi momenty. Jejich hlavní výhoda je invariantnost vůči velikosti a otočení. Podle rovnice 3.2 jsem našel centroid. Z něho jsem následně dopočítal centrální momenty podle rovnice 3.3, normalizové centrální momenty a nakonec samotné invarianty podle rovnic 3.5. Tím jsem získal vektor 7 hodnot z oboru reálných čísel. Kvůli příliš malým číslům, a také malým rozdílům mezi nimi, jsem všechny hodnoty převedl na kladné pomocí absolutní hodnoty a transformoval

Tabulka 4.2: Test Huových momentů

	test pozitivní	test negativní
pozitivní	32	7
negativní	14	18

logaritmem o základu 10. Vyšli mi například následující hodnoty:

$$\{2.4269, 0.6016, 3.4939, 3.4656, 6.9580, 6.2109, 7.9592\} \tag{4.3}$$

V těchto hodnotách jsem manuálně hledal omezující podmínky, které by dostatečně dobře oddělovaly tvary zobáků od ostatních. Takový postup není úplně optimální, neboť jsem podobnosti hledal na tvarech z testovací množiny, a proto nemusí být následný test nezaujatý. Na druhou stranu testovací množina obsahuje široké spektrum různých tvarů, které si navzájem nejsou tolik podobné. To je také důvod, proč se mi nepodařilo najít podmínky, které spolehlivě oddělí nechtěné tvary. Jediný moment, který měly všechny zobáky podobný, byl ten první a pátý. Omezil jsem je podle rovnice:

$$\begin{aligned} &\phi_1 < 2,7 \wedge \phi_1 > 2,3 \\ &\phi_5 > 6 \end{aligned} \tag{4.4}$$

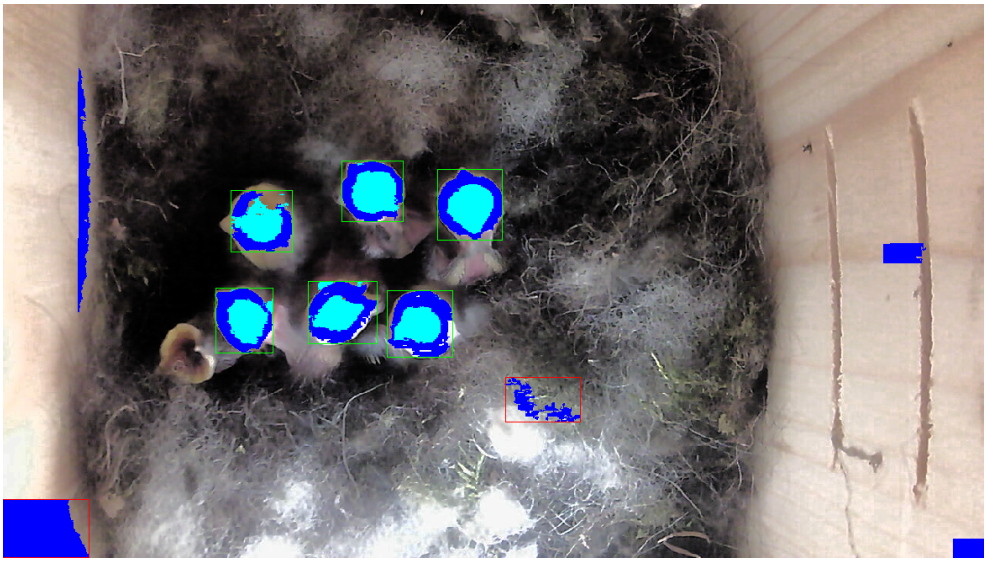
Přesto jsem zkusil aplikovat metodu na testovací množinu a vyšly mi hondoty v tabulce 4.2. Senzitivita tedy vychází 0,82, což znamená, že Huovi momenty správně rozpoznají více zobáků než první metoda. Naopak specificita vychází jenom 0,56, takže celkově označí mnohem více tvarů jako zobáky.

4.6.5 Využití barevných vlastností

Tento postup spoléhá na to, že kromě výrazných žlutých barev na okrajích zobáku se vždy uvnitř vyskytuje červená barva. Proto pro všechny pixely unvitř ohraničujícího obdélníku spočítám počet pixelů odpovídající červené barvě modelu HSV podle vzorce 4.5. Na něm je vidět vlastnost modelu HSV, kde se červený odstín nachází jak kolem 0°(v Java implementaci 0), tak kolem 360°(1), proto je nutné vytvořit jednostranný interval z obou stran. Hodnoty pro odstín červené barvy jsem získal analýzou testovacích snímků. Aby se nezapočítávaly i příliš tmavé body a body s nízkou sytostí barvy, omezil jsem hodnotu jasu a saturace.

$$\begin{aligned} &H_{x,y} < 0.07 \wedge H_{x,y} > 0.93 \\ &S_{x,y} > 0.3 \\ &V_{x,y} > 0.3 \end{aligned} \tag{4.5}$$

Kombinace odstínu žluté a červené uvnitř je velmi silný předpoklad zobáku. Jako další podmínku jsem nastavil minimální nutný počet takových

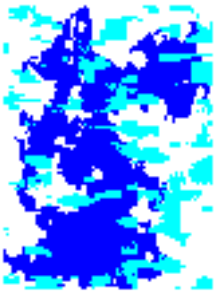


Obrázek 4.6: Úspěšně nalezeno 6 shluků (modře vyznačeno) s odpovídající barvou uvnitř (světle-modře vyznačeno)

vnitřních pixelů, aby se jednalo o zobák. Nicméně kvůli velmi světlým odstínům na zobácích a širokému spektru červené (některá dřeva jsou zabarvena do červené) se při určitém nasvícení budky nalezly i nechtěné oblasti, které tuto podmínku splnily. Na obrázku 4.7 je jeden takový tvar. Kvůli těmto případům jsem zvolil další omezující podmínku na rozptyl.

Rozptyl neboli druhý centrální moment jsem spočetl zvlášť pro souřadnici x a y, pro všechny body uvnitř ohraničujícího obdélníku splňující podmínky na odstín červené barvy, podle vzorce 3.3, kde k je rovno 2. Poměr rozptylu pro souřadnici x a y by se měl pro vnitřek zobáku blížit 1, protože pixely jsou rozmístěné do kruhu. Zároveň jsem omezil každý rozptyl zvlášť, přesněji relativní rozptyl, tedy rozptyl vydělen celkovým počtem odpovídajících pixelů. Toto opatření by mělo omezit výskyt zmiňovaných nechtěných tvarů, které mají odpovídající pixely náhodně rozprostřené a mají tedy velký rozptyl v poměru ku své celkové velikosti.

Po spuštění metody na moji testovací množinu tvarů vyšly dobré výsledky, zapsané v tabulce 4.3. Pro tyto hodnoty vychází sensitivita 0,97 a specificita 1. Tedy pouze jediný tvar byl chybně klasifikován. Jediný zobák, který byl určen jako tvar nechtěný, byl z úhlu trochu pootevřený, proto tvar vnitřních pixelů připomínal spíše elipsu a rozptyl jedné souřadnice vyšel vyšší než té druhé. Naopak u ostatních neúplně otevřených zobáků nebo neúplných tvarů se tato metoda ukázala jako poměrně úspěšná.



Obrázek 4.7: Neodpovídající tvar s odpovídající barvou uvnitř

Tabulka 4.3: Test metody hledání barev vnitřku zobáku

	test pozitivní	test negativní
pozitivní	37	1
negativní	0	33

4.7 Výsledky

Cílem práce nebylo nalézt počet mláďat v jednom snímku, nýbrž v celém videozáznamu. Nabízí se tedy dvě možnosti jak výslekdy jednotlivých snímků zkombinovat pro výslednou hodnotu. První možnost je počítat průměr nalezených hodnot ze všech snímků záznamu. Druhá možnost je vzít maximum nalezených hodnot. Vzhledem k tomu, že mláďata nemají nutně po celou dobu krmení otevřené zobáky, je vhodnější jako výsledek použít druhou zmiňovanou možnost. I přesto se může stát, že v jednom záznamu nebudou mít na žádném snímku všechna mláďata otevřený zobák najednou. V takovém případě moje metody selžou. Další nevýhodou zvolení maxima je fakt, že stačí jeden snímek, ve kterém metoda špatně označí velké množství tvarů za zobáky, a přestože tahle hodnota bude velmi vychýlena od průměru, bude zvolena za výslednou.

Z výše uvedených tabulek je zřejmé, že nejúspěšnější metoda je poslední popsaná, která měla téměř optimální přesnost. Reálné výsledky ale budou nejspíš horší kvůli zmiňovaným nedostatkům snímků.

Závěr

Pro úspěšné splnění zadané úlohy jsem mohl vybrat několik postupů. Zdaleka ne všechny z nich jsem popsal v této práci, protože by se tak její obsah znělikanásobil. Vybral jsem tedy ty, které mi přišly jako nejvhodnější z každé kategorie zpracování obrazu. Z nich jsem dále vybíral nejlépe reálně fungující pro můj problém. Tímto jsem se naučil a vyzkoušel spousty zajímavých algoritmů a detailně nahlédl do teorie spojené s počítačovým viděním. Ukázalo se, že hledání počtu ptačích mláďat je tak specifická úloha, že obecnější a zároveň robustnější metody selhaly nebo nedosáhly takových výsledků jako metody, které jsem konkrétně pro tuto úlohu vytvořil. Osobně se domnívám, že cíl vytvoření vyhovujícího detektoru počtu mláďat v hnízdě byl splněn.

Literatura

[1] Žára, J.; Beneš, B.; Sochor, J.; aj.: *Moderní počítačová grafika*. Brno: Computer Press, 2004, ISBN ISBN 80–251–0454–0.

[2] Gonzalez, R. C.; Woods, R. E.: *Digital Image Processing (2nd Ed)*. Prentice Hall, 2002.

[3] Szeliski, R.: *Computer Vision: Algorithms and Applications*. New York, NY, USA: Springer-Verlag New York, Inc., první vydání, 2010, ISBN 1848829345, 9781848829343.

[4] Kumar, V.: *Cluster Analysis: Basic Concepts and Algorithms [online]*. University of Minnesota, [cit. 2017-04-23]. Dostupné z: `https://www-users.cs.umn.edu/~kumar/dmbook/ch8.pdf`

[5] Bishop, C. M.: *Pattern Recognition and Machine Learning*. Springer, 2006, ISBN 978-0387-31073-2.

[6] Raju, P. R.; G.Neelima: Image Segmentation by using Histogram Thresholding. *International Journal of Computer Science & Engineering Technology*, ročník 2, 2012.

[7] Canny, J.: A Computational Approach to Edge Detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, ročník 8, č. 6, Červen 1986: s. 679–698, ISSN 0162-8828, doi:10.1109/TPAMI.1986.4767851. Dostupné z: `http://dx.doi.org/10.1109/TPAMI.1986.4767851`

[8] Stegmann, M. B.; Gomez, D. D.: A Brief Introduction to Statistical Shape Analysis. 2002, [cit. 2017-05-07]. Dostupné z: `https://graphics.stanford.edu/courses/cs164-09-spring/Handouts/paper_shape_spaces_imm403.pdf`

[9] Barua, J.; Chirgaiya, S.: Shape Recognition & Matching using Chain Code. [cit. 2017-05-07]. Dostupné z: `http://www.irdindia.in/journal_ijacte/pdf/vol2_iss5/2.pdf`

[10] Gerig, G.: Shape Analysis, Moment Invariants. 2010, [cit. 2017-05-03]. Dostupné z: http://www.sci.utah.edu/~gerig/CS7960-S2010/handouts/CS7960-AdvImProc-MomentInvariants.pdf

[11] Hu, M.-K.: Visual pattern recognition by moment invariants, computer methods in image analysis. *IRE Transactions on Information Theory*, ročník 8, 1962.

[12] Schönemann, P. H.: A generalized solution of the orthogonal procrustes problem. *Psychometrika*, ročník 31, 1966.

[13] PtaciOnline.cz: O projektu [online]. [cit. 2017-04-25]. Dostupné z: http://www.ptacionline.cz/o-projektu/o-projektu

[14] Ministerstvo životního prostředí: Ptáci online: Sledujte záběry z „chytré ptačí budky“ na budově MŽP [online]. Červenec 2016, [cit. 2016-12-10]. Dostupné z: http://www.mzp.cz/cz/news_160608_Ptaci_online

Příloha **A**

Seznam použitých zkratek

RGB Red, green, blue (Červená, zelená, modrá)

HSV Hue, saturation, value (Barevný tón, saturace, jas)

PNG Portable Network Graphics (grafický formát)

AWT Abstract window toolkit

FN False negative (Falešně negativní)

TP True positive (Skutečně pozitivní)

FP False positive (Falešně pozitivní)

TN True negative (Skutečně negativní)

Příloha B

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `jar` ..... adresář se spustitelnou formou implementace
- `Testovací snímky` ..... adresář s ukázkovými snímky
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `BP_Pavel_Suma_2017.pdf` ..... text práce ve formátu PDF