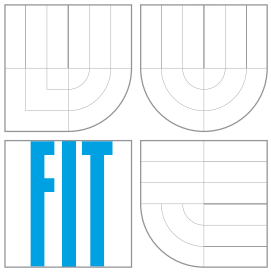
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

# RYCHLÁ DETEKCE PŘÍRODNÍCH OBJEKTŮ

FAST DETECTION OF NATURAL OBJECTS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MARTIN ZELINKA
AUTHOR

VEDOUCÍ PRÁCE — Ing. MARTIN ŠIMON
SUPERVISOR

BRNO 2016

*Zadání bakalářské práce/18559/2015/xzelin18*

**Vysoké učení technické v Brně - Fakulta informačních technologií**

Ústav počítačové grafiky a multimédií Akademický rok 2015/2016

# Zadání bakalářské práce

Řešitel: **Zelinka Martin**
Obor: Informační technologie
Téma: **Rychlá detekce přírodních objektů**
**Fast Detection of Natural Objects**
Kategorie: Zpracování obrazu

Pokyny:

1. Seznamte se s metodami detekce objektů v obraze schopnými detekovat přírodní objekty (listy, houby, zvířata, apod.).
2. Vyberte vhodné metody pro rychlou detekci přírodních objektů a detailně je popište.
3. Navrhněte a implementujte systém schopný rychlé detekce vybraných přírodních objektů.
4. Na vhodných datech vyhodnoťte implementaci.
5. Diskutujte dosažené výsledky a vhodnost použitých metod. Navrhněte další pokračování práce.

Literatura:

- dle pokynů vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- První dva body zadání

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

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

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

Vedoucí: **Šimon Martin, Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 18. května 2016

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

doc. Dr. Ing. Jan Černocký
*vedoucí ústavu*

# Abstrakt

Práce se zabývá detekcí a lokalizací psů v obraze. Rozebírá postupy počítačového vidění vhodné k řešení této problematiky. Je navrhnuto a následně implementováno řešení. V rámci vypracování práce je vytvořena datová sada obrázků reprezentující hledaný objekt a datová sada negativních obrázků. Výsledné řešení je nakonec podrobeno trénování za účelem zjištění nejlepších parametrů pro použité algoritmy. Výsledný program je součástí přiloženého DVD.



# Abstract

Thesis deals with the detection and localization of dogs in the image. It analyzes computer vision techniques suitable for solving this problem. It is designed and then implemented solution. Data set of images representing the object of the search and the data set of negative images are created. The resulting solution is finally subjected to training in order to determine the best parameters for used algorithms. The resulting program is included in the attached DVD.



# Klíčová slova

Počítačové vidění, Rozpoznávání, Přírodní objekty, Lokalizace

# Keywords

Computer vision, Recognition, Natural objects, Lacalization


# Citace

Martin Zelinka: Rychlá detekce přírodních objektů, bakalářská práce, Brno, FIT VUT v Brně, 2016

# Rychlá detekce přírodních objektů

## Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Martina Šimona. Dále prohlašuji, že jsem důsledně uvedl veškerou literaturu a další zdroje, ze kterých jsem čerpal.

.....
Martin Zelinka
18. května 2016


## Poděkování

Tímto bych chtěl poděkovat vedoucímu bakalářské práce Ing. Martinu Šimonovi, který poskytoval cenné rady a odbornou pomoc, která ve velké míře pomohla k úspěšnému vypracování bakalářské práce.

# Obsah

# Kapitola 1

# Úvod

Počítačové vidění umožňuje interpretovat obraz na vstupu a získávat z něj důležité informace, které donedávna byl schopen určit pouze člověk. Díky počítačovému vidění jsme schopni detekovat z obrazu známé objekty, jejich pozici, stav a další informace. K rozvoji tohoto oboru zároveň napomáhá snižování pořizovacích cen kamer a zvyšování výkonu současné techniky. Díky příznivým okolnostem může dnes počítačové vidění zlepšovat lidskou činnost, nebo ji dokonce úplně nahradit v některých činnostech. V praxi se využívá např. k automatizaci výrobních linek, monitorování okolí automobilu a detekování nebezpečí, či k autentizaci uživatele. Dalších možností využití je možné nalézt spoustu a rozsah činností počítačového vidění se do budoucna bude určitě dále rozšiřovat.

Má práce se zabývá rychlou detekcí přírodních objektů. Konkrétněji je zaměřena na rozpoznání psů v obraze. Kromě detekce hledaného objektu je zapotřebí určit jeho lokalizaci. Práce může nalézt uplatnění v bezpečnostním kamerovém systému automobilu, který by při detekci psa ve vozovce byl schopen upozornit řidiče a v nevyhnutelné případě zasáhnout do řízení a zamezit kolizi. Další možnost využití je v bezpečnostním kamerovém systému, který monitoruje veřejný prostor, kde je zákaz vodění psů.

V další kapitole 2 naleznete teoretický úvod do algoritmů počítačového vidění, které se dále využívají při řešení zadání. Následuje kapitola 4, kde je uveden způsob řešení a popis samotné implementace. Kapitola 5 se zabývá testování implementovaného řešení. Poslední kapitola 6 shrnuje dosažené výsledky práce.

# Kapitola 2

# Počítačové vidění

Tato kapitola je teoretickým úvodem do technik a postupů počítačového vidění, které jsou zvažovány při řešení tématu práce. Jednotlivé podkapitoly poskytují nezbytné minimum pro pochopení výsledného způsobu řešení.

Mezi uvedená témata podkapitol patří obrazové příznaky 2.1 dále jsou popsány základní klasifikátory 2.3 a v poslední části kapitoly je věnována pozornost možným způsobům lokalizace objektů v obraze 2.6.

## 2.1 Obrazové příznaky

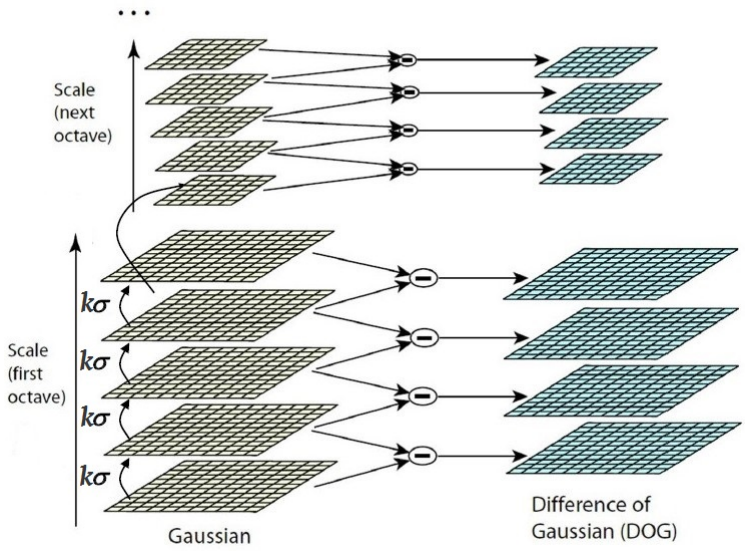
Vstupní obraz ve své původní podobě, kdy je reprezentovaný mřížkou pixelů je příliš nestrukturovaný a zašumělý pro účely strojového učení, proto existuje několik způsobů, díky kterým je možno reprezentovat obraz v podstatně menší velikosti, aniž by došle ke ztrátě důležitých informací, které jsou nezbytně nutné pro povahu dané úlohy kterou řešíme.

Jedním ze způsobů získání důležitých informací z obrazu jsou lokální obrazové příznaky. Obrazové příznaky jsou základním stavebním prvkem mnoha algoritmů počítačového vidění. Jejich vyhodnocení se skládá z několika kroků. Zaprvé jsou detekovány klíčové body v obrazu. Klíčové body nalézají zajímavá místa v obraze - nejčastěji se jedná o rohy a hrany. Poté co jsou detekovány klíčové body, je nutno je převést do podoby vektoru (deskriptoru). Fáze převedení klíčového bodu na deskriptor se nazývá extrakce příznaku. Proces extrakce příznaku využívá hodnoty okolních pixelů daného klíčového bodu. Pro získání lokálních příznaků je možno použít několik algoritmů, kdy každý je vhodný do jiné situace, a proto je potřeba jejich volbu důkladně zvážit.

## 2.2 SIFT

Populární algoritmus pro získání obrazových deskriptorů. Generované deskriptory jsou invariantní vůči změně velikosti, rotaci a jasu. Pro zjednodušení procesu získání výsledného deskriptoru je možné algoritmus rozdělit do pěti částí:

- Vytvoření scale-space pyramidy
- LoG aproximace
- Získání klíčových bodů
- Přiřazení orientace klíčovým bodům.
- Extrakce deskriptoru




Obrázek 2.1: Znázornění scale-space pyramidy, která je vytvořena algoritmem SIFT. Převzato z [2] a upraveno.

Prvním krokem je vytvoření scale-space pyramidy. Tato pyramida reprezentuje vstupní obrázek v několika rozlišeních (oktávách), kdy obraz následující oktávy je vždy poloviční rozlišení oproti předchozí vrstvě. V rámci jedné oktávy je vytvořena sekvence snímku tak, že postupně aplikujeme Gaussovský filtr se zvyšující hodnotou parametru $\sigma$, který udává míru rozmazanosti snímku. Míra nárůstu parametru $\sigma$ je dána konstantou $k$. Grafická podoba pyramidy je znázorněna na obrázku 2.1. Rovnice 2.1 znázorňuje vytvoření snímku $L$ jako konvoluci původního obrázku $I$ s Gaussovským filtrem $G$.

$$L(x, y, k\sigma) = G(x, y, k\sigma) * I(x, y) \tag{2.1}$$

V další fázi je proveden rozdíl dvou sousedních snímků (Rovnice 2.2). Tímto způsobem je získán rozdíl dvou Gaussiánu (DoG[1]), který je považovaný za aproximaci filtru LoG[2].

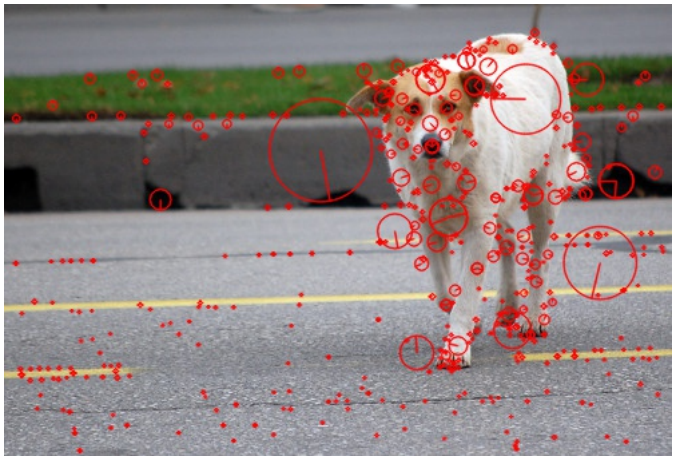
[1] Difference of Gaussians
[2] Laplacian of Gaussian

Přímému použití filtru LoG se záměrně vyhýbáme, jelikož je výpočetně náročný při počítání druhých derivací.

$$D(x, y, \sigma) = L(x, y, k_i\sigma) - L(x, y, k_j\sigma)) \tag{2.2}$$

Jakmile nalezneme DoG snímky, tyto snímky prohledáme a zjistíme lokální minima a maxima, které prohlásíme za klíčové body. Jelikož je algoritmus SIFT invariantní vůči rotaci, je ke každému klíčovému bodu zjištěna jeho orientace pomocí histogramu gradientů (jeden bin je 10° z 360°). Pro ilustraci je uveden obrázek 2.2, kde jsou znázorněny detekované klíčové body.

Poslední krokem je získání deskriptoru klíčového bodu. Za tímto účelem vezmeme okolí 16x16 kolem klíčového bodu a dále jej rozdělíme na 16 buněk o velikost 4x4. Pro každou buňku vytvoříme histogram gradientů o osmi binech, takže celkově je pro jeden klíčový bod vytvořeno 128 binů (4x4x8). Jednotlivým histogramům jsou poté přiřazeny váhy tak, aby ty co jsou blíž ke klíčovému bodu měly vetší váhu než ty vytvořené z okrajových buněk. Tímto je zajištěna větší robustnost deskriptoru vůči obrazovým transformacím [10].



Obrázek 2.2: Detekované lokální příznaky pomocí SIFT algoritmu.

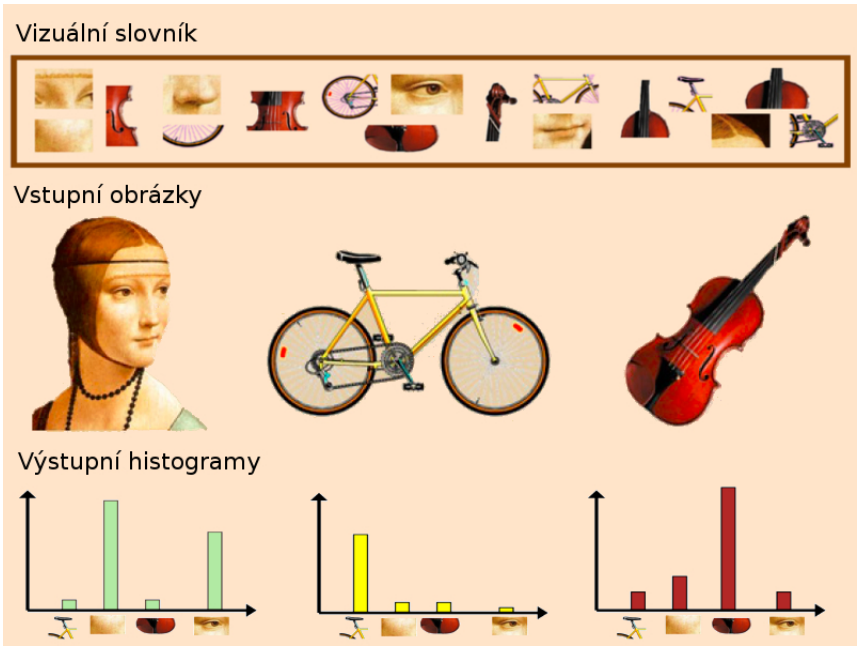
## 2.3 K-means shlukování

Je technika sloužící k vytvoření $k$ shluků v euklidovském prostoru – pro SIFT deskriptory se jedná o 128 dimenzionální prostor. Každý bod je přiřazený ke klastru k němuž má nejblíže. Výsledná pozice klastrů je určená tak, aby součet vzdáleností bodů patřících k nejbližším klastrům, byl co nejmenší. Výsledná poloha klastrů je dána minimalizací rovnice (2.3) kde $x$ je d-dimenzionální vektor patřící ke klastru $S_i$, $\mathbf{S}$ je množina všech klastrů $(S_1, S_2, \ldots, S_k)$ a $\mu_i$ je středem i-tého klastru.

$$\underset{\mathbf{S}}{\operatorname{argmin}} = \sum_{i=1}^{k} \sum_{x \in S_i} \|x - \mu_i\|^2 \tag{2.3}$$

## 2.4 Slovník vizuálních slov

Slovník vizuálních slov je zaužívaný český překlad pro algoritmus Bag of the word dále jen BoW. Základní jednotkou se kterou BoW pracuje jsou vizuální slova. Množina všech vytvořených vizuálních slov vytváří vizuální slovník. Vizuálním slovem je chápán shluk příznaků vytvořený např. pomocí techniky k-means. V praxi vizuální slova vytvořená z trénovací sady obrázků reprezentují charakteristické rysy objektu, který je obsažen v naší datové sadě.



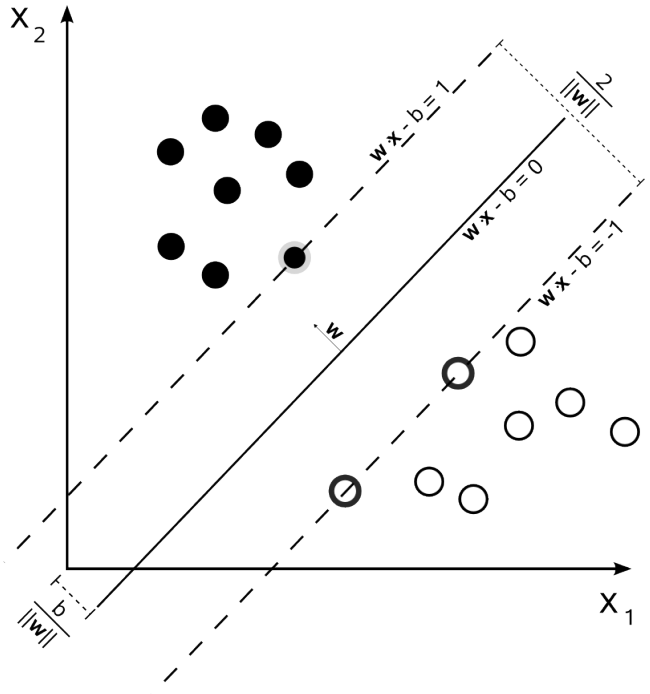

Obrázek 2.3: Názorně proces získání histogramů ze třech vstupních obrázků

Obrázek, který je na vstupu je dále převeden na histogram četnosti vizuálních slov. V tomto histogramu reprezentuje jedena třída počet výskytů vizuálního slova. Je-li vizuální slovo obsažené ve vstupním obrázku je možno určit porovnáváním příznakového vektoru s daným vizuálním slovem. Porovnání počítá s určitou mírou aproximace - nikdy nenastane absolutní shoda. Poté co je vypočítán histogram, je provedena jeho normalizace. Celý postup je znázorněn na obrázku 2.3

## 2.5 Support Vector Machines

Support Vector Machines je populární algoritmus strojového učení určený pro klasifikaci dat. Cílem SVM je nalézt nadrovinu v prostoru příznaků, která co nejlépe dokáže oddělit data dvou tříd. Výsledná nadrovina je lineární funkce $w \cdot x - b = 0$ jak je znázorněno na obrázku 2.4. Aby byla výsledná nadrovina co nejspolehlivější pro klasifikaci, hledá se taková, která dosahuje největší vzdálenosti (*margin*) od nejbližších bodů. Body, které mají nejblíže k lineární nadrovině jsou označované jako podpůrné vekrory (*support vectors*), od čehož je odvozen název klasifikačního algoritmu.

Trénovací data vstupující do algoritmu jsou ve formátu $(x_i, y_i)$, kde $x_i \in R^d$ je příznakový vektor, $y_i$ značí příslušnost k jedné ze dvou tříd $\mathbb{Y} = \{-1, +1\}$ a $i$ je pořadové označení trénovací dvojice v datové sadě o $n$ vzorcích [3][5].




Obrázek 2.4: Nadrovina oddělující data dvou tříd - kolečka bíle, černé výplně

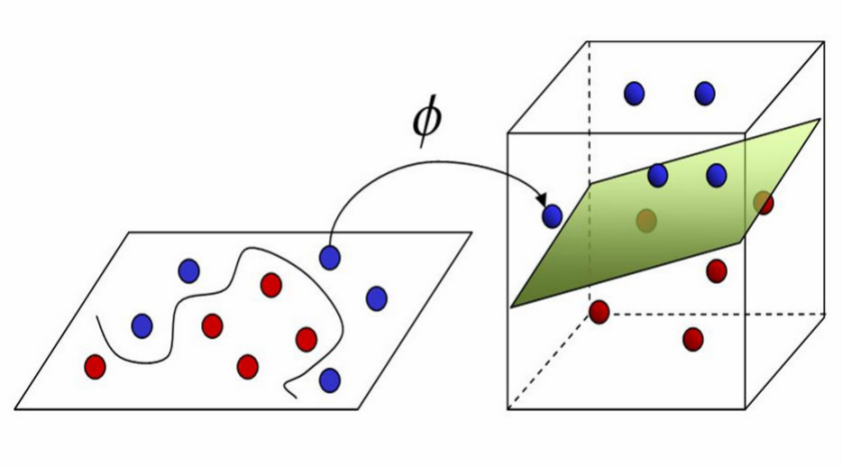
Poté, co je nalezena nadrovina, je možné provést binární klasifikaci vstupního vektoru $x$. Výsledná třída $y \in \{-1, +1\}$ vstupního vektoru je dána jako výsledek funkce $w \cdot x - b$ (2.4) (2.5).

$$y = +1 \quad pro \quad w \cdot x - b \geq +1 \tag{2.4}$$

$$y = -1 \quad pro \quad w \cdot x - b \leq -1 \tag{2.5}$$

### Lineárně neseparovatelná data

V určitých případech není možné využít lineární funkce k rozdělení dat dvou tříd a to ani za použití penalizace za špatně klasifikovaná data. Další možností je použití nelineárního SVM, které přidává datům další dimenzi. Poté co je zvýšena dimenzionalita, je vytvořen nový prostor dat, ve kterém se znovu hledá separující lineární funkce. Vše je znázorněno na obrázku 2.5, kde byla přidána třetí dimenze pro snadnější oddělení dat rozdílných tříd.




Obrázek 2.5: Transformace prostoru pomocí funkce $\phi$, která přidává datovým bodům další dimenzi za účelem lepší separace. Obrázek převzat [9] a upraven.

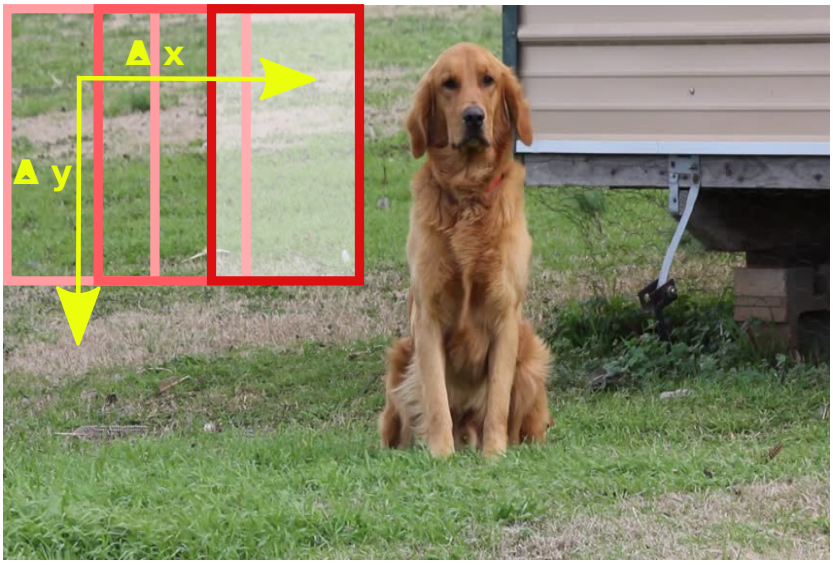
Ke zvýšení dimenze prostoru jsou využity jádrové funkce. Mezi nejčastěji používané patří:

- Polynomicke jadro - $K(x_i, x_j) = \exp(\gamma\, x_i^T x_j + r)^d;\ \gamma > 0$
- Sigmoida - $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$
- RBF(Radial Base Function) - $K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|)$

## 2.6 Sliding window

Detekce a lokalizaci objektu lze provést jednoduchou a účinnou metodou Sliding window v českém překladu plovoucí okno. Pohyb plovoucího okna je znázorněn na obrázku níže 2.6.

Tato metoda nalézá uplatnění v případě, kdy hledaný objekt má charakteristický vzhled a nedochází u něj k velkým deformacím tzn. lze natrénovat klasifikátor z positivních a negativních obrázků, který jej dokáže spolehlivě rozpoznat. Pozitivní obrázky jsou fixní velikosti $(n \times m)$ a obsahují instanci hledaného objektu - nejlépe ve středu a v co největší velikosti. Negativní obrázky obsahují cokoliv, co není náš hledaný objekt - nejlépe časté útvary, které obklopují hledaný objekt. Jako klasifikátor bývá často využíván SVM uvedený výše 2.5.

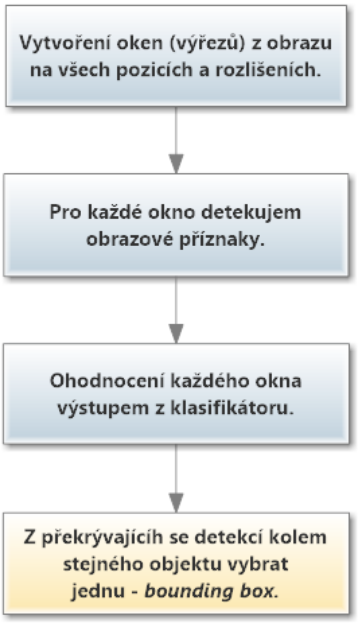



Obrázek 2.6: Sliding window - průchod obrazem.

Za předpokladu, že máme natrénovaný binární klasifikátor můžeme přejít k samotné metodě sliding window - její jednotlivé fáze jsou znázorněny v diagramu 2.7. Prvním krokem je vytvoření množiny oken velikosti $(n \times m)$, které získáme postupným posouváním klouzavého okna o $\Delta x$ a $\Delta y$. Problém může nastat v případě, když se hledaný objekt vyskytuje ve více možných velikostech. Tuto situaci je možné řešit vytvořením Gaussovi pyramidy a provést hledání $(n \times m)$ oken na všech vrstvách této pyramidy. Další řešení spočívá v postupném zvětšování délek hran klouzavého okna o konstantu $s$ na $(sn \times sm)$.

V další fázi je pro každé okno, vytvořené z předešlé fáze posouváním klouzavého okénka, získána množina obrazových příznaků. Příznaky daného okna jsou dále vstupem do binárního klasifikátoru, který určí zdali je okno pozitivní, nebo negativní vůči výskytu hledaného objektu.

V praxi často nastává situace, kdy je hledaný objekt vícekrát detekován - více oken kolem jednoho objektu. Nesprávným řešením by bylo uvažovat o vyladění klasifikátoru, který detekuje objekt pouze pokud je zarovnán přesně na střed okna $(n \times m)$. Tento způsob by v praxi nefungoval, jelikož nikdy nejsme schopni zaručit, že střed klouzavého okna bude taktéž ve shodě se středem hledaného objektu. Problém více překrývajících se detekcí kolem jednoho objektu, je řešen hledáním lokálního maxima, které vybere okno s největší pozitivní odezvou z klasifikátoru a zbylá překrývající se okna jsou zahozena [4]. Metoda Sliding




Obrázek 2.7: Sliding window - jednotlivé fáze lokalizační metody.

window je díky své jednoduchosti a spolehlivosti hojně využívaná. Vyhledání nedosahuje vysoké rychlosti, ale v ostatních ohledech se jedná o dodnes nepřekonaný přístup. Pro lepší představu je uveden algoritmus v pseudokódu 1.

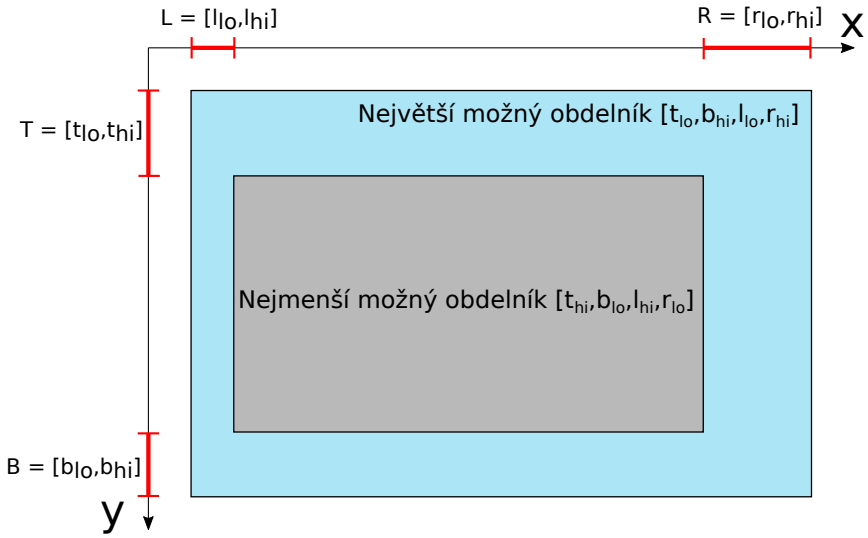
**Algorithm 1** Sliding window

Natrénování klasifikátoru na $n \times m$ obrázcích.
Nastav práh $t$ a krok $\Delta x$ a $\Delta y$ ve směru $x$ a $y$
Konstrukce Gaussovy pyramidy
**for** Pro každou vrstvu pyramidy **do**
    Aplikuj klasifikátor na každé okno $n \times m$ s krokem $\Delta x$ a $\Delta y$ na dané úrovni a získej odezvu $c$.
    **if** $c > t$ **then**
        Vlož odkaz na okno do seznamu $\mathcal{L}$ společně s hodnotou $c$
    **end if**
**end for**
**for** Pro každé okno $\mathcal{W} \in \mathcal{L}$ - vybíráme od největší hodnoty odezvy $c$ **do**
    Okna nalezená ve vyšší vrstvě pyramidy roztáhni na původní velikost
    **for** Pro okna $\mathcal{U} \neq \mathcal{W}$ **do**
        Vymaž okna, která se výrazně překrývají s $\mathcal{W}$
    **end for**
**end for**
$\mathcal{L}$ nyní obsahuje seznam detekovaných objektů

## 2.7 Effective subwindow search

Metoda Effective subwindow search dále jen ESS se zabývá efektivním vyhledáním objektu v obraze. V rozsáhlém množství možných kandidátních regionů (obrazových oken) se zaměřuje pouze na ty, které skutečně mohou obsahovat hledaný objekt. Namísto kompletního prohledání všech možností, je hledání zaměřeno na regiony s vysokým ohodnocením, které je získáno jako výstup funkce kvality regionu $f$. Regiony s nízkým skóre jsou ignorování, díky čemuž je zajištěna vyšší efektivita oproti sliding window 2.6.

Další odlišností oproti metodě sliding window je způsob notace výřezu obrazu. Zatímco Sliding window pracuje s oknem, které má přesně dané souřadnice v osách x,y - horní, spodní a levý, pravý okraj (t,b,l,r), tak ESS používá u každého rozměru dvě hodnoty (nízkou a vysokou) resp. *Low* a *High*, díky tomuto vymezení intervalů u každé hrany pracuje ESS s množinou oken (regionem) namísto jednoho okna. Výsledná reprezentace regionu je daná enticí $[T, B, L, R]$, kde $T = [t_{lo}, l_{hi}]$ atd. Pro lepší představu je uveden obrázek 2.8.




Obrázek 2.8: Reprezentace regionu $[T, B, L, R]$ v algoritmu ESS.

### Mezní funkce kvality

Pro ohodnocení okna využíváme funkci kvality $f$ - vyšší hodnota vetší pravděpodobnost výskytu hledaného objektu. ESS používá mezní funkci kvality $\hat{f}$, která udává maximální výstup funkce kvality $f$ (Rovnice 2.6) v rámci regionu (množiny oken). Dále je jedno okno značené $R$ a region oken $\mathcal{R}$. Mezní funkce kvality musí splňovat následující podmínky:

$$\hat{f}(\mathcal{R}) \geq \max_{R \in \mathcal{R}} f(R) \tag{2.6}$$

$$\hat{f}(\mathcal{R}) = f(R) \qquad \text{když okno R je jediné v množině } \mathcal{R} \tag{2.7}$$

**Algorithm 2** Effective Subwindow Search

**Require:** obrázek $I \in \mathbb{R}^{n \times m}$
**Require:** mezní funkce kvality $\hat{f}$
**Ensure:** $(t_{max}, b_{max}, l_{max}, r_{max}) = \arg\max_{R \subset I} f(R)$
  initialize P ▷ Prázdná prioritní fronta
  $[T, B, L, R] = [0, n] \times [0, n] \times [0, m] \times [0, m]$
  **repeat**
    split $[T, B, L, R] \rightarrow [T_1, B_1, L_1, R_1] \cup [T_2, B_2, L_2, R_2]$
    ▷ Nové regiony vložíme do prioritní fronty P spolu se skóre
    push $([T_1, B_1, L_1, R_1], \hat{f}([T_1, B_1, L_1, R_1]))$ into P
    push $([T_2, B_2, L_2, R_2], \hat{f}([T_2, B_2, L_2, R_2]))$ into P
    retrive top state $[T, B, L, R]$ from P
  **until** $[T, B, L, R]$ ▷ Až obsahuje pouze jeden obdelník
  $(t_{max}, b_{max}, l_{max}, r_{max}) = [T, B, L, R]$ ▷ Výsledná lokalizace

### Algoritmus

Základním kostrou ESS metody je algoritmus 2 uspořádaného prohledávání založený na branch-and-bound [8]. Algoritmus v každém cyklu rozdělí aktuální region na dva disjunktní regiony, které jsou po ohodnocení mezní funkcí kvality zařazeny zpět do prioritní fronty - jako priorita je považovaný právě výstup z mezní funkce kvality $\hat{f}(\mathcal{R})$. Samotné rozdělení regionu je provedeno v rozměru největšího intervalu jak je naznačeno na obrázku 2.9. Pro další průchod cyklem je z fronty vyjmut nejvíce prioritní region. Tímto způsobem je cyklus prováděn až do doby, kdy nejvíce prioritní region ve frontě obsahuje pouze jedno okno tzn. nelze dále dělit. V tom případě je nalezené okno globálním maximem jak znázorňuje rovnice 2.7 [7].

### Konstrukce funkce kvality

Funkce kvality vychází z BoW (Podkapitola 2.4) a SVM (Podkapitola 2.5) klasifikátoru se základním lineárním jádrem (Rovnice 2.8), kde vstupními daty jsou histogramy $h^i$ přes $i$ trénovacích dat a $\langle ., . \rangle$ značí skalární součin v $\mathbb{R}^K$ s histogramem $h$ aktuálně hodnoceného okna $R$. Dále $\alpha_i$ a $\beta$ jsou natrénované parametry SVM klasifikátoru. Rovnice 2.8 lze převést na rovnici 2.9 díky linearitě skalárního součinu a použití vztahu $w_j = \sum_i \alpha_i h^i_j$, který vytváří váhy $w_j$ pro každou třídu histogramu $j$.

$$f(R) = \beta + \sum_i \alpha_i \langle h, h^i \rangle \tag{2.8}$$

$$f(R) = \beta + \sum_{j=1}^{n} w_{c_j} \tag{2.9}$$

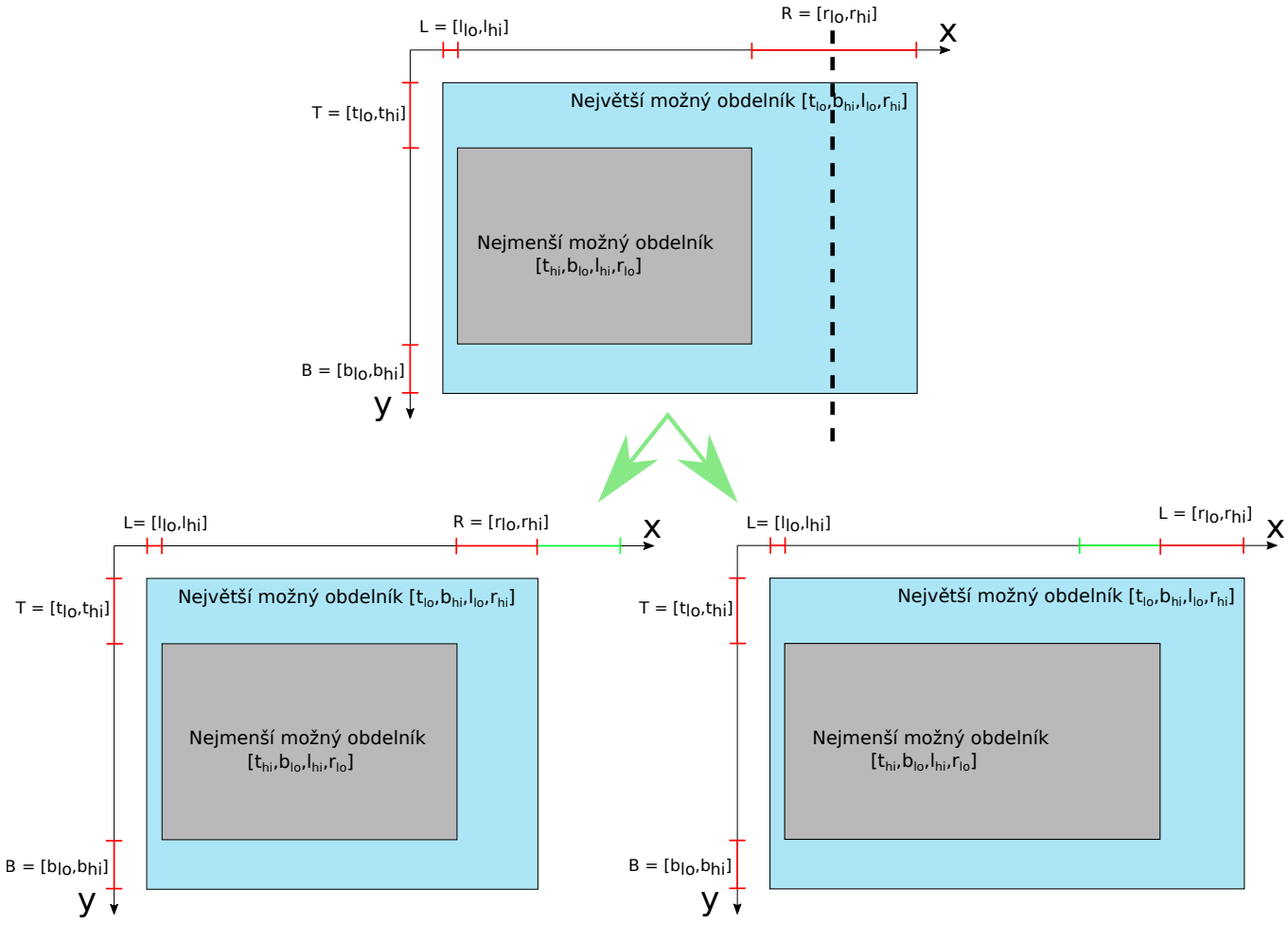
Druhá rovnice 2.9 reprezentuje příspěvek každého klíčového bodu z okna $R$ k celkové hodnotě funkce kvality. Zde je $c_j$ označení třídy do které patří detekovaný klíčový bod $x_j$. V případě použití s BoW je třídou myšleno vizuální slovo, ke kterému je klíčový bod $x_j$ přiřazen na základě podobnosti. Celkový počet klíčových bodů je $n$. Uvedený přístup nám umožňuje vyhodnocovat funkci kvality u oken, které jsou výřezy původního obrazu ($R \subset I$), díky tomu, že provádíme sumu pouze přes klíčové body nacházející se v dané podoblasti.

### Konstrukce mezní funkce kvality

Pokud máme definovanou funkci kvality (Rovnice 2.9), konstrukce mezní funkce kvality $\hat{f}$ už není složitá. Její definici je $\hat{f} = f^+ + f^-$, kde $f^+$ značí sumu přes klíčové body, které mají pozitivní váhu $w_{c_j}$ (Rovnice 2.9) a naopak $f^-$ je suma přes klíčové body s negativní váhou. Výsledná rovnice mezní funkce kvality má podobu:

$$\hat{f}(\mathcal{R}) = f^+(R_{max}) + f^-(R_{min}) \tag{2.10}$$

$R_{max}$ a $R_{min}$ jsou největší a nejmenší okna z regionu $\mathcal{R}$ (Obrázek 2.8). V konečném důsledku je možné z funkce kvality odstranit konstantu $\beta$, jelikož na výsledek porovnání dvou mezních funkcí různých regionů nemá žádný vliv.



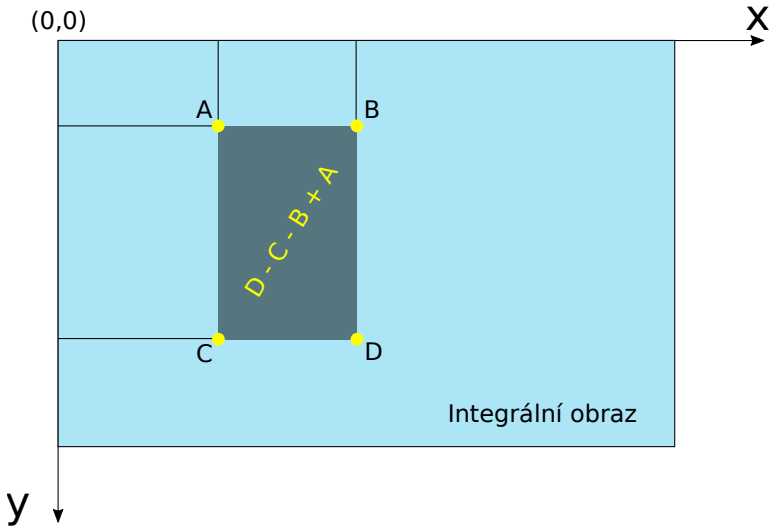

Obrázek 2.9: Dělení regionů podél největšího okraje.

### Vnitřní implementace mezní funkce kvality

Pro rychlý výpočet mezní funkce kvality, je používán integrální obraz. Vstupem pro jeho konstrukci jsou detekované pozice vizuálních slov včetně jejich váhy, která je získaná po natrénování SVM. Je vytvořena matice o velikosti vstupního obrazu, kde je na pozici danou pozicí detekce vizuálního slova vložena jeho váha. Z takto vytvořená matice $m$ je vypočten integrální obraz $I_{\sum}$ podle rovnice 2.11, kde $I_{\sum}(\mathbf{x})$ je pozice v integrálním obraze $\mathbf{x} = (x, y)$ [11].

$$I_{\sum}(\mathbf{x}) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} m(i, j) \tag{2.11}$$

Jinak řečeno integrální obraz na pozici $(x, y)$ obsahuje součet vah, které jsou v obdélníku definovaném počátkem souřadnic $(0, 0)$ a bodem $(x, y)$. Pokud je takto vytvořen integrální obraz, je možné jej použít pro výpočet vah detekovaných vizuálních slov v rámci jakékoliv oblasti v obraze, jak je znázorněno na obrázku 2.10.




Obrázek 2.10: Výpočet mezní funkce pro dané okno v obraze pomocí integrálního obrazu.

# Kapitola 3

# Návrh řešení

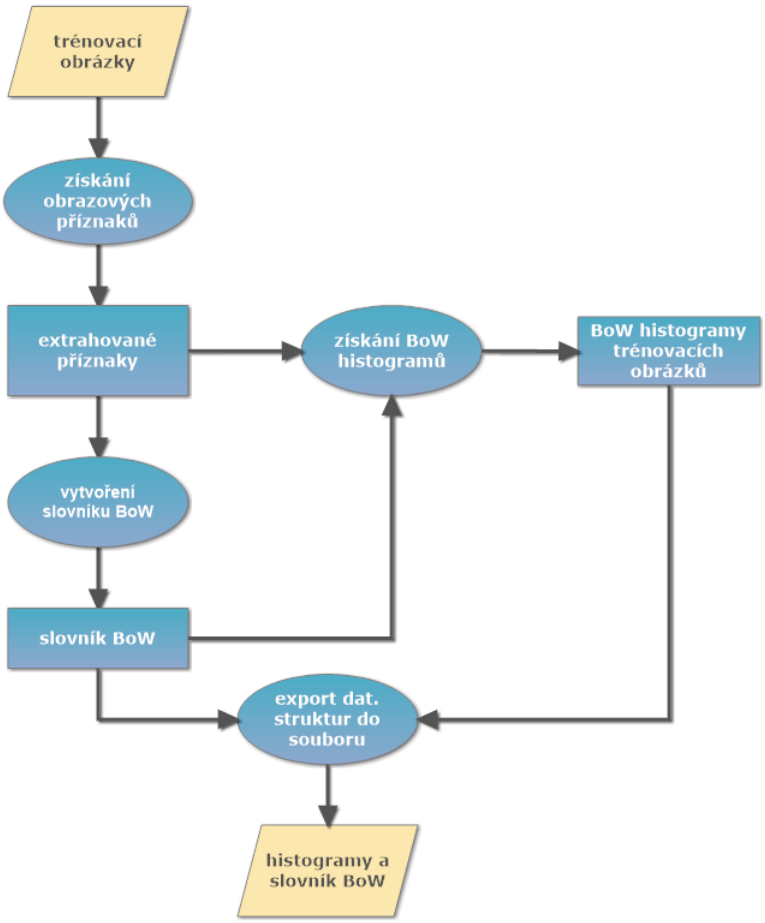
Hlavním cílem této práce je vytvořit aplikaci, schopnou detekovat a lokalizovat psa ve fotografiích, které jsou předloženy na vstupu aplikace. V předešlé kapitole 2 byl uveden popis vybraných technik a postupů počítačového vidění, které jsou zvažovány jako možné řešení práce. Tato kapitola obsahuje návrh řešení, za použití výběru z postupů uvedených dříve. Navržené řešení se opírá o tyto algoritmy: BoW (Podkapitola 2.4), K-means (Podkapitola 2.3) a ESS (Podkapitola 2.7). Kapitola se dále zaobírá rozdělením navrhovaného řešení na několik vzájemně provázaných částí. V poslední podkapitole 3.2 je popsáno získání trénovací a lokalizační sady fotografií.

## 3.1 Obecný návrh

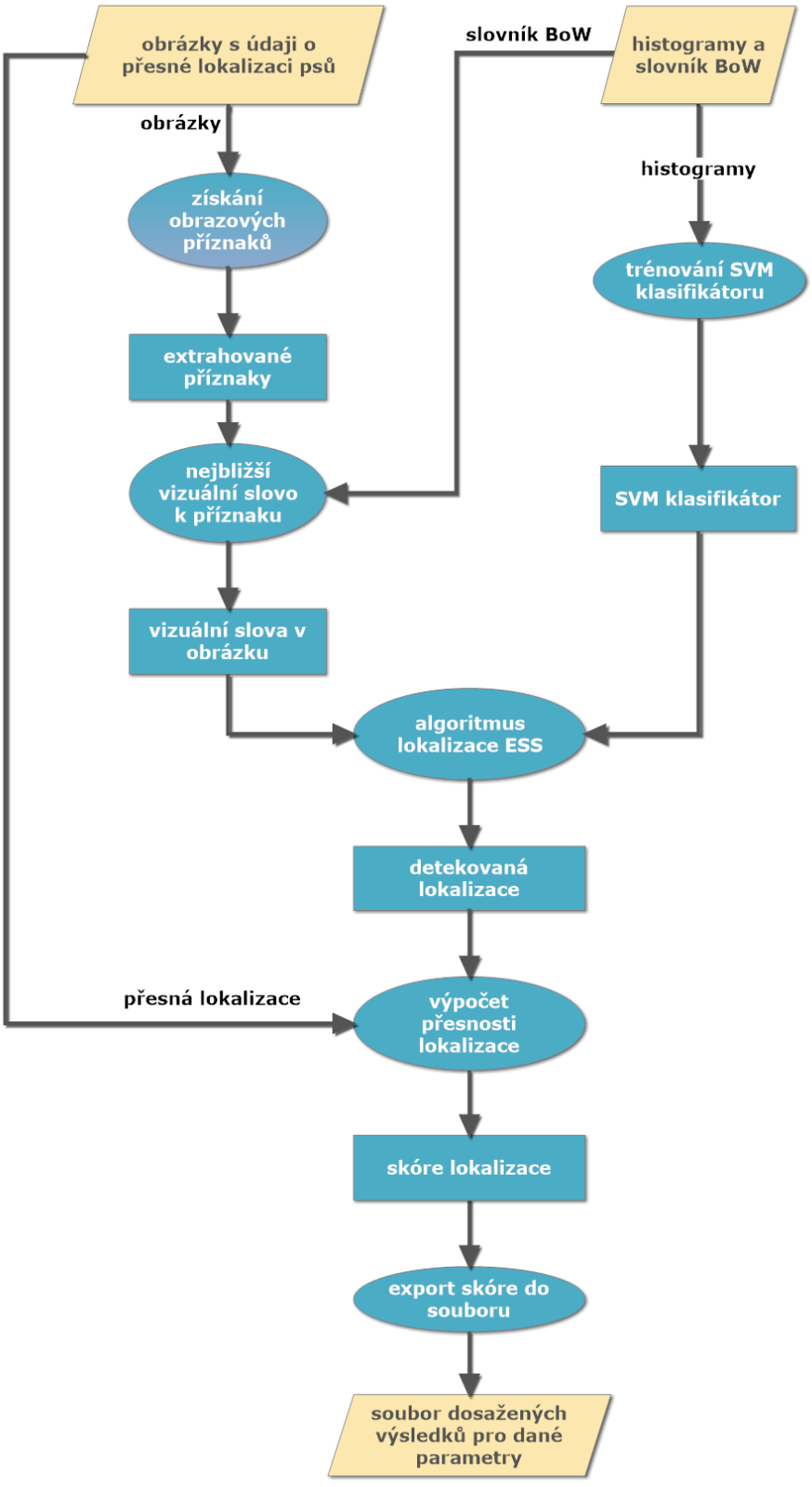
Činnost aplikace je rozdělena do několik částí (fází). Klíčové je rozdělení činnosti aplikace na část starající se o trénování a část lokalizační, která využívá natrénovaných modelů z předešlé fáze k vyhledání cílového objektu v obraze. Část trénovací se dále dělí na fázi trénování slovníku vizuálních slov (BoW) a fázi trénování lokalizace. Níže jsou uvedeny všechny fáze, jako výčet položek, v pořadí jak následují za sebou pro správnou činnost aplikace. U jednotlivých fází je uvedena jejich funkce spolu s očekávanými vstupy a výstupy:

- **Fáze trénování BoW** (Obrázek 3.1)
  - *Vstup*
    - Trénovací obrázky - positivní, negativní
  - *Funkce*
    - Vytvoření BoW modelu
    - Ke každému trénovacímu obrázku je vytvořen histogram vizuálních slov
  - *Výstup*
    - Slovník BoW
    - Histogramy vizuálních slov

- **Fáze trénování lokalizace** (Obrázek 3.2)
  - *Vstup*
    - Slovník BoW
    - Histogramy vizuálních slov
    - Obrázky určené k lokalizaci psa
    - Souřadnice výskytů psů v naší datové sadě
  - *Funkce*
    - Natrénování SVM klasifikátoru
    - Extrahování obrazových příznaků ze vstupních obrázků
    - Přiřazení obrazového příznaku k nejbližšímu vizuálnímu slovu z BoW
    - Provedení ESS lokalizačního algoritmu
    - Výpočet přesnosti lokalizace
  - *Výstup*
    - Zápis úspěšnosti lokalizace spolu s aktuálním nastavením parametrů do souboru výsledků




Obrázek 3.1: Diagram fáze trénování BoW - slovníku vizuálních slov.




Obrázek 3.2: Diagram fáze trénování lokalizace.

- **Fáze lokalizace**
  - *Vstup*
    - Obrázky určené k lokalizaci psa
  - *Funkce*
    - Použití nejpřesnějšího modelu z předešlé fáze
    - Vyhledá kde v obraze s největší pravděpodobností je pes
  - *Výstup*
    - Vytvořena kopie vstupního obrázku se zaznačenou detekcí

Výsledek práce má podobu jedné konzolové aplikace. Fází je v tomto případě myšleno spuštění konzolové aplikace s danými parametry. Výčet parametrů a jejich význam je popsán v další kapitole 4.

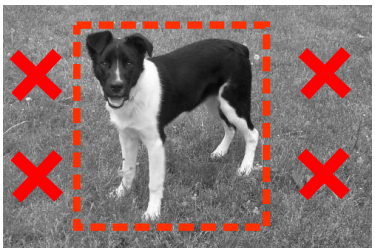
## 3.2 Datová sada

Pro vytvoření slovníku vizuálních slov je potřeba patřičně velká datová sada obrázků. V případě, že detekujeme jeden objekt - v našem případě psa, je potřeba vytvořit dvě sady obrázků. Jedna sada obrázků se označuje jako pozitivní a obsahuje instance hledaného objektu. Druhá sada obrázků se nazývá negativní a obsahuje obrázky, které vyobrazují cokoliv jiného než hledaný objekt. Je vhodné zkonstruovat negativní datovou sadu tak, aby obsahovala obrázky okolí, ve kterém je typicky zasazen hledaný objekt.

Hlavním zdrojem obrázků psů pro tuto práci je webový projekt *kaggle* vyhlašující klasifikační soutěže. U jedné konkrétní klasifikační soutěže [1], ve které je vyhlášena soutěž o nejlepší výsledek klasifikace mezi kočkami a psy, je volně dostupná datová sada čítající cca 12500 fotografií psů. Tyto obrázky nejsou použity přímo, ale jsou ještě dále upraveny. Posloupnost provedených úprav je zobrazena níže (Obrázek 3.6).

Obrázek 3.3: Originální podoba.

Obrázek 3.4: Převedení na stupně šedi.



Obrázek 3.5: Oříznutí nadbytečného okolí.

Obrázek 3.6: Posloupnost úprav pozitivních trénovacích fotografií

Negativní sada obrázků se skládá z fotografií přírody, městského prostředí a lidí. Většina fotografií je stažena z webového projektu *imageNet*[1], který zpřístupňuje rozsáhlou databází fotografií v různých kategorií.

Poslední datová sada důležitá pro řešení práce je sada fotografií psů, kde není provedeno ořezání nadbytečného okolí, namísto toho je zapsáno umístění psa ve fotografii do souboru s lokacemi. Tato datová sada je uplatněna ve fázi trénování lokalizace, která vypočtenou lokaci porovnává s opravdovou, aby bylo možné dále ladit parametry modelu. Tabulka 3.1 pro přehlednost zobrazuje všechny použité datové sady obrázků spolu s umístěním v rámci složky `Images` a počtem obrázků.

| **Datová sada** | **Počet obrázků** | **Umístění** |
|---|---|---|
| Pozitivní | 861 | `Images\Train\Dog` |
| Negativní | 1542 | `Images\Train\Other` |
| Lokalizační | 100 | `Images\TestESS` |

Tabulka 3.1: Tabulka všech použitých datových sad včetně počtu obrázků.

### Výběr obrazových příznaků

Jelikož se vyskytuje hledaný objekt v datové sadě v různých velikostech a je různě natočen, byly vybrány jako obrazové příznaky SIFT a SURF, které odolávají obrazovým transformacím.

[1] http://image-net.org/

# Kapitola 4

# Implementace

Obsahem kapitoly je popis implementace, která vychází z návrhu uvedeného v předešlé kapitole 3. Výsledkem implementace je konzolová aplikace schopná detekce a lokalizace psa ve fotografii. Jednou ze základních součástí aplikace je knihovna počítačového vidění OpenCV. Za účelem seznámení se s možnostmi OpenCV byl použit jazyk Python, který umožňuje rychlí vývoj, avšak za cenu menší běhové rychlosti. Z důvodu potřeby zvýšit rychlost běhu aplikace byl následně zvolen jazyk C++, který kompiluje aplikaci do nativního kódu díky čemuž je o poznání rychlejší než za použití Pythonu.

## 4.1 Použité vývojové prostředí

Jelikož je výsledná konzolová aplikace cílena pro Windows a je napsána v C++, bylo zvoleno jako vývojové prostředí Microsoft Visual Studio Community[1], které je zdarma pro samostatné vývojáře, opensource projekty, vědecký výzkum, vzdělávání a malé profesionální týmy. Mezi další možné alternativy vývojových prostředí umožňující vývoj v C++ na Windows patří např. Code::Blocks[2] a NetBeans[3]. Mezi hlavní důvody výběru Microsoft Visual Studio Community patří kvalitní debugger, který v přehledné formě umožňuje přístup k aktuálním hodnotám proměnných po spuštění aplikace. Mezi další často využívanou funkci patří IntelliSense, které napomáhá správnému volání hlaviček funkcí a zápisu proměnných.

## 4.2 OpenCV

OpenCV (Open source Computer Vision) je multiplatformní knihovna zaměřená na počítačové vidění a zpracování obrazu v reálném čase. OpenCV byla vyvinuta firmou Intel, jako iniciativa se záměrem zrychlit pokrok v počítačovém vidění a dopomoci při vývoji aplikací zpracovávající obraz. Knihovna je dostupná pod licencí BSD, což umožňuje její využití pro komerční i nekomerční účely. Knihovna samotná je napsaná v optimalizovaném C/C++ a při svém běhu dokáže využívat vícejádrové procesory. OpenCV disponuje několika rozhraními, díky kterým je umožněn vývoj v jazycích C, C++,Python, Java a to na několika platformách Windows, Linux, Mac OS, iOS a Android. Knihovna obsahuje více než 2500

[1] https://www.visualstudio.com/products/visual-studio-community-vs
[2] http://www.codeblocks.org/
[3] https://netbeans.org/

algoritmů, které kromě počítačového vidění a zpracování obrazu zpřístupňují také algoritmy strojového učení [6].

## 4.3 Rozhraní aplikace

Výsledná aplikace je spustitelný soubor `dogLocalization.exe` určený pro spuštění z příkazové řádky. Nastavením parametrů při spuštění definuje chování aplikace. Níže je uveden příklad spuštění v příkazové řádce systému Windows:

```
dogLocalization.exe [Parametry]
```

### Parametry

V předešlé kapitole byly definovány tři fáze činnosti programu a to fáze trénování BoW, fáze trénování lokalizace a fáze lokalizace. V této kapitole je fáze chápána jako **stav programu** a je nastavena jedním z uvedených parametrů:

- Fáze trénování BoW nastavena parametrem `-trainBOW`
- Fáze trénování lokalizace nastavena parametrem `-trainLocalization`
- Fáze lokalizace `-localization` nebo `-localize`

Po nastavení parametru stavu je potřeba nastavit další parametry, které jsou specifické pro jednotlivé stavy. Výjimku tvoří poslední stav `-localize`, který pro svou činnost nevyžaduje žádné další parametry.

**Stav `trainBOW`**

| Parametr | Význam |
|---|---|
| `-w [N1] [N2]` | `Počet vizuálních slov modelu BoW, které se mají vytvořit. N1 počet pro třídu dog a N2 pro other` |
| `-SIFT` | `Detekce obrazových příznaků pomocí SIFT algoritmu.` |
| `-SURF` | `Detekce obrazových příznaků pomocí SURF algoritmu.` |
| `-SURFe` | `Detekce obrazových příznaků pomocí SURF algoritmu v rozšířené verze s deskriptorem velikosti 128b` |

Tabulka 4.1: Tabulka parametrů kombinující se s parametrem `-trainBOW`.

Parametry stavu `trainBOW` (Tabulka 4.1) jsou důležité pro specifikaci BoW modelu, který je výstupem tohoto spuštění programu. Parametr specifikující algoritmus detekce obrazových příznaků je vybrán pouze jeden. Další tabulka 4.2 zobrazuje parametry pro stav `trainLocalization`. V tomto stavu je načten z disku BoW model, vybraný parametry `-w` a příznakovým algoritmem. Po načtení BoW modelu, je natrénovaný SVM klasifikátor s penalizační konstantou danou parametrem `c` a je provedena lokalizace na testovací lokalizační sadě obrázků.

**Stav `trainLocalization`**

| Parametr | Význam |
|---|---|
| `-w [N1] [N2]` | `Specifikace parametrů modelu BoW, který bude načten ze souboru` |
| `-SIFT` | `Detekce obrazových příznaků pomocí SIFT algoritmu.` |
| `-SURF` | `Detekce obrazových příznaků pomocí SURF algoritmu.` |
| `-SURFe` | `Detekce obrazových příznaků pomocí SURF algoritmu v rozšířené verze s deskriptorem velikosti 128b` |
| `-s nebo -show` | `Zakreslit detekovanou lokalizaci do obrázku a uložit pro náhled.` |
| `-c` | `Penalizační parametr pro vytvoření SVM klasifikátoru` |

Tabulka 4.2: Tabulka parametrů kombinující se s parametrem `-trainBOW`.

Zpracování parametrů je zajištěno voláním funkce `getParams`, která vrací zpracované parametry ve struktuře `tParams`.

## 4.4 Přehled důležitých částí implementace

### Detekce a extrakce příznaků

K detekci a extrakci příznaků jsou využity prostředky z knihovny OpenCV. Zkompilovaná knihovna OpenCV 3.1, která je dostupná ke stažení, nabízí k použití pouze některé obrazové příznaky Pro účel této práce jsou potřeba obrazové příznaky SURF a SIFT, které je nutné začlenit do zdrojových kódů OpenCV a poté provést kompilaci celé knihovny vlastnoručně.

K uložení obrázku do paměti se používá funkce `imread()`, která obrázek daný parametrem uloží do matice (Třída `Mat`). Matice je jedním ze základních datových typů OpenCV spolu s vectorem (`cv:Vec`), bodem (`cv:Point`) a dalšími.

Detektor příznaků je vytvořen, ihned po tom co jsou zpracovány parametry příkazové řádky. Detektor reprezentuje třída `FeatureDetector`, která má metodu `detect()` pro získání pozic klíčových bodů ze vstupního obrázku. Poté co jsou detekovány klíčové body, je využita další funkce detektoru a to funkce `compute()`, která klíčové body převede na deskriptory. Výsledné deskriptory jsou vektory pevné délky, uloženy jako řádky do výsledné matice deskriptorů.

### Vytvoření Bag of Words reprezentace a trénovací histogramy

Pro vytvoření BoW je využita třída z OpenCV `BOWKMeansTrainer`. Při vytváření instance této třídy, je nutné zadat počet vizuálních slov, kolik si jich přejeme vytvořit a metodu vytvoření vizuálních slov - v našem případě `KMEANS_PP_CENTERS`. Objekt třídy `BOWKMeansTrainer` obsahuje metodu `add()`, která přebírá matici deskriptorů a přidává ji k deskriptorům vloženým dříve. Poté, co vložíme deskriptory ze všech trénovacích obrázků do objekt BoW, je možné zavolat metodu `cluster()`, která za pomocí metody K-means vytvoří $K$ středů v prostoru deskriptorů tzn. zadaný počet vizuálních slov. Volání metody `cluster()` vrátilo matici, kde každý řádek reprezentuje jedno vizuální slovo. Jelikož je jedním z parametrů aplikace počet vizuálních slov pro jednu a druhou klasifikační třídu, je

provedeno natrénování BoW dvakrát - pro každou třídu zvlášť. Nakonec jsou dvě matice deskriptorů spojeny a tím je vytvořen kompletní slovník vizuálních slov.

Poté, co je vytvořený slovník vizuálních slov, je možné znovu projít všechny trénovací obrázky a vytvořit pro každý z nich histogram výskytu vizuálních slov. Vytvoření takových histogramů zajišťuje další třída OpenCV `BOWImgDescriptorExtractor`, která při vytváření instance vyžaduje stejný objekt detektoru `FeatureDetector`, jaký byl použit při vytváření deskriptorů pro BoW. Metodou `setVocabulary()` třídy `BOWImgDescriptorExtractor` je nastaven slovník vizuální slov získaný dříve. Nyní už tedy nastává další průchod přes trénovací obrázky, kdy je pro každý získán histogram pomocí metody `compute()`. Všechny vypočítané histogramy jsou ukládány jako řádky do jedné matice. Současně je provedeno vložení čísla - označující třídu trénovacího obrázku, do matice `labels`. Číslo v matici `labels` slouží k rozpoznání, ke které třídě patří histogram z daného řádku matice histogramů.

Všechny tři matice - matice vizuálních slov, histogramů a matice `labels`, jsou uloženy do souboru. Název souboru je tvořen hodnotami parametrů s jakými byl vytvořen BoW. Celý popsaný proces je vykonán, je-li aplikace ve stavu `trainBOW`. Vytvořený soubor je vstupem do aplikace spuštěné ve stavu `trainLocalization` popsané v další sekci.

### SVM klasifikátor

Aplikace spuštěná ve stavu `trainBOW` nám vytvoří všechny matice dat, které jsou nezbytné pro natrénování SVM klasifikátoru. Knihovna OpenCV obsahuje také algoritmy strojového učení, mezi které patří i SVM. Algoritmus SVM je dostupný pod stejnojmenným názvem třídy `SVM`. Po vytvoření objektu této třídy, je nastaven typ SVM na `SVM::C_SVC` a jádro SVM na lineární `SVM::LINEAR`. Jako Poslední je nastavena penalizační konstanta předaná jako parametr `-c`. Pro samotné natrénování je volaná metoda `train()` třídy `SVM` s maticí trénovacích histogramů a maticí `labels`. Obě matice jsou načtené ze souboru.

### Transformace dat pro ESS

Transformace dat pro ESS následuje poté, co je provedeno natrénování SVM klasifikátoru. Na objektu třídy `SVM` není provedena standardní metoda `predict()` pro klasifikaci, ale celý objekt je vstupem do vlastní funkce `getSvmWeightVector()`, která z natrénovaného SVM vyextrahuje váhový vektor, udávající váhy jednotlivých vizuálních slov z našeho BoW.

Nyní nastává cyklus přes všechny obrázky v lokalizační sadě. V cyklu je pro každý obrázek volána funkce `getDataForESS()`, která získá všechny potřebné data z obrázku pro provedení ESS lokalizace. Tělo funkce detekuje obrazové příznaky a následně je volána metoda `compute()` třídy `BOWImgDescriptorExtractor`, její výstup je dále upraven, za účelem získání nejbližšího vizuálního slova ke každému obrazovému příznaku.

Pozice detekovaných obrazových příznaků spolu s přiřazeným klíčovým bodem a váhovým vektorem SVM klasifikátoru, jsou vstupme do funkce `pyramid_search()`, která provede ESS a vrací souřadnice konečné lokalizace ve struktuře `Box`.

# Kapitola 5

# Testování

V této kapitole je uveden postup testování aplikace a zaznamenání výsledků, aby následně mohlo dojít k vybrání parametrů, při kterých je dosaženo nejlepších výsledků.

## 5.1 Meření přesnosti lokalizace

Pro měření přesnosti lokalizace (Obrázek 5.1) je použita rovnice 5.1, kde $A_1$ je načtená správná oblast lokace psa, $A_2$ je detekovaná oblast, která je výstupem aktuálního modelu a $s$ je výsledná přesnost, jejíž obor hodnoty je $\langle 0, 1 \rangle$. Výsledek $s = 0$ znamená, že nebyl vytvořen žádný průsečík s cílovým objektem, naproti tomu $s = 1$ značí absolutní shodu detekce s hledaným objektem. Někde mezi těmito extrémy leží práh $t$, který když není překročen hodnotou $s$ (Rovnice 5.3) je daná detekce považovaná za neúspěšnou (*false positive*), jinak je detekce považována za úspěšnou (*true positive*). Nastavení prahu je na hodnotu $t = 0.45$, aby i částečné detekce byly brány jako úspěšné (Obrázek 5.2).

$$s = \frac{A_1 \cap A_2}{A_1 \cup A_2} \tag{5.1}$$

$$s \geq t \quad \rightarrow \quad \text{TP true positive} \tag{5.2}$$

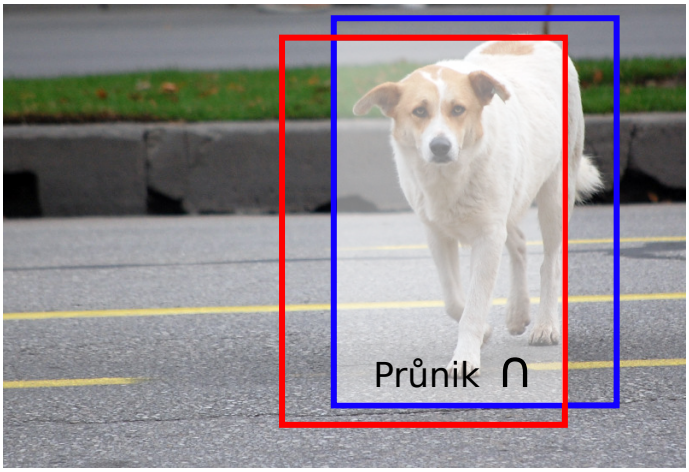
$$s < t \quad \rightarrow \quad \text{FP false positive} \tag{5.3}$$

### Celková přesnost

Celkové přesnost lokalizace na všech 100 obrázcích lokalizační datové sady je dána jako součet přesností lokalizací z každého obrázku. Celková přesnost je počítána při spuštění aplikace ve stavu `trainLocalization`. Po vypočtení je zaznamenána do CSV souboru `results.csv` spolu s parametry BoW a SVM. Pro lepší představu je níže uvedena tabulka 5.1, která znázorňuje obsah výstupního CSV souboru.

| VW Dogs | VW Other | Feature | SVM C | Score | Precision |
|---|---|---|---|---|---|
| 250 | 250 | SIFT | 350 | 33 | 38.437 |
| 300 | 300 | SIFT | 400 | 31 | 36.572 |
| 150 | 150 | SURF | 50 | 30 | 36.790 |

Tabulka 5.1: Tabulka znázorňující obsah CSV souboru result.csv.




Obrázek 5.1: Modře je znázorněná správná lokalizace a červeně je výstupní lokalizace.

Sloupec `score` udává počet detekcí, které měly přesnost lokalizace vetší než $0,5$. První 4 sloupce jsou parametry modelu, které zadáváme při spouštění aplikace. Poslední sloupec `precision` je celková přesnost lokalizace.

## 5.2 Vyhodnocení výsledků

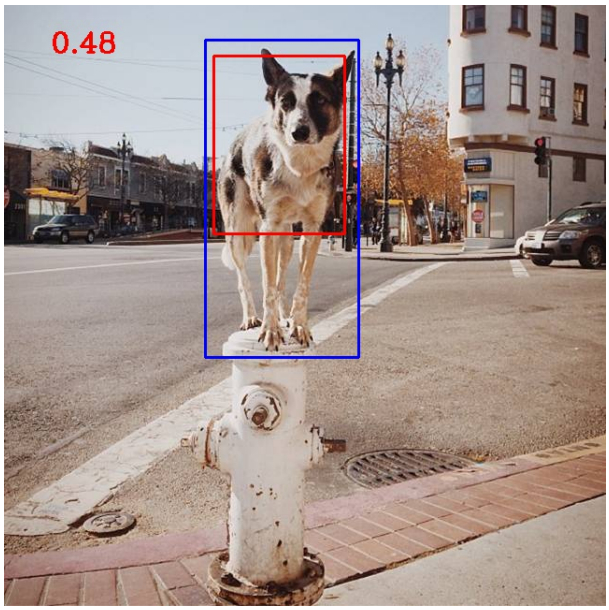
Vyhodnocení výsledků se skládá ze zpracování souboru `result.csv` a nalezení největší dosahované přesnosti lokalizace. Mezi hledané parametry patří:

- Typ obrazových příznaků (SIFT, SURF 64b/128b)
- Počet vizuálních slov na jednu třídu pro BoW
- Penalizační parametr SVM klasifikátoru

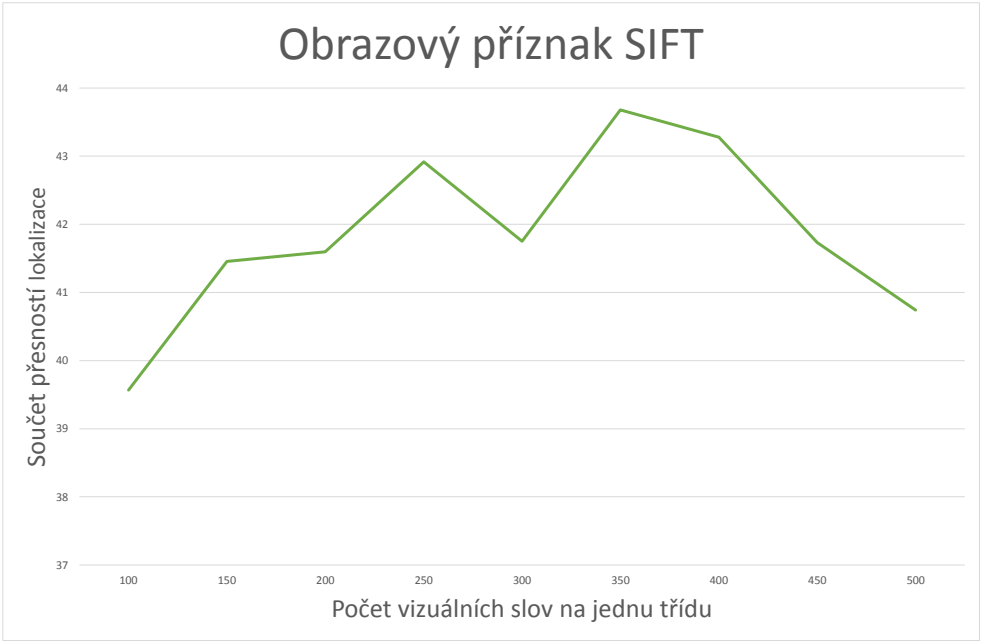
Z obsahu CSV souboru jsou vytvořeny tři grafy. Každý graf reprezentuje jeden typ obrazového příznaku. Na vodorovné ose jsou BoW modely s rozdílným počtem vizuálních slov na jednu třídu. Jako hodnota je vynesena největší celková přesnost, které bylo dosaženo u daného BoW při testování s různým penalizačního parametrem SVM. Obrázek 5.3 zobrazuje graf přesnosti lokalizace pro SIFT příznaky a obrázek 5.4 pro SURF příznaky - ve dvou verzích s 64b a 128b deskriptorem.

### Nejlepších výsledků bylo dosaženo s parametry:

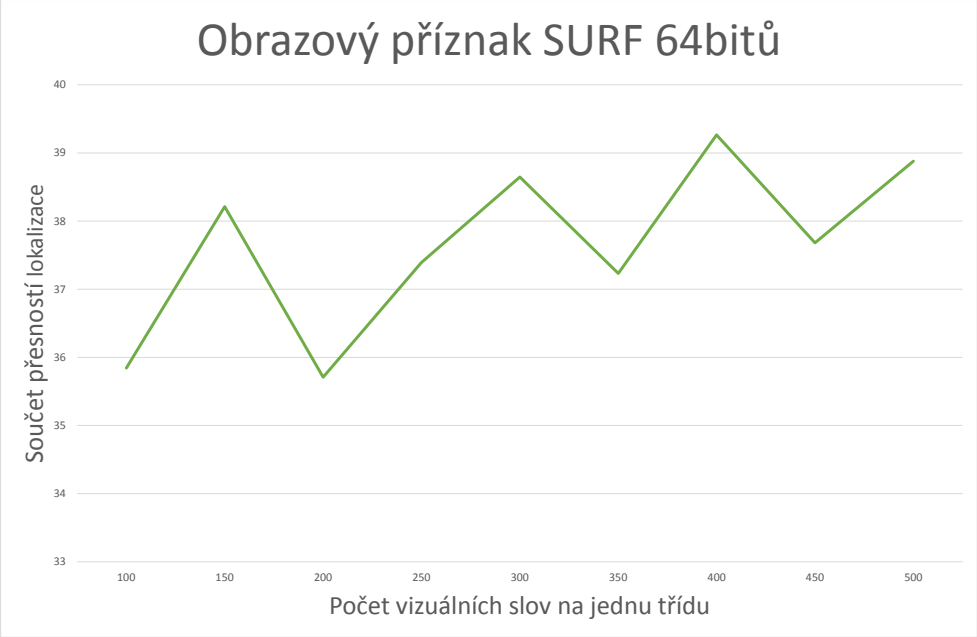
| | |
|---|---|
| Počet vizuálních slov na třídu | 350 |
| Obrazový příznak | SIFT |
| Penalizační konstanta SVM | 25 |
| Práh přesnosti lokalizace | 0,45 |
| **Průměrná rychlost lokalizace** | **0,679 ms** |
| **Celková úspěšnost** | **47 %** |

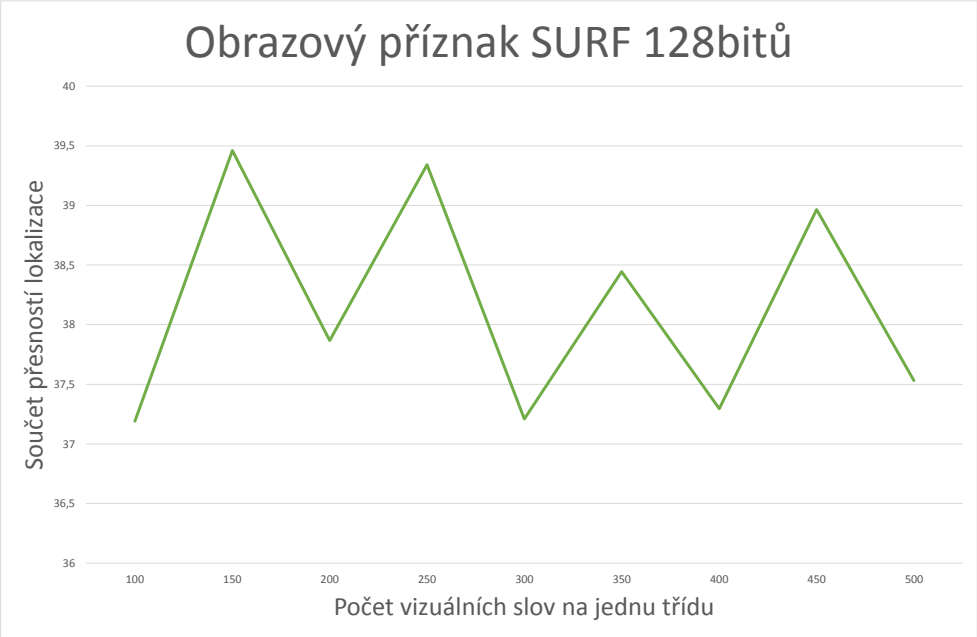



Obrázek 5.2: Příklad výsledné lokalizace naší aplikací (červený rám) přesnost lokalizace 0.48




Obrázek 5.3: Trénování modelů za použití SIFT obrazových příznaků.




Obrázek 5.4: Trénování modelů za použití SURF obrazových příznaků

# Kapitola 6

# Závěr

Cílem této práce bylo seznámit se s principy detekce a lokalizace objektů v obraze a následně uplatnit nabyté znalosti, při návrhu a implementaci řešení zadaného tématu. Tématem práce byla *Rychlá detekce přírodních objektů* a nabízela k výběru ze spousty možných objektů k rozpoznávání. Nakonec byla jako cíl detekce a lokalizace zvolena zvířata - konkrétněji psi.

Po nastudování možných způsobů řešení, byla zvolena metoda Bag of Words, vytvářející histogramy vizuálních slov, které jsou dále použity s SVM algoritmem k natrénování klasifikátoru. Dalším cílem práce bylo použití detekčního a lokalizačního algoritmu, který by svou rychlostí předčil často využívanou metodu Sliding Window (metoda klouzavého okénka). Z tohoto důvodu byl pro detekci a lokalizaci použit algoritmus Effective subwindow search, který při hledání objektu v obraze zaměřuje pozornost na ty regiony, kde je více pravděpodobný výskyt psa, díky tomu bylo dosaženo lepšího průchodu obrazem, než v případě Sliding Window a tedy i vyšší rychlosti.

Současně byla vytvořena datová sada fotografií psů o velikosti 861 obrázků a další sada 100 fotografií, s údajem o pozici psa, určená k trénování přesnosti lokalizace. Z trénovací sady fotografií je patrné, že hledaný pes se může nacházet v různém rozlišení a různě natočený, proto byly zvoleny jako obrazové příznaky algoritmy SIFT a SURF, která disponují velkou robustností vůči obrazovým transformacím.

Výsledné řešení má podobu konzolové aplikace. Průměrná rychlost detekce a lokalizace je **0,679 ms**, což je uspokojivé. Maximální dosažená úspěšnost lokalizace psa v obraze je **47 %**, zde je prostor pro další zlepšování.

Do budoucna je vhodné najít další obrazové příznaky, které napomohou k vytvoření přesnějšího klasifikátoru. Dále by ke zlepšení výsledků přispělo rozšíření datové sady fotografií.

# Literatura


[1] Kaggle - Dogs vs. Cats. [online], 2015-9-25 [cit. 2016-5-10].
URL https://www.kaggle.com/c/dogs-vs-cats

[2] OpenCV - Introduction to SIFT. [online], 2015-12-18 [cit. 2016-5-9].
URL http://docs.opencv.org/3.1.0/da/df5/tutorial_py_sift_intro.html

[3] OpenCV - Introduction to Support Vector Machines. [online], [cit. 2016-5-9].
URL http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html

[4] Forsyth, D. A.; Ponce, J.: *Computer Vision: A Modern Approach*. Pearson, druhé vydání, 2011, ISBN 01-360-8592-X, 519-520 s.

[5] Harrington, P.: *Machine learning in action*. Shelter Island, N.Y.: Manning Publications Co., c2012, ISBN 1617290181, 101-127 s.

[6] Laganiére, R.: *OpenCV 2 computer vision application programming cookbook*. Brimmingham: Packt Publishing, 2011, ISBN 9781849513241.

[7] Lampert, C.; Blaschko, M.; Hofmann, T.: Efficient Subwindow Search: A Branch and Bound Framework for Object Localization. *IEEE Transactions on Software Engineering*, ročník 31, 2009: s. 2129–2142, doi:10.1109/TPAMI.2009.144.

[8] Lawler, E.; Wood, D.: Branch-and-bound methods: A survey. *Operations Research*, ročník 14, č. 4, 1966: s. 699–719.

[9] Prateek, J.: *Support Vector Machines*. [online], 2012-8-24 [cit. 2016-5-3].
URL https://prateekvjoshi.com/2012/08/24/support-vector-machines/

[10] Rey Otero, I.; Delbracio, M.: Anatomy of the SIFT Method. *Image Processing On Line*, ročník 4, 2014: s. 370–396.

[11] Wikipedia: Summed area table. [online], [cit. 2016-5-9].
URL https://en.wikipedia.org/w/index.php?title=Summed_area_table

# Přílohy

# Seznam příloh

# Příloha A

# Obsah DVD

- **Bin/** Obsahuje spustitelný program včetně `README.txt`
- **Dll/** OpenCV knihovny
- **Images/** Datové sady fotografií
- **Latex/** Zdrojový kód bakalářské práce v LaTeXu
- **Lib/** Instalaci OpenCV 3.1 a VS C++
- **Pdf/** Text bakalářské práce v pdf
- **Source/** Zdrojové soubory, projekt v Microsoft Visual Studio Community
- **Tests/** Soubory vygenerované při testování programu