VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

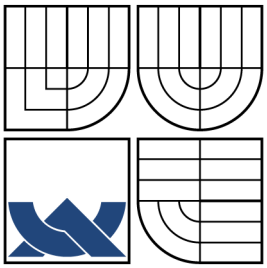
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

VYHLEDÁVÁNÍ OBJEKTŮ V OBRAZE NA ZÁKLADĚ PŘEDLOHY

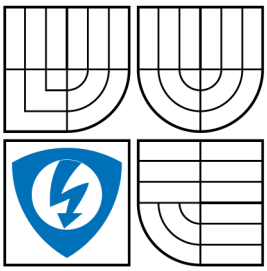
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. PAVEL NOVÁK
AUTHOR

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND
COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

VYHLEDÁVÁNÍ OBJEKTŮ V OBRAZE NA ZÁKLADĚ PŘEDLOHY

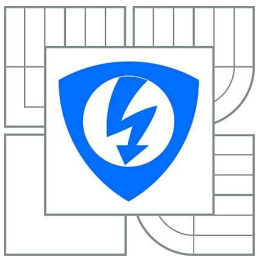
IMAGE OBJECT DETECTION USING TEMPLATE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. PAVEL NOVÁK
AUTHOR

VEDOUCÍ PRÁCE — Ing. RADIM BURGET, Ph.D.
SUPERVISOR

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Pavel Novák	***ID:***	115247
Ročník:	2	***Akademický rok:***	2013/2014

NÁZEV TÉMATU:

Vyhledávání objektů v obraze na základě předlohy

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s metodami vyhledávání objektů v obraze a na základě experimentů je srovnejte. Navrhněte a statisticky ověřte metodu, která na základě omezené trénovací množiny vyhledá tento objekt v jiných obrázcích. Statisticky zhodnoťte dosaženou přesnost a na vhodných obrázcích demonstrujte funkčnost či případné nedostatky.

DOPORUČENÁ LITERATURA:

[1] Anuj Mohan , Constantine Papageorgiou , Tomaso Poggio, Example-Based Object Detection in Images by Components, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2001
[2] P. Moreels, P. Perona A Probabilistic Cascade of Detectors for Individual Object Recognition (14 pages pdf) European Conference on Computer Vision (ECCV) Marseille, France 2008

Termín zadání:	10.2.2014	***Termín odevzdání:***	30.5.2014

Vedoucí práce: Ing. Radim Burget, Ph.D.
Konzultanti diplomové práce:

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Tato práce se zabývá detekcí objektů v obraze na základě předlohy. Hlavním přínosem práce je nová metoda extrakce příznaků histogramu orientovaných gradientů používající sadu komparátorů pro extrakci dat. V práci jsou popsány použité metody komparace a extrakce. Hlavní část je věnována především metodě histogramu orientovaných gradientů, ze které vycházíme. V práci je užita malá sada trénovacích obrazů (celkem 100) ověřená křížovou validací, následně ověřená na reálných scénách. Dosažená úspěšnost křížové validace je až 98 % pro SVM algoritmus.

KLÍČOVÁ SLOVA

Detekce objektu, počítačové vidění, histogram orientovaných gradientů, porovnání obrazu, sada komparátorů, extrakce dat, křížová validace

ABSTRACT

This Thesis is focused to Image Object Detection using Template. Main Benefit of this Work is a new Method for sympthoms extraction from Histogram of Oriented Gradients using set of Comparators. In this used Work Methods of Image comparing and Sympthoms extraction are described. Main Part is given to Histogram of Oriented Gradients Method. We came out from this Method. In this Work is used small training Data Set (100 pcs.) verified by X-Validation, followed by tests on real Sceneries. Achieved success Rate using X-Validation is 98 % for SVM Algorithm.

KEYWORDS

Object Detection, Computer Vision, Histogram of Oriented Gradients, Image Comparing, Set of Comparators, Data extraction, X-Validation

NOVÁK, Pavel *Vyhledávání objektů v obraze na základě předlohy*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2014. 61 s. Vedoucí práce byl Ing. Radim Burget, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Vyhledávání objektů v obraze na základě předlohy“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

(podpis autora)

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Radimu Burgetovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Dále bych rád poděkoval panu Ing. Janu Maškovi za odborné rady, kterými nemalým dílem přispěl k vypracování této práce.

Brno ....

(podpis autora)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

PODĚKOVÁNÍ

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .

(podpis autora)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM VLOŽENÉHO KÓDU

SEZNAM PŘÍLOH

ÚVOD

Základní otázkou je, co je to vlastně rozpoznávání objektu v obraze. Pokud bychom uvažovali např. o obrázku hrajících si dětí, kde lidé snadno rozliší postavy, poznávají rodiče, děti, předměty, jako jsou například míče, lopatka a kbelíček, nákladní Tatra, se kterou chlapec vozí písek. Lidé i snadno charakterizují obličej, nacházejí různé rysy jako tvar obličeje, vrásky, tvary obočí, uší, úst a nosu, dokáží také rozpoznávat barvy, bílá zeď, hnědá hlína, zelená tráva.

Prakticky vše, co nás zajímá, se dá označit za objekt zájmu v obraze, ať už se jedná o jakoukoliv scénu abstraktní, konkrétní, či detail obličeje. Jedná se tedy o nalezení konkrétního objektu v obraze, určení o koho nebo o co se jedná (v případě osob resp. věcí) a vymezení hranic tohoto objektu.

Detekce objektu v obraze nám začíná být čím dál blíž. Ať se jedná o rozpoznání čísel a písmen na SPZ automobilu při vjezdu do garáží, nebo vyhledávání obličeje v databázi hledaných zločinců. Můžeme sem zařadit i varování před podezřelými osobami na letišti. Nebo nám mnohem bližší praxe – přihlášení se do našeho operačního systému pomocí funkce detekce obličeje.

K těmto procesům detekce, stejně jako člověk používá mozek a své znalosti (paměť chceme-li), používá počítač umělou inteligenci a neuronové sítě. Jedná se o systém algoritmů, který se na základě vstupních dat (např. dat vyňatých z předloh obsahujících postavu a neobsahujících postavu) dokáže naučit detekovat postavu s určitou pravděpodobností. Hlavním problémem současných metod je, že těchto učících dat je zapotřebí tisíce, aby byla schopnost detekce na přijatelné úrovni.

V této práci byla vytvořena aplikace na detekci objektů v obraze na základě předlohy. Práce vychází z metody histogramu orientovaných gradientů, která byla prezentovaná v [7]. Hlavní část práce se věnuje navrženým komparátorům, kde je porovnán testovaný histogram s histogramem vzorového objektu, a tím získán jeden příznak pro natrénovaný model. Při tvorbě aplikace bylo postupováno různými směry, například jedním z nich byl použití kompletní databáze MIT [6], nebo použití 100 vzorů pro natrénování modelu. Následně byly použité metody ověřeny testem pomocí křížové validace [25], nebo i nasazením aplikace na reálné scény.

Hlavním přínosem této práce je nová metoda na extrakci příznaků histogramu orientovaných gradientů za použití sady komparátorů. Pomocí naší metody a porovnání výsledků pomocí křížové validace se dosáhlo přesnosti detekce 98% za použití algoritmu SVM. V porovnání úspěšnosti detekce se současnými metodami je prezentovaná metoda na přibližně stejné úrovni (nejlepší metody dosahují 98% – necelých 100% úspěšnosti), ale naše metoda dosáhla této úspěšnosti na velmi malé sadě učících vzorů (100 obrázků – 50 pozitivních, 50 negativních) oproti 1 – 2 tisícům učících vzorů pro detekci v ostatních pracích.

Tato práce je členěna následovně. V kapitole 1 je nejprve popsáno, co to je počítačové vidění, jsou zde uvedeny základy porovnávání obrazu a představena metoda histogramu orientovaných gradientů, která je základem této práce. V kapitole 2 je pak přiblížena vytvořená aplikace a jsou zde také blíže popisovány vytvořené metody. Tím je myšlen rozbor postupu tvoření aplikace a stručný přehled vytvořených tříd a metod. V následující kapitole 3 jsou stručně přiblíženy dosažené výsledky práce.

1 ZÁKLADY POROVNÁNÍ OBRAZU

Nejprve je důležité přiblížit, co to vlastně je počítačové vidění (dále jen CV). Nutno zdůraznit, že počítač nevidí obrazy, ale pouze data, a je tedy zprvu nutné si představit obraz jako soubor čísel. Ty je možno číselně vyjádřit pomocí různých modelů, např. v odstínech šedi (to může být 2 rozměrné pole o hodnotách buňky 0-255), nebo RGB (pole 2x3 rozměrů: 1. rozměr - červená, 2. rozměr zelená, 3. rozměr - modrá). Z těchto dat je již možné už určovat rozdíly.

Jako jednoduchý příklad je uveden obrázek mravence 1.1, který má ve vlastnostech uvedenu velikost 1545x1024 pixelů, a užívá barevný prostor sRGB[1]. V jednoduchosti to znamená, že každý jeho pixel je vyjádřen třemi složkami: R (červená), G (zelená), B (modrá).

Obr. 1.1: Mravenec

Je tak možné vyjádřit část obrázku 1.1 znázorněného žlutým rámem. Pokud by byl obrázek převeden na odstíny šedi, hodnoty pixelů uvnitř části označené žlutým obdélníkem by vypadaly jako v tabulce 1.1.

[1]Omezený standard RGB využívaný převážně na webovou prezentaci [26]

Tab. 1.1: Úrovně šedi

xy	0	1	2	3	4	5	6	7	8	9
0	59	85	89	68	56	67	48	51	49	74
1	36	52	64	49	40	60	63	48	52	54
2	60	30	59	84	57	54	44	60	50	61
3	58	40	84	72	52	73	48	33	69	54
4	53	47	31	61	27	88	54	67	43	45
5	56	79	71	59	59	60	59	41	62	53
6	68	88	80	53	58	42	61	53	54	38
7	76	68	76	61	67	49	33	38	41	46
8	71	57	59	85	75	49	65	38	41	37
9	82	85	54	41	50	58	59	45	40	34

1.1 Metody porovnání obrazu

Pro jednoduchost je předpokládán obrázek v odstínech šedi, jak tomu bylo v předchozí kapitole. Jelikož se data sestávají z tabulky čísel, první možností, která se sama nabízí pro hledání rozdílu mezi obrazy, je samotný matematický rozdíl, a poté klasifikace výsledku. Například je tak možné získat:

- Střední kvadratická odchylka (rozptyl) – Mean Square Error (MSE)[15]:

$$MSE = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} [x(m,n) - x'(m,n)]^2 \tag{1.1}$$

  kde M je počet pixelů v ose x, N je počet pixelů v ose y, m je pořadí pixelu v ose x a n pořadí pixelu v ose y, x a x' jsou obrazová data k porovnání.

- Střední absolutní chyba – Mean absolute error (MAE)[15]:

$$MAE = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} |x(m,n) - x'(m,n)| \tag{1.2}$$

- Směrodatná odchylka – Root Mean Square Error (RMSE)[16]:

$$RMSE = \sqrt{MSE} \tag{1.3}$$

- Variance [10]:

$$s^2 = \frac{1}{n-1} \sum_{i=0}^{n} |(x_i - \overline{x})^2| \tag{1.4}$$

1.2 Histogram orientovaných gradientů

Histogram orientovaných gradientů (dále jen HoG) je metoda pro nalezení hran a vytvoření tabulky pro jednotlivé buňky obrázku (o velikostech x, y pixelů). Rozhodujícími parametry pro správnou detekci objektu v obraze může být počet a rozměr buněk, ale také počet košů. Tím je myšleno pole o velikosti $0 - n$, kde n je počet košů, který dělí rozmezí úhlu $0\,° - 180\,°$. Kupříkladu aplikace popisovaná v kapitole 2 počítá s rozmezím 6 košů, což znamená, že do nultého prvku budou spadat úhly $0\,° - 29\,°$.

Jak lze vidět v obrázku 1.2, tak nejprve jsou určeny koše a rozměry buněk, viz 1.2b. Pak je provedena derivace obrázku podle os (X i Y). To se v obrazových datech nejlépe provede pomocí konvoluce [21]:

$$f \approx [-1, 0, 1] * f \tag{1.5}$$

Kde vektor $[-1, 0, 1]$ je tzv. konvoluční jádro. Tím získáme velikost 1.2d:

$$Magnitude(x, y) = f'_x(x, y)^2 + f'_y(x, y)^2 \tag{1.6}$$

a směr úhlu 1.2e pro každý pixel:

$$Orientation(x, y) = \arctan \frac{f'_y(x, y)}{f'_x(x, y)} \tag{1.7}$$

kde x a y jsou pozice pixelu v obrázku. Zbývá rozřadit vypočtené úhly do patřičných košů každé buňky a sečíst jejich velikost. Zjištění, do kterého koše buňky spadá daný pixel, se dá vypočítat pomocí vzorce:

$$bin = Orientation(x, y)/(180/n)) \tag{1.8}$$

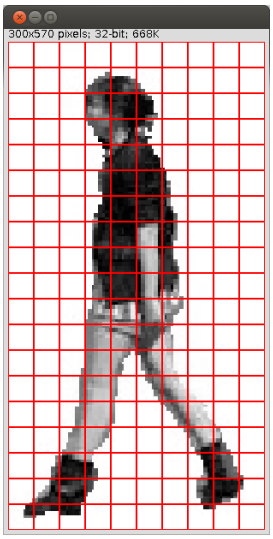
kde n je zvolený počet košů. Tím je možné se dostat k vyobrazení výsledného histogramu orientovaných gradientů 1.2f, kde pro každou buňku jsou vykresleny všechny koše.

HoG využívá mnoho prací, kterými byla tato práce inspirována. Například práce [7], ve které je využito 16 x 16 bloků s 50% překrytím, kde blok je 2 x 2 buňky (8 x 8 pixelů). Je zde použitá kvantizace do 9 košů a vykazovaná přesnost 89 %.

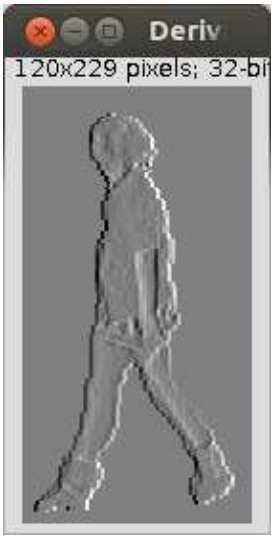
V práci [22] je popsána detekce člověka po částech. Nejprve jsou detekovány v obraze objekty jako trup, hlava, ruce, nohy, a pak je zjišťováno, zda jsou ve správné geometrické kompozici. Teprve posléze je určeno, zda je objekt osobou či nikoli. To však vyžaduje hodně objektů a hodně učení. Práce má přibližně 98% úspěšnost.

Ve zdroji [24] je využit HoG na detekci obličeje. Detekce obličeje je z pohledu počítačového vidění brána jako snazší, jelikož detekovaný obličej se nevyskytuje v tolika tvarech (osoba je detekována v jakémkoli postoji) a barvách (mnoho barev

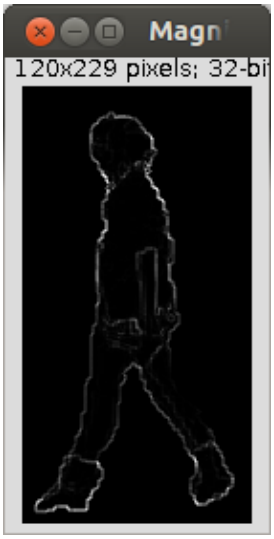
(a) Vzorek



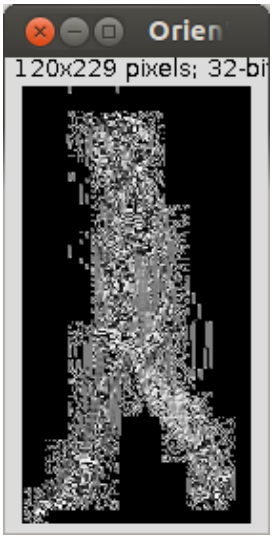
(b) Stanovení mřížky



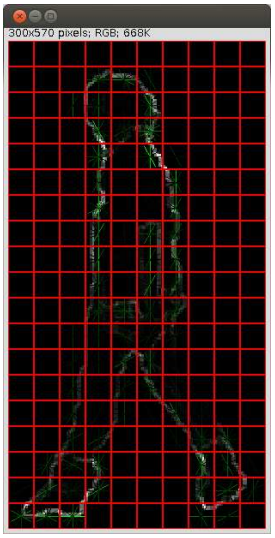
(c) Derivace vzorku



(d) Velikosti úhlů



(e) Orientace úhlů



(f) Výsledný HOG

Obr. 1.2: Postup výpočtu histogramu

oblečení). Detekce obličeje probíhá pouze z čelního pohledu na rozdíl od detekce osob. V této práci je vytvořena větší škála bloků s rozdílnou velikostí pro detekci hran, na kterých je naučen trénovací algoritmus AdaBoost[13]. Po vybrání nejlepších bloků pasujících na detekci byla vytvořena kaskáda algoritmů, kdy každý z algoritmu mohl ukončit kaskádu. To urychlilo výpočet natolik, že výsledky se blížily reálnému času. Systém mohl propustit od 5 do 30 snímků za vteřinu. Jako výsledek bylo uvedeno, že účinnost je přibližně stejná jako ve výchozích pracích, ale prezentovaná metoda je přibližně 70 krát rychlejší.

V práci [19] je popsáno, že s každým košem je zacházeno jako s vlastností a každý koš je použit jako základní stavební element pro množinu vlastností. Je zde použita substituce *haar* metody za HoG a tím je zachována rychlost. Je dosaženo větší přesnosti (cca 99 %) než [30], ale za stávající rychlosti.

Práce [8] je zaměřena na detekci objektů v datovém toku. Tím je myšlen filmový záběr. Autoři zde použili deskriptory kombinované s HoG. Snažili se zachytit pohyb končetin a zabránit zachycení nežádoucích pohybů. Použili kombinaci se vzhledovým deskriptorem pro omezení chybových alarmů. Dosáhli dobrých výsledků s použitím statických deskriptorů. Výsledná ztrátovost byla okolo 8 %.

Práce [3] je zaměřena na detekci obrazu pomocí 2 kamerových stereosystémů: 2 vzdálené infračervené kamery a 2 kamery na denní světlo. Pro detekci chodce je scéna rozdělena na několik menších oblastí, kde je vypočítán HoG, ve kterém jsou následně vyhledány tvary. Detekované tvary jsou pak podrobeny analýze, v níž je rozhodnuto, zda jde o chodce, či nikoli. V práci je prezentovaná účinnost 91 % pro detekci chodce v nočních snímcích a 80 % pro detekci denních snímků.

Práce [27] je zaměřena na kompletní systémem detekce chodce založený na metodě HoG aplikovatelný na infračervené snímky. V práci je nejprve rozebrána jednosnímková detekce chodce pomocí histogramu a SVM klasifikátoru. Jako trénovací sada jsou použity množiny 10, 100 a 1000 vzorků z množiny 2200 pozitivních a stejného množství negativních vzorků. Prezentovaná úspěšnost pro sadu 1000 trénovacích vzorků je přibližně 99 %.

Práce [1] se zabývá detekcí objektů pomocí rozpoznání podobností tvarů. V první části se práce zabývá shodou 2 bodů mezi objekty. V následující části se zaměřuje na využití korespondujících bodů k vyrovnávací transformaci. Výsledky práce prezentují na siluetách, známkách, ručně psaných číslicích nebo COIL [5] databázi.

Práce [12], [11] jsou zaměřeny na vyhledávání shody obrázkových struktur v obrazech. Obrázková struktura je kolekce částí poskládaná v deformovatelné konfiguraci. Konfigurace je reprezentována spojením mezi těmito částmi. Obrázkovou strukturu je možno si představit jako paži deformovatelnou ve spojení, kterým je v tomto případě kloub.

Práce [4] je založená na hledání vhodných kandidátů na části těl. Po nalezení

vhodných kandidátů jsou tvořeny sestavy ve vhodné kompozici. Je zde uváděna vysoká úroveň špatné pozitivní detekce. Práce je tedy aplikovatelná pouze v některých případech.

Práce [14] je založena na tvarové detekci aplikovatelné pro palubní zařízení užívající distanční transformaci. V ní je použitá hierarchie vzorů k získání různých objektových tvarů. Prezentovaná účinnost je 75–85 %.

Práce [23] se zabývá detekcí lidí ve statických obrazech založené na vlnovém vzoru (Haarově vlnce [21]), ve které je definován tvar objektu na podmínkách podmnožiny vlnkových koeficientů obrazu. Je zde dokázáno, že invariance a výpočetní efektivita dělají z vlnkového vzoru účinný nástroj s 69,6–81,7 % účinností detekce.

Práce [2] je zaměřena na detekci chodců pro vozidla za použití infračerveného vidění upevněného k vozidlu s detekcí od 7 do 43,5 m. U této metody je finální validační proces založen na detekci lidských tvarů. Je zde prezentována nízká úroveň falešně pozitivní detekce – 0,2 falešně pozitivní na jednu detekovanou scénu.

Práce [9] se zabývá detekcí obličeje a je rozdělena do 3 částí. V první části je zaměřena na kompenzaci chyb při detekci obličeje způsobené absorpcí, pozicí světla nebo změnou světelnosti. Ve druhé části je zaměřena na fúzi HoG deskriptorů různých velikostí umožňujících zachytit důležité struktury pro rozpoznání obličeje. V třetí části je prezentována důležitost poskytnutí dimenzionální redukce k odstranění šumu a snaha udělat proces méně náchylný na přeučení. Dále je zde uvedena schopnost získat významný nárůst (až 13 %) ve výkonu při rozpoznání na standardní databázi FERRET [28].

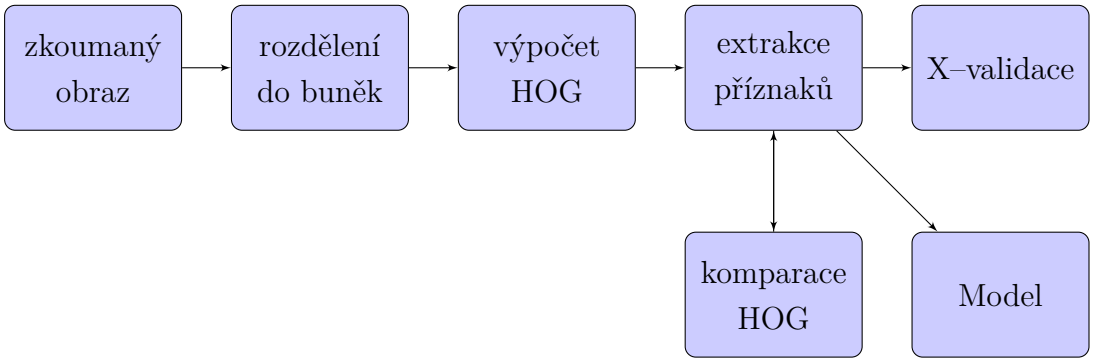
2 ROZBOR VYTVOŘENÉ APLIKACE

V této kapitole je uveden popis funkce aplikace. V prvé řadě je to výpočet histogramu a následná extrakce dat z histogramu. V druhé řadě pak detekce osob ve scéně na základě vzoru.

Zjednodušený postup extrakce dat je možné vidět na obr. 2.1. Tím je zvolení velikosti buněk obrázku a jeho následné rozdělení do těchto zvolených buněk. Dále následuje vypočtení HoG. Některé metody použité v této aplikaci pak využívají přímo parametry HoG, některé před extrakcí ještě porovnají testovaný HoG s předlohou. Po extrakci příznaků jsou pak data předána k dalšímu zpracování, ať už ke křížové validaci pro získání procentuální úspěšnosti zvolených metod, nebo detektoru scény, který zpracovává detekovanou scénu.

Zjednodušený postup detekce scény je možné vidět na obr. 2.2. Zkoumaná scéna je postupně procházena detekčním oknem. Pro každý vzorek detekčního okna jsou extrahovány příznaky, které jsou následně předány naučenému modelu k ohodnocení. Pozitivně označené vzorky jsou pak zobrazeny uživateli.

V prvé části této kapitoly jsou uvedeny použité knihovny, v druhé myšlenkový postup vytvoření tříd, ve třetí podrobný popis metod extrakce, a ve zbylých třech je popis procesu křížové validace, popis detekce scény a popis složky s ukázkami.

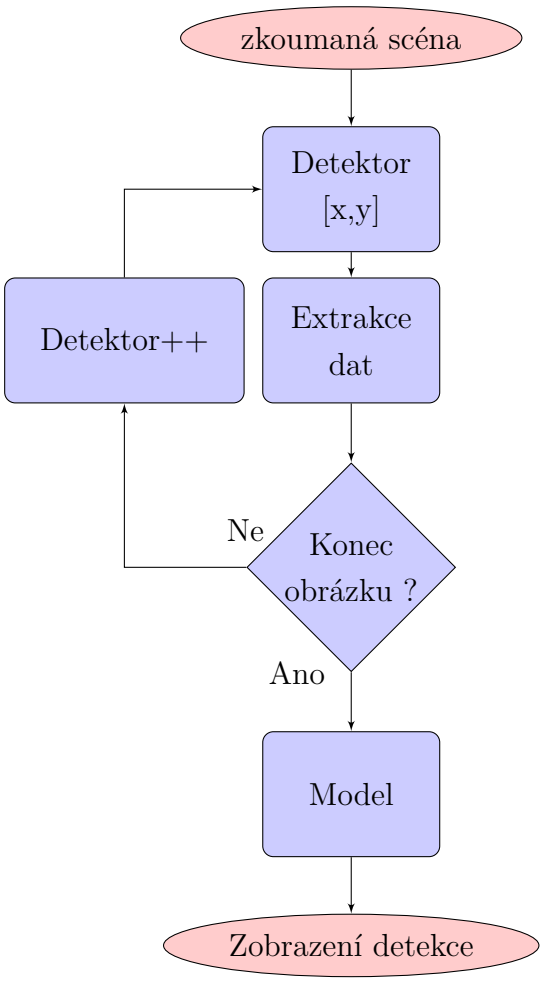


Obr. 2.1: Postup extrakce dat

2.1 Popis použitých knihoven

Jako nejdůležitější je knihovna **ImageJ** [17], která obsahuje nástroje na zpracování obrazu pro vývoj appletů, servletů nebo aplikací:

- je OpenSource [29] nástroj psaný v jazyce Java a lze ho tedy spustit v různých operačních systémech



Obr. 2.2: Postup detekce scény

- světově nejrychlejší program psaný čistě v jazyce Java určený na zpracování obrazu
- filtrace orazu až $2048 * 2048$ pixelů v 0,1 vteřině. To je 40 milionů pixelů za vteřinu.
- bohatá uživatelská komunita – více jak 1700 uživatelů a vývojářů po celém světě udržující korespondenci
- užití maker (více jak 300 dostupných), k tomu vázaného recorderu maker a debuggeru maker
- rozšiřitelnost pomocí pluginů a vestavěného text editoru a java compileru

Jako další byly použity knihovny z balíku **RapidMiner** [25], který bude rozebrán později v sekci 2.4, a **java.io** [18] a to hlavně:

BufferedWriter – Vypíše text jako proud znaků. Ukládá znaky, pole a pole znaků do vyrovnávací paměti. Může být použit spolu s třídou *FileWriter* pro zapsání toku znaků do souboru.

FileWriter – Třída pro pohodlné zapisování výstupního toku do souboru.

File – Třída použitelná pro abstraktní reprezentaci souboru nebo cesty v kódu.

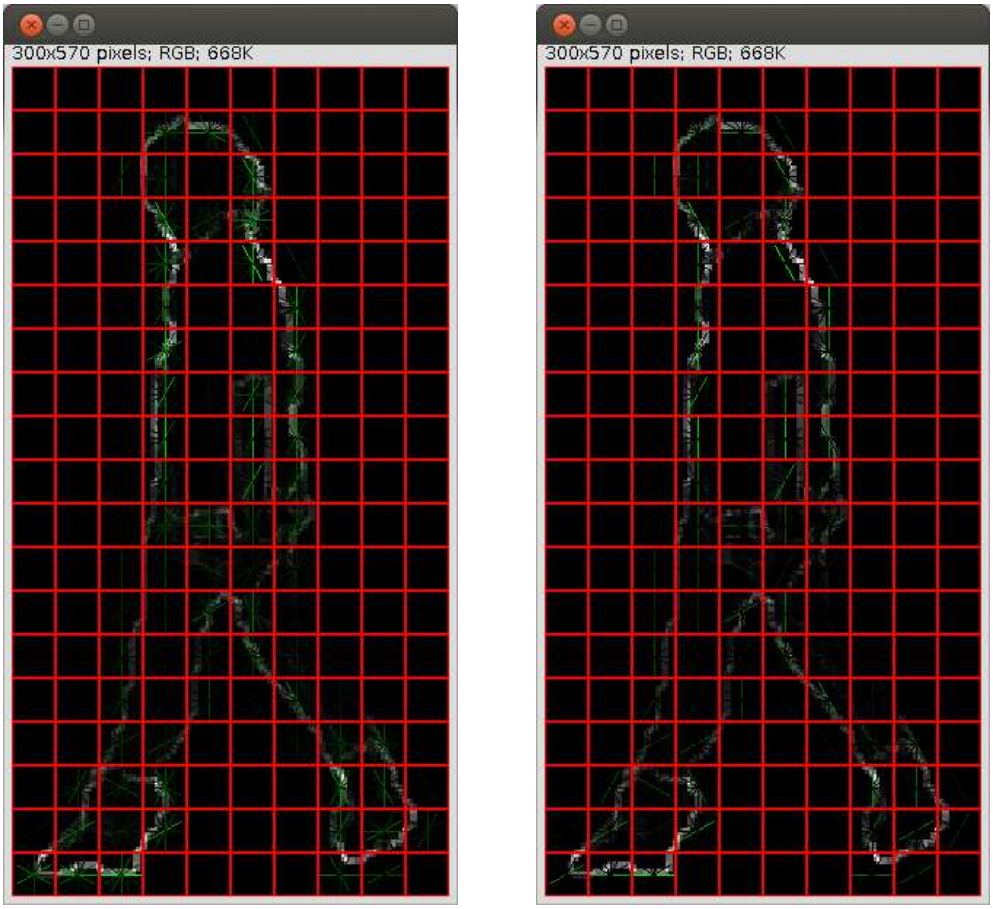
IOException – Třída signalizující výjimky operací se soubory.

2.2 Struktura balíku

V této práci je použita metoda HoG. V tom je zahrnuta určitá množina výpočtů, která proto byla ponechána v jedné třídě v kořenu balíku *cz.vutbr.feec.novak68/* **HOG.java**. Základní operace s třídou **HOG.java** je možné vidět v příkladě **HOGExample.java** v balíku **cz.vutbr.feec.novak68/** ve složce **example/**. Zde jsou vytvořeny metody na zobrazení jednotlivých fází výpočtu HoG.

V HoG mohou být po výpočtu obsaženy všechny koše. To by mohlo být vhodné, ale také by to mohlo působit rušivě, proto byly do balíku přidány filtry. Pro přehlednost jsou filtry obsaženy v balíku *cz.vutbr.feec.novak68.hogfilters*. Na obrázku 2.3a je vidět čerstvě vytvořený HoG. Lze si všimnout, že v jedné buňce může být obsaženo více (někdy i všech) košů, proto byl zaveden filtr **StrengthElementFilter** pro vyfiltrování pouze nejsilnějších košů z buňky. Výsledek může být viděn na obrázku 2.3b. Pro jednoduché využití všech i budoucích filtrů byl navržen model s rozhraním HOGFilter.

V této fázi by se dalo říct, že se HoG nachází ve finálním stádiu. Je tedy nutné ho nějakým způsobem porovnat s trénovacím vzorem. Pro tyto účely byl vytvořen balík *cz.vutbr.feec.novak68.hogcomparators/*. Jako první lze vycházet z kapitoly 1.1 a vytvořit tak třídu porovnávající dva HoG na základě RMSE každého koše.



(a) HoG bez filtru (b) HoG s filtrem

Obr. 2.3: Rozdíl mezi HoG s filtrem a bez

Jako výsledek je použit aritmetický průměr všech košů. Takto byla vytvořena třída *SimpleHOGComparator*.

Další metodou porovnání může být počet košů rovnoběžných s osou X (*HorizontalElementComparator*), s osou Y (*VerticalelementComparator*), porovnání všech košů (*AllElementComparator*), porovnání pouze košů nevertikální a nehorizontální polohy (*OtherElementComparator*). Všechny tyto komparátory porovnávající polohu prvku implementují rozhraní *ElementBaseComparator*.

Za předpokladu, že většina osob je ve stojící nebo ležící poloze (vertikální nebo horizontální), lze hledat i počet symetrických prvků v osách X a Y. V ose X je to *SimpleXSymetryComparator*, v ose Y je to *SimpleYSymetryComparator*. Je možné také spočítat součet čtverců těchto symetrií – *SimpleSymetryComparator*. Ve všech komparátorech symetrie je implementováno rozhraní *SimpleBaseComparator*. Dále je možné postupovat například vytvořením třídy, ve které je porovnán počet nejsilnějších košů každé buňky – *SimpleStrengthComparator*. Ve všech výše popsaných komparátorech je implementováno základní rozhraní *HOGComparator*.

Další, co je možné udělat, je extrakce dat pro **RapidMiner**. Zde jsou dva základní pohledy: extrakce dat z HoG nebo použití komparátorů. Je možné například z každého HoG extrahovat celkovou velikost košů (tzn. výslednou sumu příspěvků každé buňky pro každý koš). To je zahrnuto v *SimpleHogbinExtractor*, ale také je možné extrahovat velikost každého koše pro každou buňku (*SimpleHogcellExtractor*.

Jelikož je nejlepší získat různé kombinace komparátorů vs. extraktorů pro všechny testovací obrázky, jsou všechny instance tvořeny ve třídě *SimpleHogPropertyExtractor*, ve které je obsažena metoda vracející data, jak pro souběžně běžící proces aplikace **RapidMiner**, tak i zápis dat do souboru. Pro možnost budoucího rozšíření je zde vytvořeno rozhraní *HogExtractor* a třída obsahující základní parametry pro extrakci HoG – *HOGbaseExtractor*.

V této fázi je v aplikaci obsaženo vše potřebné pro snadnou extrakci dat do souboru a možné další zpracování v jiných aplikacích. To by ale byl pro detekci poněkud zdlouhavý proces, proto je knihovna **RapidMiner** implementována do aplikace. Jelikož je potřeba proces inicializovat, spouštět a předávat mu data, byla vytvořena třída *myProcess* v balíku *cz.vutbr.feec.novak68.rapidminer/*, ve které je výše uvedené umožněno.

Protože je získávání dat ze scény je složitější proces, je vhodné zde vytvořit uživatelské rozhraní, ve kterém je skryta složitost kroků vedoucích k získání dat. Těmi kroky jsou například rozdělení scény do malých oken podle zadaných koordinátů, nebo vytvoření tabulky pro **RapidMiner**. Proto je zde vytvořena třída *ImageFacade* v balíku *cz.vutbr.feec.novak68.imageProcessor/*.

Jelikož ve třídě *ImageFacade* jsou použity pro orientaci ve scéně určité koordináty, mezi kterými jsou zahrnuty pozice detekčního okna, jeho střed, nebo i šířka a výška, a v neposlední řadě i poměr velikosti detekčního okna k velikosti vzorového obrázku, je zde vytvořena třída *Coordinate* v balíku *cz.vutbr.feec.novak68.tools/* pro snazší předávání parametrů mezi objekty.

Nyní je vhodné, aby byly výše jmenované třídy skryty. Za předpokladu, že bude známa přibližná velikost detekovaných objektů, krok, se kterým bude detekční okno posunuto, a dále i některé vlastnosti detekce, je pro tyto případy vytvořena třída *DetectorImpl* ve které je implementováno rozhraní *Detector* v balíku *cz.vutbr.feec.novak68.detector/*.

2.3 Popis tříd a metod

V této sekci bude podrobněji popsáno, co která třída balíku vykonává. Celkovou strukturu vytvořené aplikace je možné vidět na obr. 2.19, obsah adresáře se vzorovými spustitelnými třídami na obr. 2.20. Jako první je zde uvedena třída na výpočet HoG, která je vytvořena podle [7]. Třída *VarianceComputer* v sekci 2.3.8 byla inspirována bulletinem [10], ostatní prezentované třídy byly navrženy a vytvořeny pro tuto aplikaci.

2.3.1 Kořenový balík

(a) HoG (b) Support

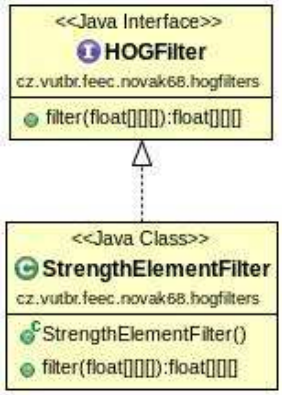
Obr. 2.4: Obsah balíku novak68

V tomto balíčku jsou umístěny třídy *HOG* a *Support*. Ve třídě *Support* je obsažena pouze jediná statická metoda *matrixMultiply(Matice A, Matice B)*, která slouží pro vynásobení 2 matic prvek po prvku. Viz rovnice 2.1.

$$C_{m,n} = A_{m,n} . * B_{m,n} = \begin{pmatrix} a_{1,1} \cdot b_{1,1} & a_{1,2} \cdot b_{1,2} & \cdots & a_{1,n} \cdot b_{1,n} \\ a_{2,1} \cdot b_{2,1} & a_{2,2} \cdot b_{2,2} & \cdots & a_{2,n} \cdot b_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} \cdot b_{m,1} & a_{m,2} \cdot b_{m,2} & \cdots & a_{m,n} \cdot b_{m,n} \end{pmatrix} \tag{2.1}$$

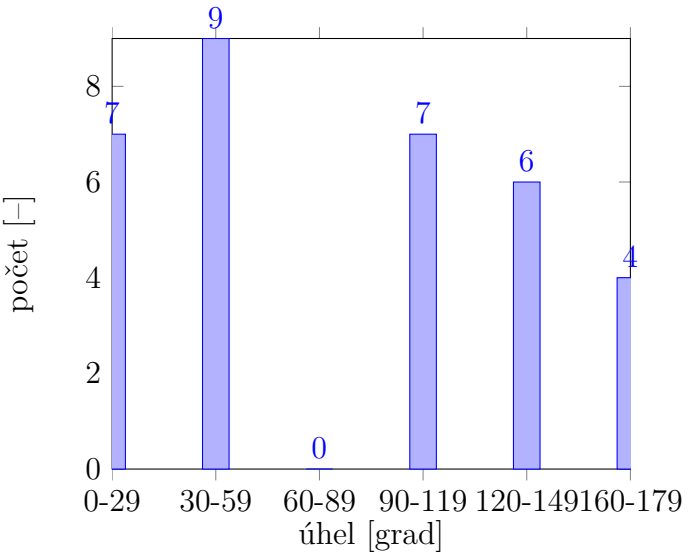
Výpočet histogramu je podrobně uveden v kapitole 1.2, proto je zde uveden pouze odkaz na UML diagram třídy se seznamem veřejných metod – obr. 2.4a. Jako zásadní metoda je zde uvedena *compute()*, ve které je proveden výpočet histogramu podle kapitoly 1.2. V případě použití filtru je metodě *useFilter(HOGFilter)* předán filtr, který je ihned použit na daný HoG. Ostatní metody jsou zaměřeny na grafické zobrazení vypočtených výsledků, až na metodu *getMagnitude(int, int, int)*, jejíž návratová hodnota je velikost koše, který je specifikován vstupními parametry. Návratovou hodnotou metody *getCells()* je vypočtený histogram. V metodě *getImageSize()* je užita jako návratová hodnota velikost obrázku použitého k výpočtu histogramu, a v metodě *normalizeL2()* je aplikována na HoG L2 normalizace popisovaná v [7].

2.3.2 Balík filtrů

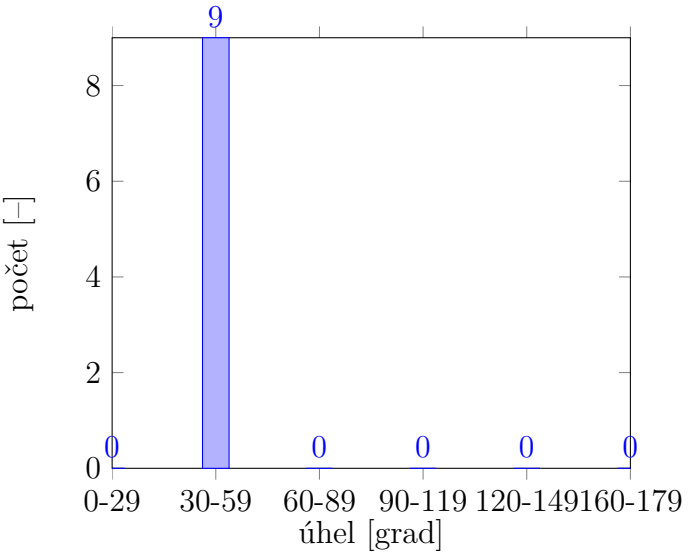


Obr. 2.5: HogFilter

V balíku *cz.vutbr.feec.novak68.hogfilters/* je obsažen pouze jediný filtr, ale je počítáno s rozšířením tohoto balíčku o další filtry v budoucnu, proto je v aplikaci implementováno rozhraní *HOGFilter*. Vytvořeným filtrem je *StrengthElementFilter*, ve kterém jsou procházeny všechny koše v každé buňce a ponechány pouze vítězné koše. Ostatní jsou nulovány, viz následující obrázek, nebo obr. 2.3.



(a) Obsah koše před



(b) Obsah koše po

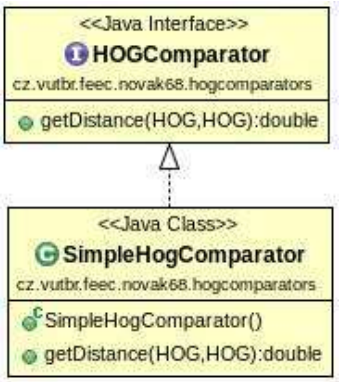
Obr. 2.6: Účinek filtru *StrengthElementFilter*

2.3.3 Balík komparátorů

Nejpočetnější skupinou je skupina komparátorů. Ty je možné rozdělit do 4 skupin:

- porovnání HoG
- porovnání jednotlivých elementů
- porovnání na základě symetrie
- ostatní metody porovnání

Porovnání HoG



Obr. 2.7: SimpleHogComparator

Tento jednoduchý komparátor je založený na metodě MSE a RMSE blíže rozebrané v sekci 1.1. Komparátor je pouze upraven pro použití na histogram místo použití na obrázek. Zde je uveden výpočet pro jediný koš:

$$MSE(a) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} [x(m,n,a) - x'(m,n,a)]^2 \tag{2.2}$$

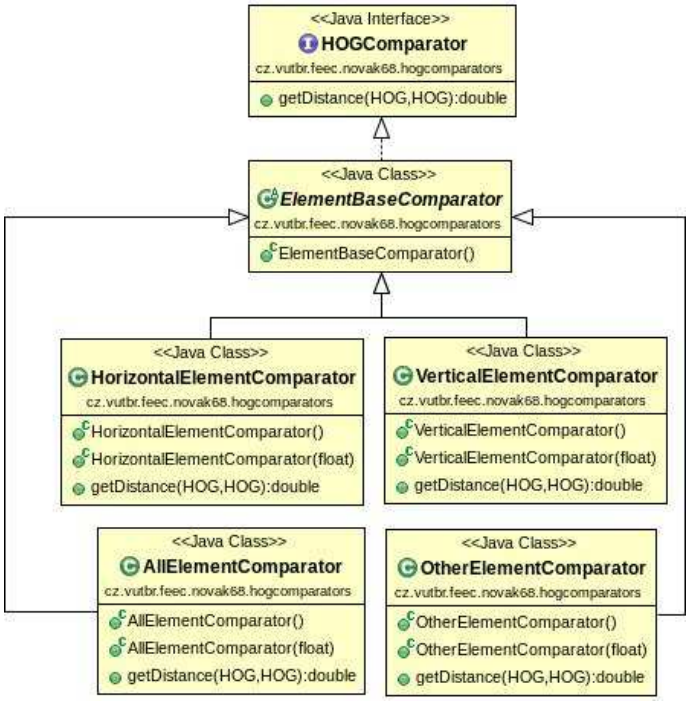
kde M, N jsou pozice buňky a a je pozice koše. Tímto se vypočte MSE pro všechny koše, a dále se z nich vypočte RMSE:

$$RMSE = \frac{1}{a_{max}} \sum_{a=0}^{a_{max}-1} \sqrt{MSE(a)} \tag{2.3}$$

kde a je pořadí koše a a_{max} je celkový počet košů.

Porovnání elementů

Tato skupina pracuje na principu porovnání prvků. Ve třídě *HorizontalElementComparator* je vypočítán počet prvků v nultém koši nad celým HoG, a následně



Obr. 2.8: ElementBaseComparator

porovnán s druhým HoG. Pro každý nenulový prvek je zvýšena hodnota o 1:

$$Horizontal = \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} HoG_1(0) - \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} HoG_2(0) \tag{2.4}$$

kde x a y jsou pozice buněk v HoG. Ve třídě *VerticalElementComparator* je postupováno obdobně s tím rozdílem, že koš, který je použit, je vypočítán buď jako polovina počtu košů, nebo jako nejbližší vyšší a nižší číslo poloviny košů. V tomto případě by byly použity oba koše pro součet prvků. Pro každý nenulový prvek je zvýšena hodnota o 1. Zde je uveden příklad výsledné komparace:

$$Vertical = VERTICAL_1 - VERTICAL_2 \tag{2.5}$$

Ve třídě *AllElementsComparator* je procházen HoG, a pro každý nenulový prvek je zvýšen výsledek o 1. Jako výsledek komparace je tedy použit rozdíl počtu nenulových prvků:

$$AllElements = Elements_1 - Elements_2 \tag{2.6}$$

Ve třídě *OtherElementsComparator* jsou využity výsledky předchozích komparátorů:

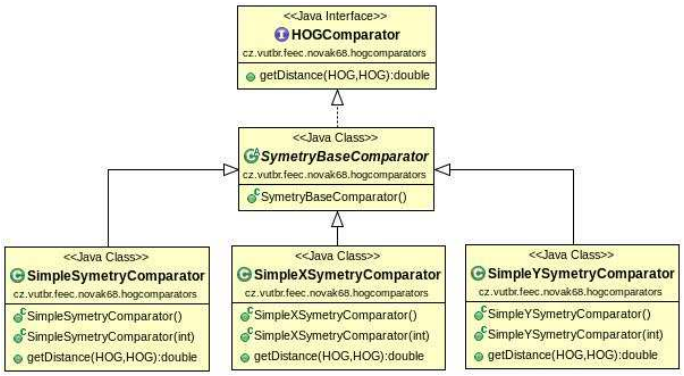
$$OtherElements_1 = AllElements_1 - VERTICAL_1 - Horizontal_1 \tag{2.7}$$

Následně jsou pak tyto výpočty použity pro výslednou komparaci:

$$OtherElements = OtherElements_1 - OtherElements_2 \tag{2.8}$$

U každého porovnání elementů je možné nastavit práh. Význam prahu je odstranit hodnoty nacházející se pod tímto prahem. V aplikaci je tento práh nastavený vždy na hodnotu 1000, čili slabý a nevýrazný prvek je brán jako 0.

Porovnání na základě symetrie



Obr. 2.9: SymetryBaseComparator

Všechny tyto komparátory jsou zaměřeny na získání počtu prvků, které jsou navzájem v osové symetrii. Tyto prvky musí být obsaženy nejméně 4 krát. To znamená 2 na jedné straně osy a 2 na druhé straně osy. Zde je uvedena část kódu pro *SimpleXSymetryComparator* jako příklad (z důvodu lepší čitelnosti byl použit znak "_" pro zalomení dlouhého řádku, kód tedy není v této podobě funkční):

Výpis kódu 2.1: Ukázka části kódu výpočtu prvků v X symetrii

```
boolean [][][] cellsMask = new boolean[cells.length] _
  [cells[0].length][cells[0][0].length];
  for (int x=0;x<cells.length; x++)
    for(int y=0;y<cells[0].length;y++ )
      for(int a = 0; a<cells[0][0].length;a++)
        if(cells[x][y][a] > threshold){
          for (int y2 = y+1;y2<cells[0].length;y2++)
            if(cells[x][y2][a] > threshold){
              for(int x2=x+1;x2<cells.length;x2++){
                if(cells[x2][y][a] >threshold && _
                  cells[x2][y2][a]>threshold && _
```

```
                cells[x][y2][a]>threshold){
            //znamena, ze se ulozi vzdy 2 nasledujici
                cellsMask[x][y][a]=true;
                cellsMask[x2][y][a]=true;
                cellsMask[x][y2][a]=true;
                cellsMask[x2][y2][a]=true;
            }
            else{
                break;
            }
        }}}
```

Z kódu je lépe zřejmé, že jsou vždy kontrolovány 4 prvky. To znamená 2 na jedné straně osy a dva na pomyslné druhé straně. V případě nalezení jisté symetrie je vyplněna maska o velikosti HoG, a po skončení hledání symetrických prvků je výsledná maska sečtena – výsledný počet prvků v symetrii v ose x. Výsledek komparátoru je pak dán rozdílem symetrických prvků v ose x:

$$XSymetryComparator = XSymetry_1 - XSymetry_2 \tag{2.9}$$

Obdobně pracuje *SimpleYSymetryComparator* s rozdílem přehození os, ve kterých je hledaná shoda. Jako poslední komparátor hledající symetrii je uveden *SimpleSymetryComparator*, ve kterém jsou kombinovány předchozí 2 komparátory.

$$SymetryComparator = \sqrt{XSymetryComparator^2 - YSymetryComparator^2} \tag{2.10}$$

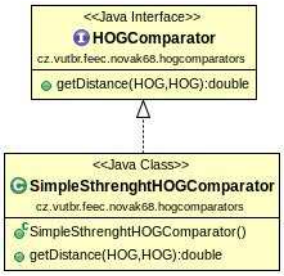
I u komparátorů porovnávajících symetrické prvky lze nastavit práh, a tím odfiltrovat nízké prvky. V prezentované aplikaci jsou nastaveny na *1000*.

Ostatní metody porovnání

Mezi ostatní metody je možné zařadit pouze jeden komparátor – *SimpleSthrenghtHOGComparator*. U této metody je na HoG nejprve aplikován filtr 2.5 na získání pouze největších košů buňky, a poté jsou porovnány příslušné prvky:

$$Strength = STRENGTH_1 - STRENGTH_2 \tag{2.11}$$

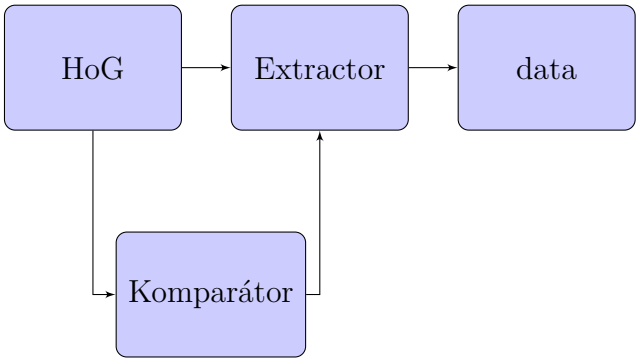
kde $STRENGTH_1$ a $STRENGTH_2$ je počet nenulových prvků HoG po použití filtru.



Obr. 2.10: SimpleSthrenghtHOGComparator

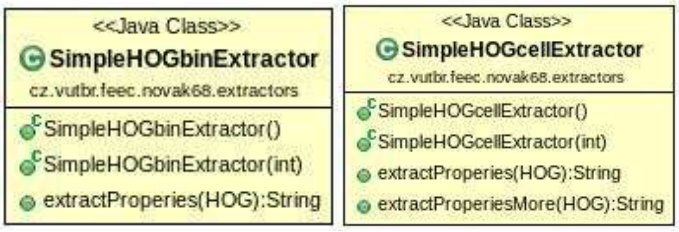
2.3.4 Balík na extrakci dat

Jak již bylo psáno výše, obsah tohoto balíku je zaměřen na extrakci dat pro aplikaci **RapidMiner**. Jedná se tedy o rozhraní mezi HoG a daty. Extraktor může využívat jak samotná data HoG, tak použít data komparátoru.



Obr. 2.11: Spojení HoG vs. data

Nejprve budou představeny extraktory, které využívají pouze data z vypočteného histogramu. Těmi jsou *SimpleHOGbinExtractor* a *SimpleHOGcellExtractor*.



(a) SimpleHOGbinExtractor (b) SimpleHOGcellExtractor

Obr. 2.12: Extraktory využívající HoG

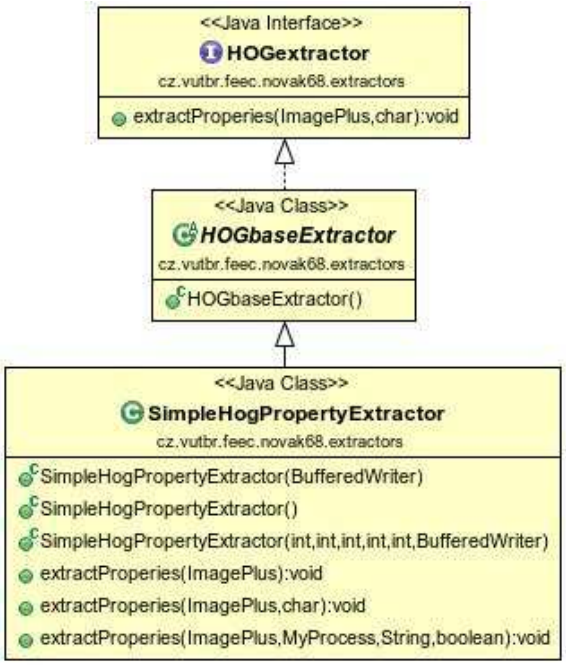
Ve třídě *SimpleHOGbinExtractor* je procházen celý histogram, a v případě nalezení prvku většího než nastavený práh, je přičtena k výsledku 1. To znamená, že výsledek je počet všech prvků košů filtrovaných pouze nastaveným prahem:

$$binExtractor(x, y, a) = \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} \sum_{a=0}^{A-1} \begin{cases} 1, & HoG(x, y, a) > threshold \\ 0, & HoG(x, y, a) < threshold \end{cases} \tag{2.12}$$

kde x a y je pozice prvku v histogramu, a a je koš. Ve třídě *SimpleHOGcellExtractor* je provedena také kontrola každého prvku, ale data nejsou sečtena. Každý prvek je uložen do pole. Dalo by se to znázornit obdobným vzorcem za předpokladu rozložení histogramu do jedné osy:

$$cellExtractor(x) = \sum_{x=0}^{X-1} \begin{cases} 1, & HoG(x) > threshold \\ 0, & HoG(x) < threshold \end{cases} \tag{2.13}$$

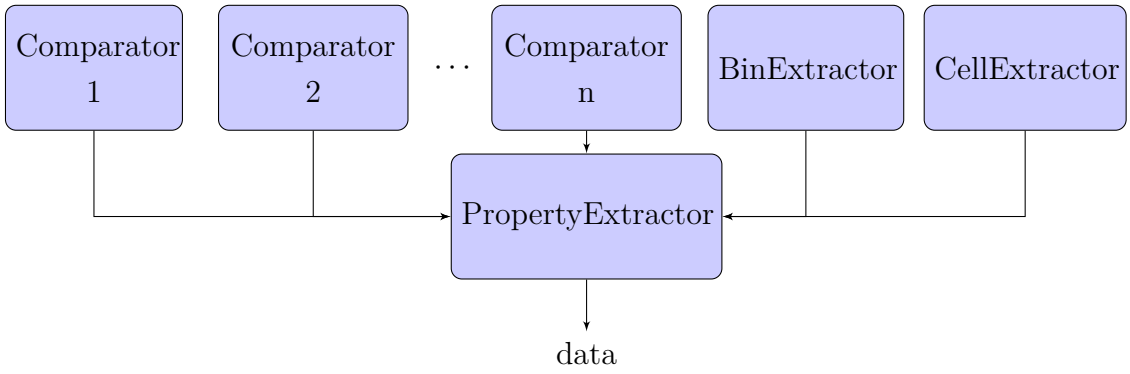
V této třídě je obsažena ještě jedna metoda, a to číselná extrakce, kdy je extrahována hodnota každého prvku, která je vyšší než nastavený práh.



Obr. 2.13: Hlavní extraktor

Třída *SimpleHogPropertyExtractor* je brána jako hlavní třída k extrakci dat. Jsou uvnitř vytvářeny instance předchozích extraktorů a všech komparátorů. V metodě *extractProperies(ImagePlus img, char folder)* je extrahován histogram vypočtený z daného obrázku *img*, a na první pozici dat je umístěn znak *folder*. V této

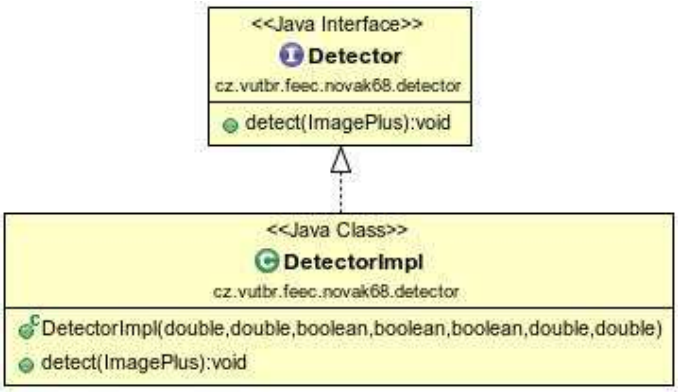
metodě jsou z extraktorů přebírána data v podobě textového řetězce, z komparátorů v podobě čísla. Data jsou následně ukládána do textového souboru, kde každá položka je oddělená znakem ",". Naopak v metodě *extractProperies(ImagePlus img, MyProcess process, String coordinates, boolean useVariance)* je pracováno s celou scénou. V metodě je vypočten HoG v místě určeném koordináty *coordinates* a data jsou předána danému procesu. V závislosti na hodnotě *useVariance* je na konec dat přidána hodnota variance (viz rovnice 1.4) obrázku či nikoliv.



Obr. 2.14: Funkce hlavního extraktoru

2.3.5 Balík detektor

Jak již bylo popsáno výše, *Detector* je použit pro snadné detekování objektů ve scéně. Jeho diagram lze vidět na obrázku 2.15. Pro jeho použití stačí zavolat konstruktor s patřičnými argumenty, jako jsou např.: hodnota prahu pozitivní i negativní detekce, hodnota kroku detekčního okna a poměrná velikost detekčního okna k velikosti detekované scény. Další argumenty jsou booleovského typu a určují, zda a jak se mají prahy použít, nebo zda se má počítat variance detekčního okna. V této třídě je obsažena pouze jediná metoda *detect(ImagePlus sourceImage)*, ve které je inicializován kvalifikační proces, daná scéna je poslána objektu *ImageFacade*, a po zpracování scény je spuštěn kvalifikační proces. Po skončení kvalifikačního procesu jsou procházena označená data, ve scéně jsou označeny pozitivní detekce, a následně je označená scéna zobrazena uživateli.

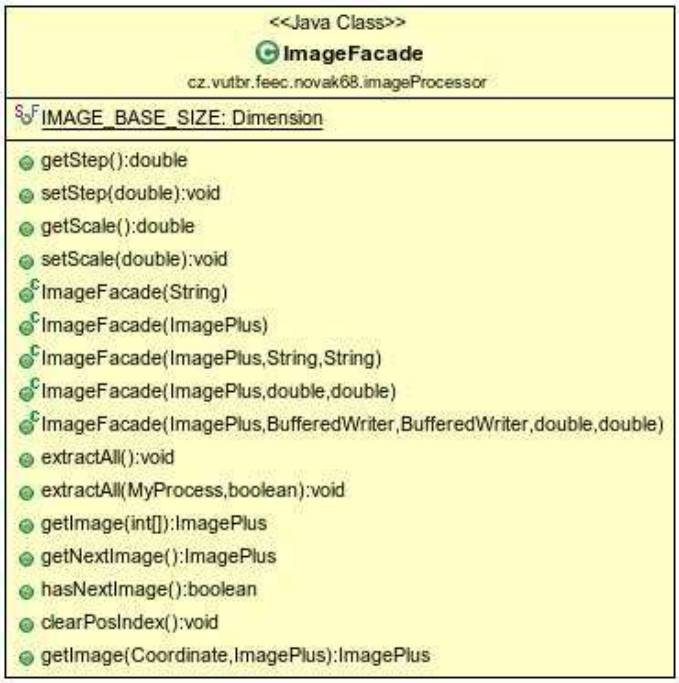


Obr. 2.15: Detector

2.3.6 Balík imageProcessor

V tomto balíku je obsažena třída *ImageFacade*, ve které se nacházejí metody sloužící pro usnadnění operací s obrazem. Její uml diagram lze vidět na obr. 2.16. V této třídě je vytvořen přetížený konstruktor, který je určen pro dva způsoby využití. Jeden *ImageFacade (ImagePlus, BufferedWriter, BufferedWriter,double, double)* je pro použití spolu s knihovnou *BufferedWriter* pro extrakci do souboru, druhý (*ImageFacade (ImagePlus,double, double)* je pro vytvoření datové sady pro proces aplikace **RapidMiner**. Hned při vytváření objektu *ImageFacade* jsou vypočteny všechny možné pozice detekčního okna pro extrakci dat. Návratová hodnota metody *getImage(int[])* je obraz na pozici $int\{x, y\}$. Dále je zde uvedena metoda *extractAll()*, ve které jsou buď extrahovány příznaky ze všech předem vypočítaných oken, a zapsány do souboru, nebo v případě varianty s argumentem *MyProcess* předány vypočtené

příznaky danému procesu. Metody *getNextImage()* a *hasNextImage* jsou použity pro procházení pole souřadnic vypočtených při vytváření objektu.

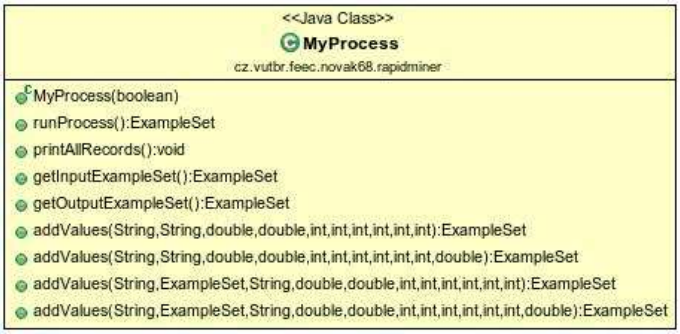


Obr. 2.16: ImageProcessor

2.3.7 Balík rapidminer

Další důležitou součástí aplikace je obsah balíku *cz.vutbr.novak68.rapidminer*. Zde se nachází třída *MyProcess*, ve které je zahrnuta operace s aplikací **RapidMiner**. Tím je myšlen například vytvoření procesu, vytvoření hlavičky tabulky dat, přidání dat do tabulky, spuštění kvalifikačního procesu.

V konstruktoru třídy *MyProcess(boolean useVariance)* je možno si zvolit, zda má jít o proces využívající varianci obrázku jako další příznak pro klasifikátor. Hlavními metodami jsou *runProcess()* a přetížená metoda *addValues(...)*. V metodě *runProcess()* je spuštěn kvalifikační proces pro předem vytvořenou datovou sadu a její návratovou hodnotou je datová sada oznámkovaná. V metodě *addValues(...)* jsou přidávána data do vstupní tabulky procesu. Tato tabulka musí být vyplněna před samotným spuštěním procesu. Zde je možno zvolit jednu z možných variant podle toho, zda do dat zahrneme i varianci, nebo ne.

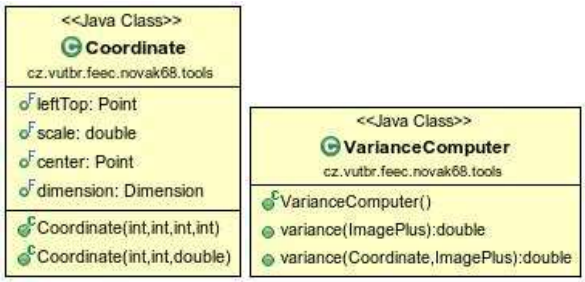


Obr. 2.17: MyProcess

2.3.8 Balík tools

V balíku *cz.vutbr.novak68.tools* jsou obsaženy pomocné třídy *Coordinate* a *VarianceComputer*, viz obr. 2.18. Třída *Coordinate* je použita jako přepravní třída pro snadnou manipulaci s koordináty detekčního okna. Jsou v ní obsaženy souřadnice levého horního rohu, souřadnice středu detekčního okna, dimenze detekčního okna a poměr jeho velikosti k základnímu vzorovému obrazu. Všechny atributy této třídy jsou veřejné, ale zároveň konečné a neměnné. Jsou použity ke zpracování oznámkovaných dat příchozích z výstupu kvalifikačního procesu *MyProcess*, jehož instance je ve třídě *DetectorImpl*, kde jsou koordináty po analyzování dat převedeny právě na tuto formu pro snadnou manipulaci. Dále pak předány metodám k označení pozitivní detekce ve scéně.

Jako další je zde uvedena třída *VarianceComputer*. V této třídě je prováděn výpočet variance daného obrázku:



(a) Coordinate (b) VarianceComputer

Obr. 2.18: Obsah balíku tools

2.4 RapidMiner – křížová validace

Program **RapidMiner** je používán pro prediktivní analytiku [25]. Je v něm obsaženo množství učících algoritmů a nástrojů k dolování dat (Data mining), jako jsou například mřížková optimalizace, optimalizace pomocí generace nebo optimalizace pomocí selekce. Vzhledem k malé množině testovacích obrázků je použita pro porovnání úspěšnosti navržených metod křížová validace. Pro křížovou validaci je zapotřebí mít data pozitivní i negativní. To znamená data, kdy HoG byl vytvořen z obrazu postavy, a data, kdy byl vytvořen z čehokoli jiného. Princip je naznačen na obrázku 2.21. Jako první jsou data rozdělena na 10 rovných dílů (díly [1] – [10]), poté je vybrán 1. díl, který slouží pro ověření úspěšnosti učícího algoritmu, a zbylých 9 dílů slouží jako data pro natrénování algoritmu. Ve chvíli, kdy je algoritmus naučen na data, je postoupeno k testování na vybrané části dat. Výsledkem testování je úspěšnost vyhodnocení. Poté je vybrán další díl dat jako testovací, a na zbylých 9 dílech je algoritmus natrénován. Po skončení deseti cyklů jsou výsledky zprůměrovány:

$$Performance = \sum_{k=1}^{K} Performance_k \tag{2.14}$$

2.5 Zdrojová složka example/

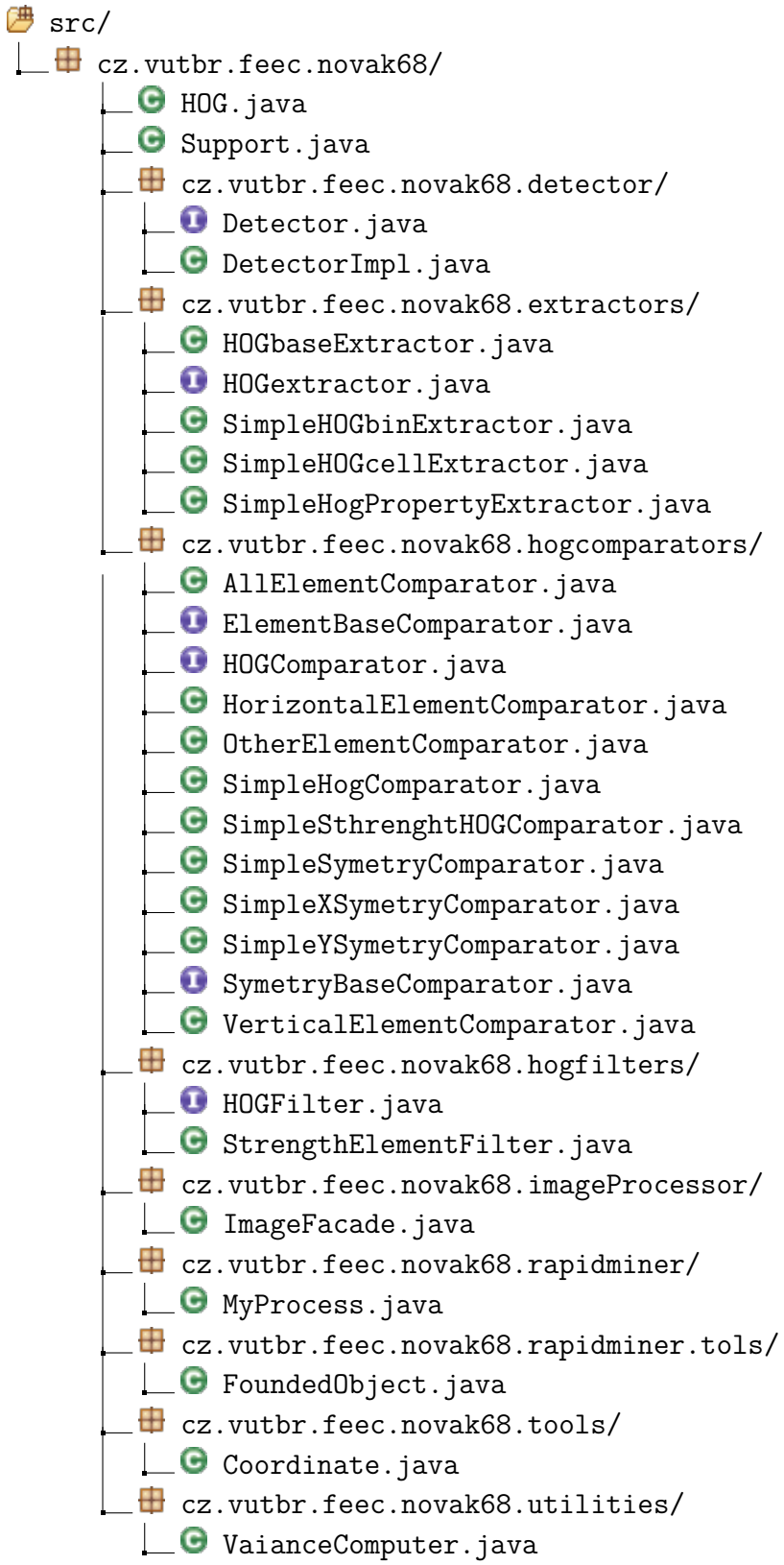
Poslední část, která by zde měla být uvedena, je složka *example/*, ve které jsou obsaženy 2 třídy. Třída *HOGexample* obsahující pokusné kódy, ve které je uvedeno vytvoření histogramu a jeho grafického zobrazení, a třída *GetProperties*, ve které je obsažena metoda **main()**, je tedy spustitelná, a je v ní vytvářena instance hlavního extraktoru *SimpleHogPropertyExtractor*. V tomto extraktoru byla v rámci práce řešena extrakce dat použitá pro **křížovou validaci**. Zde byly za pomocí komentářů vybrány pouze ty metody, které budou uvedeny v závěru. Vzorová část kódu je zde:

Výpis kódu 2.2: Ukázka části kódu třídy SimpleHogPropertyExtractor

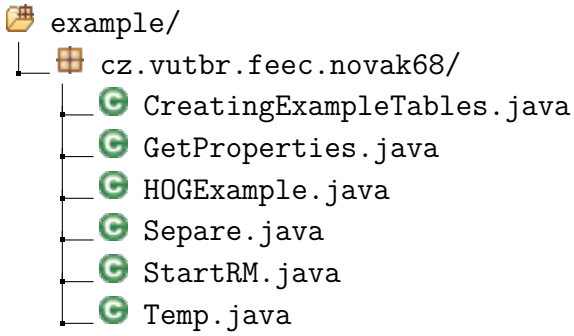
```
//finStr.append(binExtractor.extractProperies(hogTest));
//finStr.append(", ");
//finStr.append(cellExtractor.extractProperies(hogTest));
//finStr.append(", ");
finStr.append(simpleC.getDistance(hogTrain, hogTest));
finStr.append(", ");
finStr.append(strenghtC.getDistance(hogTrain, hogTest));
finStr.append(", ");
```

V tomto kódu je uveden příklad, jak se pomocí objektu *StringBuilder* sestavují data. Vytváření instancí objektů v této třídě je naznačeno také v uml diagramu A.1 v příloze A.

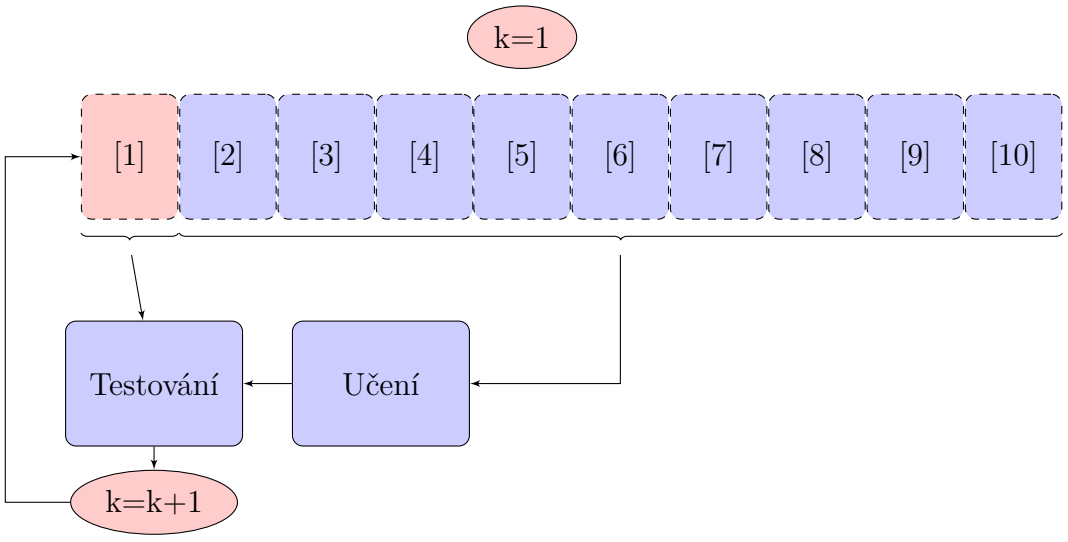
Další podstatnou třídou v této aplikaci je *Separe*, ve které je vytvářena instance objektu *Detector* pro všechny testovací scény uložené ve složce *data/-test/*. V tomto detektoru je spouštěn kvalifikační proces pro každou scénu, jsou zde vyznačeny detekované objekty a výsledek je zobrazen uživateli.



Obr. 2.19: Navržená struktura balíku



Obr. 2.20: Vzorové třídy



Obr. 2.21: Princip křížové validace

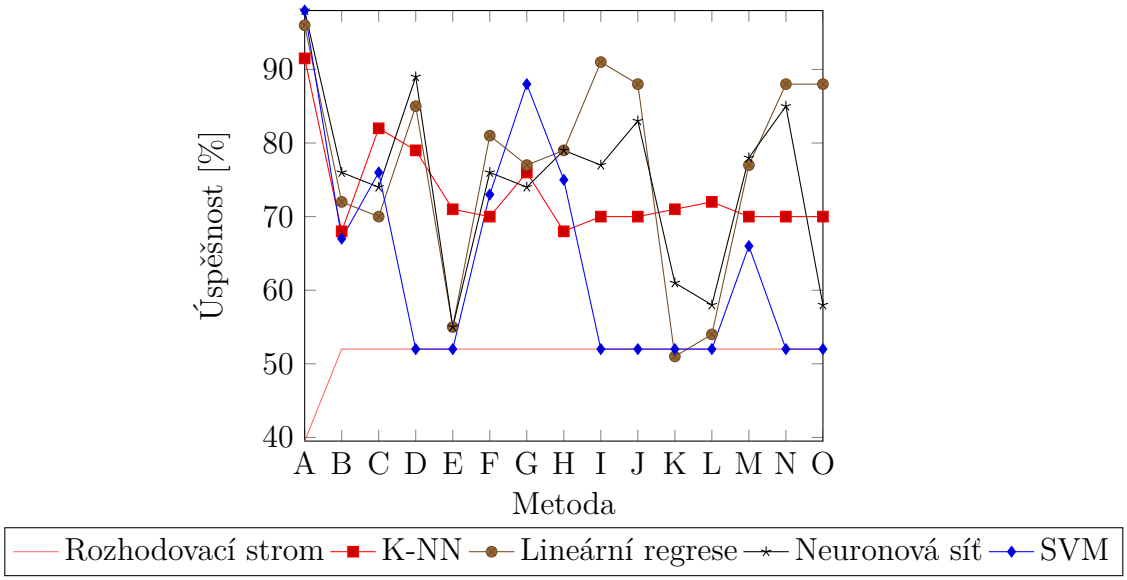
3 VÝSLEDKY

Pro získání přehledu o účinnosti zvolených metod byl použit program Rapid Miner [25]. S jeho pomocí bylo možno na získaných datech natrénovat tyto algoritmy: rozhodovací strom, náhodný les, k-NN, SVM, neuronové sítě, lineární regrese. Úspěšnost rozhodování algoritmů je uvedeno v tabulce 3.1. Seznam metod je uveden v příloze A.

Tab. 3.1: Úspěšnost detekce v procentech pro metody A–O

	Rozhodovací strom	K-NN	Lineární regrese	Neuronová síť	SVM
A	39.50	**91.50**	**96.00**	**98.00**	**98.00**
B	52.00	68.00	72.00	76.00	67.00
C	52.00	82.00	70.00	74.00	76.00
D	52.00	79.00	85.00	89.00	52.00
E	52.00	71.00	55.00	55.00	52.00
F	52.00	70.00	81.00	76.00	73.00
G	52.00	76.00	77.00	74.00	88.00
H	52.00	68.00	79.00	79.00	75.00
I	52.00	70.00	**91.00**	77.00	52.00
J	52.00	70.00	88.00	83.00	52.00
K	52.00	71.00	51.00	61.00	52.00
L	52.00	72.00	54.00	58.00	52.00
M	52.00	70.00	77.00	78.00	66.00
N	52.00	70.00	88.00	85.00	52.00
O	52.00	70.00	53.00	58.00	52.00

Jak lze vidět z tabulky, největšího úspěchu detekce bylo dosaženo s pomocí metody **A** (žlutě zvýrazněný řádek) a algoritmů SVM a neuronové sítě. Dalším úspěchem by mohla být metoda **I** a algoritmus lineární regrese, ale z ostatních výsledků je spíše patrné, že došlo k zavedení chyby do výpočtů. Přehled úspěšnosti lze vidět i na obr. 3.1, kde stabilního výsledku dosahuje již zmíněná metoda **A**, metody **B** a **C** (cca 70% – 80%) a metoda **G** (cca 75%). Dále lze vidět, že po použití filtru u metody **B** byly výsledky spíše zhoršeny. Metoda **B** sestává ze základního setu komparátorů za předchozího použití filtru silných prvků. Lze tedy soudit, že použití filtru silných prvků tedy nebylo správnou cestou, a ostatní kombinace, ač s větším počtem dat, spíše detekci zhoršily.

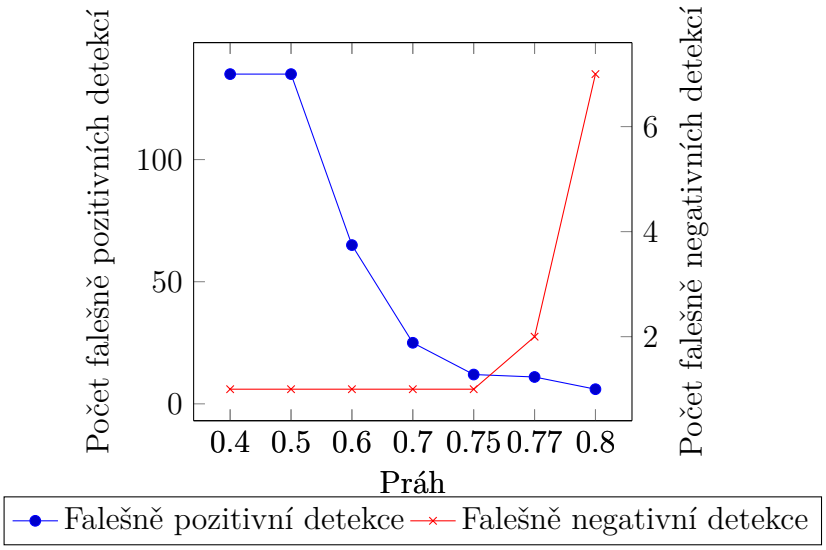


Obr. 3.1: Graf míry úspěšnosti jednotlivých metod při křížové validaci

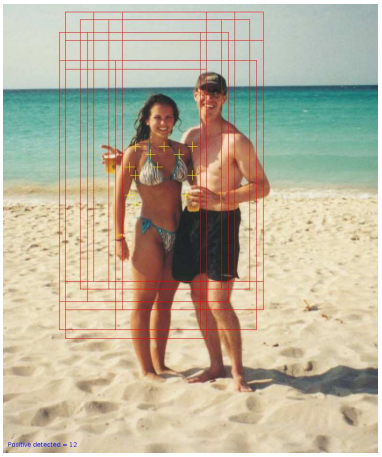
Vzhledem k tomu, že neúspěšnější metoda byla metoda **A** obsahující pouze sadu 9 komparátorů, bylo dále využito především této metody. K testování byl použit model natrénován na 50 pozitivních vzorků (chodci ve stejném úhlu jako vzor) a 50 negativních vzorků. Výsledek testu na konkrétních scénách je uveden na obrázku 3.2, kde je na levé ose y uveden součet všech chybných pozitivních detekcí, na straně pravé součet všech chybných negativních detekcí vzhledem k nastavenému prahu zvoleného **SVM** modelu. Výsledek detekce je možné vidět na obrázku 3.3.

Další snahou bylo tyto výsledky vylepšit zvětšením databáze trénovacích vzorů. K tomu byla použita kompletní databázi **MIT** [6]. Z obrázku 3.4 lze vidět, že výsledek se rapidně zhoršil. Proto byla tato trénovací množina zavržena a bylo přistoupeno zpět k množině původních 100 vzorků, ale s použitím variance obrázku. Při testech této metody opět nebylo dosaženo očekávaného úspěchu, jak ukazuje graf 3.6. Proto byla použita normalizace **L2** popsaná v [7]. S použitím normalizace bohužel zdaleka nebylo dosaženo takových úspěchů jako v předchozích případech a detektor nebyl schopen správně detekovat. Viz obrázek 3.8

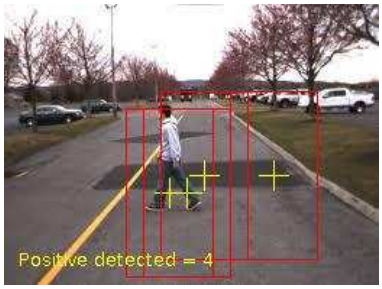
Jako nejlepší možné řešení ze všech provedených pokusů se jevil model natrénovaný na 100 trénovacích vzorech bez použité normalizace, nebo přidaných příznaků 3.2. Bohužel je zde jistá nepřesnost týkající se objektů v nepředvídané kompozici, jako je např. obrázek 3.3c. Dále je možné vidět chybnou pozitivní detekci na obr. 3.3b, kde by se dal detekční práh zpřísnit pro daný případ použití, či obr. 3.3f, kde by pomohlo také zpřísnění detekčního okna, nebo popř. metody na předzpracování obrazu, které odstranily pozadí za případným detekovaným předmětem.



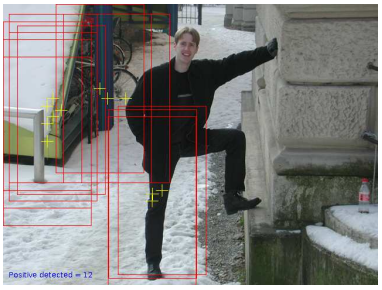
Obr. 3.2: Počet chybných detekcí s naučeným modelem na 100 trénovacích vzorů otestovaným na 25 scénách



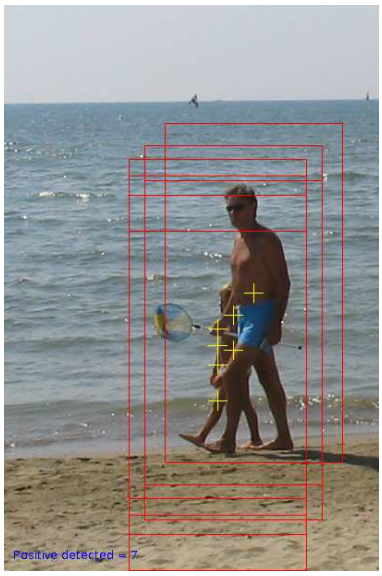
(a) Vzorek č. 02



(b) Vzorek č. 03



(c) Vzorek č. 10

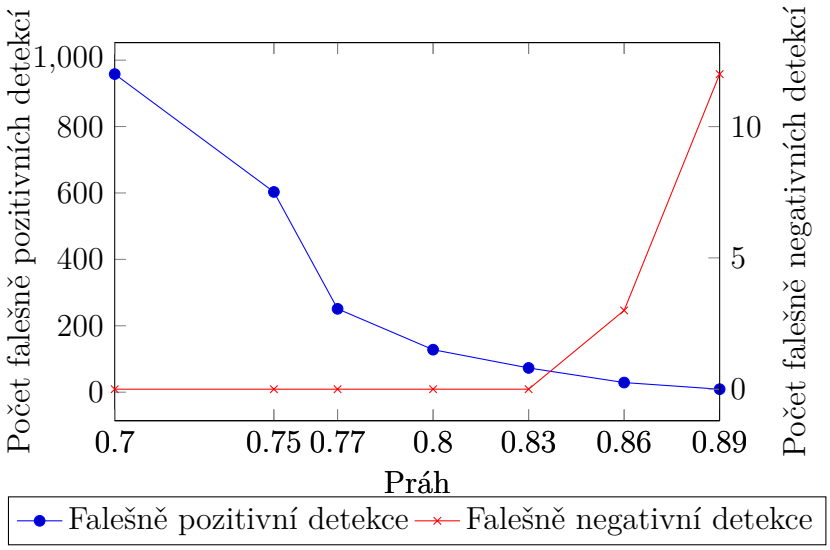


(d) Vzorek č. 13

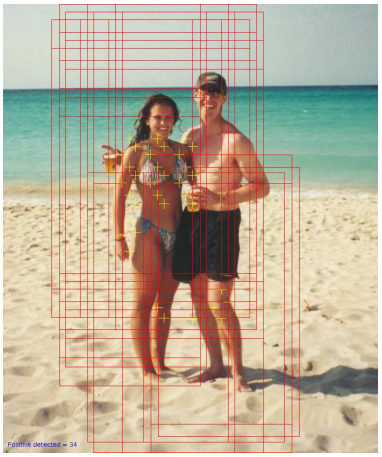
(e) Vzorek č. 20

(f) Vzorek č. 24

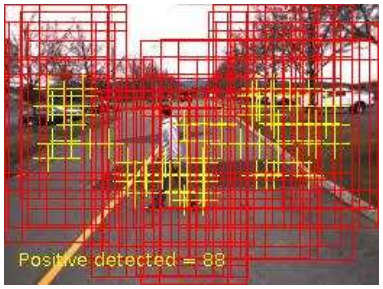
Obr. 3.3: Výsledek detekce za použití 100 trénovacích vzorů a nastavení prahu 75 %



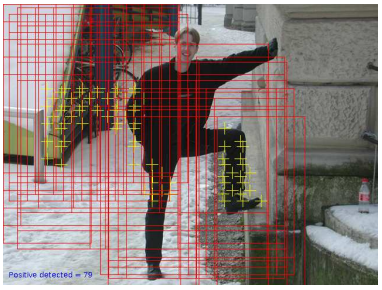
Obr. 3.4: Počet chybných detekcí s naučeným modelem na kompletní databázi MIT otestovaným na 25 scénách



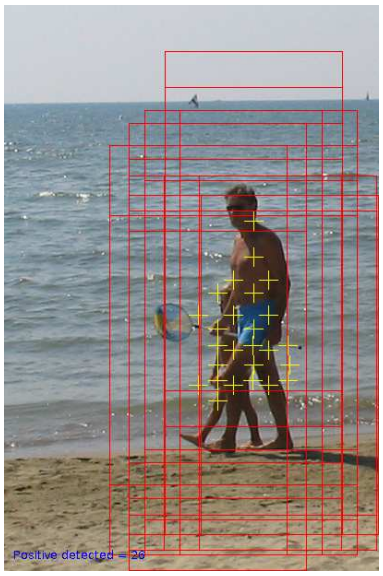
(a) Vzorek č. 02



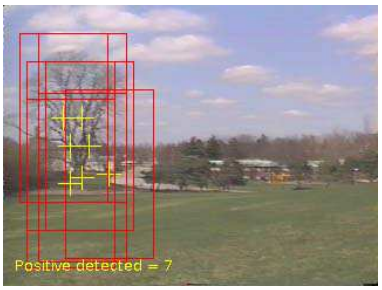
(b) Vzorek č. 03



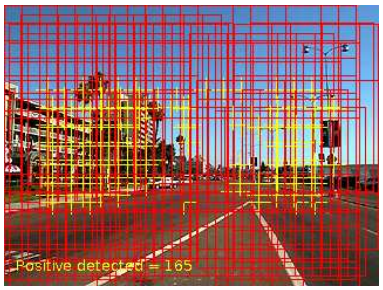
(c) Vzorek č. 10



(d) Vzorek č. 13

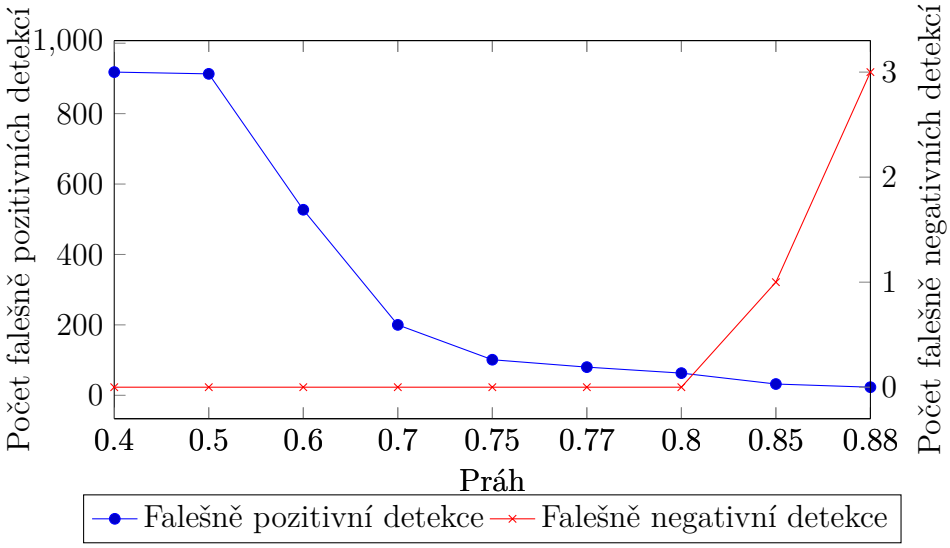


(e) Vzorek č. 20

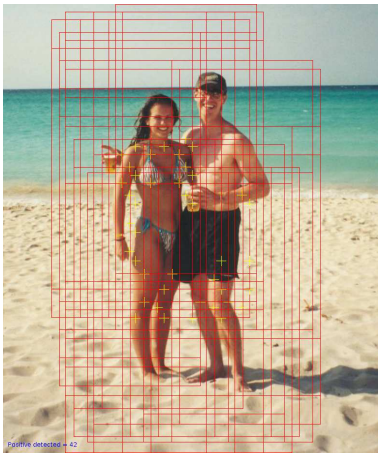


(f) Vzorek č. 24

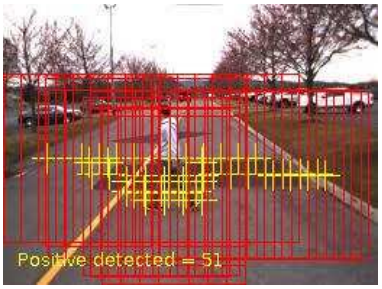
Obr. 3.5: Výsledek detekce za použití MIT databáze a nastavení prahu 75 %



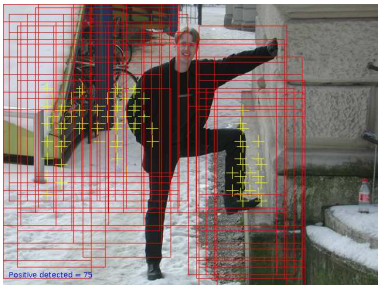
Obr. 3.6: Počet chybných detekcí s naučeným modelem na databázi 100 trénovacích vzorů s použitím variance otestovaným na 25 scénách



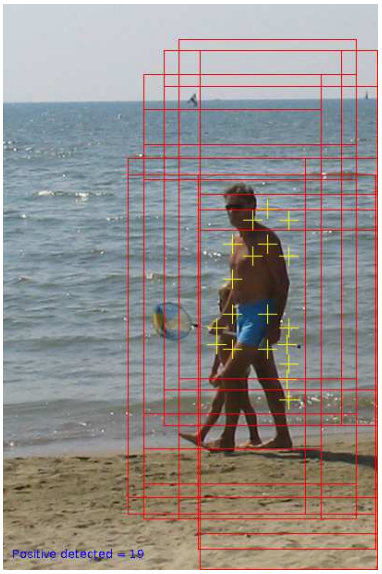
(a) Vzorek č. 02



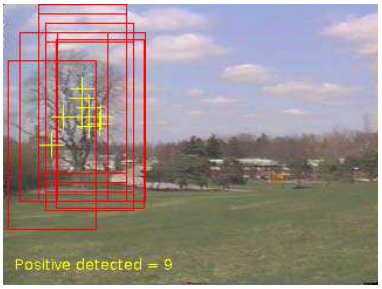
(b) Vzorek č. 03



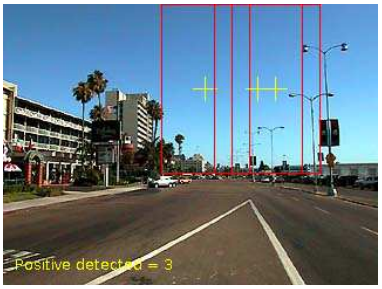
(c) Vzorek č. 10



(d) Vzorek č. 13

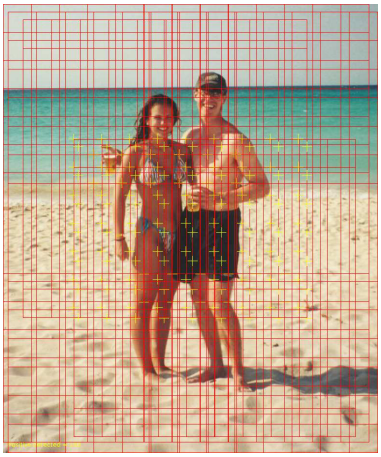


(e) Vzorek č. 20

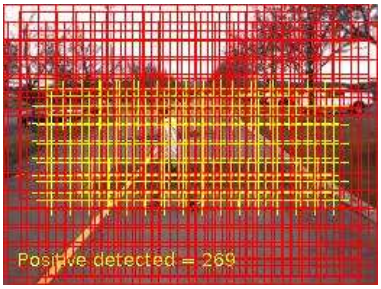


(f) Vzorek č. 24

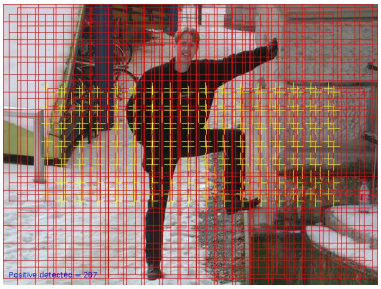
Obr. 3.7: Výsledek detekce za použití databáze se 100 vzory s přidanou variancí a nastaveným detekčním prahem na 75 %



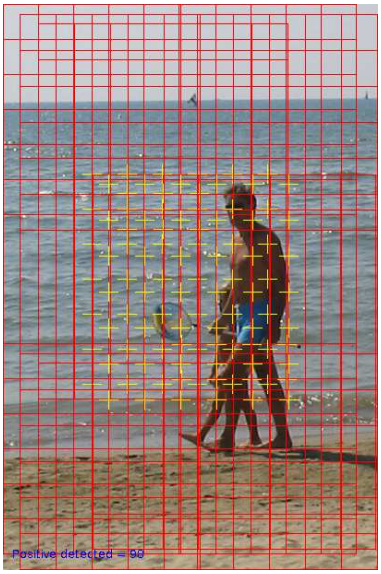
(a) Vzorek č. 02



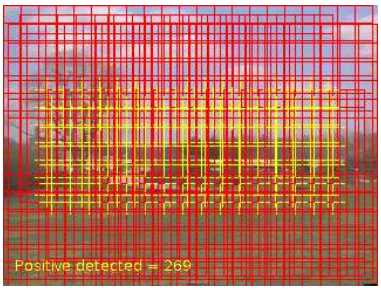
(b) Vzorek č. 03



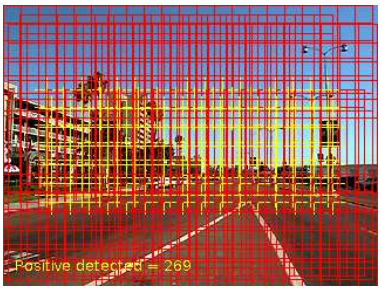
(c) Vzorek č. 10



(d) Vzorek č. 13



(e) Vzorek č. 20



(f) Vzorek č. 24

Obr. 3.8: Výsledek detekce za použití databáze se 100 vzory s použitím L2 normalizace a nastaveným detekčním prahem na 70 %

4 ZÁVĚR

Úkolem této práce bylo seznámit se s metodami vyhledávání objektů v obraze a navrhnout vlastní metody na detekci objektu v obraze pomocí předlohy. Práce vycházela z metody histogramu orientovaných gradientů. Další metody, kterými je tato práce inspirována, byly popsány v 1. kapitole. V následující 2. kapitole byla popsána vytvořená aplikace implementující metodu histogramu orientovaných gradientů s popisem metod extrakce příznaků, které byly v rámci práce vytvořeny. Tyto metody byly následně ověřeny pomocí křížové validace. Nejúspěšnější metoda křížové validace byla následně otestována na reálných scénách s použitím různých trénovacích sad.

Hlavním přínosem této práce je nová metoda na extrakci příznaků histogramu orientovaných gradientů za použití sady komparátorů. Pomocí navržené metody a následného porovnání výsledků křížové validace bylo dosaženo přesnosti detekce 98 % za použití algoritmu SVM. V porovnání úspěšnosti detekce se současnými metodami je naše metoda na přibližně stejné úrovni (nejlepší metody dosahují 98 % – necelých 100 % úspěšnosti), ale naše metoda dosáhla této úspěšnosti na velmi malé sadě trénovacích vzorů (100 obrázků – 50 pozitivních, 50 negativních) oproti 1–2 tisícům trénovacích vzorů pro detekci v ostatních pracích.

Při testech na reálných scénách metoda vykazovala určitou chybovost v oblasti pozitivní detekce. Ta mohla být způsobena zvoleným univerzálním nastavením detekčního okna. Dala by se tedy odstranit speciálním nastavením pro speciální případ nasazení detektoru v praxi. Také chybovost při detekci postav v nepředvídatelné kompozici by se dala řešit zvýšením počtu trénovacích vzorů různých postavení těl.

Navržená metoda je alternativou současných metod pro použití v místech, kde je nevhodné, či nemožné získat potřebné množství trénovacích vzorů. Detekce zatím používá pouze jeden vzorový objekt – tím je chodec v chůzi. Další možností je rozšířit data o obrazy chodců v různých pozicích chůze z různých úhlů, popřípadě osob v různých pozicích. Dále by bylo možné nakombinovat naše metody s metodami z aktuálních prací (např. překrývání buněk apod.). Bylo by také možné se pokusit aplikaci rozšířit o různé barevné filtry, které by usnadnily identifikaci osob.

LITERATURA

[1] BELONGIE, S., J. MALIK a J. PUZICHA. Matching shapes. *Proceedings Eighth IEEE International Conference on Computer Vision*. ICCV 2001. IEEE Comput. Soc, 2001, s. 454-461. DOI: 10.1109/ICCV.2001.937552. Dostupné z: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=937552>

[2] BERTOZZI, M., A. BROGGI, A. FASCIOLI, T. GRAF a M. MEINECKE. Pedestrian Detection for Driver Assistance Using Multiresolution Infrared Vision. *IEEE Transactions on Vehicular Technology*. 2004, vol. 53, issue 6, s. 1666-1678. DOI: 10.1109/TVT.2004.834878. Dostupné z: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1360127>

[3] BERTOZZI, M., A. BROGGI, M. Del ROSE, M. FELISA, A. RAKOTOMAMONJY a F. SUARD. A Pedestrian Detector Using Histograms of Oriented Gradients and a Support Vector Machine Classifier. *2007 IEEE Intelligent Transportation Systems Conference*. IEEE, 2007, s. 143-148. DOI: 10.1109/ITSC.2007.4357692. Dostupné z: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4357692>

[4] BURGES, Christopher J.C. Probabilistic Methods for Finding People. *Data Mining and Knowledge Discovery*. vol. 2, issue 2, s. 121-167. DOI: 10.1023/A:1009715923555. Dostupné z: <http://link.springer.com/article/10.1023/A:1011179004708#page-1>

[5] CAVE: Databases. COLUMBIA UNIVERSITY. *Columbia University: Computer Vision Laboratory* [online]. [cit. 2014-05-01]. Dostupné z: <http://www.cs.columbia.edu/CAVE/databases/>

[6] CBCL Pedestrian Data. *MIT - Massachusetts Institute of Technology* [online]. 2000 [cit. 2014-05-12]. Dostupné z: <http://cbcl.mit.edu/software-datasets/PedestrianData.html>

[7] DALAL, N. a B. TRIGGS. Histograms of Oriented Gradients for Human Detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. IEEE, 2005, č. 1, s. 886-893. DOI: 10.1109/CVPR.2005.177. Dostupné z: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1467360>

[8] DALAL, N., B. TRIGGS a C. SCHMID. Human Detection Using Oriented Histograms of Flow and Appearance. s. 428. DOI: 10.1007/11744047_33. Dostupné z: <http://www.springerlink.com/index/10.1007/11744047_33>

[9] DÉNIZ, O., G. BUENO, J. SALIDO a F. DE LA TORRE. Face recognition using Histograms of Oriented Gradients. *Pattern Recognition Letters*. 2011, vol. 32, issue 12, s. 1598-1603. DOI: 10.1016/j.patrec.2011.01.004. Dostupné z: <http://linkinghub.elsevier.com/retrieve/pii/S0167865511000122>

[10] DOHNAL, Gejza. *Informační Bulletin České statistické společnosti*. Praha, 1995, roč. 6, č. 3. ISSN 1210-8022. Dostupné z: <http://www.statspol.cz/bulletiny/ib-95-3.pdf#page=15

[11] FELZENSZWALB, Pedro F. a Daniel P. HUTTENLOCHER. Pictorial Structures for Object Recognition. *International Journal of Computer Vision*. 2005, vol. 61, issue 1, s. 55-79. DOI: 10.1023/B:VISI.0000042934.15159.49. Dostupné z: <http://link.springer.com/10.1023/B:VISI.0000042934.15159.49>

[12] FELZENSZWALB, Pedro F. a Daniel P. HUTTENLOCHER. Efficient matching of pictorial structures. *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*. CVPR 2000 (Cat. No.PR00662). IEEE Comput. Soc, 2000, s. 66-73. DOI: 10.1109/CVPR.2000.854739. Dostupné z: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=854739>

[13] FREUND, Yoav, a spol. Experiments with a new boosting algorithm. V: ICML. 1996. s. 148-156. Dostupné z: <http://web.eecs.utk.edu/~parker/Courses/CS425-528-fall12/Handouts/AdaBoost.M1.pdf>

[14] GAVRILA, D.M. a V. PHILOMIN. Real-time object detection for "smart"vehicles. *Proceedings of the Seventh IEEE International Conference on Computer Vision*. IEEE, 1999, 87-93 vol.1. DOI: 10.1109/ICCV.1999.791202. Dostupné z: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=791202>

[15] HEGMON, Jiří. *Automatická anotace obrazu* [online]. Brno, 2013 [cit. 2013-11-26]. 54 l. Dostupné z: <https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=67452> Diplomová práce. Vysoké učení technické v Brně. Vedoucí práce Ing. RADIM BURGET, Ph.D.

[16] HYNDMAN, Rob J. a Anne B. KOEHLER. Another look at measures of forecast accuracy. *International Journal of Forecasting*. 2006, vol. 22, issue 4, s. 679-688. DOI: 10.1016/j.ijforecast.2006.03.001. Dostupné z: <http://linkinghub.elsevier.com/retrieve/pii/S0169207006000239>

[17] *ImageJ: Image Processing and Analysis in Java* [online]. [cit. 2013-11-30]. Dostupné z: <http://rsb.info.nih.gov/ij/index.html>

[18] Java.io (Java Platform SE 6). *Oracle Documentation* [online]. 2011 [cit. 2013-12-01]. Dostupné z: <http://docs.oracle.com/javase/6/docs/api/java/io/package-summary.html>

[19] JIA, Hui-Xing a Yu-Jin ZHANG. Fast Human Detection by Boosting Histograms of Oriented Gradients. *Fourth International Conference on Image and Graphics (ICIG 2007)*. 2007. DOI: http://dx.doi.org/10.1109/icig.2007.53. Dostupné z: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4297169&tag=1>

[20] *JUnit* [online]. 2013 [cit. 2013-12-31]. Dostupné z: <http://junit.org/>

[21] KRÁLÍK, Martin. *Detekce objektů v obraze s pomocí rozšířené sady Haarových příznaků a histogramu*. Brno, 2012. 58 l. Dostupné z: <https://www.vutbr.cz/studium/zaverecne-prace?zp_id=52110> Diplomová práce. Vysoké učení technické v Brně. Vedoucí práce Ing. Radim Burget, Ph.D.

[22] MOHAN, A., C. PAPAGEORGIOU a T. POGGIO. Example-based object detection in images by components. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2002, svazek 23, vydání 4, s. 349-361. DOI: 10.1109/34.917571. Dostupné z: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=917571>

[23] OREN, M., C. PAPAGEORGIOU, P. SINHA, E. OSUNA a T. POGGIO. Pedestrian detection using wavelet templates. *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Comput. Soc, 1997, s. 193-199. DOI: 10.1109/CVPR.1997.609319. Dostupné z: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=609319>

[24] QIANG ZHU, MEI-CHEN YEH, KWANG-TING CHENG a S. AVIDAN. Fast Human Detection Using a Cascade of Histograms of Oriented Gradients. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2 (CVPR'06)*. IEEE, 2006, č. 2, s. 1491-1498. DOI: 10.1109/CVPR.2006.119. Dostupné z: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1640933>

[25] *RapidMiner: Predictive Analytics, Data Mining, Self-service, open source* [online]. 2013 [cit. 2013-12-01]. Dostupné z: <http://rapidminer.com/>

[26] STOKES, Michael, Matthew ANDERSON, Srinivasan CHANDRASEKAR a Ricardo MOTTA. A Standard Default Color Space for the Internet - sRGB. *World Wide Web Consortium (W3C)* [online]. [cit. 2013-11-26]. Dostupné z: <http://www.w3.org/Graphics/Color/sRGB.html>

[27] SUARD, F., A. RAKOTOMAMONJY, A. BENSRHAIR a A. BROGGI. Pedestrian Detection using Infrared images and Histograms of Oriented Gradients. *2006 IEEE Intelligent Vehicles Symposium*. IEEE, 2006, s. 206-212. DOI: 10.1109/IVS.2006.1689629. Dostupné z: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1689629>

[28] The Facial Recognition Technology (FERET) Database. *NIST: Information Technology Laboratory* [online]. 2010, 15. 4. 2014 [cit. 2014-05-01]. Dostupné z: <http://www.itl.nist.gov/iad/humanid/feret/feret_master.html>

[29] *The Open Source Initiative* [online]. [cit. 2013-12-01]. Available from: <http://opensource.org/>

[30] VIOLA, P. a M. JONES. Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001.* IEEE Comput. Soc, 2001, I-511-I-518. DOI: 10.1109/CVPR.2001.990517. Dostupné z: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=990517>

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

SPZ Státní poznávací značka

CV Počítačové vidění – Computer Vision

MAE Střední absolutní chyba – Mean absolute error

MSE Střední kvadratická odchylka (rozptyl) – Mean Square Error

RMSE Směrodatná odchylka – Root Mean Square Error

HoG Histogram orientovaných gradientů – Histogram of Oriented Gradients

COIL Columbia University Image Library

ImageJ Zpracování obrazu a analýza v Javě – Image Processing and Analysis in Java

SVM Support Vector Machine

k-NN k–nejbližších sousedů – k–Nearest Neighbor

A TABULKY A DIAGRAMY

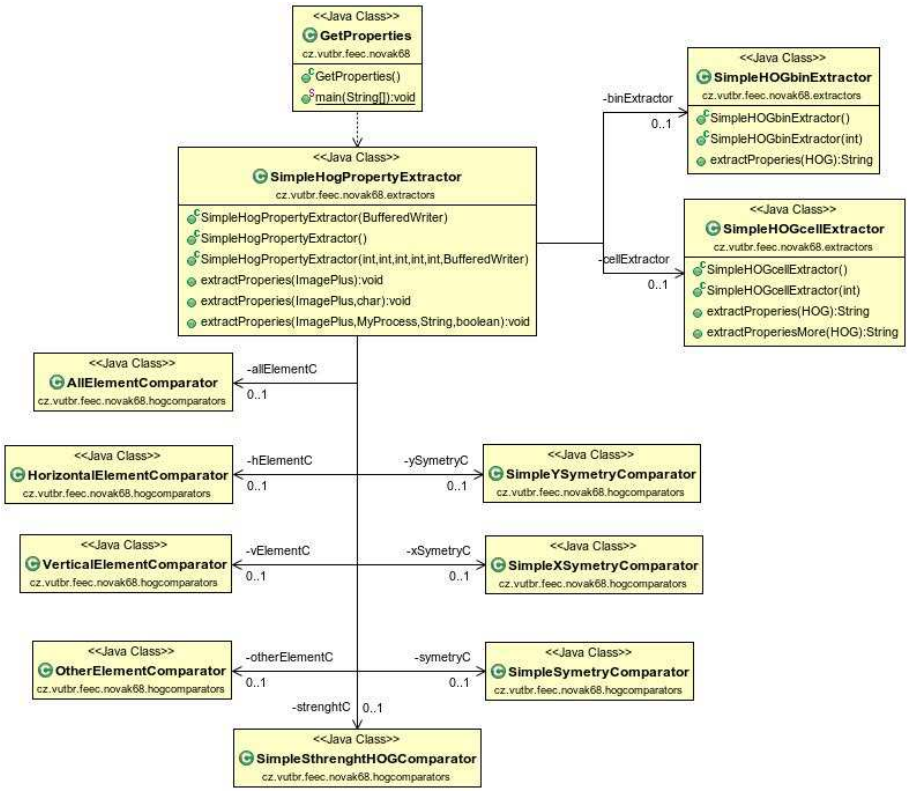
Seznam navržených metod

A	SimpleHogComparator, SimpleSthrenghtHOGComparator, SimpleSymetryComparator, SimpleXSymetryComparator, SimpleYSymetryComparator, AllElementComparator, HorizontalElementComparator, VerticalElementComparator, OtherElementComparator
B	Filtr + A
C	SimpleHOGbinExtractor
D	SimpleHOGcellExtractor.extractProperties()
E	SimpleHOGcellExtractor.extractPropertiesMore()
F	Kombinace A + B
G	Kombinace C + A
H	Kombinace C + B
I	Kombinace D + A
J	Kombinace D + B
K	Kombinace E + A
L	Kombinace E + B
M	Kombinace C + A + B
N	Kombinace D + A + B
O	Kombinace E + A + B

Tab. A.1: Použité metody

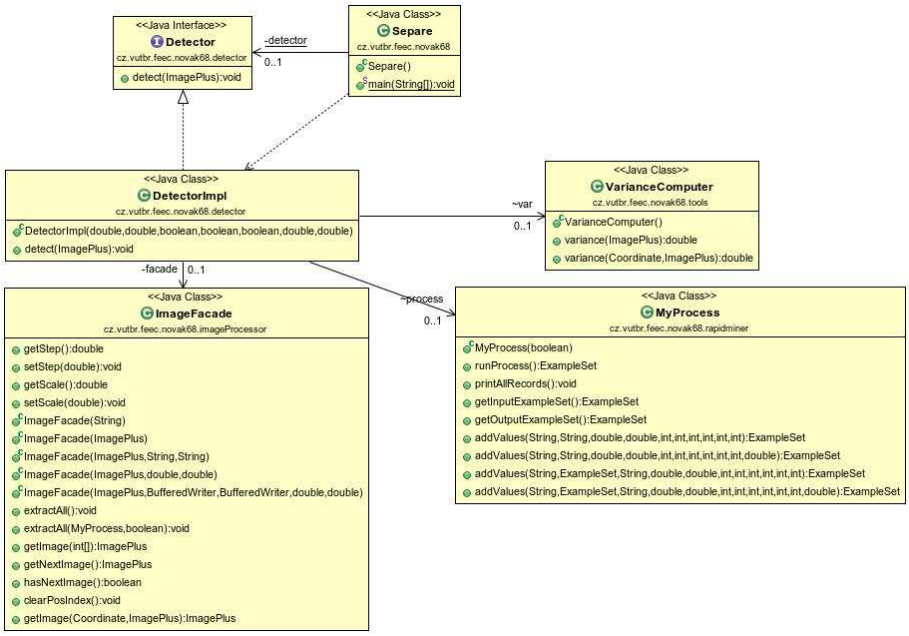
V tabulce A.1 je seznam použitých metod. K extrakci dochází postupně a data jsou vložena do řetězce pro následné uložení do souboru. Kombinace znamená, že je nejprve provedena extrakce jedné metody a za ní extrakce druhé metody.

UML diagram třídy GetProperties



Obr. A.1: UML diagram třídy GetProperties

UML diagram třídy Separe



Obr. A.2: UML diagram třídy Separe

B SEZNAM PŘÍLOH NA DVD

- Text práce (*DetekceObjektuVObraze.pdf*)
- Vytvořená aplikace ve formátu jar *xnovak68.jar* obsahující vytvořené třídy
- Dokumentaci ve složce *doc*
- Data set 100 obrázků použitý v této práci