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

UČENÍ DETEKTORŮ POMOCÍ SLEDOVÁNÍ OBJEKTŮ

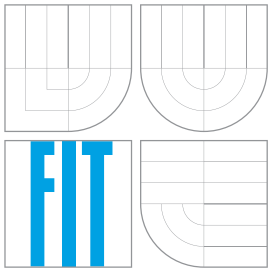
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. RADIM BUCHTELA

BRNO 2013

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

UČENÍ DETEKTORŮ POMOCÍ SLEDOVÁNÍ OBJEKTŮ

LEARNING DETECTORS BY TRACKING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. RADIM BUCHTELA

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MICHAL HRADIŠ

BRNO 2013

Abstrakt

Práce se věnuje problematice dlouhodobého sledování objektů ve video sekvenci, konkrétně oblasti učení detektorů pomocí sledování objektů. V práci jsou diskutovány metody pro sledování objektů, detekci objektů a online učení a možnosti jejich nasazení v sofistikovanějších technikách, které kombinují sledování objektu a online učení detektorů.

Abstract

This thesis is devoted to learn detectors by object tracking in video sequence. In this thesis, we discuss methods for object tracking, object detection and online learning and possibilities of their using in sophisticated techniques, which combine object tracking and online learning detectors.

Klíčová slova

dlouhodobé sledování objektu, detekce objektu, online učení detektorů, klasifikace příznaky, náhodné kapradiny, porovnávání šablonou, kaskáda klasifikátorů, Lucas-Kanade optický tok

Keywords

long-term object tracking, object detection, online learning detectors, feature-based classification, random ferns classifier, template matching, cascade of classifiers, Lucas-Kanade optical flow, Lucas-Kanade tracker

Citace

Radim Buchtela: Učení detektorů pomocí sledování objektů, diplomová práce, Brno, FIT VUT v Brně, 2013

Učení detektorů pomocí sledování objektů

Prohlášení

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

.....
Radim Buchtela
22. května 2013

Poděkování

Chtěl bych poděkovat vedoucímu mé diplomové práce za vedení a poskytnuté konzultace.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Problematika sledování objektu v obraze představuje jeden z typických úkolů vědní disciplíny – *počítačové vidění* (computer vision). Základní skupinu metod tvoří *rekurzivní sledování*, jimiž jsou dobře známé techniky využívané v mnoha aplikacích počítačového vidění. S příchodem nových rychlejších technik v oblasti detekce vznikl k řešení problematiky nový přístup zvaný *sledování detekcí*, který převádí sledování na klasifikační úlohu. Na určitých typech úloh vykazuje výrazně lepší výsledky než tradiční rekurzivní sledovací techniky. Současný vývoj v oblasti sledování směřuje ke kombinaci těchto dvou přístupů a využití jejich hlavních předností: robustnost charakteristická pro sledování detekcí a přesnost charakteristická pro rekurzivní sledování.

V této práci se zaměřujeme na pokročilé přístupy, které pro řešení dlouhodobého sledování využívají kombinace rekurzivního sledování s online učením detektoru. Tyto metody představují komplexní problém, u kterého se budeme snažit představit základní přístupy k řešení jednotlivých podúloh: *sledování objektu*, *detekce objektu* a *online učení* a možnosti jejich kombinace ve funkční celek. Přičemž důraz bude kladem na možnost nasazení těchto technik v aplikacích pracujících v reálném čase. Schopnosti těchto metod budou demonstrovány na navrženém přístupu, který bude podroben různým druhům testování.

Práce je rozčleněna do osmi kapitol. V 2. kapitole jsou popsány základní přístupy kombinace sledování s online učením detektoru. Jakým způsoben se vypořádávají s úkoly sledování, znovudetekování a přizpůsobení se změnám vzhledu objektu. V kapitole je také diskutována základní struktura našeho navrženého přístupu. Přístupy k řešení jednotlivých podúloh: rekurzivní sledování, detekce a online učení jsou popsány ve třech samostatných kapitolách 3, 4 a 5.

Kapitola 3 se zabývá rekurzivním sledováním pomocí optického toku. Je zde vysvětlena Lucas-Kanade metoda výpočtu optického toku a popsány techniky pro zpřesnění sledovacího odhadu, ať již na úrovni výpočtu optického toku nebo na úrovni lokálních příznaků.

V kapitole 4 se zabýváme návrhem kaskádového detektoru. Jsou zde rozebrány techniky klasifikace, jako je náhodný les, náhodné kapradiny či klasifikace založená na porovnávání šablonou. V závěru kapitoly je popsána technika pro potlačení nemaximálních detekcí.

Kapitola 5 popisuje způsob interakce rekurzivního sledování a detektoru. Je zde vysvětlen princip trénování online modelu pomocí P-N učení.

Kapitola 6 se zabývá návrhem testovací aplikace. Je zde popsán konkrétní způsob implementace a použité technologie.

V 7. kapitole je předvedena úspěšnost sledování navrženého přístupu na sadě testovacích videosekvencí. Struktura testů byla navržena k otestování jak činnosti samotného systému, tak i jeho jednotlivých podčástí a jejich vlivu na celkovou úspěšnost dlouhodobého sledování.

Kapitola 2

Sledování-učení-detekce

Učení detektoru pomocí sledování je poměrně nový přístup, který byl úspěšně aplikovaný na problematiku dlouhodobého sledování. Tato technika využívá pro sledování objektu detektor, který je postupně trénován na datech získaných v průběhu celého sledovacího procesu. Tento přístup využívá časoprostorové závislosti v datech k získání nových informací, které jsou využity pro zlepšení sledovacího výkonu. Na rozdíl od klasického přístupu, který operuje vždy a pouze nad dvěma snímky, čímž jsou možnosti k dolování užitečných informací silně omezeny a tím pádem také jeho sledovací výkon.

Tato kapitola diskutuje metody, které jsou vhodné pro úkoly dlouhodobého sledování. Jsou zde také zmíněny výzvy dlouhodobého sledování a problémy, kterými se všechny jmenované metody musí zabývat. Dlouhodobé sledování je komplexní problém, jehož řešení požaduje se vypořádat s úkoly sledování, znovudetekování a přizpůsobení se změnám vzhledu objektu.

2.1 Rekurzivní sledování

Rekurzivní sledování představuje standardní skupinu sledovacích metod, které provádějí odhad pozice objektu mezi dvěma snímky I_{t-1} a I_t. Sledování objektu probíhá pouze na základě informace o jeho vzhledu v referenčním snímku I_{t-1} a předpokládá se, že sledovaný objekt je viditelný po celou dobu sledování. Mezi základní přístupy patří metody, jako je *Lucas-Kanade tracker* [6] nebo *Median flow tracker* [15], které pro odhad cíle využívají optického toku [17]. Rekurzivních sledovacích metod je velké množství [27]. Každá z těchto metod se zaměřuje na optimalizaci různých aspektů zpracování, jako je rychlost [17], přesnost [6], robustnost [15] či jejich vzájemná kombinace. Nicméně žádný z těchto přístupů se přímo nevěnuje problému chování po zanesení chyby, zotavení z chyby nebo problému změny vzhledu objektu v průběhu času.

Nasazení těchto metod na problematiku dlouhodobého sledování se ukazuje jako nevhodné, jelikož jsou bez schopnosti znovudetekování. Znovudetekování je nezbytné, abychom lokalizovali objekt po selhání sledovače, zmizení objektu ze scény nebo po kompletním zakrytí objektu.

2.2 Sledování detekcí

Tyto metody převádějí problém sledování na problém klasifikace [16] nebo kombinují navzájem oba dva přístupy [1]. Všechny tyto metody potřebují znát předem objekt zájmu a

požadují velké množství trénovacích dat pro vytvoření modelu. Většinou je také potřeba znát předchozí informace o objektu pro navržení vhodného detektoru. U těchto metod je striktně oddělena trénovací a testovací fáze a nejsou schopny přizpůsobit model široké variabilitě objektu. Toto oddělení fází prakticky neumožňuje se vypořádat se změnami, proto detektor je úspěšný v detekci dobře naučených příkladů, ale není vhodný pro dlouhodobé sledování, kde se mohou vyskytovat změny vzhledu objektu nebo prostředí.

2.3 Učení detektoru pomocí sledování

Učení detektoru pomocí sledování je poměrně nový přístup aplikovaný na problematiku sledování. Jedná se o metody, které umožňují aktualizovat detektor online v závislosti na kontextu sledování. Tyto metody lze hrubě rozdělit na dvě kategorie podle typu aktualizace. Provedení aktualizace pro každý snímek je nejběžnější pro adaptivní sledovací metody [2]. U kterých se očekává, že pracují správně a pod tímto předpokladem je při každém pozorování provedena aktualizace modelu objektu. Výhodou tohoto přístupu je rychlá schopnost adaptivity, na druhou stranu však urychluje vznik chyby sledování. Selektivní aktualizace bere v úvahu, že sledování nemusí být vždy správné, proto je aktualizace vykonána pouze, pokud výsledek sledovače není daleko od modelu objektu [13].

Jako vhodný typ učení je zde aplikováno učení s částečným dozorem (semi-supervised learning) [28], které používá obojí označená a neoznačená data pro trénování modelu. Označená data jsou generována z prvního snímku, kde je vybrán objekt zájmu (pozitivní příklad představuje vybraná oblast, negativní příklady představují všechny ostatní možné oblasti) a neoznačená data jsou všechny možné oblasti získané během zpracovaní snímku videosekvence. Cílem učení je rozšířit sadu označených dat přeznačením neoznačených dat s pomocí nějakých dozorčích informací.

Nejstarším přístupem pro učení s částečným dozorem je *self-learing*. Na začátku je klasifikátor natrénován na malé označené trénovací sadě. Klasifikátor je potom vyčíslen na neoznačených datech. Příklady s největší důvěrou jsou se svými nově predikovanými značkami přidány do trénovací sady a klasifikátor je přetrénován, tato procedura je opakována.

Co-training předpokládá, že vektor příznaků popisující příklady může být rozdělen do dvou částí, zvané také pohledy; každá část je dostatečná k trénování dobrého klasifikátoru; obě dvě části jsou navzájem statisticky nezávislé. Na počátku jsou oba dva klasifikátory odděleně natrénovány na příslušném pohledu. Každý klasifikátor je potom vyčíslen na neoznačených datech a učí druhý klasifikátor s malým počtem příkladů, jimž predikoval značky s největší důvěrou. Každý klasifikátor je přetrénován s přidanými trénovacími příklady danými z druhého klasifikátoru. Procedura je opakována. V práci [14, 28] ukazují úspěšné aplikování této metody na problémy s dvěma druhy informací, jako je klasifikace textu (text a odkazy), biometrické rozpoznávací systémy (vzhled a zvuk). Nebo v problematice detekce objektu byla nasazena na rozpoznávání pohybujících objektů. V práci [14] argumentují, že co-training není dobrou volbou pro detekci objektu, pokud jsou příklady vzorkovány z jednoho kanálu informací. Příznaky extrahované z jednoho kanálu informací mohou být závislé, což narušuje předpoklad co-trainingu.

Dalším způsoben je metoda kombinující *jeden pohled + vícero učitelů s demokratickým rozhodováním*. Soubor učitelů s různou chybou predikce jsou odděleně trénováni nad kompletními příznaky označených dat. Každý učitel je vyčíslen na neoznačených datech. Pokud se většina učitelů s vysokou důvěrou shodne na třídě neoznačeného příkladu, je tento příklad označen a přidán do trénovacích dat. Všichni učitelé jsou přetrénováni na aktualizované trénovací sadě. Podobnou myšlenkou je řízen *tri-training*, který používá tři učitele. Pokud

dva z nich souhlasí na klasifikaci neoznačeného příkladu, tak klasifikace je použita na učení třetího klasifikátoru. Tento přístup nevyžaduje potřebu explicitního měření důvěry značky od žádného učitele. Lze ho aplikovat na data bez odlišných pohledů nebo odlišných typů klasifikátorů.

Z. Kalal a spol. [14] navrhli metodu zvanou *TLD* (Tracking-learning-detection), která používá příklady ležící na trajektorii k postupnému trénování detektoru. Trajektorie sledovaného objektu je získána pomocí sledovací metody založené na optickém toku. Aktualizace je vykonána pouze, pokud je příklad podobný inicializační obrazové oblasti. Na rozdíl od ostatních přístupů je výstup detektoru využit pouze k reinicializaci sledovače v případě jeho chyby, ale není nikdy použit k aktualizaci svého klasifikátoru.

Techniky využívající výstupu sledovače na trénování modelu provádějí trénování pouze na výsledku s vysokou důvěrou. Nechť trajektorie v čase t produkovaná sledovačem je $T = \{x_0, x_1 \ldots, x_t\}$, potom pro určení důvěry výstupu sledovače, x_t, existují strategie jako:

- Absolutní vzdálenost od prvního příkladu – důvěra je určena jako vzdálenost aktuální oblasti x_t od inicializační oblasti x_0 [13, 14].
- Rozdíl mezi následujícími příklady – důvěra je vypočtena jako vzdálenost mezi aktuální oblastí x_t a předchozí oblastí x_{t-1} [13]. Tato strategie je tolerantní k malým změnám ve vzhledu.
- Vzdálenost od online modelu – důvěra je vypočtena jako vzdálenost aktuální oblasti x_t od online modelu $\mathcal{M}_{t-1}$ [20].
- Interní odhad sledovače – sledovač obsahuje vlastní mechanismus pro odhad důvěry, který je vrácen spolu s predikovaným výstupem x_t [26].
- Časová struktura – představuje retroaktivní přístup, který umožňuje zpětně změnit hodnotu důvěry. Převádí trajektorii sledovače na sekvenci vzdáleností od online modelu a potom hledá jisté vzory v této sekvenci. Tento vzor se nazývá *uzavřená smyčka* a je definován následovně: začíná se od příkladu, který je podobný online modelu, jeho podobnost je θ, a očekává se, že po několika snímcích se opět stane podobný. Tato strategie využívá vlastnost adaptivního sledovače a přitom odolává vzniku driftingu. Pokud se vzhled navrátí, je tu silný předpoklad, že zvýšená změna odlišnosti byla způsobená změnami vzhledu sledovaného objektu či jinými obrazovými odchylkami. Tato strategie je tolerantní k silným vzhledovým odlišnostem, které však stále reprezentují objekt [13]. Tato technika je použitelná v kombinaci s jinými technikami.

Pro úplnost uveďme, že ohodnocení důvěry výstupu detektoru je většinou interní záležitostí jeho samotného, respektive jeho klasifikátoru [14, 13, 20].

2.4 Rámec práce

V rámci této práce následujeme přístup Tracking-learning-detection od Z. Kalali a spol. [14]. Jako rekurzivní sledovací metodu jsme zde zvolili Lucas-Kanade sledovač [6], který je založen na výpočtu optického toku pomocí Lucas-Kande metody [17]. Lucas-Kanade sledovač je rozšířen o techniku měření dopředně-zpětné chyby [15] pro zvýšení robustnosti. Jako detektor jsme zde použili kaskádový detektor, který jsme navrhli jako třístupňový s přihlédnutím ke zvýšení výkonu systému. Kaskáda je složena s variančního filtru [20], kapradinového

klasifikátoru [22] a porovnávající šablonou klasifikátoru, který v rámci detektoru tvoří verifikátora. Detektor a sledovací metoda běží nezávisle vedle sebe. Výstup sledovací metody a detektoru je ohodnocen důvěrou pomocí porovnávajíc šablonou klasifikátoru obsaženého uvnitř detektoru [20]. Klasifikátor si udržuje online model z 15×15 px normalizovaných šablon [13], tyto šablony představují jak pozitivní příklady, tak i negativní příklady. Sloučením výsledku sledovací metody a detektoru na základě velikosti důvěry je získán finální výsledek. Pokud výsledek sledovací metody má malou důvěru a důvěra k výsledku detektoru je vyšší, je provedena reinicializace sledovače. Trénování modelu probíhá pouze, pokud jako finální výsledek je vybrán výsledek sledovače a jeho důvěra je vysoká. Pro trénování je použito P-N učení [14], které je aplikováno jak na kapradinový klasifikátor, tak i na porovnávající šablonou klasifikátor. Tato procedura je opakována pro každý snímek videosekvence.

Přístup sledování-učení-detekce se skládá z jednotlivých podúloh jako je sledování, detekce a P-N učení včetně hypotézy sloučení výsledků, kterým se podrobně věnují následující kapitoly 3, 4 a 5.

2.5 Shrnutí

V této kapitole byly rozebrány základní přístupy k problematice sledování objektu. Přičemž zvláštní pozornost byla věnována oblasti učení detektoru pomocí sledování, kde byly naznačeny různé strategie řešení. Jedná se o techniky, díky kterým získáváme schopnosti pro dlouhodobé sledování a zároveň pro budování spolehlivého online modelu. Kombinace krátkodobého sledování a online učení detektoru vykazuje výrazně lepší výsledky [14], než je tomu u běžných metod popsaných v sekci 2.1 a 2.2.

Kapitola 3

Sledování objektu založené na optickém toku

Tato kapitola se zabývá problematikou krátkodobého sledování objektu ve videu, konkrétně rekurzivními metodami, u kterých nejsou požadovány žádné předchozí informace o vzhledu objektu a očekávají pouze pozici sledovaného objektu v předchozím snímku. Získání pozice sledovaného objektu v předchozím snímku vyžaduje externí inicializaci. V našem případě je inicializace vykonána manuálně vyznačením objektu v prvním snímku a následně výsledky produkovanými detekčním mechanismem probíhajícím nad dalšími snímky videosekvence.

Do problematiky krátkodobého sledování objektu spadá velké množství metod, které jsou v rozsáhlém díle [27] rozděleny do tří hlavních kategorií: *bodové sledování* (*point tracking*), *jádrové sledávání* (*kernel tracking*) a *sledování siluety* (*silhouette tracking*). V naší práci se zaměříme na druhou kategorii, tedy kernel tracking, konkrétně na sledovací metodu známou jako Lucas-Kanade tracker [6], která je doplněna o přístup dopředně-zpětné chyby [15].

3.1 Lucas-Kanade tracker

Lucas-Kanade tracker [6] je metoda, která pro sledování objektu mezi dvěma snímky využívá odhad optického toku. Pro výpočet optického toku je zde využita diferenciální Lucas-Kanade metoda [17], která oproti jiným technikám pro výpočet optického toku jako např. block matching metoda [24] či Horn-Schunck metoda [11] vykazuje obecně lepší výsledky z pohledu přesnosti a rychlosti zpracování, jak je demonstrováno v rozsáhlé práci [3].

3.1.1 Výpočet optického toku

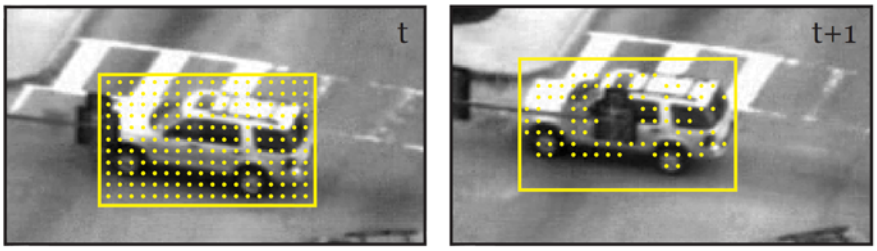
Lucas-Kanade metoda [17] pro výpočet optického toku je lokální diferenciální technika, která je založena na třech předpokladech. Prvním předpokladem je jasová stálost obrazu a je vyjádřen jako

$$I(X) = J(X + d)\,, \tag{3.1}$$

kde pixel v obraze I na pozici o dvoudimenzionálních souřadnicích X může změnit svou pozici v obraze J, ale jasová hodnota musí zůstat stejná. Vektor d je vektor posunutí.

Druhým předpokladem je časová stálost. To značí, že vektor posunutí je malý. Což v tomto případě znamená, že $J(X)$ může být aproximována

$$J(X) \approx I(X) + I'(X)d\,, \tag{3.2}$$



Obrázek 3.1: Princip rekurzivní sledovací metody. Metoda očekává ohraňující obdélník a dva snímky. Je vygenerován určitý počet bodů uvnitř obdélníku, které jsou sledovány použitím optického toku, je pro ně odhadnuta chyba a nespolehlivé body jsou odfiltrovány. Ze zůstavších bodů je odhadnut nový ohraňující obdélník. Obrázek převzat z [15].

kde $I'(X)$ je gradient obrazu I na pozici X. Odhad vektoru d je potom

$$d \approx \frac{J(X) - I(X)}{I'(X)}. \tag{3.3}$$

Pro daný libovolný pixel je rovnice nedourčená a řešením prostoru je přímka, resp. množina bodů ležící na této přímce. Třetí předpoklad, též známý jako prostorová koherence, do jisté míry řeší tento problém. Značí, že všechny pixely uvnitř okna okolo uvažovaného pixelu se pohybují souvisle. Po zařazení tohoto předpokladu, d je nalezen minimalizací výrazu

$$\sum_{x,y \in W} (J(X) - I(X) - I'(X)d)^2, \tag{3.4}$$

což je řešení přeurčených rovnic pomocí minimalizace nejmenších čtverců. Velikost W definuje uvažovanou oblast okolo každého pixelu. V [3] je ukázáno, že uzavřená forma řešení 3.4 je

$$Gd = e\,, \tag{3.5}$$

kde

$$G = \sum_{X \in W} I'(X)I'(X)^T = \sum_{X \in W} \begin{pmatrix} I_x^2(X) & I_{x,y}(X) \\ I_{x,y}(X) & I_y^2(X) \end{pmatrix} \tag{3.6}$$

a

$$e = \sum_{X \in W} (I(X) - J(X))I'(X) = \sum_{X \in W} \begin{pmatrix} (I(X) - J(X))I_x(X) \\ (I(X) - J(X))I_y(X) \end{pmatrix}, \tag{3.7}$$

kde I' je derivace obrazu I s ohledem na X a je definovaný jako sloupcový vektor $I' = [I_x\ I_y]^T$, I_x je parciální derivace obrazu I dle x-ové osy, I_y je parciální derivace obrazu I dle y-ové osy a I_{xy} je parciální derivace obrazu I dle x-ové a y-ové osy.

Výsledkem výpočtu optického toku je vektor d, který určuje posunutí pixelu mezi obrazem I a J.

Odhad vektoru d závisí na lineární aproximaci chování funkce $I(X)$ v okolí X, tady je aproximace vyjádřena v podobě funkce $I'(X)$, rovnice 3.3. Úspěch této techniky vyžaduje, aby posuv d byl dostatečně malý. Pro zpřesnění našeho odhadu posuvu d je zde aplikován iterativní přístup, kdy funkci $J(X)$ posuneme k funkci $I(X)$ přes odhadnutý vektor d, vypočteme nový odhad d a tuto proceduru opakujeme. Ideálně by naše sekvence odhadů

d měla konvergovat k nejlepšímu d. Princip iterativního vylepšování odhadu d je ukázán v alg. 1, jelikož vektor optického toku d není celé číslo je nutné pro posunutí obrazu J k I použít pokročilejší techniky warpování (např. využívající bilineární interpolaci).

Algoritmus 1 Iterativní Lucas-Kanade metoda

Vstup: I, J : obrázek
Výstup: $D = \{d_1 \dots d_n\}$: vektory opt. toku, d_i koresponduje s pixelem p_i
  $D \leftarrow \mathrm{LK}(I, J)$ ▷ výpočet opt. toku LK pro každý pixel, rov. 3.5
  opakuj
    $warpObraz \leftarrow$ nawarpování obrazu J směrem k obrazu I využívající D
    $D' \leftarrow \mathrm{LK}(I, warpObraz)$ ▷ výpočet korekce v toku
    $D \leftarrow \{d_i + d'_i \mid \forall i : d_i \in D,\ d'_i \in D'\}$ ▷ přičtení korekce
  dokud konverguje D **a zároveň** počet iterací $<$ MAX_ITER

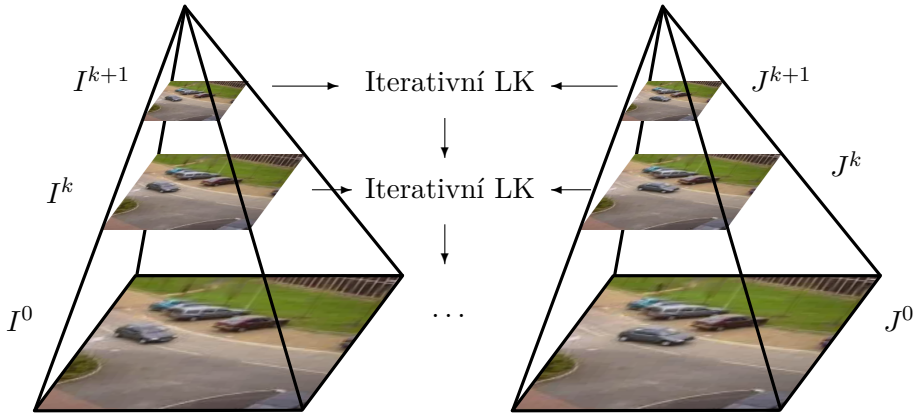
3.1.2 Hierarchický přístup

V praxi při zpracování časové sekvence obrázků je běžné, že se zde mohou často vyskytovat pohyby přes velké oblasti obrazu. Pokud se objekt nebo kamera posunuje rychle, tak by pohyby nastaly pravděpodobně všude v obraze. Tento problém je dále zvýrazněn, když snímkovací frekvence videosekvence je příliš malá, aby mohl být pohyb zachycen na více snímcích. U metod založených na porovnávání bloků jsou vypočtené hodnoty optického toku nepoužitelné, pokud pohyb v obraze je větší než vyhledávací oblast. To je také problém diferenciálních metod výpočtu optického toku, kde přesnost prostorových a časových gradientů se snižuje s velkými pohyby.

Běžným řešením je použití *hierarchického* přístupu (též označován jako *pyramidální*) [21, 6, 24], pomocí něhož je ze vstupního snímku vytvořena série snímků se snížením prostorové frekvence, např. snížením rozlišení. Výsledkem je tedy sada obrázků v různých rozlišeních, které jsou uspořádány v pyramidální datové struktuře, jako je Gaussova či Laplacova pyramida [8]. Pyramidální přístup aplikovaný na výpočet optického toku je ilustrován na obr. 3.2. Při výpočtu optického toku je nejprve odhadnuta hodnota optického toku v obraze s nejnižším rozlišením, který se vyskytuje na vrcholu pyramidy, tato hodnota koresponduje s většími pohyby. Takto získané hodnoty jsou následně promítnuty směrem dolů do další úrovně pyramidy, kde jsou použity jako inicializační hodnoty, které jsou potom dále vylepšovány. Hodnoty jsou jedenkrát vylepšeny a poté jsou dále vloženy do další úrovně. Takto proces pokračuje, dokud nedosáhneme spodní úrovně pyramidy nebo původního vstupního obrazu. Matematicky to je identické k iterativnímu zlepšování, jež očekává, že každý odhad v daném měřítku musí být nadvzorkován a interpolován (nejčastěji bilineární interpolace), než bude dále zpracován v následující kvalitnější úrovni. Algoritmická verze je dána v alg. 2.

3.1.3 Výběr příznaků

Většina metod výpočtu optického toku je navržena pro výpočet hustého pole optického toku, což znamená, že vektor toku je produkován pro významný počet pixelů (např. pro každý pixel). Výsledky testů [3] však ukazují, že ne každý pixel je vhodný pro výpočet. Přesnost u mnoha takto vypočtených vektorů je diskutabilní. V homogenních oblastech vektory toku jsou nedefinované a mohou produkovat nespolehlivé výsledky. V oblastech s hranami může být nalezena pouze ortogonální hodnota pro vektor optického toku (též známá jako normála



Obrázek 3.2: Výpočet optického toku iterativní metodou Lucas-Kanade s pyramidou. Optický tok je získán integrací optických toků v každé úrovni pyramidy.

Algoritmus 2 Pyramidální iterativní Lucas-Kanade metoda

Vstup: I, J : obrázek v pyramidě s K úrovněmi
Výstup: $D = \{d_1 \dots d_n\}$: vektory opt. toku, d_i koresponduje s pixelem p_i

```
D_K ← iterLK(I^K, J^K)                    ▷ výpočet iterativní LK v nejvyšší úrovni
pro úroveň k od K − 1 do 0 dělej
    vem optický tok D_{k+1} z úrovně k + 1
    vytvoř matici D*_k pro úroveň k o dvakrát větším rozlišení než k + 1
    přiřaď hodnoty z D_{k+1} do D*_k aplikováním bilineární interpolace
    vektory D*_k vynásob číslem 2
    warpObraz_k ← nawarpuj obrázek J^k směrem k obrázku I^k s využitím D*_k
    D'_k ← iterLK(I^k, warpObraz_k)       ▷ výpočet korekce v toku
    D_k ← {d*_i + d'_i | ∀i : d*_i ∈ D*_k, d'_i ∈ D'_k}      ▷ přičtení korekce
konec pro
D ← D_0
```

optického toku). Komponenta podél hran chybí kvůli tzv. *štěrbinovému problému* (*aperture problem*). V obraze pouze tyto body, které obsahují dostatek gradientů, nejméně ve dvou směrech, budou produkovat spolehlivé vektory toku.

Intuitivním řešením je zde jednoduše ignorovat všechny nespolehlivé vektory a zabývat se pouze některými. Toto řešení je motivováno metodami založenými na příznacích a přinášejí další dva podproblémy, jako je detekce příznaků a určení jejich korespondence mezi obrazy.

Základní podmínkou pro výběr bodů vyplívá přímo z definice 3.5. Vektor optické toku d může být vypočten pouze, když je matice G invertibilní (není singulární). Matice je invertibilní pokud platí:

$$min(\lambda_1, \lambda_2) \neq 0\,, \tag{3.8}$$

kde λ_1 a λ_2 jsou vlastní hodnoty matice G. Aby byla matice G dostatečně přesná, je potřeba mít velké obě dvě hodnoty λ_1 a λ_2, to nastává v případě, kdy existují gradienty ve dvou směrech. Proto my zde používáme přísnější podmínku

$$min(\lambda_1, \lambda_2) > \lambda \tag{3.9}$$

jako první kritérium pro spolehlivé sledování bodů.

Další možnou metodou pro výběr vhodných bodů pro sledování, které využívají vysoké křivosti dvou dimenzionálních příznaků (např. rohy), je použití běžných detekčních příznakových metod, jako je Moravcův operátor [19] nebo jeho vylepšená verze zvaná Plesseyův operátor navržený Harrisnem a Stephensnem [9].

3.1.4 Korespondence příznaků

Pro porovnávání příznakových bodů mezi obrazy bývají nejčastěji použity korelační metody [21] jako:

- Suma absolutních hodnot rozdílů (SAD) – je jedna z nejjednodušších metod měření podobnosti, která je vypočtena jako suma rozdílů pixelů z čtvercového okolí P_1 pixelu p_1 a pixelů z čtvercového okolí P_2 pixelu p_2 mezi referenčním snímkem I a cílovým snímek J. Pokud jsou oblasti v obou obrázcích stejné, potom je výsledkem roven nule.

$$\mathrm{SAD}(P_1, P_2) = \sum_{x=1}^{N} |P_1(x) - P_2(x)| \tag{3.10}$$

- Suma čtverců rozdílů (SSD) – je spočtena jako suma druhých mocnin rozdílů pixelů z čtvercového okolí P_1 pixelu p_1 a pixelů z čtvercového okolí P_2 pixelu p_2 mezi referenčním snímkem I a cílovým snímek J. Tato metoda větší měrou penalizuje pixely si méně podobné než pixely si více podobné. SSD podává dobré výsledky, pokud obě dvě korespondující oblasti mají stejnou intenzitu a kontrast.

$$\mathrm{SSD}(P_1, P_2) = \sum_{x=1}^{N} \Big((P_1(x) - P_2(x))\Big)^2 \tag{3.11}$$

- Normalizovaná vzájemná korelace (NCC) – výhodou této metody oproti výše zmíněným je, že je invariantní vůči lokálním změnám jasové intenzity. Obor hodnot leží v intervalu [0,0; 1,0], kde 1 značí, že jsou dané oblasti stejné.

$$\mathrm{NCC}(P_1, P_2) = \frac{\sum_{x=1}^{N} P_1(x) \cdot P_2(x)}{\sqrt{\sum_{x=1}^{N} P_1^2(x) \cdot \sum_{x=1}^{N} P_2^2(x)}} \tag{3.12}$$

Všechny tyto zmíněné korelační metody vykazují často dobré výsledky. Dále ještě bývají zpracovány a vyhlazeny, ale korelace je základní metoda pro určení podobnosti. V naší aplikaci jsme se přiklonili k použití metody NCC a jako velikost okolí jsme zvolili 10×10 pixelů.

3.1.5 Dopředně-zpětné měření chyby

Při sledování bodů se často setkáváme s problémem, kdy se vzhled bodů dramaticky mění či mizí mimo zorné pole kamery. Pod těmito podmínkami proces sledování vykazuje vysokou chybovost. V práci [15] je pro jejich eliminaci navrhována *dopředná* a *zpětná* technika. Tato technika je založena na tzv. dopředné-zpětné stálosti, která předpokládá, že správné sledování by mělo být nezávislé na směru časového toku.

Metoda je rozdělena do tří kroků. Nejprve je provedeno klasické dopředné sledování bodů. Za druhé probíhá validace trajektorie, nově získané polohy bodů představují počáteční body trajektorie a je na ně aplikován sledovací proces ve zpětném směru (z posledního snímku na první snímek). Posledním krokem je změření dopředně-zpětné chyby porovnáním originální a zpětně získané trajektorie, pokud jsou totožné (podobné), je originální trajektorie prohlášena za validní, v opačném případě je prohlášena za defektní.

Navržené dopředně-zpětné měření chyby ϵ je definované jako euklidovská vzdálenost

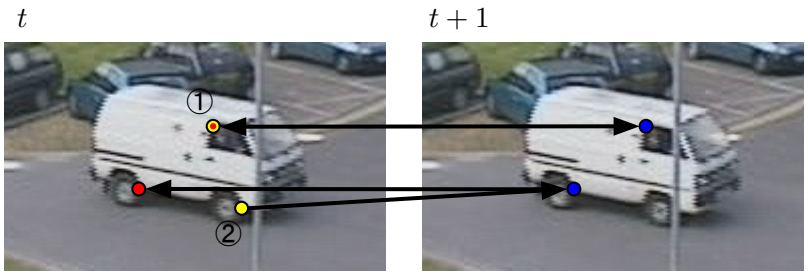
$$\epsilon = |p - p''|\,, \tag{3.13}$$

kde p'' je

$$p'' = LKtrack(LKtrack(p))\,, \tag{3.14}$$

což znamená, že metoda Lucas-Kanade je aplikovaná dvakrát na p.

Dopředně-zpětné měření chyby je použito ve spojení s dalším měřením založené na podobnosti oblastí obklopující p a oblasti obklopující výsledek sledování p'. Podobnost těchto dvou oblastí P_1 a P_2 je porovnána použitím Normalizované vzájemné korelace NCC 3.12.



Obrázek 3.3: Dopředně-zpětná metoda sloužící pro validaci trajektorie sledovaných bodů. Myšlenka vychází z pozorování, že jisté body nemohou být sledovány do své původní pozice. Převzato v upravené podobě z [15].

3.1.6 Extrakce modelu

Dle přístupu Z. Kalala a spol. [15] je pro všechny body vypočtena dopředně-zpětná chyba a změřena podobnost mezi referenčními a cílovými body. Po té je vypočten medián $median_{FB}$ všech dopředně-zpětných chyb a medián $median_{NCC}$ všech podobností a je provedena filtrace bodů, tak že se uchovají pouze ty body, které mají dopředně-zpětnou chybu menší než $median_{FB}$ a změřenou podobnost větší než $median_{NCC}$. Pokud je hodnota $median_{FB}$ větší než předdefinovaný práh τ_{FB}, pak proces sledování ukončíme bez výsledku, což lze interpretovat jako nespolehlivý výsledek sledování, v opačném případě jsou zůstavší body použity k odhadu posunutí celého ohraničujícího obdélníku.

Odhad výsledného obdélníku vychází z velikosti a pozice původního obdélníku, pozice referenčních bodů a pozici cílových bodů. Pro všechny páry referenčních bodů jsou vypočteny vzdálenosti, které jsou zprůměrovány do hodnoty $avrgDist_1$, pro všechny páry cílových bodů jsou vypočteny vzdálenosti, které jsou zprůměrovány do hodnoty $avrgDist_2$. Získaný poměr $scale = avrgDist_2/avrgDist_1$ lze interpretovat jako změnu velikosti. Posunutí ve

směru x-ové osy je spočítáno jako průměr horizontálního posuvu mezi korespondujícími body. Posunutí ve směru y-ové osy je spočítáno obdobně.

Algoritmická verze navržené sledovací metody je popsána v alg. 3. Pro generování bodů zde používáme pravidelnou mřížku $m \times n$, v oblasti vyznačené ohraňujícím obdélníkem je tedy vygenerováno $m \cdot n$ bodů určených ke sledování, typicky $m = n, m \in \{5, 10, 15\}$.

Algoritmus 3 Sledování objektu - Median flow metoda

Vstup: B_I : ohraničující obdélník, I, J : obrázek
Výstup: B_J : ohraničující obdélník

```
p_1 ... p_n ← generováníBodů(B_I)
pro všechny p_i dělej
    p'_i ← pyramidLKSledování(p_i)
    p''_i ← pyramidLKSledování(p'_i)
    ε_i ← |p_i − p''_i|
    η_i ← NCC(W(p_i), W(p'_i))
konec pro
median_FB ← medián(η_i ... η_n)
median_NCC ← medián(ε_i ... ε_n)
jestliže median_FB > τ_FB pak
    B_J ← ∅
jinak
    pro všechny p_i | p'_i ≠ ∅ ∧ ε_i < median_FB ∧ η_i > median_NCC dělej
        C ← (p_i, p'_i)
    konec pro
    B_J ← výpočetOhraničujícíhoObdélníku(B_I, C)
konec jestliže
```

3.2 Shrnutí

V této kapitole jsme se zabývali metodou, která je volána snímek po snímku pro sledování objektu našeho zájmu. Metoda nevyžaduje žádné předchozí informace o vzhledu objektu a očekává pouze pozici objektu v předchozím snímku. Je zde vysvětlen princip extrakce výsledného ohraničujícího obdélníku, který je odhadnut z optického pole. Optické pole je spočítáno iterativní diferenciální Lucas-Kanade metodou doplněnou o pyramidální přístup, díky čemuž je metoda schopna odhadnout optický tok i pro větší pohyby. Je zde detailně vysvětlena metoda Lucas-Kanade, její nedostatky a navržené přístupy pro eliminaci chyb optického toku vyplívající jednak z nedodržení předpokladů, na nichž je metoda formulovaná, jednak ze zvoleného matematického aparátu pro řešení přeurčené soustavy rovnic. Jako hlavní kritéria pro zvýšení spolehlivosti bylo uvedeno měření podobnosti mezi korespondujícími body pomocí korelace a jako další kritérium pro validaci cílových bodů byla představena metoda měření dopředně-zpětné chyby.

Kapitola 4

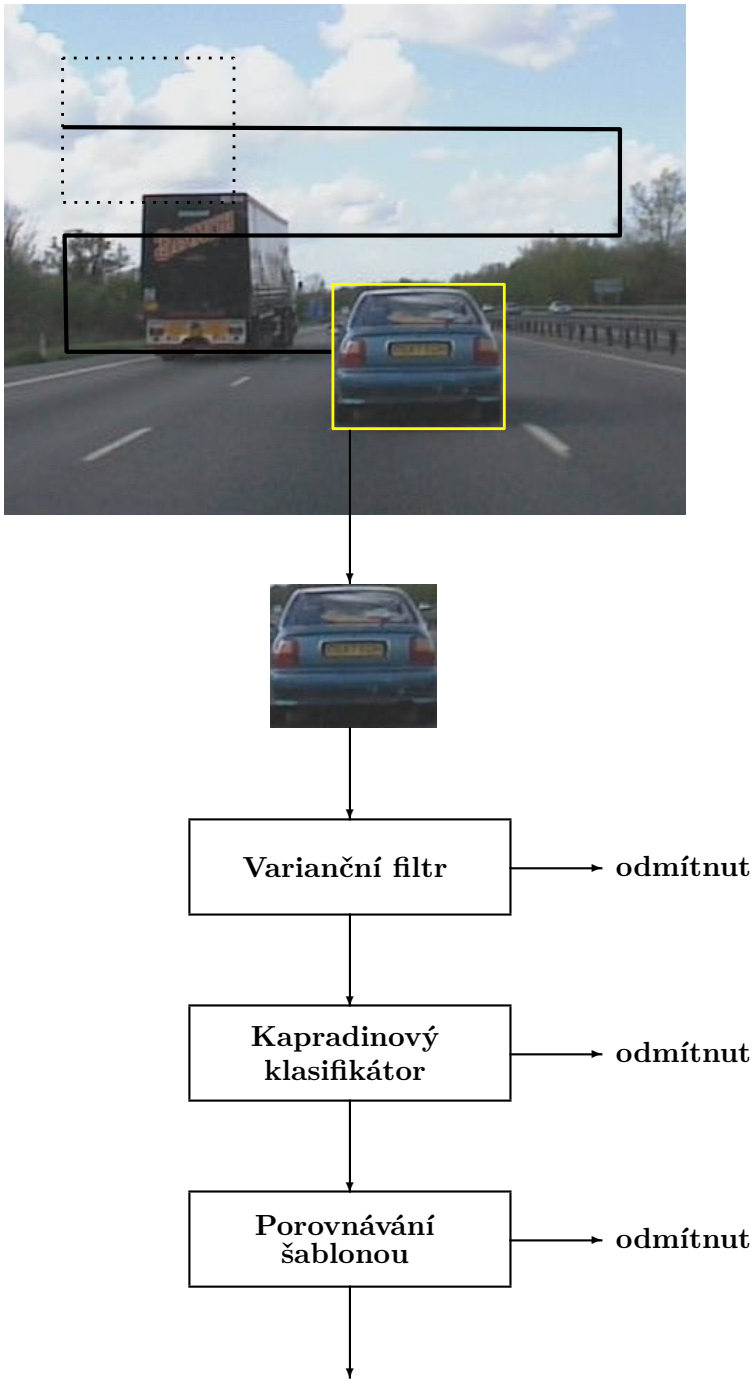
Detekce objektu kaskádou klasifikátorů

Tato kapitola se zabývá problematikou detekce objektu. Diskutujeme zde metodu, kterou rozvíjíme pro detekování objektu. Detekce objektu nám umožňuje provedení reinicializace rekurzivní sledovací metody, která si neudržuje model objektu, a proto není schopna se zotavit z chyby. Zatím co sledovací metoda závisí na pozici objektu v předchozím snímku, tak detekční mechanismus zde prezentovaný provádí podrobné hledání v celém obraze, aby nalezl objekt.

Běžným přístupem k obecné detekci/lokalizaci objektu je použití klouzavého okna, které je posouváno po obraze a každé takové lokální okno je klasifikováno jako pozadí nebo cíl. Tento přístup byl úspěšně použit v aplikacích [25, 23, 14]. Přirozeným rozšířením tohoto přístupu je použití klasifikátorů s klouzavým oknem k detekci částí objektu a po té sestavení těchto částí do celého objektu [18]. Dalším populárním přístupem je extrakce lokálních bodů zájmu z obrazu a klasifikace každého z regionů okolo těchto bodů, raději než by se hledalo ve všech možných podoknách [5].

V naší práci používáme detektor založený na technice klouzavého okna. Při této technice dochází ke generování určitého počtu podoken, ten je závislý na inicializační velikosti objektu, velikosti kroku a počtu velikostí klouzavého okna. Pro každý snímek o rozlišení 640×480 je to typicky 50 000 až 200 000 podoken, které musí být vyčísleny, což pro systém představuje výraznou výpočetní zátěž. Pro snížení výpočetní náročnosti se jako možné řešení ukazuje využití strategie kaskády [25]. Kaskáda je sekvence postupně všestrannějších klasifikátorů. Každý stupeň kaskády je navržen, aby odmítl vysoké procento všech negativních příkladů přicházejících do tohoto stupně, zatímco všechny pozitivní příklady zůstávají a jsou vloženy do dalšího stupně. Snahou kaskády je zamítnutí co největšího množství negativních příkladů s co nejmenším výpočetním úsilím.

Strategie kaskády byla přejata i do našeho přístupu. Náš detektor jsme navrhli jako třístupňový kaskádový detektor, ukázán na obr. 4.1. Prvním stupněm kaskády je varianční filtr, který propouští podokna, které mají vysoký rozptyl obrazové funkce. Druhý stupeň je tvořen kapradinovým klasifikátorem (ferns classifier), který představuje souborovou metodu složenou z náhodných kapradin. Poslední třetí stupeň kaskády je založen na metodě porovnávání šablonou (template matching), která pro měření podobnosti používá normalizovanou korelaci (NCC). V závěrečném zpracování se zabýváme technikou pro řízení detekovaných navzájem se překrývajících podoken pomocí strategie na potlačení nemaximálních detekcí. Výsledkem této strategie je podokno, které nejlépe reprezentuje příslušnou skupinu.



Obrázek 4.1: Navržený kaskádový detektor. Každé vygenerované okno prochází tří stupňovou kaskádou, při jejímž průchodu je v každém stupni v závislosti na výsledku testu odmítnuto nebo předáno do dalšího stupně. Po úspěšném průchodu celé kaskády je obrazová oblast klasifikována jako objekt.

4.1 Technika klouzavého okna

Přístup založený na technice klouzavého okna [25], kdy při posuvu po obraze dochází ke generování množiny oblastí v různých velikostech. Při tomto procesu dochází ke generování obrovského množství oblastí, které dramaticky ovlivňují výkon celého systému. Pro zredukování počtu podoken zde využíváme proporcionální strategii, jež předpokládá určitou stálost velikosti objektu.

Při proporcionální strategii je velikost klouzavého okna volena v závislosti na velikosti původního inicializačního obdélníku vybraného manuálně uživatelem v prvním snímku videosekvence. Měřítko zvětšení pro klouzavé okno, které je aplikované na velikost původního obdélníku, je typicky definované exponenciální funkcí, takovými příklady jsou: $scale = 1{,}2^a, a \in [-10, 10] \subset N$ z [20] nebo $scale = 1{,}1^a, a \in [-5, 5] \subset N$ z [26]. Minimální velikost klouzavého okna je omezena na hodnotu 20 pixelů ve smyslu $min(width, height) \geq 20$. Posuv klouzavého okna je vždy nastaven proporcionálně k jeho aktuální velikosti. V práci [26] navrhují krok v x-ové ose nastavit na 10% šířky klouzavého okna, $\Delta x = 0{,}1 \cdot width$, a krok v y-ové ose nastavit na 10% výšky klouzavého okna, $\Delta y = 0{,}1 \cdot height$. My navrhujeme x-ový a y-ový posuv nastavit na 10% té z menší strany, tj. $\Delta x = \Delta y = 0{,}1 \cdot min(width, height)$. Velikost množiny všech podoken $\mathcal{S}$ zredukované navrženými omezeními je potom [20]:

$$|\mathcal{S}| = \sum_{scale \in 1,2^{\{-10,\ldots,10\}}} \left\lfloor \frac{imgW - scale(w + \Delta x)}{scale \cdot \Delta x} \right\rfloor \left\lfloor \frac{imgH - scale(h + \Delta y)}{scale \cdot \Delta y} \right\rfloor , \tag{4.1}$$

kde $imgW$ je šířka snímku, $imgH$ je výška snímku, w a h jsou šířka a výška inicializačního obdélníku.

4.2 Varianční filtr

V této fázi je změřena variance každé obrazová oblast pro zjištění variability. Měření variability je rychlým způsobem získání základní charakteristiky obrazové oblasti, která umožňuje vzájemné porovnání dvou obrazových oblastí. V této sekci je představena technika výpočtu variability. Pomocí níž dochází k filtrování oblastí, při kterém jsou přijaty pouze ty oblasti, které mají hodnotu variability vyšší než var_{min} a zároveň menší než var_{max}. Pomocí variančního filtru dochází k významné redukci testovaných oblastí s nízkým výpočetním úsilím. Neumožňuje však od sebe rozlišit dobře strukturované objekty. Variabilita má několik pro nás vítaných vlastností, jako je invariantnost vůči změně intenzity světla a vůči afinním transformacím.

Předpis pro přijetí obrazové oblasti P ve variančním filtru je definován splněním podmínky:

$$0{,}5 \cdot var_{init} < \text{variabilita}(P) < 2 \cdot var_{init} , \tag{4.2}$$

kde variabilita(P) je variabilita obrazové oblasti P a var_{init} je konstanta, která je nastavena na hodnotu variability inicializačního obdélníku P_{init}, $var_{init} = variabilita(P_{init})$. Např. v práci [20] mají předpis definovaný pouze spodní hranicí, naše testy však ukazují, že definicí doplněnou o horní omezení lze účinnost filtrace významně zvýšit, aniž by to mělo negativní vliv na úspěšnost detekce. Na obr. 4.2 je ilustrovaná účinnost tohoto filtru v počtu redukovaných obdélníků.

Pro určení variability existují mnohé míry, v naší práci uvádíme dvě nejběžnější:

- *Variační rozpětí* – je jednou z nejjednodušší charakteristiky variability. Varianční rozpětí R je rozdílem mezi největší hodnotou, max_P, a nejnižší hodnotou, min_P, obrazové plochy P, vyjádřeno vztahem:

$$R(P) = max_P - min_P\,. \tag{4.3}$$

  Variační rozpětí udává interval, ve kterém se pohybují jednotlivé hodnoty obrazové plochy. Nevýhodou této míry, že je počítána jen na základě dvou krajních hodnot, které mohou být nahodilé a extrémní. Zároveň nezohledňuje nikterak vnitřní proměnlivost hodnot v oblasti.

- *Variance* též *rozptyl* – je nejpoužívanější mírou pro vyjádření charakteristiky variability. Je počítána ze všech hodnot obrazové oblasti, při které se zohledňuje odlišnost jednotlivých hodnot od střední hodnoty obrazové oblasti a zároveň odlišnost jednotlivých hodnot obrazové plochy vůči sobě navzájem. Rozptyl σ^2 je definován jako aritmetický průměr ze čtverců odchylek jednotlivých pixelů, vyjádřeno vztahem:

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2, \tag{4.4}$$

  kde x je hodnota pixelu, n je počet pixelů v obrazové ploše P, $n = |P|$, a μ je střední hodnota obrazové plochy definovaná:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i\,. \tag{4.5}$$

  Alternativní zápis rov. 4.4 je:

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} x_i^2 - \mu^2. \tag{4.6}$$

Obrázek 4.2: V levém obrázku jsou ukázány oblasti P_3, P_4 s vysokou mírou variability a oblasti P_1, P_2 s nízkou mírou variability. Vpravo na obrázku je vyobrazena mapa variancí σ^2 spočítaná pro obdélníkové oblasti o velikosti 80×70 pixelů. Obdélníky jsou v mapě referencované svým středem. V případě, že bychom oblast P_3 vzali jako referenční, poté bychom podle rov. 4.2 mohli zredukovat počet všech obdélníkových oblastí o velikosti 80×70 pixelů cirka o 90%.

Pro výpočet variance je možné použít efektivní mechanismus popsaný v práci [20]. Běžný přístup pro výpočet variance rov. 4.6 pro obrazovou oblast o velikosti n pixelů vyžaduje n přístupů do paměti. Pokud vezmeme v úvahu, že se zde vyskytuje množství překrývajících se oblastí, které jsou opakovaně vyčíslovány zvlášť pro každou oblast, tak je možné výpočet urychlit eliminací těchto redundantních operací. V práci [20] je ukázán způsob výpočtu variance σ^2 pro obrazovou oblast za použití dvou integrálních obrazů vzniknuvších ze vstupního obrazu I. Tento způsob vyžaduje nově pouze 8 přístupů do paměti.

Integralní obraz někdy taky znám jako *tabulka sumované oblasti* představuje efektivní techniku, s jejíž pomocí je možné součet bodů pod libovolným obdélníkem v obraze spočítat v konstantním čase. Výpočet vyžaduje pouze 4 přístupy do paměti a 3 aditivní operace (1× součet, 2× rozdíl). Způsob získání součtu hodnot bodů uvnitř ohraničené oblasti v integrálním obraze je ilustrován na obrázku 4.3. Integrální obraz má stejné rozměry jako původní obrázek a pro každý bod integrálního obrazu o souřadnicích (x, y) platí rovnice:

$$I_\Sigma(x, y) = \sum_{x' \leq x, y' \leq y} I(x', y') \,. \tag{4.7}$$

Integrální obraz je spočítán v jednom průchodu přes obraz $I(x, y)$ dle předpisu:

$$I_\Sigma(x, y) = I(x, y) + I_\Sigma(x - 1, y) + I_\Sigma(x, y - 1) - I_\Sigma(x - 1, y - 1) \,. \tag{4.8}$$

Potom součet bodů v oblasti definované obdélníkem P s parametry (x, y, w, h) je dán předpisem:

$$\Sigma(P) = I_\Sigma(x+w-1, y+h-1) - I_\Sigma(x+w-1, y-1) - I_\Sigma(x-1, y+h-1) + I_\Sigma(x-1, y-1), \tag{4.9}$$

kde x, y jsou souřadnice levého horního rohu ohraničené oblasti P, w a h je šířka a výška ohraničené oblasti P.

Efektivní metoda výpočtu variance tedy využívá integrální obraz $I_\Sigma(x, y)$ a integrální obraz, který je konstruován nad čtverci hodnot obrazu $I(x, y)$ dle předpisu:

$$I'_\Sigma(x, y) = \sum_{x' \leq x, y' \leq y} I(x', y')^2. \tag{4.10}$$

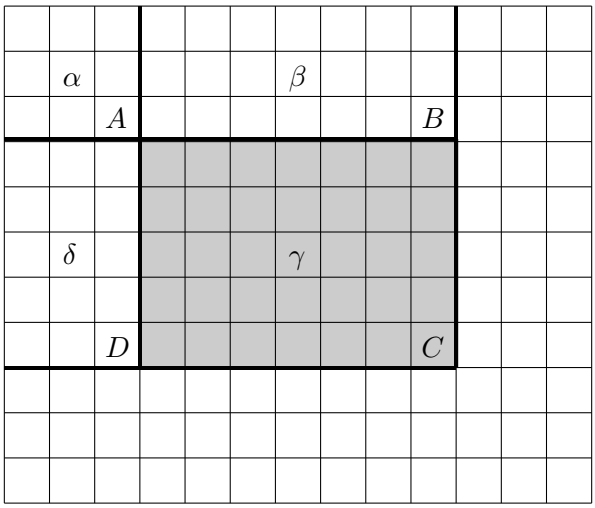
Analogicky s rov. 4.9 definujeme součet čtverců hodnot bodů v oblasti ohraničené obdélníkem P předpisem:

$$\Sigma'(P) = I'_\Sigma(x+w-1, y+h-1) - I'_\Sigma(x+w-1, y-1) - I'_\Sigma(x-1, y+h-1) + I'_\Sigma(x-1, y-1). \tag{4.11}$$

Dosazením rov. 4.9 a rov. 4.11 do rov. 4.6 dostáváme předpis:

$$\sigma^2 = \frac{1}{n}\Sigma'(P) - \left(\frac{1}{n}\Sigma(P)\right)^2. \tag{4.12}$$

Varianční filtr nasazený v prvním stupni našeho kaskádového detektoru porovnává obrazové oblasti na základě jejich variability, zde vyjádřené v podobě variance σ^2. Pro výpočet variance je použita rov. 4.12, která využívá jen 8 přístupů do paměti a tím významně šetří čas výpočtu procedury.



Obrázek 4.3: Pomocí integrálního obrazu může být jednoduše spočítán součet hodnot bodů uvnitř oblasti γ jako $\gamma = C - D - B + A$. Hodnota integrálního obrazu v bodě A je rovna součtu bodů v oblasti α, v bodě B je hodnota integrálního obrazu rovna $\alpha + \beta$, v bodě D je hodnota integrálního obrazu rovna $\alpha + \delta$ a bodě C je hodnota integrálního obrazu rovna $\alpha + \beta + \gamma + \delta$.

4.3 Kapradinový klasifikátor

V této sekci se zabýváme kapradinovým klasifikátorem (ferns classifier), který tvoří druhý stupeň kaskádového detektoru. V této fázi dochází ke změření příznaků v každé vstoupivší obrazové oblasti aplikováním příznakových operátorů. Na základě detekovaných příznaků je obrazové oblasti přiřazena pravděpodobnost, která vyjadřuje míru důvěry, že jde o objekt našeho zájmu. Pomocí takto získané pravděpodobnosti dochází k filtrování oblastí, při kterém jsou přijaty pouze ty oblasti, u nichž je hodnota pravděpodobnost vyšší než $konf_{min}$. Přístup založený na porovnávání příznaků je schopen od sebe odlišit i dobře strukturované objekty oproti porovnávání oblastí založeného na měření variability, což však s sebou přináší vyšší výpočetní nároky. Při výběru vhodných příznaků je tato technika invariantní vůči změnám intenzity a vůči afinním transformacím.

Předpis pro přijetí obrazové oblasti P v tomto stupni je definován splněním podmínky:

$$\text{konfidence}(P) > 0{,}5. \tag{4.13}$$

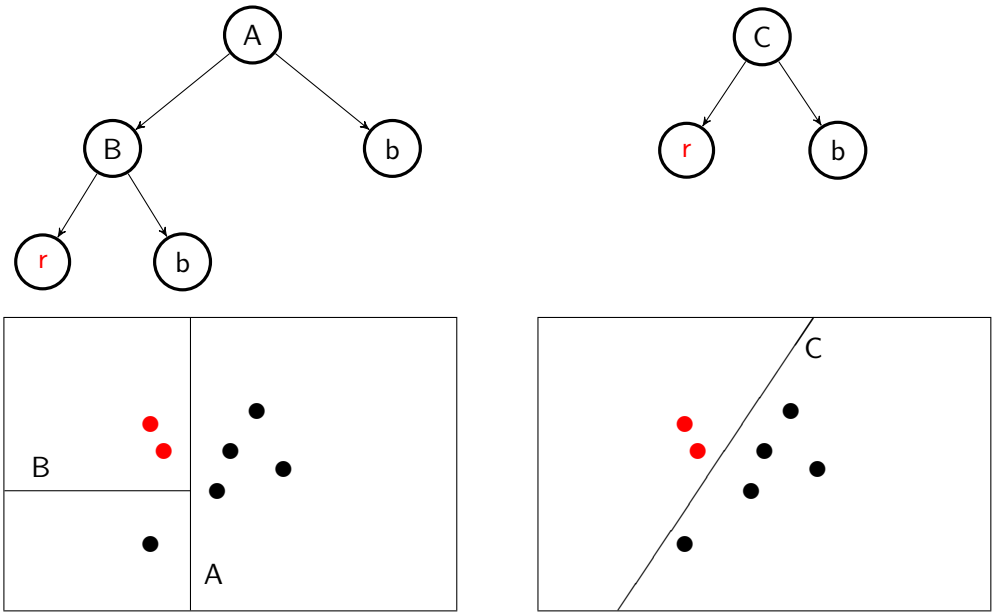
Důvěra k oblasti P je získána jako výstup klasifikace kapradinového klasifikátoru.

Mezi dva základní přístupy pro detekci objektu založené na rozeznávání příznaků patří *náhodný les* a *náhodné kapradiny*. Které byly v poslední době aplikovány v implementacích běžících v reálném čase [13, 14, 20]. Oba dva tyto přístupy patří do kategorie souborových metod. Souborové metody používají více modelů a kombinací jejich výstupů dochází k zlepšení predikčního výkonu. Použití vícenásobných klasifikátorů umožňuje často kompenzovat systematickou chybu jednoho klasifikátoru. Která může vzniknout při nenatrénování klasifikátoru všemi trénovacími daty či zanedbáním určitých faktorů při návrhu klasifikátoru. Dalším charakteristickým rysem těchto dvou přístupů je využití náhodnosti jako silného nástroje pro zlepšení predikčního výkonu. Pomocí něho zavádíme rozdíly do jednotlivých klasifikátorů.

4.3.1 Náhodný les (random forest)

Náhodný les [10, 7] je složen z vícenásobných modelů obvykle z *rozhodovacích stromů* (decision tree). Kde každý strom představuje vlastní model, jenž je závislý na náhodně generovaných příznacích. Je zde žádoucí, aby náhodně generované příznaky byly vybrány nezávisle a se shodným rozložením pro všechny stromy v lese. Během klasifikačního procesu každý strom volí nejpravděpodobnější třídu jako např. objekt, pozadí. Třída, jež má největší počet hlasů, je prohlášena za výsledek. Rozhodovací stromy si získaly svou oblibu pro tyto výhody: princip je intuitivní, trénování je často dopředné a ze všeho nejlepší klasifikace je extrémně rychlá.

Když je strom konstruován a trénován jsou předložená data (příklad) klasifikovány průchodem stromu shora dolů. Při průchodu stromu je v každém nekoncovém (rozhodovacím) uzlu použito rozlišovací kritériu (obvykle tvořeno jedním příznakem) pro rozhodnutí, do které větvě je následně přiřazen. Geometricky to lze interpretovat, tak že je bod přiřazen na jednu stranu nadroviny, která je rovnoběžná s jednou osou příznakového prostoru. Geometrický význam stromu je ilustrován na obr. 4.4.



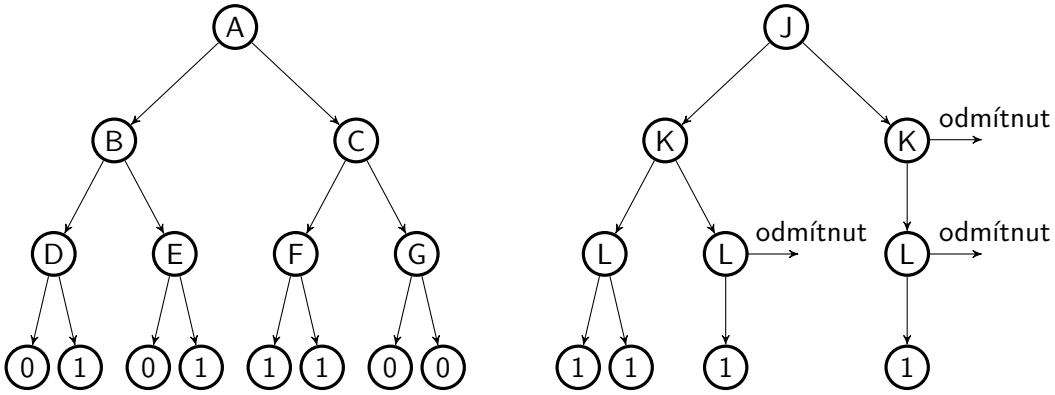
(a) Rozhodovací strom (b) Šikmý rozhodovací strom

Obrázek 4.4: Geometrická interpretace stromu. Rozlišovací kritérium A, B, C nekoncového uzlu stromu lze interpretovat jako nadrovinu, která dělí prostor na dva poloprostory.

V práci [10] používají *šikmý rozhodovací strom* (oblique decision tree), který je obecnější v tom, že nadrovina není nezbytně rovnoběžná s žádnou z os příznakového prostoru. Každá nadrovina je reprezentována lineární funkcí příznakových komponent. Použití šikmé nadroviny vždy produkuje menší strom, který umí data plně rozdělit do listů obsahujících jednu třídu. Přínos ve zmenšení stromu tkví v urychlení procesu klasifikace. Autoři ve své práci pro konstrukci stromu navrhují dvě základní metody: metodu *Central axis projection*, která využívá centrální projekční osy pro nalezení nadroviny v každém nekoncovém uzlu a metodu *Perceptron training*, jež využívá algoritmus trénování perceptronu pro nalezení

nadroviny. Jedná se však o off-line trénovací metody, které nejsou vhodné pro inkrementální trénování. V našem případě by to vyžadovalo vždy u každého vstoupivšího příkladu přepočítání celého stromu.

Z. Kalal a spol. v práci [13] navrhují *Sekvenční náhodný les*, který umožňuje inkrementální trénování. Klasifikátor má klasickou formu náhodného lesu, který je tvořen *sekvenčními stromy*. V tomto přístupu je strom konstruován pouze pro pozitivní příklady. Na začátku každý strom obsahuje jednu větev definovanou inicializačním příkladem, potom s každým novým pozitivním příkladem je vytvořena větev, která je přidána do stromu. V případě lesu je tento proces aplikován na každý strom v lese. Zde ještě podotkněme, že každému stromu v lese je přiřazena různá sada náhodně vybraných příznaků (testů), velikost sady je však pro každý strom shodná. Odezva každé sady je reprezentovaná rozlišovacím vektorem, který nazýváme větví stromu. Ve výsledku tento mechanismus umožňuje snadné přidání, popř. odebrání větve asociované s konkrétním příkladem. Při klasifikaci, pokud vstupní příklad projde rozhodovacím stromem až do listového uzlu, je klasifikován jako objekt našeho zájmu, v opačném případě je zamítnut a označen za pozadí. Konečný výsledek je vyřknut na základě sečtení hlasů od jednotlivých stromů lesu.

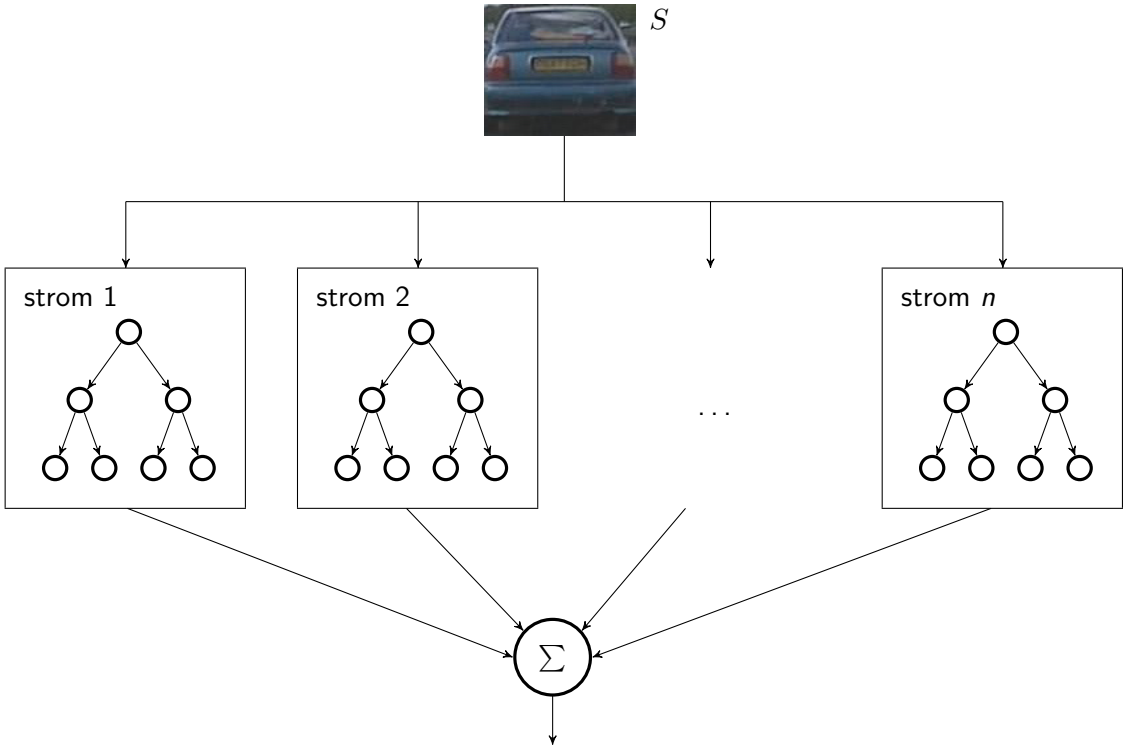


(a) Náhodný rozhodovací strom (b) Náhodný sekvenční rozhodovací strom

Obrázek 4.5: (a) rozhodovací strom složen s náhodně vybraných rozlišovacích kritérií A–G, listové uzly obsahují třídu objektů, 0 - pozadí, 1 - objekt. (b) ukazuje příklad sekvenčního stromu s náhodně vybranými rozlišovacími kritérii J–L, uzly ve stejné úrovni stromu obsahují stejné rozlišovací kritérium.

4.3.2 Náhodné kapradiny (random ferns, ferns)

Náhodné kapradiny představují poměrně nový přístup, podobně jako náhodný les jsou také vícenásobným klasifikátorem. Tento přístup byl poprvé představen v práci M. Özuysala a spol. [22]. Oproti náhodnému lesu, kapradiny tvoří nehierarchickou strukturu, kde každá entita (kapradina) je v zásadě sada testů. V náhodném lese je testovací sada každého stromu soubor různých testů rozmístěných podél uzlů, které formují strom. Kvůli ploché struktuře je testovací sada v každé kapradině tvořena jednoduchým vektorem testů (příznaků). Neboli na každý vstupní příklad kapradiny je aplikovaná stejná sekvence testů. Výhodou je, že příznaky (testy) této sekvence mohou být počítaný nezávisle a v libovolném pořadí, což přináší dobrý předpoklad pro paralelizaci. Další pro nás vítanou výhodou je, že umožňuje



Obrázek 4.6: Náhodný les složený z n stromů. Při klasifikaci náhodným lesem je vstupní příklad S vložen do jednotlivých stromů, kde je oklasifikován. Následně jsou sečteny hlasy od jednotlivých stromů, na jehož základě je uděleno konečné rozhodnutí.

inkrementální trénování, je jednoduchý a snazší na implementovatelnost. Oproti stromu je však hůře interpretovatelný.

Při klasifikaci příkladu dochází k změření příznaků v každém uzlu kapradiny. V původní práci [22] je jako rozlišovací kritérium zvolen binární příznak f_i, který pouze závisí na porovnání intenzit dvou pixelů o pozici $d_{i,1}$ a $d_{i,1}$ obrazové oblasti. Vyjádřeno předpisem:

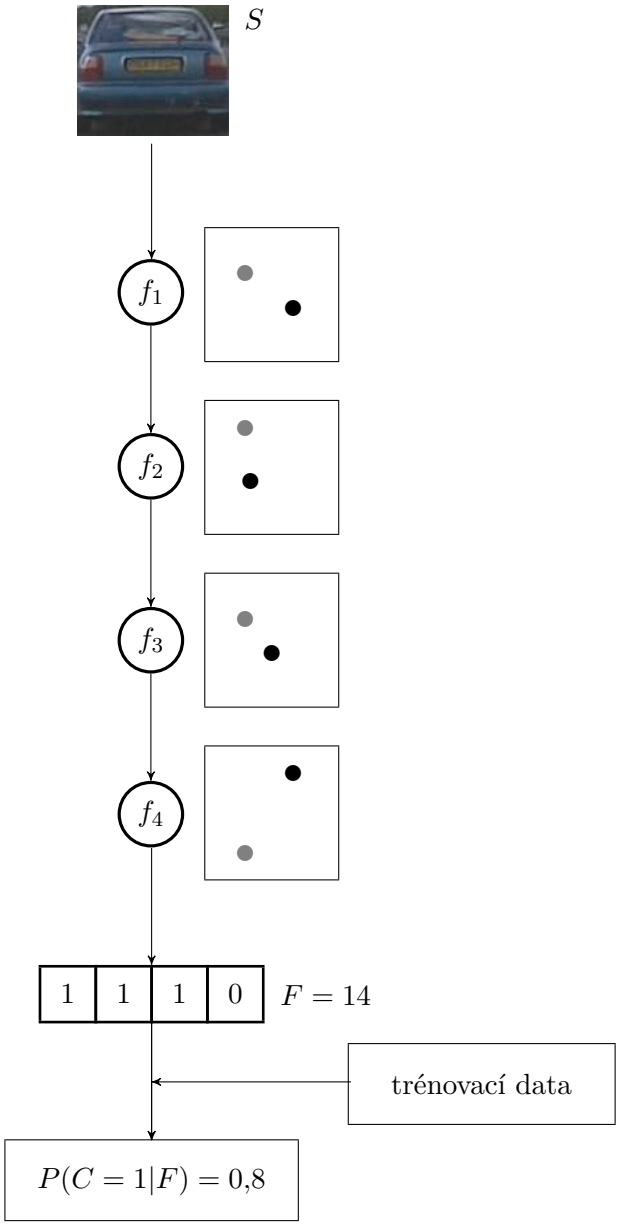
$$f_i = \begin{cases} 1 & \text{jestliže } I(d_{i,1}) < I(d_{i,2}) \\ 0 & \text{jinak,} \end{cases} \tag{4.14}$$

kde I reprezentuje obrazovou oblast, $d_{i,1}$ a $d_{i,1}$ jsou náhodně vybrané pozice. Kapradinu tedy lze popsat n-tící příznaků $F = (f_1, \ldots, f_n)$, po změření příznaků pro vstupní příklad získáme n-tici nul a jedniček, kterou můžeme interpretovat jako binární číslo, kde i-tý bit koresponduje s i-tým příznakem f_i. Toto binární číslo zde dále označované jako příznakové číslo, slouží jako index pro získání posteriorní pravděpodobnosti. Princip klasifikace jednou kapradinou je znázorněn na obr. 4.7, kde je vyobrazena kapradina tvořená čtyřmi příznaky. Z vyčíslených příznaků nad vstupním obrázkem je získáno příznakové číslo F v binární formě 1110, dekadicky 14, pro něhož je získána posteriorní pravděpodobnost $P(C = 1 \mid F) = 0{,}8$.

Kapradina si u každého příznakového čísla F_k zaznamenává počet pozitivních p a negativních n příkladů, které byly vyčísleny tímto příznakovým číslem během trénování. Posteriorní pravděpodobnost je spočítána potom dle vztahu:

$$P(C = 1 \mid F_k) = p/(p + n)\,, \tag{4.15}$$

pokud je počet příkladů u tohoto příznakového čísla nulový, $p + n = 0$, pak posteriorní pravděpodobnost je rovněž nulová, $P(C = 1 \mid F_k) = 0$.



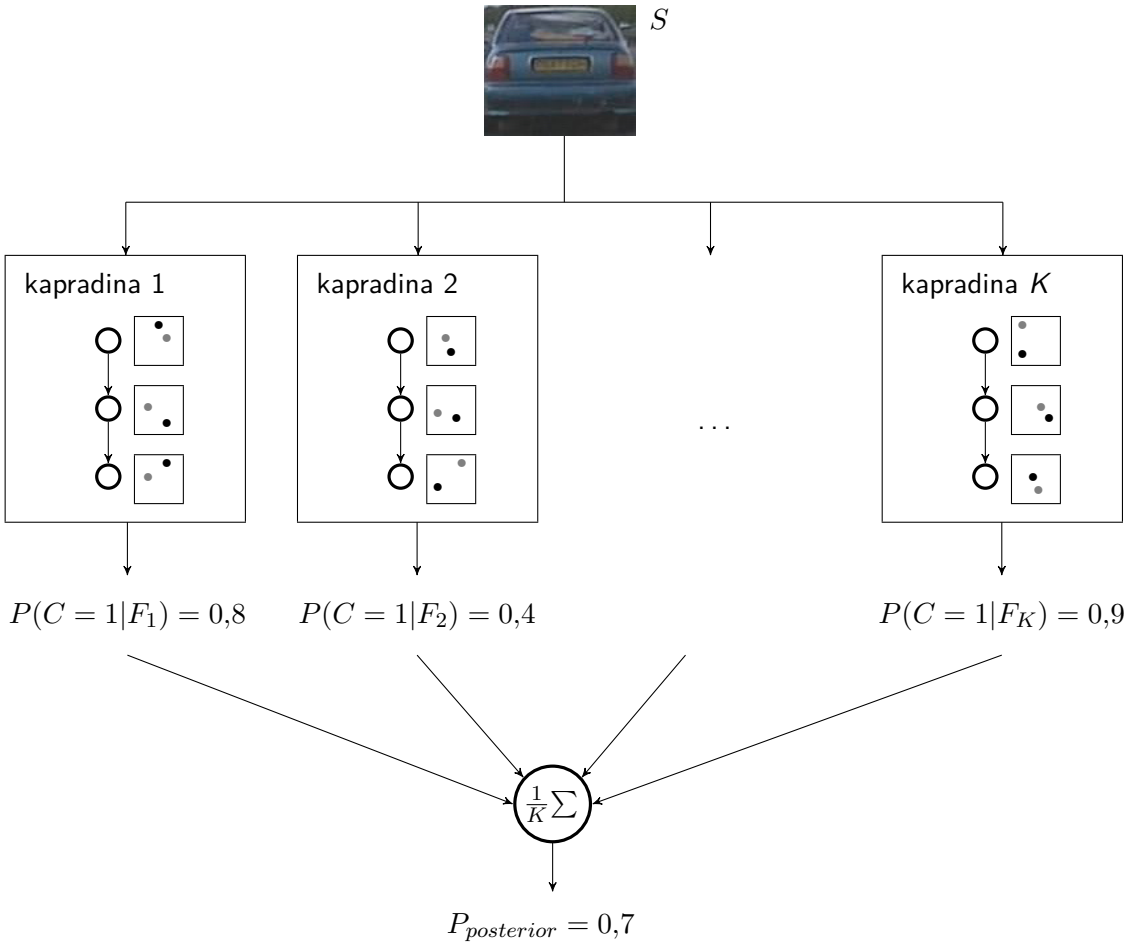
Obrázek 4.7: Klasifikace jednou kapradinou. Je vypočteno příznakové číslo F na základě vyčíslení příznaků $f_1, \ldots, f_4$, kde i-tý příznak koresponduje s i-tým bitem příznakového čísla. Příznakové číslo F slouží jako index pro získání posteriorní pravděpodobnosti $P(C = 1|F_k)$.

Při klasifikaci s využitím vícero náhodných kapradin je vstupní příklad nezávisle oklasifikován jednotlivými kapradinami, výsledná hodnota $P_{posterior}$ je vypočtena jako průměr posteriorních pravděpodobností $P(C = 1 \mid F_k)$ každé kapradiny. Matematicky vyjádřeno:

$$P_{posterior} = \frac{1}{K} \sum_{k=1}^{K} P(C = 1 \mid F_k), \tag{4.16}$$

kde K je označuje počet kapradin. Příklad klasifikátoru s náhodnými kapradinami je ukázán na obr. 4.8. Každá kapradina je tvořena odlišnou sadou příznaků, lze zde pozorovat vliv náhodnosti ve výběru příznaků, kde každá kapradina pro stejný vstupní obrázek generuje odlišné posteriorní pravděpodobnosti.

Kapradinový klasifikátor můžeme popsat dvojicí (K, N), přičemž K je počet kapradin a N je počet příznaků, z nichž je každá kapradina tvořena. V naší práci máme obvykle $K = 10$ a $N = 10$.



Obrázek 4.8: Klasifikace s použitím náhodných kapradin. Při klasifikaci je vstupní příklad S vložen do jednotlivých kapradin, kde je oklasifikován. Finální hodnota $P_{posterior}$ je získána zprůměrováním posteriorních pravděpodobností $P(C = 1 \mid F_k)$ každé kapradiny.

4.3.3 Výběr příznaků

U klasifikátorů založených na porovnávání příznaků hraje významnou roli výběr samotných příznaků, které ovlivňují výkon klasifikace, ať již z pohledu rychlosti či přesnosti. Mezi hlavní kritéria při výběru příznaků patří rychlost vyčíslení, odolnost vůči změnám intenzity a některým afinním transformacím, jako je změna měřítka a rotace. Mezi základní typy příznaků patří:

- *Dvoubodový příznak* – jedná se o binární příznak, který závisí na porovnání intezit dvou pixelů (bodů) o pozici d_1 a d_2 v obrazové oblasti I. Matematicky vyjádřeno:

$$f = \begin{cases} 1 & \text{jestliže } I(d_1) < I(d_2) \\ 0 & \text{jinak.} \end{cases} \tag{4.17}$$

  Jedná se o jednoduchý příznak, který byl úspěšně aplikovaný v práci [22] pro konstrukci kapradinového klasifikátoru. Mezi jeho přednosti patří extrémně rychlé vyčíslení, invariance vůči změnám intenzity a změně měřítka.

- *Dvouoblastní příznak* – binární příznak, definován porovnáním intenzit dvou obdélníkových oblastí P_1 a P_2 s parametry (d, w, h) v obraze I:

$$f = \begin{cases} 1 & \text{jestliže } I(d_1, w, h) < I(d_2, w, h) \\ 0 & \text{jinak,} \end{cases} \tag{4.18}$$

  kde d_1 a d_2 jsou pozice obdélníků v obraze, w a h je šířka a výška obdélníku. Tento příznak je odolnější na nahodilé výkyvy hodnot způsobené např. šumem.

- *Haarovy příznaky* – jsou založeny na porovnání intenzit obdélníkových oblastí P_{black} a P_{white}. Oproti předchozímu příznaku jsou porovnávané oblasti umístěny vedle sebe tak, že na sebe přímo navazují.

$$f = \begin{cases} 1 & \text{jestliže } I(P_{black}) < I(P_{white}) \\ 0 & \text{jinak.} \end{cases} \tag{4.19}$$

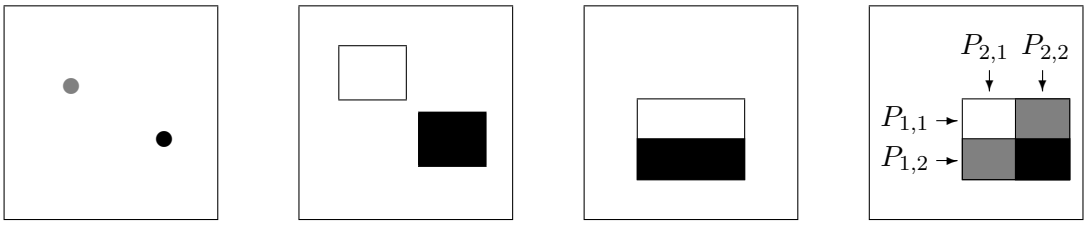
  Tento typ příznaků byl s úspěchem využit při konstrukci Viola-Jones detektoru [25].

- *2bitBP příznak* – je dvou bitový příznak, který je inspirován Haarovými příznaky. Nechť je dán obdélník, který je rozdělen na dvě horizontální oblasti a dvě vertikální oblasti. Příznak 2bitBP je potom definován na základě porovnání intenzit dvou horizontálních oblastí $P_{1,1}, P_{1,2}$ a dvou vertikálních oblastí $P_{2,1}, P_{2,2}$. Výsledek těchto dvou porovnání je reprezentován dvěma bity, kde každý bit koresponduje s jedním typem porovnání. Dáno předpisem:

$$\begin{aligned} f_i &= \begin{cases} 1 & \text{jestliže } I(P_{i,1}) < I(P_{i,2}) \\ 0 & \text{jinak} \end{cases} \\ f &= (f_1, f_2). \end{aligned} \tag{4.20}$$

  Tento příznak využili autoři v práci [13] pro konstrukci klasifikátoru založeného na sekvenčním náhodném lesu.

Základní optimalizační technikou k vyčíslení výše uvedených příznaků je využití integrálního obrazu pro výpočet součtu intenzit uvnitř obrazové oblasti. S integrálním obrazem se tento výpočet redukuje pouze na tři operace. Princip integrálního obrazu je vysvětlen v sekci 4.2.



(a) Jednobodový příznak (b) Dvouoblastní příznak (c) Haarův příznak (d) 2bitBP příznak

Obrázek 4.9: Ukázky příznaků.

4.4 Porovnávající šablonou klasifikátor

Poslední stupeň kaskádového detektoru je tvořen klasifikátorem založeným na porovnávání šablonou. Na rozdíl od příznakového klasifikátoru popsaného v předchozí sekci dochází k porovnání celých oblastí, pixel po pixelu. Na jehož základě je obrazové oblasti přiřazena míra důvěry. Pomocí takto získaných hodnot dochází ke konečnému filtrování vstupních oblastí.

V této fázi jsou všechny obrazové oblasti normalizovány, zmenšením na velikost 15×15 pixelů. Pro porovnání dvou obrazových oblastí P_1 a P_2 je využito normalizované vzájemné korelace (NCC). Definice NCC může být vyjádřena:

$$\mathrm{NCC}(P_1, P_2) = \frac{\sum_{x=1}^{N} P_1(x) \cdot P_2(x)}{\sqrt{\sum_{x=1}^{N} P_1^2(x) \cdot \sum_{x=1}^{N} P_2^2(x)}}. \tag{4.21}$$

Oborem hodnot této funkce je interval [0,0; 1,0]. Pro podobné oblasti funkce NCC nabývá hodnot blízko 1. Potom pro vyjádření vzdálenosti $d(P_1, P_2)$ mezi oblastí P_1 a P_2 definujeme vztah:

$$\mathrm{d}(P_1, P_2) = 1 - \mathrm{NCC}(P_1, P_2). \tag{4.22}$$

V rámci klasifikátoru si udržujeme šablony pro pozitivní třídu a pro negativní třídu. Třídu s pozitivními šablonami je označena P^+, třída s negativními šablonami je označena P^-. Šablony jsou získány online učením. Pro vstupní obrazovou oblast P, neznámé třídy, je vzdálenost k pozitivní třídě určena jako:

$$d^+ = \min_{P_i \in P^+} \mathrm{d}(P, P_i) \tag{4.23}$$

a vzdálenost k negativní třídě:

$$d^- = \min_{P_i \in P^-} \mathrm{d}(P, P_i). \tag{4.24}$$

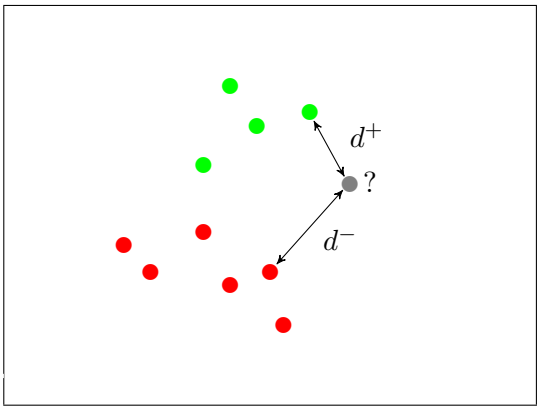
Grafické znázornění výpočtu vzdáleností d^+ a d^- je ukázáno na obr. 4.10. Z hodnot d^+ a d^- je získaná výsledná hodnota p^+ dle vztahu:

$$p^+ = \frac{d^-}{d^- + d^+}, \tag{4.25}$$

přičemž pro $P^+ = \emptyset$ je $p^+ = 0$, pro $(P^- = \emptyset \wedge P^+ \neq \emptyset)$ je $p^+ = 1$. Hodnota p^+ vyjadřuje důvěru, že obrazová oblast P patří do pozitivní třídy. V této fázi kaskády je přijata každá

obrazová oblast, jejíž důvěra p^+ je větší než předefinovaný práh τ^+, v naší práci je $\tau^+ = 0{,}65$. Poté tedy předpis pro přijetí oblasti P je definován splněním podmínky:

$$p^+ > 0{,}65. \tag{4.26}$$



Obrázek 4.10: Klasifikace neznámého objektu na základě vzdálenosti nejbližšího souseda z pozitivní a negativní třídy.

V práci [20] používají model, který je tvořen šablonami jak pozitivní, tak negativní třídy, které jsou získávány během učení. Jejich velikost není nikterak omezena. Což v průběhu aplikace způsobuje znatelný nárůst výpočetních nároků. Opačným příkladem je model [14], který je tvořen pouze jednou pozitivní šablonou, která obsahuje objekt vybraný v prvním snímku videosekvence. My navrhujeme počet pozitivních šablon omezit na velikost $m = 5$, a to tak, že model obsahuje inicializační šablonu, která představuje objekt vybraný v prvním snímku video sekvence, tato šablona zůstává po celou dobu součástí modelu. Pokud počet pozitivních šablon v průběhu učení převýší m, potom je z modelu odstraněna nejstarší pozitivní šablona (s výjimkou inicializační šablony). Počet šablon negativní třídy navrhujeme omezit na $n = 10$, princip při překročení velikosti je obdobný.

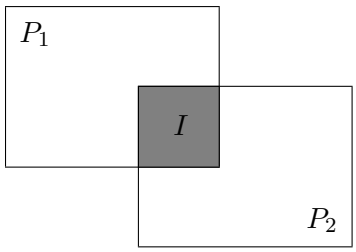
4.5 Potlačení nemaximálních hodnot

Detekční algoritmus produkuje typicky větší počet překrývajících se obdélníků pro objekt ve scéně. V závěrečné fázi zpracování se snažíme nemaximální obdélníky potlačit v závislosti na jejich prostorovém překryvu. Pro tento účel zde rozvíjíme techniku potlačení nemaxim založenou na shlucích [20]. Při této technice jsou všechny obdélníky rozděleny do shluků v závislosti na jejich prostorovém překryvu, potom pro každý shluk je vypočítán maximální obdélník pomocí zprůměrování obdélníků daného shluku. Překryv dvou obdélníku P_1, P_2 máme definován:

$$\text{overlap}(P_1, P_2) = \frac{P_1 \cap P_2}{P_1 \cup P_2} = \frac{I}{P_1 + P_2 - I}, \tag{4.27}$$

výsledek léží v intervalu $[0{,}0; 1{,}0]$.

Pro vytvoření shluků je zde použita aglomerativní hierarchická shluková metoda [12] konstruující shluky zespoda nahoru. Algoritmická podoba metody je na alg. 4, kde vzdálenost mezi shluky A, B je určena vzdáleností dvou nejbližších obdélníků shluků A, B,



Obrázek 4.11: Překryv oblastí P_1 a P_2 je vyjádřen jako podíl jejich průniku I a jejich sjednocení $P_1 + P_2 - I$.

matematicky vyjádřeno:

$$\text{Dist}(A, B) = \min\{\text{overlap}(a, b) : a \in A, b \in B\}. \tag{4.28}$$

A podmínka pro ukončení shlukování je splněna, pokud dva nejbližší shluky mají vzdálenost větší než 0,5. Matematicky vyjádřeno:

$$\min_{A,B \in S, A \neq B} \text{Dist}(A, B) > 0{,}5\,, \tag{4.29}$$

kde S je množina shluků.

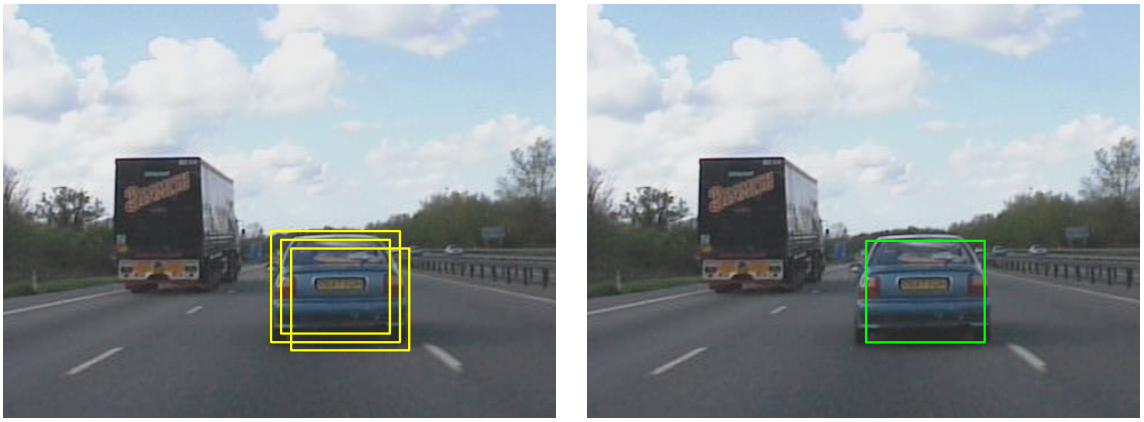
Algoritmus 4 Aglomerální hierarchická shluková metoda nejbližšího souseda

Vstup: množina obdélníků
Výstup: množina shluků
pro každý obdélník vytvoř jeden shluk
dokud počet shluků > 1 **dělej**
nalezni dva nejbližší shluky A a B, pro výpočet vzdálenosti použij rovnici 4.28
jestliže vzdálenost mezi A a $B > 0{,}5$ **pak**
ukonči cyklus
konec jestliže
spoj shluky A a B
konec dokud

4.6 Shrnutí

V této kapitole jsme se zabývali detekcí objektu, představili možné přístupy. Popsali běžný přístup, který je založen na technice klouzavého okna. Poukázali na její nedostatky a navrhli řešení v podobě využití kaskády, kterou jsme koncipovali jako třístupňovou. Pro první stupeň kaskády byl použit varianční filtr, který provádí prvotní klasifikaci na základě měření variance. Představuje jednoduchou rychlou metodu pomocí, které lze odmítnou velké procento negativních příkladů s nízkým výpočetním úsilím. Pro druhý stupeň byl zvolen příznakový klasifikátor založený na náhodných kapradinách. Spolu s náhodným lesem patří mezi základní přístupy. Oproti náhodnému lesu vyniká svou jednoduchostí a snazší implementovatelností. Při této příležitosti byly uvedeny 4 typy příznaků, mezi jejichž hlavní přednosti patří rychlá vyčíslitelnost. Jako poslední stupeň je použit klasifikátor založený na

porovnávání šablonou. Který je v rámci kaskádového detektoru pasován do role verifikátora. Pro závěrečnou fázi zpracování je uvedena metoda, která provádí potlačení nemaximálních obdélníků v závislosti na jejich prostorovém překryvu.



Obrázek 4.12: Potlačení nemaximálních obdélníků. Všechny překrývající se obdélníky jsou zprůměrovány do jednoho výsledného obdélníku.

Kapitola 5

Online učení detektoru

Při zpracování obrazu běží detektor a sledovací metoda nezávisle vedle sebe. V této kapitole je popsán způsob kombinace těchto dvou komponent k získání jednoho výsledku. Dále je zde představen proces online P-N učení aplikovaný na kaskádový detektor. V rámci kaskádového detektoru dochází k učení kapradinového klasifikátoru a porovnávající šablonou klasifikátoru. Podotkněme, že varianční filtr je neadaptivní a během učení detektoru zůstává nezměněn.

5.1 Sloučení a validace

V této sekci popisujeme interakci detektoru a sledovače, u které dochází k validaci výsledků sledovací metody a detektoru. V naší práci následujeme schéma popsané v alg. 5. Vstupem je výsledek sledovací metody, obdélník B_{track} o důvěře p^+_{track}, a výsledek detektoru, množina detekovaných obdélníků B_{detect} o důvěře p^+_{detect}. Důvěra p^+_{track}, p^+_{detect} je změřena pomocí porovnávající šablonou klasifikátoru, který je aplikován na výsledky B_{track} a B_{detect}. Jako finální výsledek označíme výsledek detektoru, $B_{final} \leftarrow B_{detect}$, pokud je důvěra výsledku detektoru vyšší než výsledek od sledovací metody a zároveň překryv výsledků B_{track} a B_{detect} je malý. Tuto podmínku interpretujeme, že dochází k selhávání sledovací metody a je nutná reinicializace sledovací metody. V opačném případě je jako finální výsledek vzat výsledek sledovací metody, $B_{final} \leftarrow B_{track}$.

V případě, že selhala sledovací metoda a je nezbytné ji reinicializovat. Potom je reinicializace provedena pouze tehdy, pokud detektor vrátil jeden detekovaný obdélník, $|B_{detect}| = 1$. Poznamenejme, že detektor vrací jako výsledek pouze obdélníky, jejichž důvěra je větší než τ^+, rov. 4.26, pro nás $\tau^+ = 0{,}65$.

Dalším krokem je určení události pro zahájení učení. Učení je zahájeno pouze tehdy, pokud finální výsledek je výsledkem sledovací metody a jeho stupeň důvěry je vysoký, v takovém případě zde mluvíme o validním výsledku. Konkrétně validní výsledek je takový, jehož důvěra je velká, $p^+_{track} > \tau^+$, pro nás $\tau^+ = 0{,}65$, anebo jeho důvěra je o něco menší ale stále dostatečně velká a současně je podpořena faktem, že předchozí výsledek sledovací metody byl validní, $p^+_{track} > \tau^- \ \wedge \ \text{valid}(B_{track,t-1})$, pro nás $\tau^- = 0{,}5$.

Algoritmus 5 Sloučení a validace

Vstup: $B_{track,t}$, $B_{detect,t}$: množina obdélníků
Výstup: $B_{final,t}$: množina obsahující finální obdélník

```
B_final,t ← ∅
valid(B_final,t)← false
jestliže B_track,t ≠ ∅ pak
    jestliže |B_detect,t| = 1 ∧ p+_detect,t > p+_track,t ∧ overlap(B_detect,t, B_track,t) < 0,5 pak
        B_final,t ← B_detect,t
    jinak
        B_final,t ← B_track,t
        jestliže p+_track,t > 0,65 pak
            valid(B_final,t) ← true
        jinak jestliže valid(B_final,t−1) ∧ p+_track,t > 0,5 pak
            valid(B_final,t) ← true
        konec jestliže
    konec jestliže
jinak jestliže |B_detect,t| = 1 pak
    B_final,t ← B_detect,t
konec jestliže
```

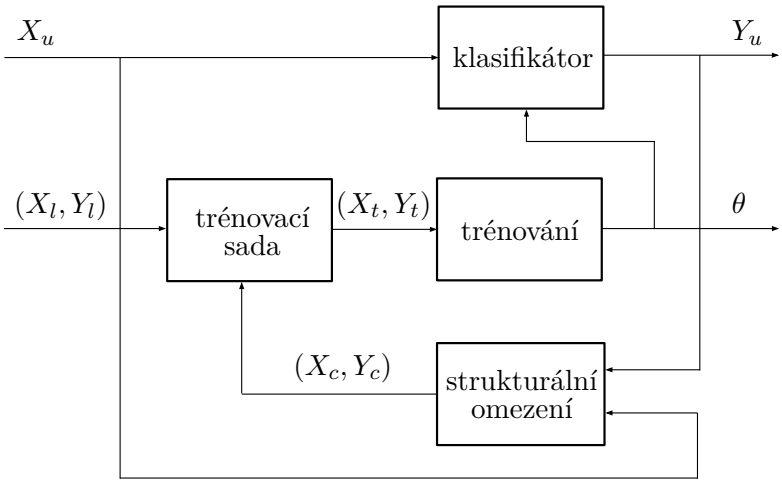
5.2 P-N učení

Pro trénování detektoru zde používáme učení s částečným dozorem (semi-supervised learning). Je to typ strojového učení, při kterém je známo malé množství označených příkladů a je zde klíčové navýšit počet označených dat. U P-N učení je získávání nových označených dat prováděno pomocí pozitivních (P) a negativních (N) omezení. Pozitivní omezení jsou použity k identifikování příkladů, které byly označeny klasifikátorem chybně jako negativní. Tyto příklady jsou přidány jako pozitivní příklady do trénovací sady. Negativní omezení je použito k identifikaci příkladů, které byly označeny klasifikátorem chybně jako pozitivní. Tyto příklady jsou přidány jako negativní příklady do trénovací sady. Vliv P, N omezení na vzhled modelu objektu je ilustrován na obrázku 5.2.

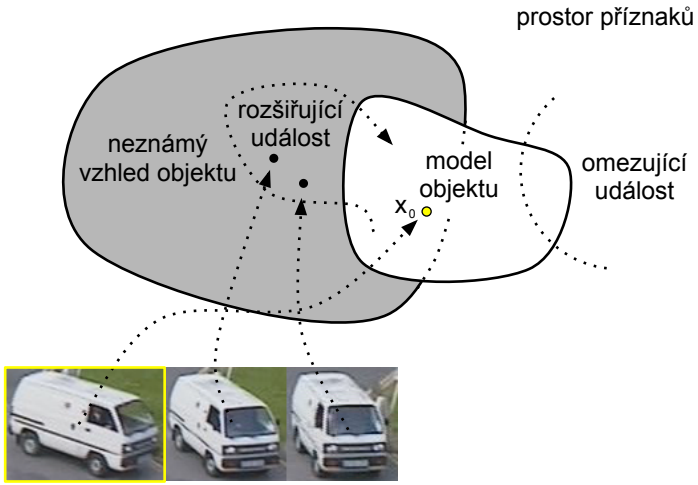
Princip P-N učení je popsán diagramem na obrázku 5.1. Nechť x je příklad z příznakového prostoru $\mathcal{X}$ a y je značka z prostoru značek $\mathcal{Y} = \{1, -1\}$. Sada příkladů je označena X, sada korespondujících značek je Y a (X, Y) je označená sada. Úkolem P-N učení je naučit klasifikátor $f : \mathcal{X} \to \mathcal{Y}$ z předchozí označené sady (X_l, Y_l). Na počátku trénovaní je klasifikátor inicializován předchozí označenou sadou (X_l, Y_l), tento krok lze přirovnat klasickému učení s dozorem. Poté proces probíhá iterativně. V každé iteraci je klasifikátorem provedeno označení neoznačených dat. Poté jsou identifikovány a přeznačeny příklady, které porušují strukturální omezení. Takto verifikovaná sada je přidána do trénovací sady a model je přetrénován. Proces učení klasifikátoru probíhá v jednom čase vždy nad jedním příkladem. Tady nejsou žádné omezení ohledně unikátnosti příkladů v sadě, proto omezení mohou přidávat stejné příklady do trénovací sady v průběhu času vícekrát a to dokonce s odlišnými značkami.

Pro učení detektoru zde používáme P, N-omezení navržené v práci [14]. P-omezení požaduje, aby všechny obrazové oblasti, které se vysoce překrývají s finálním výsledkem B_{final}, byly klasifikovány jako pozitivní příklady. N-omezení požaduje, aby všechny obrazové oblasti, které se nepřekrývají s finálním výsledkem, musí být klasifikovány jako negativní

příklady. V našem případě vysoký překryv je definován jako: overlap$(B_{final}, B_i) > 0.6$, a nepřekrývající se oblasti definujeme jako: overlap$(B_{final}, B_i) < 0.2$.



Obrázek 5.1: P-N učení nejprve natrénuje klasifikátor z označených dat a potom iterativně opakuje: klasifikátor označí neoznačená data, strukturální omezení verifikují značky a přeznačují ty, které porušují strukturální omezení, opravené data jsou přidány do trénovací sady a model je přetrénován. Převzato z [14].



Obrázek 5.2: Online model je inicializován příkladem x_0 vybraným v prvním snímku. Model je rozšiřován příklady ležícími na trajektorii objektu pomocí P-omezení a ořezáván příklady generovanými pomocí N-omezení. Model pomalu konverguje směrem k neznámému vzhledu objektu. Převzato v upravené podobě z [13].

5.3 Integrace komponent

Tato sekce představuje interakci komponent systému, která probíhá dle alg. 5. Vstupem systému je videosekvence, $I_0, \ldots, I_N$, a ohraničující obdélník objektu vybraného v prvním snímku videosekvence, B_0. Základní funkcionalita je rozdělena do 4 hlavních procedur, které jsou postupně volány pro každý snímek videosekvence. Na konci každé iterace je vrácen finální výsledek, $B_{final,t}$, který obsahuje nejpravděpodobnější pozici objektu našeho zájmu pro příslušný snímek videosekvence.

Algoritmus 6 Integrace komponent systému

Vstup: $I_0, \ldots, I_N$: videosekvence, B_0 : obdélník objektu
  $B_{final,0} \leftarrow B_0$
  učení$(I_0, B_{final,0})$
  pro t **od** 1 **do** N **dělej**
    $B_{track,t} \leftarrow$ sledování$(I_{t-1}, I_t, B_{final,t-1})$
    $B_{detect,t} \leftarrow$ detekce(I_t)
    $B_{final,t} \leftarrow$ sloučení&validace$(B_{track,t}, B_{detect,t})$
    jestliže valid$(B_{final,t})$ **pak**
      učení$(I_t, B_{final,t})$
    konec jestliže
    print $B_{final,t}$
  konec pro

5.4 Shrnutí

V této kapitole byly popsány hypotézy, které formulují způsob kombinace výsledků z detektoru a sledovače k získání finálního výsledku. Důvěra výsledků sledovače a detektoru je ohodnocena pomocí porovnávajíc šablonou klasifikátoru. Trénování probíhá pomocí P-N učení a to pouze nad validním výsledkem. Proces učení je aplikován na kapradinový a porovnávajíc šablonou klasifikátor, tak aby došlo ke zlepšení modelu. Pro učení jsou použity P, N omezení v původní podobě, jak byly představeny v práci [14].

Kapitola 6

Návrh a implementace

Tato kapitola se zabývá návrhem a implementací aplikace, jejíž snahou je demonstrování činnosti navržené TLD metody kombinující sledování a detekci spolu s online P-N učením popsaných v předchozích kapitolách. Při návrhu aplikace byl hlavně kladen důraz na funkčnost a efektivitu TLD metody před uživatelským rozhraním samotné aplikace.

Navržená aplikace byla implementována pomocí programovacích jazyků C a C++ s použitím volně dostupné knihovny OpenCV, což z ní činí multiplatformní aplikaci. Knihovna OpenCV zde zajišťuje jednoduchý přístup k obrazovým datům videosekvence. Jako vstupní datový proud je použit video soubor uložený ve formátu .avi. Jako zdroj je také možné použít přímý datový proud z připojené kamery.

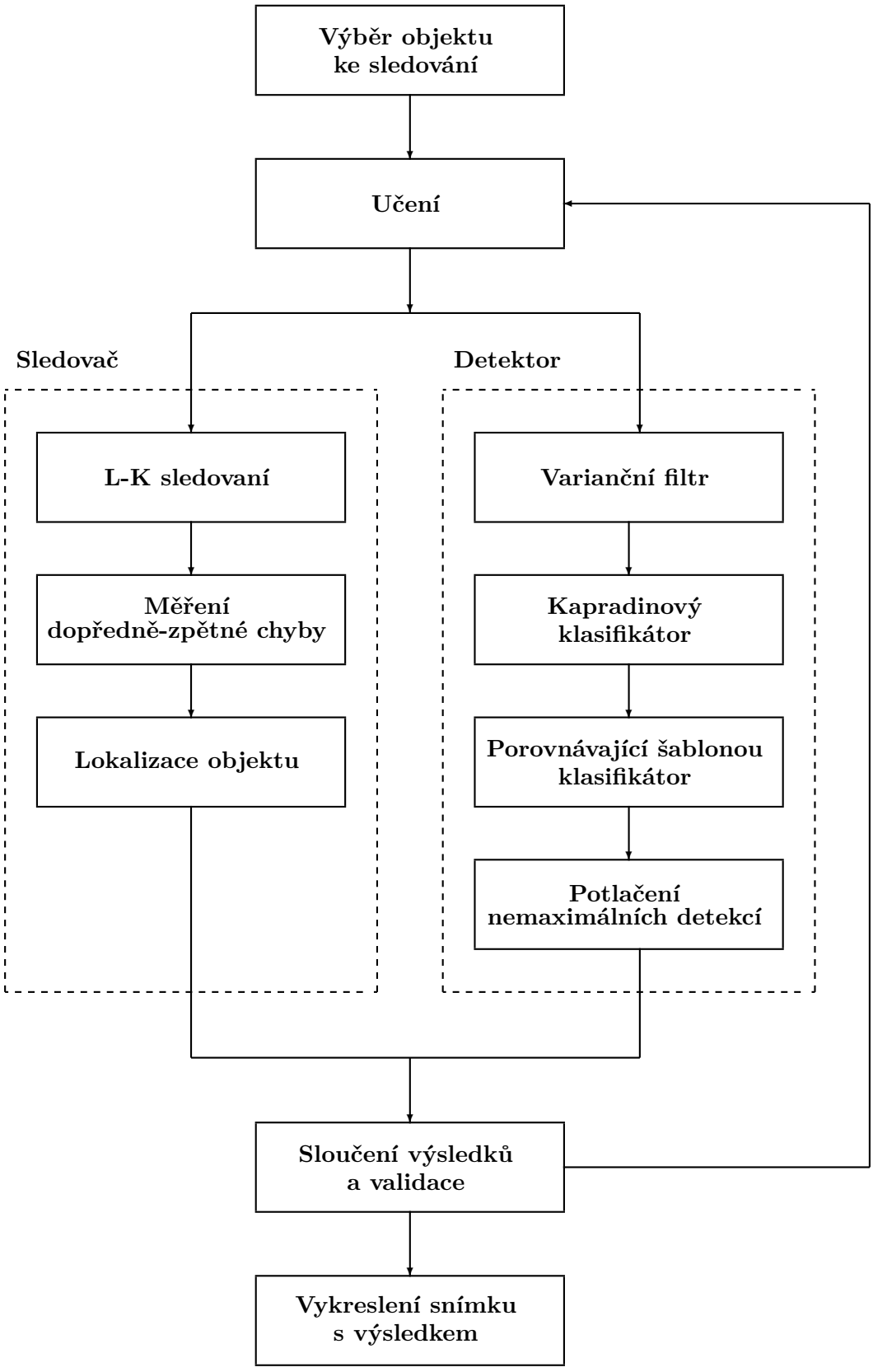
6.1 Návrh aplikace

Aplikace byla navržena tak, aby byla schopna uspokojit naše požadavky ohledně sběru, zpracování a vizualizace výstupních dat. Při návrhu aplikace bylo nutné řešit problémy jako je zajištění přístupu k obrazovým datům, efektivní manipulace s daty a možnost jejich vizualizace, způsob komunikace mezi jednotlivými komponentami TLD metody a v neposlední řadě také volba vhodné formy prezentace výstupních dat.

Na základě vztahů mezi jednotlivými komponenty byla navržena struktura aplikace, která je popsaná diagramem datových toků na obrázku 6.1.

Vstupem aplikace je načtená videosekvence. Zdrojem této videosekvence může být jednak připojená kamera nebo video soubor uložený ve formátu .avi. Díky možnosti připojení různých zdrojů lze lépe analyzovat vlastnosti detekčního algoritmu. Pokud se zabýváme analýzou algoritmu pracujícího v reálném čase, pak lze jako zdroj využít připojenou kameru. Pro hlubší analýzu jednotlivých komponent metody je přínosnější pracovat s uloženým video souborem, nad kterým lze provádět opětovná vyhodnocování na základě různého nastavení parametrů algoritmu.

Činnost TLD metody je zde rozdělena do čtyř kroků: sledování, detekce, sloučení výsledků a validace, učení. Metoda je inicializována na základě předložené videosekvence a objektu našeho zájmu. Výběr objektu je proveden manuálně uživatelem aplikace na samotném počátku procesu. Následně další kroky probíhají již nezávisle na uživateli pomocí vnitřní logiky TLD algoritmu. Posloupnost kroků je prováděna iterativně, kdy iterace je provedena pro každý nový snímek videosekvence. Na konci každé iterace dochází k vizualizaci výsledků, při které je vykreslen právě zpracovávaný snímek a výsledek sledování v podobě žlutého obdélníku, který koresponduje s detekovaným objektem.



Obrázek 6.1: Diagram datových toků navržené aplikace

6.2 Implementace

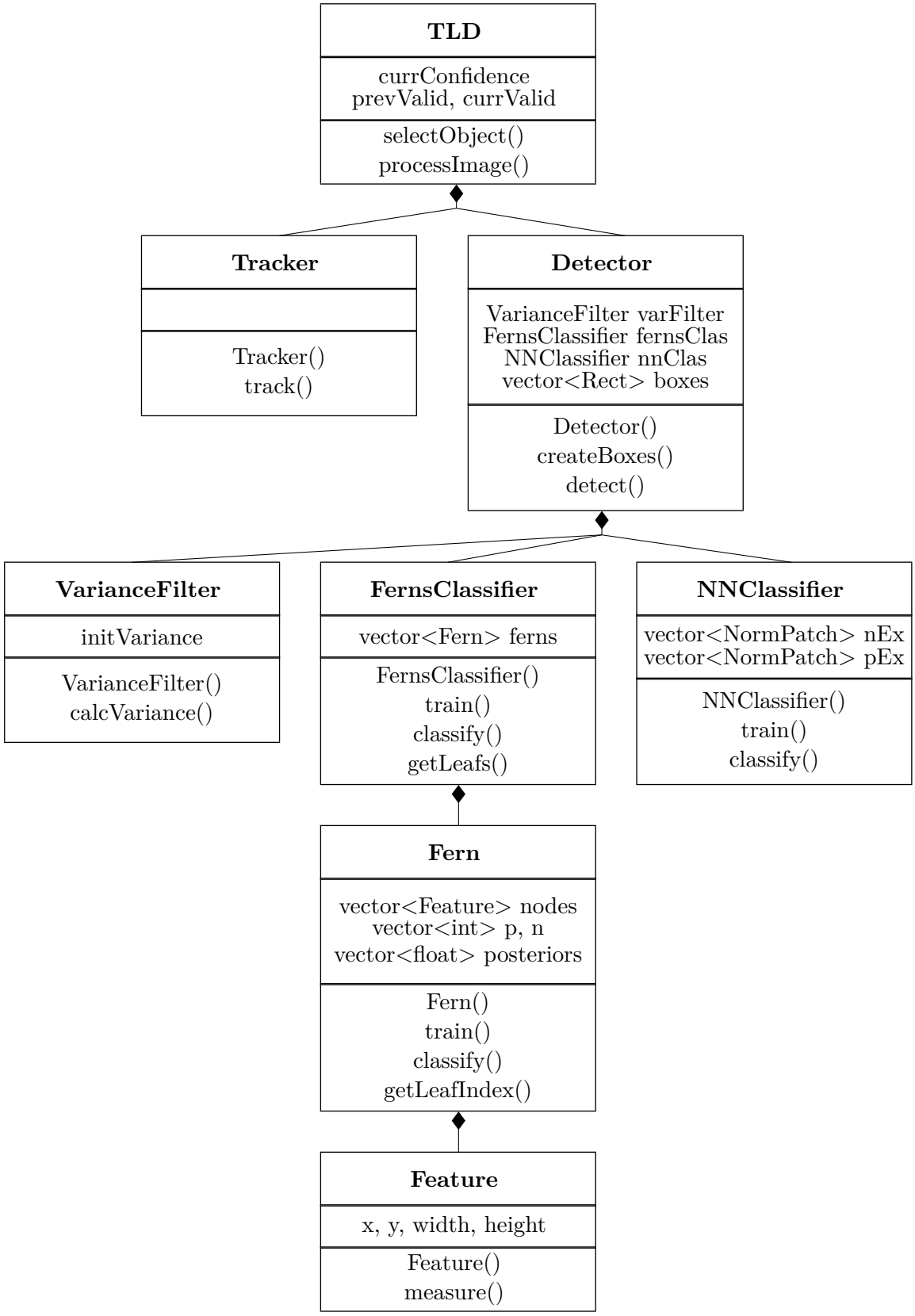
Aplikace je implementována pomocí programovacích jazyků C a C++. Pro práci s obrazovými daty zde byla použita volně šiřitelná knihovna OpenCV, která poskytuje nástroje pro přístup k videosekvenci, funkce pro manipulaci s obrazovými daty a jednoduchý aparát pro tvorbu GUI.

Z pohledu navržené aplikace je pro nás vhodné části aplikace implementovat jako samostatné komponenty, což nám přinese vetší nezávislost a jednodušší údržbu těchto celků. K tomu účelu je využito vlastností objektového programování, kterými disponuje jazyk C++. Implementace TLD metody je popsána pomocí diagramu tříd zobrazeného na obr. 6.2.

Základní komponentu zde tvoří třída `TLD`, která zapouzdřuje navrženou TLD metodu a poskytuje interface k její manipulaci. Vlastnosti metody TLD implementované v třídě `TLD` jsou popsány sadou atributů a operací. Mezi důležité atributy patří: `currBB` obsahující výsledný detekovaný objekt, zde reprezentovaný obdélníkem o parametrech (x, y, width, height), `currValid` indikující validitu takto získaného obdélníku v podobě booleovské hodnoty a `currConf` vyjadřující důvěru výsledku. Činnost TLD metody je zde popsána pomocí operací, které lze nad třídou vykonávat. Pro správný chod TLD metody musí nejprve dojít k nastavení proměnných, to zde zajišťuje konstruktor třídy. Ke kompletní inicializaci dochází až voláním metody `selectObject()`, která podává informace o rozměrech snímku vstupní videosekvence a informace o objektu našeho zájmu prostřednictvím svých vstupních parametrů. Tyto informace jsou vnitřní strukturou šířeny i mezi další komponenty, jako je sledovací a detekční metoda, jimiž jsou také inicializovány. Samotný proces běhu TLD metody je zajišťován voláním metody `processImage()`. Výsledek této operace je uložen v atributech `currBB`, `currValid`, `currConf` uvnitř třídy, jejich význam byl popsán již výše.

6.3 Základní charakteristika aplikace

Aplikace je navržena jako konzolová aplikace komunikující pomocí příkazového řádku. Aplikace umožňuje zpracovávat video data uložená ve formátu .avi. V případě potřeby je také možné využít přístup k datovému proudu z připojené kamery. Pro potřeby detailnější analýzy TLD metody je aplikace rozšířena o možnost zaznamenávání výstupu, který je ukládán do video souboru ve formátu .avi. Aplikace podporuje dva režimy přehrávání: plynulý a krokovaný. Výstup aplikace vizualizován pomocí jednoduchého GUI okno, které obsahuje aktuálně zpracovaný snímek video sekvence. Kompletní seznam přepínačů spolu s instrukcemi k překladu jsou umístěny v souboru `README` na přiloženém CD.



Obrázek 6.2: Diagram tříd.

Kapitola 7

Testování

Tato kapitola se snaží shrnout výsledky implementované TLD metody a poukázat na její hlavní přednosti, popř. nedostatky na základě jednotlivých testů. V prvním testu se zaměřujeme na dopředně-zpětnou metodu pro odhad sledovaného objektu. Další test je zaměřen na varianční filtr, konkrétně na jeho účinnost redukce obrazových oblastí procházejících kaskádovým detektorem. Další testy se zabývají výběrem příznaků a volbou parametrů kapradinového klasifikátoru z pohledu jejich vlivu na efektivitu detektoru.

Testovací data jsou tvořena sadou videosekvencí snímající reálnou scénu v rozlišení 320×240 pixelů. Hlavním zdrojem je TLD[1], PETS2001[2] a AVSS2007[3] databáze.

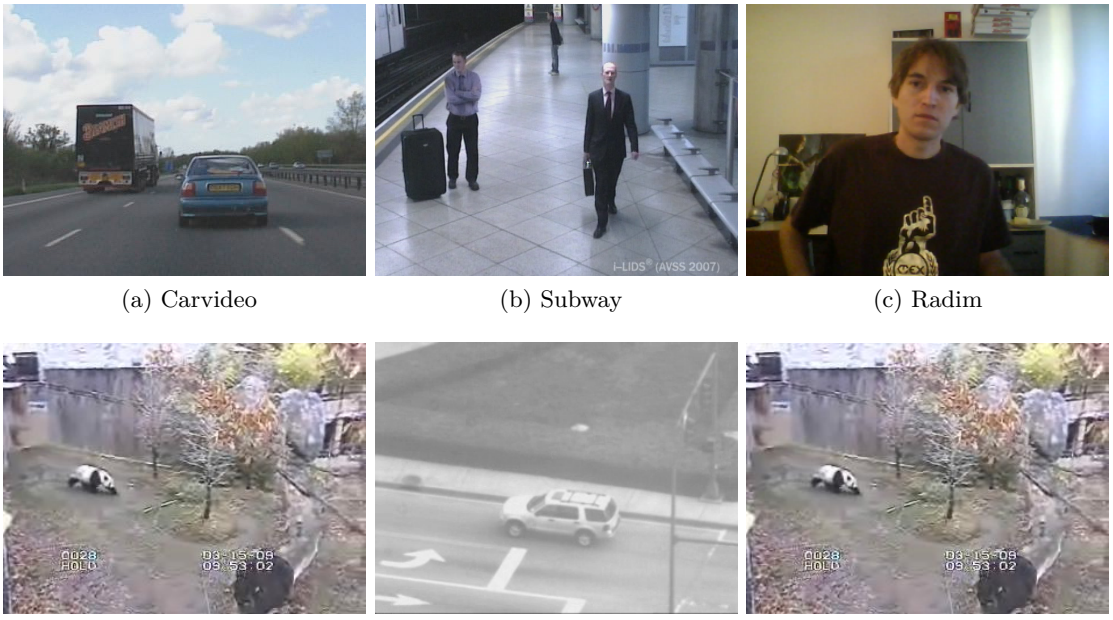
Testování a vyhodnocení výsledků probíhalo jednak na základě vizuálního porovnání, jednak pomocí anotovaných videí a porovnávacího skriptu. Výsledky jsou zde prezentovány pomocí následující metriky:

- FN (false negative) – vyjadřuje počet nedetekovaných objektů,
- TP (true positive) – vyjadřuje počet správně detekovaných objektů,
- FP (false positive) – vyjadřuje počet chybně detekovaných objektů,
- FAR (False Alarm Rate) = FP / (FP + TP),
- DR (Detection Rate) = TP / (TP + FN),

kde *Detection Rate* (DR) vyjadřuje míru počtu správných detekcí k celkovému počtu všech potencionálně možných detekcí, *False Alarm Rate* (FAR) vyjadřuje míru počtu chybně provedených detekcí k celkovému počtu všech provedeným detekcí. Objekt je správně detekovaný, pokud se jeho obdélníková hranice shoduje s referenčním objektem. Shoda je zde definovaná tak, že těžiště detekovaného obdélníku leží uvnitř referenčního obdélníku. Více informací ohledně metrik a algoritmů pro vyčíslení výkonu detekčních a sledovacích metod je možné nalézt v práci [4].

Měření bylo prováděno vždy v deseti opakováních, získané výsledky byly následně zprůměrovány. K výpočtu byl použit stroj: notebook s procesorem Core 2 Duo 2.10GHz.

[1] dostupné z http://personal.ee.surrey.ac.uk/Personal/Z.Kalal/TLD
[2] dostupné z http://www.cvg.cs.rdg.ac.uk/PETS2001/pets2001-dataset.html
[3] dostupné z http://www.eecs.qmul.ac.uk/~andrea/avss2007_d.html



(a) Carvideo (b) Subway (c) Radim

(d) Panda (e) Car (f) Pedestrian1

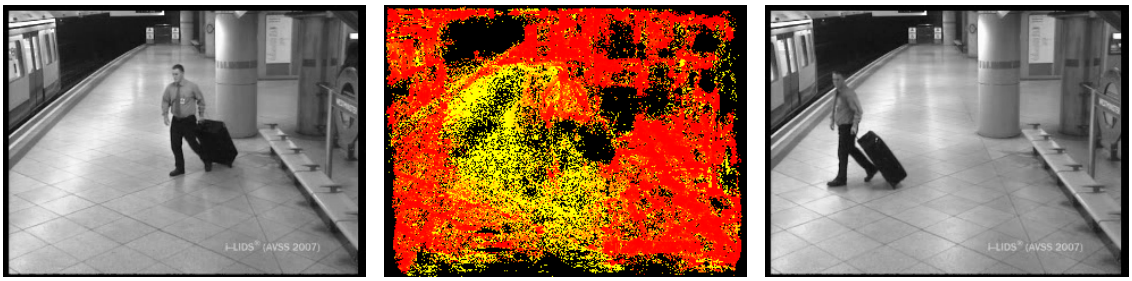
Obrázek 7.1: Testovací data

7.1 Dopředně-zpětné měření chyby pro odhad objektu

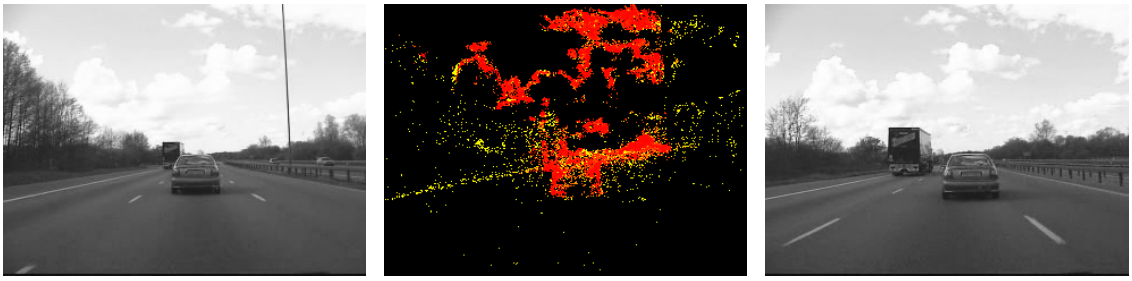
Dopředně-zpětná metoda může být použita pro validaci trajektorie sledovaného objektu nebo pro výběr příznakových bodů pro sledování, jak je prezentováno v práci [15]. Výpočet spolehlivých bodů je ukázán na mapě dopředně-zpětné chyby, kde každý pixel z počátečního snímku t byl sledován po celou videosekvenci, pro trajektorii byla vypočtena chyba, která byla následně přiřazena příslušnému pixelu na mapě. Výpočet dopředně-zpětné chyby je prezentován na dvou videosekvencích na obrázku 7.2. Kde (a), (b) jsou počáteční snímky videosekvencí, (e), (f) jsou konečné snímky sekvence, (c), (d) ukazují mapu chyb. Červená barva představuje nízkou dopředně-zpětnou chybu. Body vybrány z této oblasti vykazují vysokou spolehlivost při sledování během celé videosekvence. Světlejší (žlutá) barva označuje oblasti, které je složité či nemožné sledovat, např. na mapě (b) je to zapříčeno zakrytím této oblasti pohybujícím se objektem. Černá barva označuje ztracené body během sledování nebo body, které není možné sledovat. Příkladem je část oblohy na (d), která je tvořena homogenní oblastí, která není vhodná ke sledování.

Výsledky ukazují, že výběr bodů identifikovaných pomocí této techniky umožňuje zvýšit přesnost a prodloužit běh sledovacího algoritmu. Výsledky sledování a vliv dopředně-zpětné chyby na jejich úspěšnost jsou ilustrovány na obrázku 7.3. Je zde porovnána technika, která pro validaci sledovaných bodů používá měření podobnosti pomocí NNC, technika, která pro validaci používá dopředně-zpětnou chybu (DZ) a technika, která kombinuje oba dva přístupy (NCC+DZ). Na získaných výsledcích je patrné, že technika kombinující NCC a DZ je z uvedených možností nejlepším řešením.

Testování zároveň potvrdilo slabé dlouhodobé sledovací schopnosti rekurzivních technik. Které jsou náchylné na šum, stíny, odlesky a různé obrazové odchylky, jež způsobují chybu ve sledování. Ta se s časem zvětšuje a konečným výsledkem je ztráta objektu.

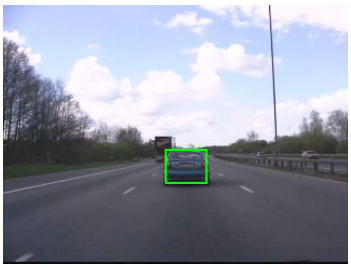


(a) Snímek t (b) Mapa dopředně-zpětné chyby (c) Snímek $t + 2\,\mathrm{s}$

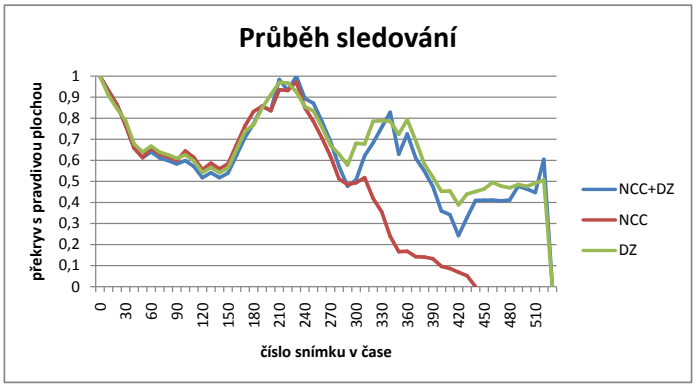


(d) Snímek t (e) Mapa dopředně-zpětné chyby (f) Snímek $t + 4\,\mathrm{s}$

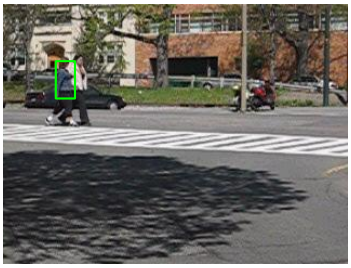
Obrázek 7.2: Výběr vhodných sledovacích bodů pomocí měření dopředně-zpětné chyby. Dopředně-zpětná chyba změřená pro každý pixel přes celou videosekvenci, kde (a) je počáteční a (c) je koncový snímek sekvence. Změřené chyby jsou ukázány na mapě (b), červená barva znázorňuje body s malou dopředně-zpětnou chybou.



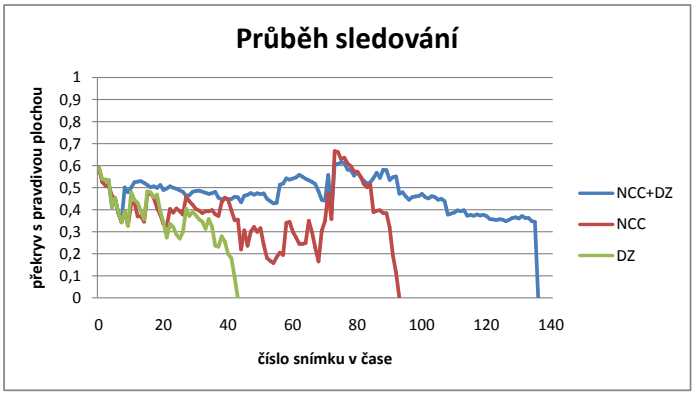
(a) Počáteční snímek



(b) Průběh sledování na videosekvenci carvideo



(c) Počáteční snímek



(d) Průběh sledování na videosekvenci pedestrian1

Obrázek 7.3: Průběh sledování v závislosti na zvolené technice validace sledovaných bodů – NCC (normalizovaná korelace), DZ (dopředně-zpětné měření chyby).

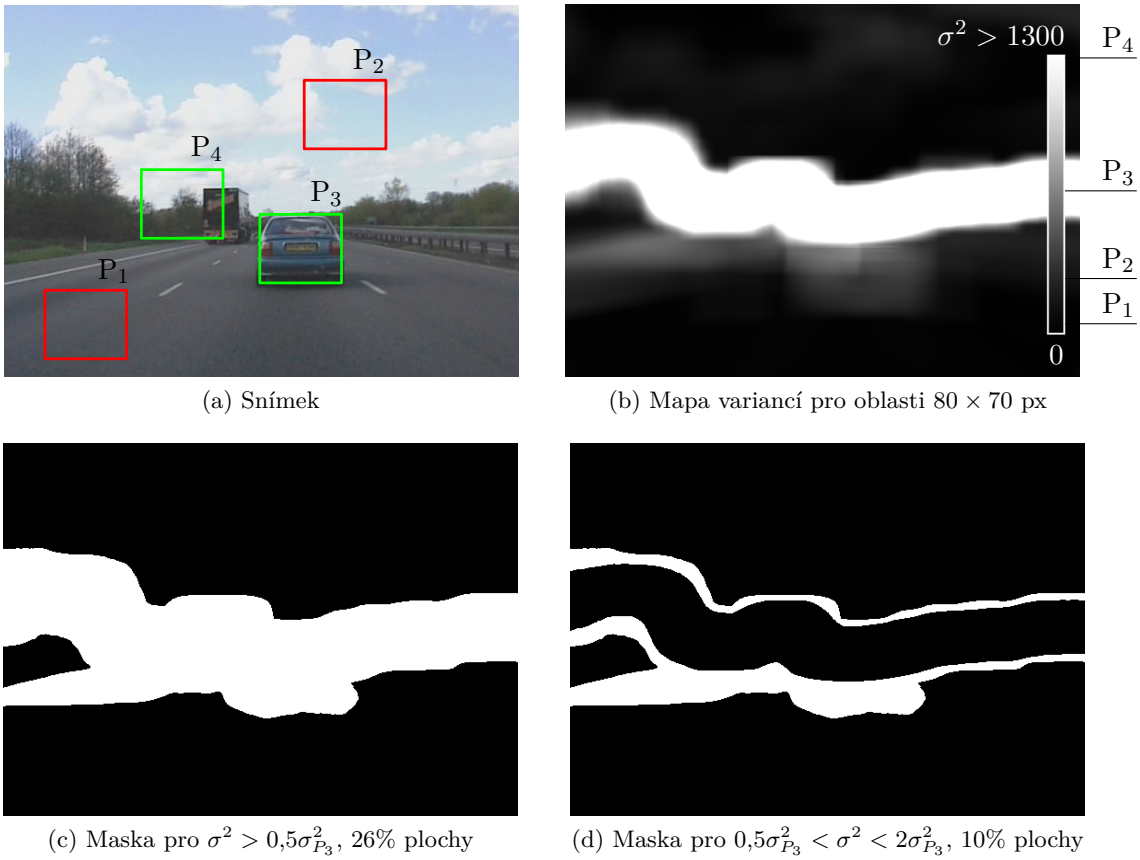
Metoda	**Body ke sledování**	**Velikost NCC okna**	**Čas zpracování**	**Snímků za vteřinu**
DZ	100	–	0,00763s	131
DZ + NCC	100	10×10px	0,08466s	11
DZ + NCC	100	20×20px	0,08765s	11

Tabulka 7.1: Výpočetní náročnost sledování

7.2 Varianční filtr

Varianční filtr tvoří první stupeň našeho kaskádového detektoru. Jeho úkolem je filtrování vstupních oblastí s co nejmenším výpočetním úsilím. K tomuto úkolu je zde využita variabilita vyjádřena pomocí variance σ^2. Při tomto testu se zaměříme na účinnost filtrace z pohledu počtu redukovaných oblastí.

Na výsledcích testu je vidět zřetelný přínos variančního filtru, s jehož pomocí je možné odfiltrovat 50–80% vstupních oblastí. Účinnost filtrace je ilustrována na obrázku 7.4, kde je na obrázku (a) znázorněn aktuální snímek videosekvence a k němu korespondující mapa variancí σ^2, která byla spočtena pro oblasti 80×70 px. V případě, že bychom jako referenční hodnotu vzali varianci oblasti P_3 a aplikovali filtraci dle podmínky 4.2, je možné odfiltrovat až 90% vstupních oblastí. Pokud bychom aplikovali pouze dolní omezení na velikost variance, účinnost filtrace se sníží, v tomto případě na 74%.



(a) Snímek

(b) Mapa variancí pro oblasti 80×70 px

(c) Maska pro $\sigma^2 > 0{,}5\sigma^2_{P_3}$, 26% plochy

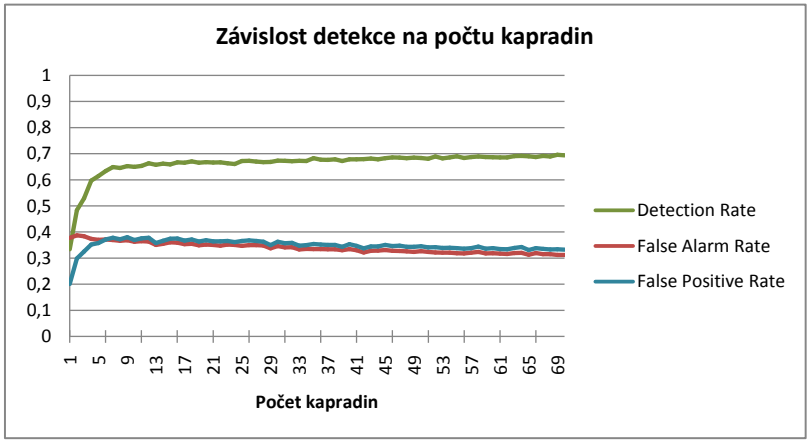
(d) Maska pro $0{,}5\sigma^2_{P_3} < \sigma^2 < 2\sigma^2_{P_3}$, 10% plochy

Obrázek 7.4: Účinnost filtrace na základě variability oblastí, variabilita vyjádřená pomocí variance σ^2. Na obrázku (d) při aplikovaní podmínky 4.2 dochází k filtraci 90% oblastí.

7.3 Parametry kapradinového klasifikátoru

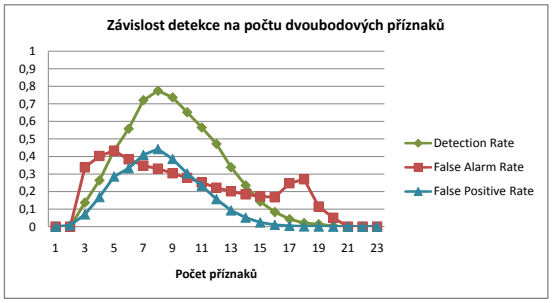
V tomto testu se zaměřujeme na vliv proměnných parametrů pro kapradinový klasifikátor. Konkrétně analyzujeme dva parametry: počet kapradin, K, a počet příznaků v kapradině, N. V práci [7] zabývající se náhodnými stromy je ukázáno, že zvýšení počtu náhodných rozhodovacích stromů nemá vliv na přetrénování. To znamená, že zvýšení počtu kapradin nebude mít vliv na snížení míry detekce (detection rate), což je ukázáno na výsledcích testu obr. 7.5. Testování bylo prováděno až do počtu 130 kapradin. Jako vhodný kompromis mezi úspěšností detekce a výpočetním výkonem se jeví $K = 12$.

Kapradiny podobně jako rozhodovací stromy mají tendenci k přetrénování pro zvyšující se počet příznaků. Což je zapříčiněno korelací trénovacích dat. Malý počet příznaků ignoruje

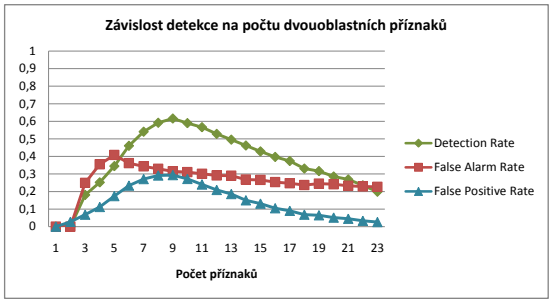


Obrázek 7.5: Průběh detekce v závislosti na počtu kapradin

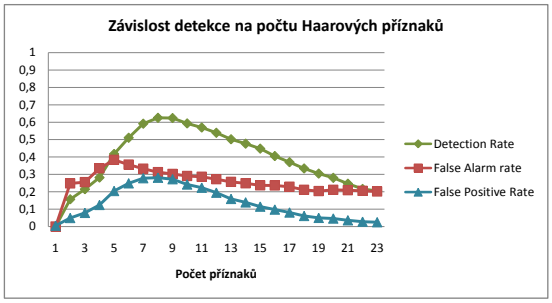
tuto korelaci, ale zároveň dochází k nárůstu chybných detekcí. Proto dalším cílem bylo zjištění základních charakteristik různých typů příznaků a jejich vlivu na účinnost detekce. Pro testování byly zvoleny čtyři typy příznaků: dvoubodový, dvouoblastní, 2BitBP příznak a Haarovy příznaky. Příznaky byly popsány v kapitole 4.3.3. Testování probíhalo na třech různých datových sadách: CD-ped (třída chodci a nechodci, 18×36 px, námi využito 40 000 příkladů), Faces (třída obličeje a neobličeje, 19×19 px, námi využito 6 000 příkladů), ObjectCategories (třída obličeje a chodci, 64×128 px, 900 příkladů).



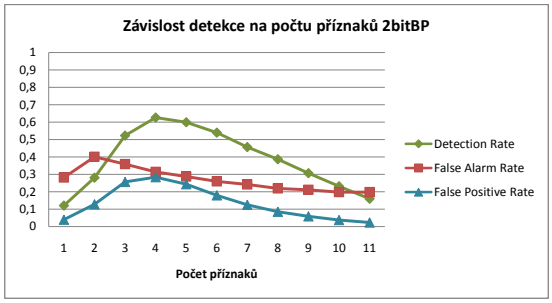
(a) Dvoubodový příznak



(b) Dvouoblastní příznak



(c) Haarovy příznaky



(d) 2BitBP příznak

Obrázek 7.6: Průběh detekce v závislosti na typu a počtu příznaků

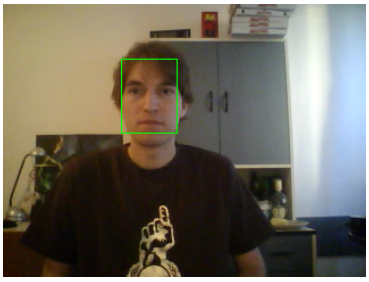
Výsledky zobrazené na obr. 7.6 ukazují, že vhodný počet příznaků u typu příznaku (a), (b), (c) leží shodně na 9, u příznaku (d) se jako nejlepší počet jeví 5. Při dalším zvyšování počtu příznaků dochází k postupnému snižování detekční míry. Testování zároveň ukázalo, že příznaky (b), (c), (d) mají velmi podobnou detekční charakteristiku. Výjimku tvoří příznak (a), který má cirka o 20% vyšší míru detekce, která je však zároveň doprovázena vyšší mírou chybných detekcí.

Kapradina složená z příznaku (a), (b), (c) vyžaduje 2^N položek pro uložení posteriorní pravděpodobnosti, u příznaku (d) je tomu $2^{2 \cdot N}$. Pokud z výsledků testů budeme předpokládat, že (a), (b), (c) mají nejlepší výkon pro $N = 9$ a (d) má pro $N = 5$. Potom můžeme konstatovat, že nejlepšího detekčního výkonu u příznaků (a), (b), (c) lze dosáhnout s nepatrně nižšími paměťovými nároky, než je tomu u příznaku (d), jelikož platí $2^9 < 2^{2 \cdot 5}$. Zde je však nutné vzít v potaz, že hranice mezi sousedními hodnotami N jsou velmi slabé.

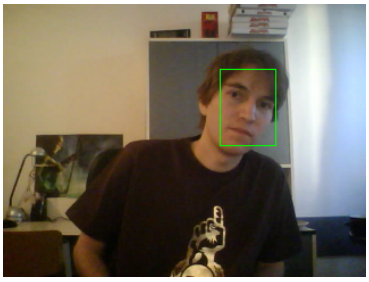
7.4 Úspěšnost sledování pomocí TLD metody

Poslední test byl zacílen na demonstrování vlastností TLD metody. Schopnosti LTD metody byly testovány na různých scénářích. První scénář byl zaměřen na schopnost znovudekování objektu, druhý scénář byl zaměřen na schopnost přizpůsobení se variabilitě sledovaného objektu.

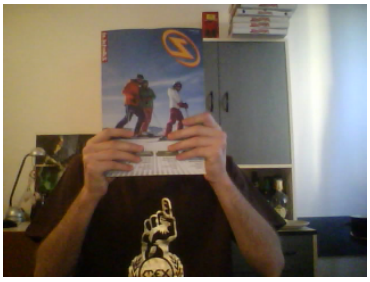
Testování ukázalo, že TLD metoda je schopna se během krátké doby naučit vzhled sledovaného objektu. Na videosekvenci obr. 7.7 je detektor schopen detekovat označený objekt po uplynutí 1-3 snímků po inicializaci (testováno s kapradinovým klasifikátorem: 2bitBP příznaky, K=10, N=10). Příklady znovudetekování jsou (d), (e), (g) na obr. 7.7. Testy také ukázaly na schopnost sledování objektu, jehož variabilita se v průběhu času mění. Tato schopnost je však omezená a silně závislá na míře změny variability. Příklady změny variability u sledovaného objektu jsou (j) a (b).



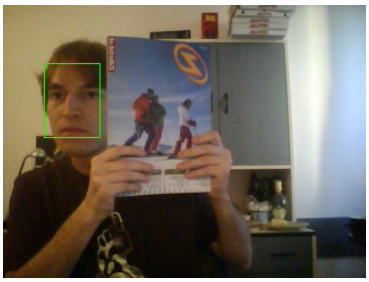
(a) Snímek 1



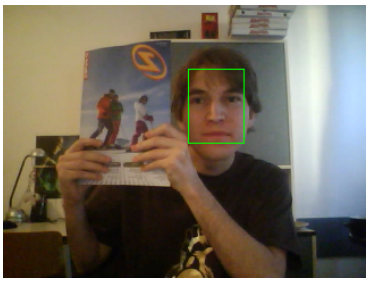
(b) Snímek 53



(c) Snímek 73

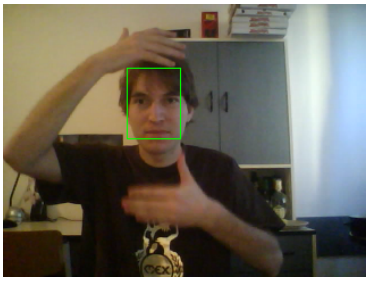


(d) Snímek 97

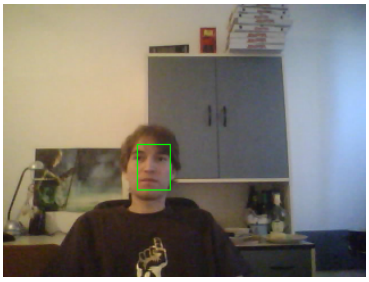


(e) Snímek 185

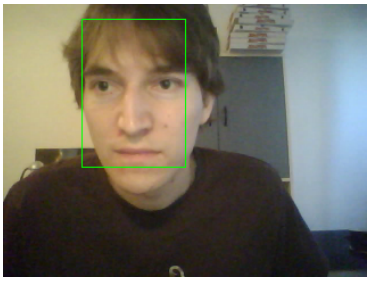
(f) Snímek 220



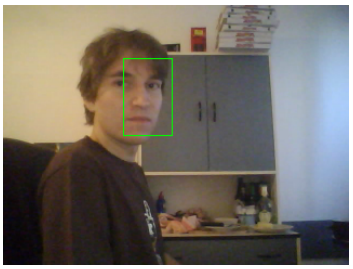
(g) Snímek 222



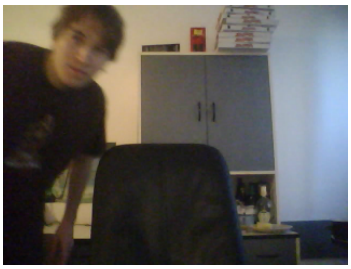
(h) Snímek 624



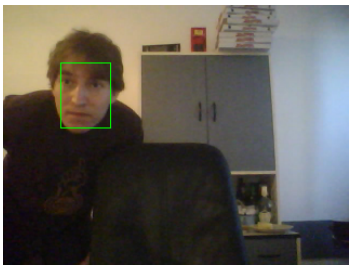
(i) Snímek 645



(j) Snímek 696



(k) Snímek 771



(l) Snímek 837

Obrázek 7.7: Ukázky sledování TLD metodou

Kapitola 8

Závěr

Cílem této diplomové práce bylo popsat principy metod kombinující online učení detektorů a sledování objektů, které se využívají pro dlouhodobé sledování objektů. Ukázalo se, že řešení této problematiky představuje komplexní problém, při kterém je nutné řešit tři významné úlohy: sledování objektu, detekce a online učení. V práci byly představeny možné techniky k řešení jednotlivých úloh a popsán způsob jejich vzájemné kombinace.

Schopnosti metod kombinující online učení detektoru a sledování byly demonstrovány na základě navržené metody. Výsledky ukazují, že přístup kombinace rekurzivního sledování a detekce s online učením je možným směrem budoucího vývoje v oblasti sledování. Jedná se o techniku, která je schopna se vypořádat s řadou nežádoucích vlivů, jako je zmizení objektu ze scény, zakrytí objektu či změna vzhledu, které omezují současné sledovací techniky. Kombinací detekce a sledování se podařilo získat z obou skupin to nejlepší, jako je přesnost z rekurzivního sledování a robustnost z detekčních metod. Testování zároveň prokázalo, že při volbě vhodných technik lze tento přístup úspěšně provozovat v aplikacích běžících v reálném čase.

Budoucí rozšíření může probíhat vícero směry. Prvním směrem by mohlo být zaměření se na výběr dalších možných přístupů pro řešení jednotlivých podčástí systému, jako je sledování, detekce a online učení, nebo jejich zdokonalení. Druhým a pro mě zajímavějším rozšířením by mohlo být zavedení další komponenty do systému, která by byla schopna z dostupných dat získat nové informace, které by přispěly ke zlepšení sledovacího výkonu. Ať již se při malé fantazii může jednat o doplnění o další sledovací či detekční metodu. V neposlední řadě by také nebylo od věci rozšířit záběr této práce a pokračovat ve zkoumání dalších strategií, které by poskytly podrobné a ucelené informace o této bezesporu zajímavé oblasti.

Literatura

[1] Andriluka, M.; Roth, S.; Schiele, B.: People-tracking-by-detection and people-detection-by-tracking. *CVPR*, 2008.

[2] Avidan, S.: Ensemble Tracking. *CVPR*, 2005: s. 494–501.

[3] Barron, J. L.; Fleet, D. J.; Beauchemin, S. S.: Performance of optical flow techniques. *International Journal of Computer Vision*, ročník 12, č. 1, 1994: s. 43–77.

[4] Bashir, F.; Porikli, F.: Performance avaluation of object detection and tracking systems. *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS)*, 2006.

[5] Bouchard, G.; Triggs, B.: Hierarchical part-based visual object categorization. *CVPR*, 2005.

[6] Bouguet, J.-Y.: Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm. *Intel Corporation, Microprocessor Research Labs*, 2000.

[7] Breiman, L.: Random Forests. *Machine Learning*, ročník 45, č. 1, 2001: s. 5–32.

[8] Burt, P.; Adelson, E.: The Laplacian Pyramid as a Compact Image Code. *Communications, IEEE Transactions on*, ročník 31, č. 4, April 1983: s. 532–540.

[9] Harris, C.; Stephens, M.: A Combined Corner and Edge Detectio. *Proceedings of The Fourth Alvey Vision Conference*, 1988: s. 147–151.

[10] Ho, T. K.: Random Decision Forests. *Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1*, 1995.

[11] Horn, B. K. P.; Schunk, B. G.: Determining Optical Flow. *Artificial Intelligence*, ročník 17, 1981: s. 185–203.

[12] Jain, A. K.; Murty, M. N.; Flynn, P. J.: Data Clustering: A Review. *ACM Computing Surveys*, 1999.

[13] Kalal, Z.; Matas, J.; Mikolajczyk, K.: Online learning of robust object detectors during unstable tracking. *On-line Learning for Computer Vision Workshop*, 2009.

[14] Kalal, Z.; Matas, J.; Mikolajczyk, K.: P-N Learning: Bootstrapping Binary Classifiers by Structural Constraints. *Conference on Computer Vision and Pattern Recognition*, 2010.

[15] Kalal, Z.; Mikolajczyk, K.; Matas, J.: Forward-Backward Error: Automatic Detection of Tracking Failures. *International Conference on Pattern Recognition*, 2010.

[16] Lepetit, V.; Lagger, P.; Fua, P.: Randomized trees for real-time keypoint recognition. *CVPR*, 2005.

[17] Lucas, B. D.; Kanade, T.: An Iterative Image Registration Technique with an Application to Stereo Vision. *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)*, 1981: s. 674–679.

[18] Mohan, A.; Papageorgiou, C.; Poggio, T.: Example based object detection in images by components. *IEEE Trans. Pattern Anal. Mach. Intell.*, ročník 23, 2001.

[19] Moravec, H.: Visual Mapping by a Robot Rover. *Proceedings of the 6th International Joint Conference on Artificial Intelligence*, 1979: s. 599–601.

[20] Nebehay, G.: Robust Object Tracking Based on Tracking-Learning-Detection. *Diplomarbeit*, Wien, Technische Universität Wien, 2012.

[21] O'Donovan, P.: Optical Flow: Techniques and Applications. The University of Saskatchewan, April, 2005.

[22] Ozuysal, M.; Fua, P.; Lepetit, V.: Fast Keypoint Recognition in Ten Lines of Code. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, ročník 0, 2007: s. 1–8.

[23] Schneiderman, H.; Kanade, T.: A Statistical Method for 3D Object Detection Applied to Faces and Cars. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference*, 2000.

[24] Tao, M. W.; Bai, J.; Kohli, P.; aj.: SimpleFlow: A Non-iterative, Sublinear Optical Flow Algorithm. *Computer Graphics Forum (Eurographics 2012)*, ročník 31, č. 2, May 2012.

[25] Viola, P.; Jones, M.: Rapid object detection using a boosted cascade of simple features. *CVPR*, 2001: s. 511–518.

[26] Vojíř, T.: Demo application for Tracking-Modeling-Detections. *Diplomová práce*, Praha, FEL ČVUT v Praze, 2010.

[27] Yilmaz, A.; Javed, O.; Shah, M.: Object tracking: A survey. *ACM Computing Surveys*, ročník 38, č. 4, December 2006.

[28] Zhu, X.: Semi-Supervised Learning Literature Survey. *University of Wisconsin — Madison, Computer Sciences TR*, 2008.

Příloha A

Obsah přiloženého CD

- Zdrojové soubory programu
- Knihovny potřebné k přeložení aplikace
- Přeložená aplikace pro Windows
- Manuál k ovládání aplikace
- Testovací videa
- Videoukázky činnosti detektoru
- Text diplomové práce v .pdf a .tex