Sem vložte zadání Vaší práce.

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra ... teoretické informatiky

Bakalářská práce

Analýza pohybu pro staticky umístěnou kameru

Ondřej Cvacho

Vedoucí práce: Ing. Radomír Polách

12. května 2014

Poděkování

V první řadě bych rád poděkoval Ing. Radomíru Poláchovi za vedení a možnost pracovat na tomto tématu. Dále rodině za finanční a morální podporu při mých studiích, přátelům a rodině Fillerových za motivaci.

Prohlášení

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

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

V Praze dne 12. května 2014 .....

Odkaz na tuto práci

Cvacho, Ondřej. *Analýza pohybu pro staticky umístěnou kameru.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2014.

Abstrakt

Počítačové vidění je rozrůstajícím se oborem v informatice a nabízí zajímavé oblasti pro zkoumání. Tato práce si bere za cíl poskytnou náhled na danou problematiku a zjištěné informace aplikovat na vývoji systému pro analýzu videa.

Klíčová slova Počítačové vidění, OpenCV, python, teplotní mapa, sledování, klasifikace objektů, odčítání pozadí

Abstract

Computer vision is a fast-growing discipline in a computer science industry and it offers interesting fields for research. This bachelor's thesis aims to provide brief overview about approaches for solving problems related to computer vision and how to apply these knowledge to develop a system for video analysis.

Keywords Computer vision, OpenCV, python, heat map, tracking, object classification, background subtraction

Obsah

Seznam obrázků

Úvod

Počítačové vidění je rychle se rozrůstající obor informatiky, protože nabízí využití nových možností pro aplikace. Může se jednat o interakci mezi člověkem a strojem. Příkladem může být ovládání počítače pomocí pohyby vlastního těla. Dobrým příkladem úspěchu této technologie je ovladač Kinect od společnosti Microsoft, kterým se dají ovládat kompatibilní hry pouze pohybem těla a přístroj zaznamenal velký úspěch [1]. Stejně u handicapovaných lidí se dá použít počítačové vidění pro umožnění ovládat počítač snímáním pohybů jejich očí nebo rukou.

Ale počítačové vidění se dá využít i v dalších oblastech. Jednou z nich je využití z bezpečnostních důvodů nebo sledování dění. Pro bezpečnost se může jednat o informace o počtu osob přítomných v objektu nebo dohled nad zakázanými oblastmi. V mobilním průmyslu se počítačové vidění používá u systémů varujících před možnými kolizemi s jinými objekty na vozovce. Jako jednou z velice zajímavých oblastí využití, je sledování osob, kdy se dají rozeznat „nervózní lidé“ z jejich chování a pohybu těla.

Dalším možným využitím je sledování pohybu za účelem zlepšení marketingu obchodníka, tímto si může zjistit frekventovaná navštěvovaná místa v obchodě a podle toho upravit rozložení nabízených produktů [2].

Kapitola 1

Cíl práce

Cílem práce je navrhnout a implementovat algoritmy, potřebné pro analýzu videa, na detekci a sledování objektů. Sledované objekty by měly být jak lidé, tak ostatní objekty reálného světa, a systém by měl dokázat jejich rozeznání. Systém řešící tento problém by měl sloužit jako analyzátor videa, který poskytne informace o pohybu lidí ve sledované oblasti. Tento výstup bude ve formě teplotní mapy (více o teplotních mapách v kapitole 2.1.6).

1.1 Popis řešeného problému

Analýza pohybu z videozáznamu může být jednoduchý problém, ale také velice složitý. Vše záleží na způsobu analýzy pohybu a vlastností, které se mají zaznamenávat. Pokud postačuje jednoduché rozhodnutí, zda je na videu nějaký pohyb či nikoli, bude aplikace řešící tento problém velice jednoduchá.

Jeden z nejsložitějších problémů v analýze obrazů a videa, je rozhodně strojová klasifikace a rozeznávání objektů. Tento problém je například pro člověka triviální. Naše neuronové sítě v mozku jsou propojeny a naučeny natolik dobře, že rozeznání obličejů nám může trvat nejhůře vteřiny a navíc naše možnosti rozpoznání jsou, i oproti dnešním nejlepším algoritmům, mnohonásobně úspěšnější. Počítač je odkázán na kvalitu implementovaných algoritmů. Ty se vyvíjejí od raných počátků počítačového vidění a za tu dobu bylo publikováno mnoho prací a postupů. Moderní metody jsou bezpochyb, oproti dřívějším, výrazně lepší, ale některé starší techniky se dnes stále hojně používají, protože nabízejí dostatečný poměr mezi kvalitou detekce, jednoduchostí a výkonem. Problém klasifikace je nejenom jeden z nejnáročnějších disciplín v počítačovém vidění, ale také jeden z nejvíce zkoumaných, protože nabízí veliké možnosti zlepšení.

Po vyřešení problému detekce objektů a jejich separace, je na řadě samotné sledování detekovaných objektů. I tento problém nabízí ohromnou škálu možných řešení. Jednoduché postupy na každém snímku videa naleznou objekty popředí, které jsou porovnávány s již detekovanými objekty dříve. Takové po-

rovnávání se může provádět podle různých vlastností objektu, jako například barva nebo tvar. Až po komplexnější postupy řešení, které staví na pravděpodobném pohybu objektu a následném zpřesnění polohy. Takový algoritmus u každého sledovaného objektu uchovává informace, podle kterých vypočítává jeho pravděpodobnou novou polohu. Mezi takové informace se řadí rychlost, směr pohybu nebo orientace objektu. K výslednému zpřesnění polohy objektu se využijí další jeho vlastnosti, stejně jako u předešlé metody barva nebo tvar, ale mohou mít přesnější reprezentaci. Díky odhadu nové polohy objektu se nemusí prohledávat celý nový snímek a stačí se omezit jen na vytyčenou oblast. To nám může ušetřit i výpočetní výkon, kdy prohledávání jen části snímku je časově méně náročné, než prohledávání celého snímku. Díky získaným informacím o objektu se může postupně přesnost sledování zvyšovat.

Mezi překážky při sledování objektů patří částečné nebo úplně překrytí objektů, náhlá změna pohybu objektu, změny vzhledu pozadí nebo sledovaného objektu. Tyto situace jsou většinou málo předvídatelné a robustní algoritmy si s nimi musí poradit.

Moderní způsoby řešení klasifikace lidí staví na přednaučení modelu, který je pak porovnáván s nalezenými objekty. Tato metoda se zdá být efektivní a nabízí vysokou úspěšnost, avšak je implementačně složitější a její úspěšnost se odvíjí od kvality trénovacího vzorku dat.

1.2 Vymezení cílů a požadavků

Hlavními požadavky na funkčnost jsou:

1. Výstup ve formě teplotních map, které zobrazují nejfrekventovanější místa pohybu objektů.
2. Systém který dokáže videozáznam zpracovávat v reálném čase.
3. Sledování objektů po dobu jejich výskytu na záznamu.
4. Klasifikace lidí a ostatních objektů, které se objeví ve sledovaném prostoru.

Systém by měl tedy pracovat v reálném čase pro libovolný vstupní videozáznam, ať uložené video nebo přímý přenos z kamery. Zpracovávání informací bude prováděno vždy jen pro jeden video vstup.

Předností systému by měla být jeho snadné používání. Předpokladem je, že systém bude možné použít bez jakýchkoliv dalších zásahů uživatele. Tedy v ideálním případě si uživatel systém stáhne, nastaví nutné informace a bez dalších zásahů systém spustí. Předpokládá se využívání systému v dlouhodobém časovém rozmezí, je tedy důležité, aby takovýto chod zvládal a postupem času nezvětšovat svojí zátěž na výpočetní výkon, ani jiné systémové prostředky.

Charakter systému umožňuje využít méně dokonalý detektor lidí, který se bude projevovat menší úspěšností správně klasifikovat osoby, na úkor menší náročnosti na výpočetní výkon.

1.3 Struktura práce

Struktura práce je rozdělena do kapitol. V kapitole 2 bude shrnut obor počítačového vidění. Základní poznatky a možná řešení problematiky sledování, segmentace a klasifikace objektů, zde bude také popsáno. V následující kapitole 3 budou představeny již existující systémy, řešící podobný problém, a představení zvoleného implementačního prostředí. Poslední kapitola 4 bude obsahovat popsání výsledné implementace systému, její nedostatky a možná budoucí vylepšení. Vše bude nakonec shrnuto v závěru.

Kapitola 2

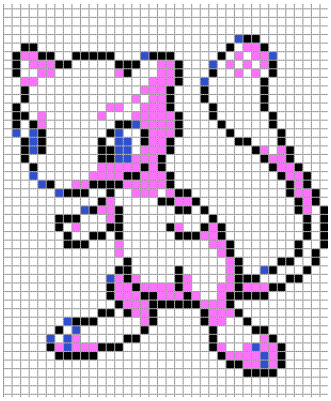
Počítačové vidění

Počítačové vidění je snaha počítač naučit vidět stejně jako lidé.

Jak již bylo napsáno, počítačové vidění je obor velice rychle rostoucí a v posledním desetiletí si získal také velikou oblibu. Rozvoj tohoto oboru se může vázat na zvyšování výpočetního výkonu počítačů a také rozšířením volně dostupných, lépe či hůře fungujících, otevřených knihoven, implementujících základní algoritmy využívané v počítačovém vidění. Díky takovým knihovnám je dnes velice snadné začít vyvíjet systémem, který je postaven na počítačovém vidění a zpracovávání grafického obsahu.

Mezi oblasti počítačového vidění patří tedy zejména sledování a rozpoznávání objektů. Ale to není jediné využití. Může se jedna o zpracování obrazů, kde slouží převážně k odstranění šumu z obrazů, jiných nežádoucích jevů nebo naopak modifikace vstupního obrazu, jako změna rozměrů či jiné morfologické operace obrazu. V medicíně se může počítačové vidění využít pro vytváření modelů orgánů, například mozku, díky nímž jdou lépe provádět diagnózy. Dále to jsou různé rekonstrukce scény, například tvorba 3D modelu ze série snímků nebo panoramatické fotografie.

Obrázek 2.1: Zleva doprava: originální obraz, obraz po přidání šumu, odstranění šumu pomocí Gaussovské distribuce



Obrázek 2.2: Příklad jak je obraz skládán z jednotlivých pixelů. [3]

2.1 Zpracování obrazu

2.1.1 Pixel

Pixel je samostatný bod reprezentující určitou barvu. Pokud budeme reprezentovat pixel ve stupních šedi, postačí nám na jeho uložení pouhý jeden *byte* (256 odstínů šedi). Při reprezentaci v barevném modelu se většinou hodnota pixelu skládá z více složek. Například v barevném modelu RGB, je pixel reprezentován třemi barvami (červená, zelená a modrá) a jejich kombinací vznikne výsledná barva, tak jak ji vnímáme lidským okem. V počítačovém vidění se většinou pracuje s intenzitou pixelu, což je hodnota kombinující všechny složky, ze kterých se pixel skládá.

2.1.2 Obraz

Obraz je složen z pixelů, které jsou uspořádány vedle sebe. Na obrazu 2.2 jsou viditelné jednotlivé pixely, které jako celek tvoří obraz. Pro účely počítačového vidění musíme naše vnímání 3D světa namapovat do počítačové reprezentace. Nejlepším možným způsobem je reprezentovat obraz jako 2D matici a jednotlivé prvky matice představují jeden pixel. Při takovém zobrazení z našeho 3D vnímání okolí do 2D reprezentace, bohužel ztrácíme některé cenné informace o scéně, například hloubku.

Obraz má vlastnosti jako barevné schéma, ve kterém jsou reprezentovány jednotlivé pixely, šířku a výšku. Šířka a výška je označována jako rozlišení obrazu a zapisována ve formátu `šířka` $\times$ `výška`. Tyto hodnoty udávají počet pixelů v daném směru. Například obraz s šířkou 1024 a výškou 720 (značeno 1024×720), obsahuje $1024 * 720 = 737280$ samostatných pixelů.

Jednotlivé pixely obrazu jsou indexovány pomocí x a y souřadnic od počátečního $(0, 0)$ bodu.

2.1.3 Video

Video je tvořeno sekvencí obrazů. Je to tedy jakýsi „kontejner“ pro obrazy, který je zobrazuje v daném pořadí a počtu snímků za sekundu. Jednotlivé obrazy ve videu jsou v této práci označovány slovem snímek. Počet snímků za sekundu (fps) udává kolik snímků se má z videa zobrazit za dobu jedné vteřiny. Video o délce 5 vteřin a 25 fps, obsahuje sekvenci 125 snímků. Všechny snímky z jednoho videa mají většinou stejné rozlišení.

Aby člověk vnímal video jako plynulé, musí mít nejméně 24 snímků za sekundu. Pokud tedy chceme aby systém, který má zobrazovat výstup, byl vnímán jako plynulý (tedy v reálném čase), musíme zpracovávat jeden snímek z video sekvence nejhůře za dobu $41ms$.

2.1.4 Feature

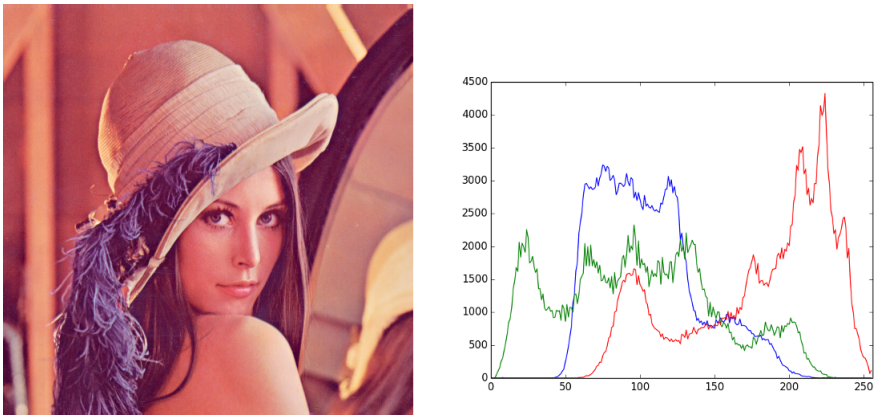
Pojem feature, v počítačovém vidění, zastupuje jakýsi obecný termín popisující informaci o vlastnosti a přesná definice vždy závisí na konkrétním problému. Obecně se jedná o struktury obsažené v obrazu (může to být bod, hrana nebo celkový objekt). Je to tedy jakási vlastnost obrazu, ale v některém kontextu by pojem vlastnost mohl být matoucí. U klasifikace, feature může zastupovat například barvu, tvar nebo rozložení a počet klíčových bodů 2.1.4.

Všechny techniky počítačového vidění jsou na features velice závislé, protože ovlivňují jejich celkovou úspěšnost. Proto kvalita features, a také kvalita jejich detekce, jsou udávajícím faktorem úspěšnosti výsledku. Je potřeba, aby vybrané features bylo možné nalézt v dalších obrazech nebo snímcích, proto se vybírají jen takové features, které mají velkou informativní hodnotu.

Hrana a Obrys

Hrana je část obrazu kde se mění jedna barva na druhou. Přesněji hranice mezi rozdílnými intenzitami barev. Takto funguje jednoduchý hranový detektor, projde celý obraz a každý pixel ohodnotí číslem, které odpovídá vzdálenosti jeho intenzity od intenzity pixelu sousedního. Poté všechny pixely, které mají ohodnocení větší než určitá mez, jsou označeny jako součást hrany. Existující hranové detektory [4, 5].

Samotná hrana nám o objektu ke kterému náleží moc neřekne. Když nalezneme hrany v obrazu, tak ještě nemusíme dostat segmentovaný objekt (pojem segmentace bude vysvětlen později 2.1.7), protože mohou být hrany více objektů vzájemně spojeny. Můžeme však nalézt celkový obrys objektu na základě nalezených hran jejich shlukováním, v binárního obrazu všechny „bilé“ plochy nebo jiných postupů [6].



Obrázek 2.3: Histogram znázorňující rozložení barev v obrazu.

Klíčový bod

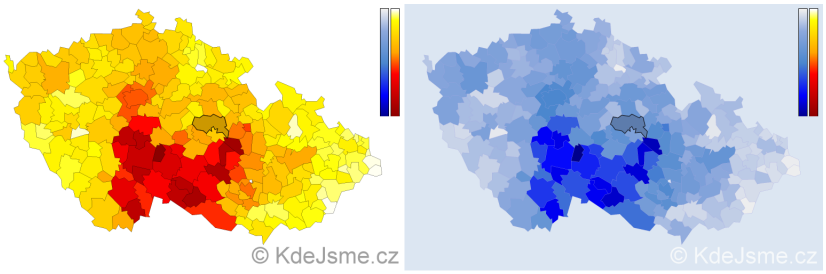
Klíčový bod (angl. *interest point*) je jednou z možných feature objektu a může být označen i jako „roh“ (angl. *corner*). Jedná se o bod v obrazu, který je unikátní ve svém okolí nebo obsahuje víc informací než jiné body. Jako klíčové body v obrazu, jsou často označovány reálné „rohy“, tedy body, kde se protíná dva a více hran. Nebo světlé body s tmavým okolím, stejně tak jakékoli body kde se mění intenzita barev, proto klíčové body často leží na hranách. Detektory klíčových bodů [7, 8, 9].

Naopak velice špatnými kandidáty na klíčové body jsou jednolité povrchy obrazu, které nejsou nijak odlišné od okolí. Jejich sledování ani nalezení na dalších snímcích je velice obtížné, protože se mohou zaměnit za jiné, velice podobné plochy.

2.1.5 Histogram

Jednou z nejvyužívanějších věcí v počítačovém vidění jsou histogramy. Histogramy jsou ale používané i v jiných oblastech a jsou velmi často využívány pro zobrazení nejrůznějších naměřených informací. Pomocí nich se jednoduše dají znázorňovat data o výskytu hodnot. Jedná se o graf, nebo grafické znázornění, určité pozorované vlastnosti. V počítačovém vidění je příklad histogram obrazu, který poskytuje grafické znázornění celkového rozložení všech barev.

Řekněme, že máme obraz složen ze čtyř barev s rozlišení 5×5. Každý pixel je reprezentován jednou z těchto barev. Pro každou barvu zjistíme počet pixelů s danou barvou. Dostáváme čtyři skupiny s počtem pixelů, které je rozložení barev v obrazu a dá se graficky znázornit. Histogram takto funguje a výsledek může zobrazit jako graf 2.3.



Obrázek 2.4: Teplotní mapa podle hustoty příjmení „Dvořák“ v jednotlivých oblastech ČR s použitím dvou barevných schémat. Převzato z [10].

2.1.6 Teplotní mapa

Teplotní mapy umožňují graficky znázorňovat 2D data, jak je ukázáno na obrazu 2.4. Barevně zobrazuje data podle jejich hodnoty, kdy nízké hodnoty dat mají jinou barvu než vysoké hodnoty. To se provádí tak, že se nalezne největší a nejmenší hodnota v datech. Toto rozmezí se rozdělí do několika menších intervalů, kde každému bude přiřazena barva. Poté se postupně procházejí hodnoty dat a podle toho, do kterého intervalu hodnota patří je odpovídající pixel ohodnocen barvou intervalu.

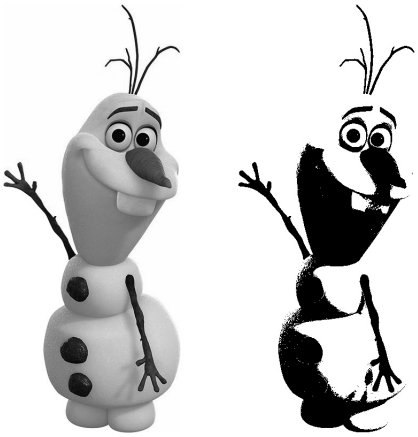
Barvy přiřazené intervalům dohromady tvoří barevné schéma teplotní mapy a pro různá data a různému účelu, může být vždy použité schéma jiné. Například barevné schéma, kde nízké hodnoty jsou reprezentovány světlou modrou barvou, poté přecházejí do sytějších barev, až po sytě rudou barvou kterou jsou ohodnoceny nejvyšší hodnoty. Takovéto schéma je ideální reprezentací pro člověka, protože poskytuje na první pohled patrné rozdíly mezi oblastmi s nízkými hodnotami a těmi s vysokými.

Teplotní mapy se také využívají například pro vizualizaci oblastí webových stránek, kterou jsou nejvíce „čteny“ návštěvníky, nebo na které části je nejvíce klikáno.

2.1.7 Segmentace

Pro potřeby dalšího zpracování obrazů a videa potřebujeme od sebe rozlišit oblasti, které mají stejné nebo podobné vlastnosti. Pomocí segmentace se takto dá předzpracovat obraz a dále na něm provádět další potřebné kroky pro získání potřebného výstupu.

Podle potřeb dalšího zpracování a výsledku se také musí vybrat postup, jak potřebné segmentace dosáhnout. Pokud chceme z obrazu získat plochy konkrétní barvy, využijeme segmentaci založenou na barvě, která nám tyto oblasti zvýrazní. Nebo můžeme segmentovat na základě hran obrazu, takto dostáváme možnost od sebe oddělit objekty jednoduchým způsobem.



Obrázek 2.5: Obraz v odstínech šedi byl pomocí prahování upraven na binární. Originál převzat z [11]

Prahování

Prahování je operace, která modifikuje vstupní obraz. Vstupní parametry algoritmu je hodnota prahu T, která rozděluje hodnoty obrazu na dvě skupiny, minimální hodnota min a maximální hodnota max.

Všechny hodnoty, které jsou menší než práh T, jsou převedeny na zadanou minimální hodnotu min, a naopak, hodnoty větší než práh zase na maximální hodnotu max. Vyjádřeno rovnicí 2.1.

$$p_{new}(x,y) = \begin{cases} min & p(x,y) < T \\ max & p(x,y) > T \end{cases} \tag{2.1}$$

$p(x,y)$ značí hodnotu daného pixelu.

Příkladem je prahování obrazu ve stupních šedi. Práh byl zvolen na hodnotu 150 a výsledný obraz 2.5. Tomuto prahování se říká binární, protože výsledný obraz se skládá pouze ze dvou hodnot a to min a max.

Vylepšení metody přinesl ve své práci Otsu [12], kde bylo rozšířeno prahování o nekonstantní práh T, který je upravován na základě sousedů pixelu, a tím se docílilo lepší segmentace. Jeho přístup je dnes nejčastěji používán.

K-mean shlukování

Algoritmus sloužící pro segmentaci obrazu do k shluků [13]. k je vstupním parametrem před spuštěním algoritmu. Přiřazení pixelu do určitého shluku se provádí na základě průměrné vzdálenosti ke středu shluků. Středy shluků mohou být vybrány algoritmem náhodně anebo využity nějaké heuristické metody. K-mean shlukování se využívá, mimo zpracování obrazu, také v oblasti vytěžování dat, kde potřebujeme rozdělit vstupní skupinu dat do k oblastí.

Jedná se o techniku strojového učení a poté se využívá při klasifikaci nově napozorovaných bodů.

V počítačovém vidění jsou naopak středy shluků vybírány například jako k nejvyšších vrcholů v histogramu nebo také náhodně. Postupně jsou do shluků přidávány pixely s podobnou barevnou intenzitou jakou má jeho aktuální střed. Středy mohou být také postupně aktualizovány, kdy průměrná hodnota všech pixelů v daném shluku bude novým středem. Takto můžeme, za předpokladu známého počtu objektů s jednolitou barvou, jednoduše segmentovat obraz nebo jej zjednodušit pro další zpracování.

Detekce hran

Předchozí dvě metody využívali pro segmentaci barevné schéma a po segmentaci obraz obsahoval spojené oblasti s podobnou barevnou intenzitou. Detekce hran je alternativou pro tyto metody. Po aplikování této segmentace na obraz, dostaneme výstupem seznam všech nalezených hran. Jak bylo dříve popsáno v kapitole 2.1.4, hrana je hranice mezi dvěma oblastmi, a pro jejich nalezení se využívají dva hlavní postupy, porovnávání úseků obrazu se šablonou hran a nebo vypočítání gradientu. Více informací v [14, Kapitola 5: Edge detection].

2.2 Detekce pohybu

Algoritmy, zajišťující detekci pohybu, pracují s aktuálním zachyceným snímkem a libovolně velikou skupinou snímků předcházejících. Jedná se o formu segmentace objektů popředí od pozadí scény.

2.2.1 Odčítání pozadí

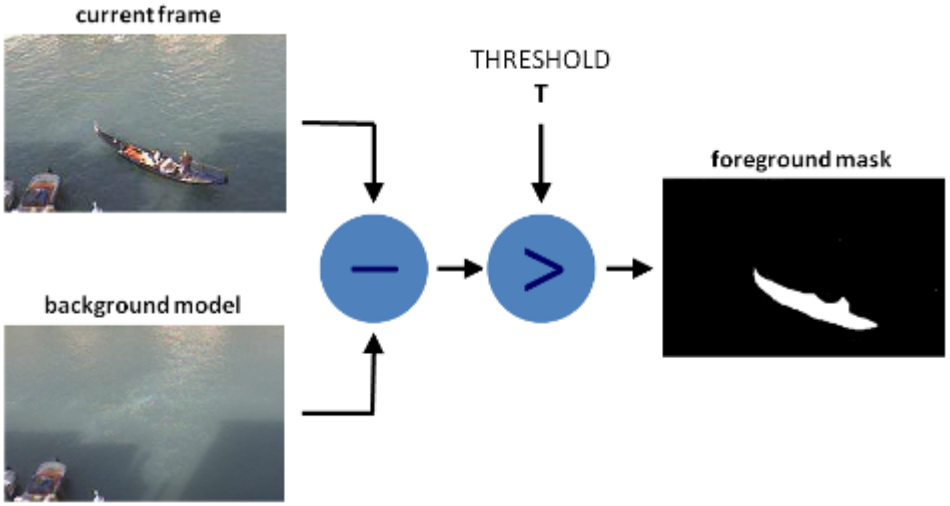
Tato metoda je jednou ze základních technik analýzy sekvence snímků. Snaha je ze sekvence snímků, ze statické kamery, detekovat všechny objekty popředí. Funguje na principu vytvoření referenčního modelu pozadí, který se poté odečte od aktuálního snímku a tím se vytvoří obraz obsahující jen objekty popředí. Odčítání pozadí pracuje s barevnou reprezentací pixelů. Na obrazu 2.2.1 je znázorněn diagram průběhu této metody.

Následující rovnice 2.2 udává obecný vztah principu této metody. Výsledný obraz je poté ještě metodou prahování zbaven šumu.

$$|snímek_t - pozadí| > práh \tag{2.2}$$

Jak si lze všimnou, výsledek je také citlivý na hodnotu zvoleného prahu.

Nejjednodušší formou metody odečítání pozadí, je aktuální snímek porovnávat jen se snímkem předcházejícím 2.3. Metoda je to velice jednoduchá a výpočetně nenáročná, ale bohužel neposkytuje velmi dobrý výsledek, protože může „zvýraznit“ jen část objektu popředí. Další překážkou u této metody



Obrázek 2.6: Diagram metody odčítání pozadí. Převzat z [15]

jsou malé opakující se pohyby, které jsou ale součástí pozadí. Například pohyby stromů ve větru, které by byly detekované jako součást popředí. Možným vylepšením, který by ovšem řešil jen část problémů, by mohlo být zvolit, jako referenční model pozadí, první snímek ze sekvence, nebo ten, který neobsahuje žádné objekty popředí a je tvořen jen pozadím. To je ale závislé na konkrétním případě využití a vstupních datech.

$$|snímek_t - snímek_{t-1}| > práh \tag{2.3}$$

Další překážkou jsou také změny v osvětlení pozadí, které by předchozí naivní metoda také označila jako objekt popředí. Může se jednat o náhlé změny nebo postupné. U postupných změň by mohla být situace vyřešena použitím „ideálního“ prahu při zpracování výsledného snímku popředí. Protože v takovém případě se hodnota pixelu může od původní lišit jen o malé číslo, vhodně zvolený práh by v takovém případě přinesl zlepšení.

Značné vylepšení vychází tedy z vytváření kvalitního modelu pozadí. Mezi existující techniky patří počítání průměru hodnot pixelu ze sekvence snímků nebo pomocí reprezentace hodnoty pixelu pomocí Gaussovské distribuce.

Počítání průměru hodnot z předchozích n snímků může zlepšit výsledný model pozadí a tak zkvalitnit detekci změn. Naivním řešením je uchovávání předchozích n snímků a z nich vždy spočítat průměr pro každý pixel obrazu. Tento přístup se dá zlepšit. Když budeme hodnotu pozadí počítat rovnou při zpracování váženým průměrem, nebudeme si muset ukládat zbytečně $n - 1$ snímků navíc. Poté je každým novým snímkem model pozadí ovlivněn rovnicí 2.4.

$$pozadí(x,y) = \alpha * snímek_t(x,y) + (1-\alpha) * pozadí(x,y)_{t-1} \tag{2.4}$$

α v tomto případě udává míru učení, tedy jak moc ovlivní aktuální hodnoty pixelu model pozadí. Pro malé hodnoty α bude pozadí méně ovlivněno a naopak pro velké hodnoty.

Podobně při hledání hodnoty pixelu lze využít histogramů. Díky reprezentaci hodnot pixelu pomocí histogramu, nalezneme tu hodnotu, která se nejvíce vyskytovala. Metoda histogramů byla rozšířena pomocí [16], kdy místo klasického histogramu je hodnota pixelu modelována Gaussovou funkcí, která slouží jako pravděpodobnostní rozdělení hodnot pixelu. Tak získáme průměrnou hodnotu pixelu a odchylku od této hodnoty. Díky tomu dostáváme rozmezí hodnot pixelu, které se považují za hodnoty pozadí. Z aktuálního snímku získáme hodnotu daného pixelu a porovnáme zda leží v tomto rozmezí hodnot pozadí, jinak se jedná o popředí 2.5.

$$\begin{array}{l} \frac{|I(x,y)_t - \mu_t|}{\sigma_t} > P \in popředí \\ \frac{|I(x,y)_t - \mu_t|}{\sigma_t} < P \in pozadí \end{array} \tag{2.5}$$

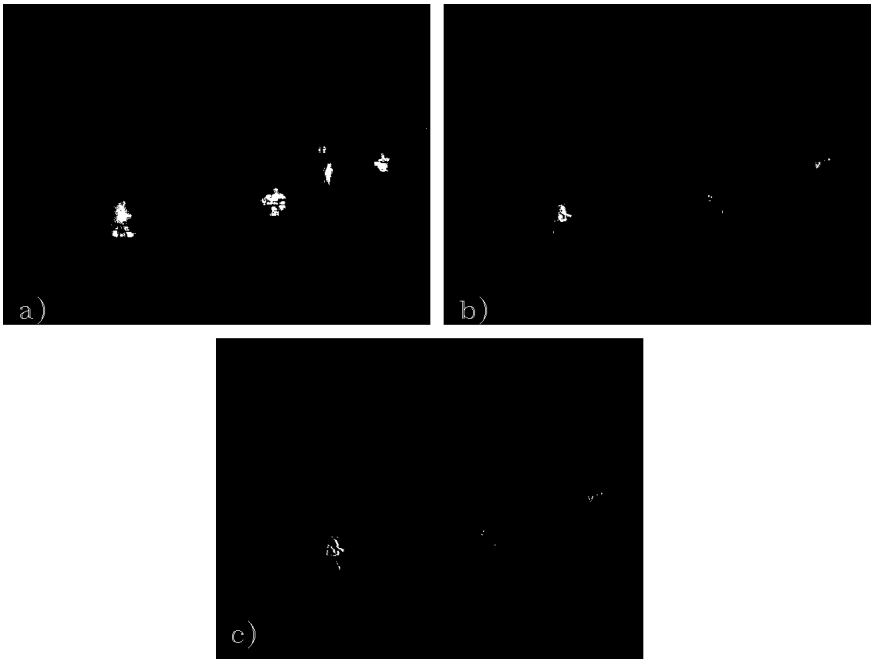
Kde $I(x,y)_t$ označuje hodnotu pixelu na aktuálním snímku, μ_t je průměrná hodnota pixelu a σ_t značí možnou odchylku od průměru. P je poměr mezi vzdáleností $I(x,y)$ od průměrné hodnoty a odchylkou distribuce. P zde plní funkci podobnou k u prahování.

Z tohoto přístupu vychází rozšíření, které nevyužívá jen jednu Gaussovu distribuci na reprezentaci hodnoty pixelu, nýbrž několik. Původně byla tato metoda představena pány Stauffer a Grimson [17]. K Gaussových distribucí, K je doporučeno malé (podle [17] číslo v rozmezí 3 až 5), kde každá reprezentuje pravděpodobnost výskytu jedné barvy a odpovídající průměrnou hodnotu a odchylku. Každá Gaussova distribuce může reprezentovat například barvu popředí nebo pozadí. Jde vlastně o K nejpravděpodobnějších barev, které se mohou v daném pixelu objevit. Rozhodnutí, zda je aktuální hodnota pixelu pozadí nebo popředí, se provádí podobně jako u předešlého metody, ale je porovnávána se všemi K Gaussovými distribucemi. Mohou nastat dva případy:

1. Shoda je nalezena v jedné z K Gaussových distribucí. V takovém případě, pokud je to Gaussova distribuce pozadí, je pixel klasifikován jako součást pozadí, v opačném případě je pixel součástí popředí.

2. Shoda v žádné Gaussovově distribuci nebyla nalezena. V takovém případě je pixel součástí popředí.

Tato metoda získala velkou oblibu. Ale i když přináší oproti předchozím značné vylepšení segmentace, bylo publikována mnoho dalších metod, které ji dále vylepšují [18, 19]. Ty se zaměřují jak na zlepšení výpočetní náročnosti,



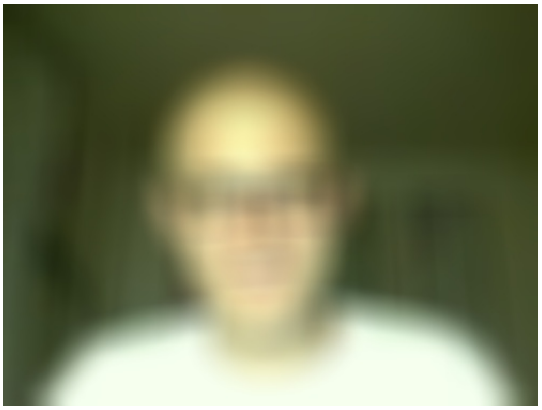
Obrázek 2.7: Srovnání metod odčítání pozadí. a) K Gaussova distribuce, b) vážený průměr, c) odčítání pouze předchozího snímku.

tak i na zlepšení segmentace popředí od pozadí. Existují i postupy tvorby pozadí, které dokáží rozeznat periodicky se opakující děje v pozadí a tak je chybně neoznačovat jako popředí, například mořské vlny. Více o metodách využívající Gaussovu distribuci se dá dočíst v článku [20].

Všechny představené metody tvorby pozadí, zpracovávají obraz pouze s ohledem na jednotlivé pixely nezávisle na sobě. To znamená, že hodnota pixelu je počítána jenom jako hodnota odpovídajícího pixelu z předchozích snímků. Existují ale i metody, které využívají informací z okolí a jsou tak méně náchylné na rapidní změny v celkovém stavu obrazu [21].

2.3 Klasifikace objektů

Po získání oblasti s možným výskytem objektu, nalezeného například pomocí metody odčítání pozadí, následuje potřeba rozeznat o jaký objekt se jedná. Nejdůležitějším faktorem u klasifikace, je potřeba vybrání takových features, které jsou pro objekt identifikující. Pokud je potřeba rozlišovat objekty základních neměnných tvarů, jako například čtverec, kruh nebo hvězda, postačí zjistit tvar objektu. Čtverec má vždy čtyři hrany a všechny mají stejnou délku, proto tento údaj postačuje pro jeho klasifikaci. Pokud je ale potřeba rozeznat složitější objekty, které navíc nemají vždy stejný tvar, musejí se využít metody,



Obrázek 2.8: Rozeznání objektu je pro člověka snadný úkol. Pro počítač je to ale mnohem náročnější.

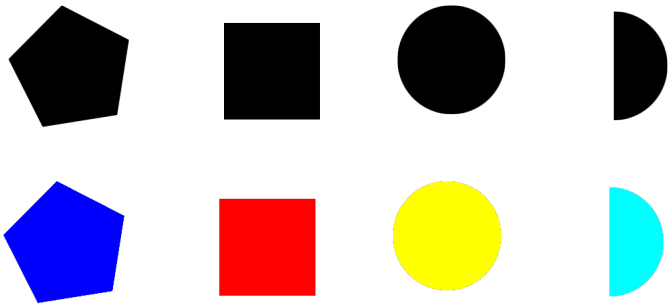
které dovolují rozlišit objekt s určitou pravděpodobností. Možným postupem při klasifikaci takovýchto objektů, může být vytvoření obecného modelu skupiny, se kterým se budou detekované objekty porovnávat.

Jako lidé nemáme problém s rozlišením velkého množství objektů v reálném prostředí ani v obrazu, i když je nějakým způsobem snížena jeho kvalita. Dokonce nejsme náchylní ani na rotace objektů nebo změny velikosti. Tento problém je však stále velkou výzvou pro strojové vnímání počítače. Na příkladu 2.8 je zachycen obličej člověka.

Musíme si uvědomit, že systémy počítačového vidění vždy dokáží rozlišit pouze předdefinované objekty neboli ty, na které byl klasifikátor navrhnut a naučen. S přihlédnutím na vstupní data, která většinou obsahují šum a mohou být rozdílná, nejsou tyto klasifikátory nikdy stoprocentní. V kontrastu od algoritmů na řazení dat, algoritmy počítačového vidění jsou úspěšné jen v některých případech a jejich autoři zveřejňují procento úspěchu na konkrétních testovacích datech.

2.3.1 Jednoduchá klasifikace

U klasifikace objektů, které mají jednotnou barvu, se použije barva jako feature a podle ní se budou objekty klasifikovat. K tomu je potřeba segmentace obrazu, která odstraní všechny oblasti s jinou než hledanou barvou a výstupem budou jen oblasti výskytu objektu. Tento přístup se dá použít u předem známých objektů, které mají vždy stejnou barvu. Například pokud je potřeba najít a klasifikovat ovoce, bude nadefinováno, že jahody mají červenou a banány žlutou barvu, postačí nejdříve vstupní obraz segmentovat pro tyto dvě barvy a najít zbylé oblasti. Tyto oblasti budou označeny jako hledané ovoce. Navíc pokud se bude testovat tato metoda na sekvenci snímků obsahující jen jahody a banány, může se dostat na stoprocentní úspěšnost při klasifikaci.



Obrázek 2.9: Geometrické objekty jsou jedinečné tvarem, pětiúhelník má pět hran a podobně u dalších.

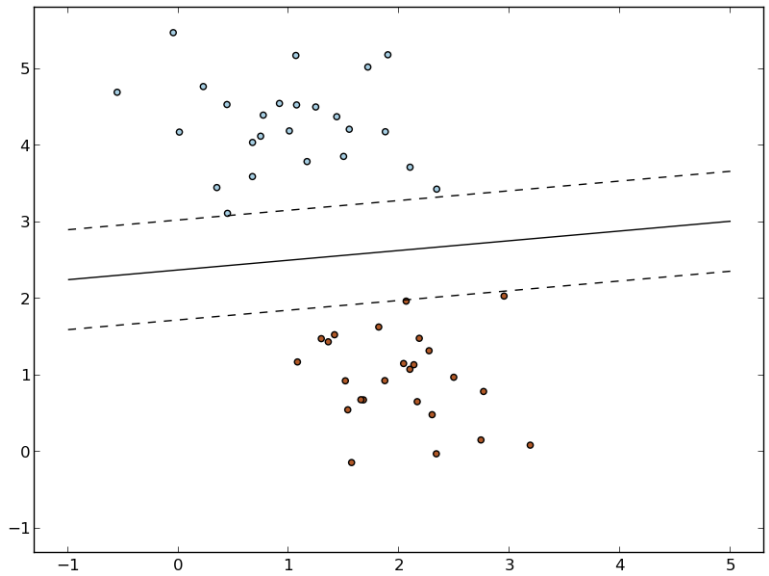
Tato klasifikace samozřejmě funguje jen pro vstupní data, která obsahují pouze hledané objekty a mají stejnou podobu. Pokud jako vstup zvolíme snímek, na kterém je červený míček, označí ho algoritmus chybně za jahodu.

Podobně se může postupovat i u klasifikaci geometrických tvarů. Vhodnou metodou segmentace obrazu se naleznou hledané oblasti, zjistí se obrys oblastí a na základě toho se objekty klasifikují. Čtverec nebo hvězdu se tímto dají jednoduše rozeznat, protože jejich tvar je přesně daný a pokud nalezená oblast nebude odpovídat, bude jisté, že to hledaný objekt není. Klasifikace, podle tvarů objektů, je znázorněna na obrazu 2.3.1. Například v [22] vypracovali metodu založenou na podobnosti tvaru objektu, a dalších heuristik, pro nalezení a klasifikaci objektů. Výslednou metodu testovali na datech siluet, ručně psaných znaků a značek.

2.3.2 Komplexnější klasifikace

V této sekci bude představena skupina metod pro klasifikaci založená na naučení modelu objektu, který se má klasifikovat. Tento postup se nazývá koordinované učení. Nejprve je potřeba zvolit podle jaké feature se bude klasifikátor učit rozlišovat objekty. Může to být například barva objektu, jeho obrys nebo vzhled reprezentovaný pomocí histogramů. Metody učení takových modelů zahrnují například neurální sítě, rozhodovací stromy, adaptive boosting nebo support vector machines.

Nevýhodou pro tyto metody je potřeba dostatečně velké kolekce vzorů pro každý objekt, podle kterých se klasifikátor naučí je rozlišovat. Navíc tyto metody vyžadují, aby všechny vzory byly označeny do jaké skupiny objektu patří. Kvalita trénovacích dat je také velice důležitá. Takto naučený klasifikátor pracuje jen za určitých podmínek, kdy je hledaný objekt plně viditelný, nebo jen částečně překrytý. Kvalitně naučený a robustně navrhnutý klasifikátor je na tyto případy sice méně náchylný, ale dostat se za každých podmínek ke stoprocentní klasifikaci, není ani v takovém případě jednoduchý úkol.



Obrázek 2.10: Nalezení nejlepšího rozdělení dvou tříd pomocí SVM a krajní body.

Support vector machine - SVM

SVM jako klasifikátor rozděluje vstupní data na dvě třídy. První třída odpovídá hledanému objektu (pozitivní vzory) a druhá ostatním datům (negativní vzory). Ukázka rozdělení 2.3.2. Proto tento klasifikátor potřebuje pro naučení také negativní vzory. Principem je nalézt takovou rovinu, která bude mít od okrajových bodů obou skupin největší vzdálenost, a takovým bodům se říká support vektory. Jedná se o lineární separaci dat, ale existují postupy, jak udělat z SVM nelineární klasifikátor.

Naučený SVM klasifikátor, pro každý nový vstup, rozhodne do jaké třídy náleží a to tak, že místo aby porovnával vstup se všemi daty, porovnává jen se support vektory.

Adaptive boosting (AdaBoost)

Základním principem AdaBoost algoritmu je zkombinovat množinu slabých klasifikátorů [1]. Příkladem může být klasifikace pohlaví člověk na základě výšky, kde osoby vyšší 180 cm budou klasifikovány jako muž a menší jako žena. AdaBoost následně slabé klasifikátory ohodnotí váhou, kterou by přispívaly do predikce, a na základě této váhy zkombinuje slabé klasifikátory do jednoho robustního klasifikátoru.

[1] klasifikátory které mají malou úspěšnost predikce, ale lepší než náhodná klasifikace

Bag-of-words (BOW)

Tento klasifikátor pracuje s features objektu jako se „slovy“. Každý trénovací vzor reprezentuje pomocí vizuálních vlastností objektu, tedy vizuálních slov. Nalezená podobná slova vzorů jsou shlukována, aby vytvořila referenční vzorek pro detekci. Následně je každá třída reprezentována podle výskytu slov, pomocí histogramu, kde jsou u jednotlivých tříd spočteny výskyty jednotlivých slov a rozhodnutí, zda testovaný objekt náleží do třídy, je založeno na porovnání histogramů. Tedy oba objekty musí obsahovat stejná slova a o stejném počtu.

BOW využívá další algoritmy pro separaci dat a jejich klasifikaci. Například metodu nejbližších sousedů nebo SVM.

Shrnutí

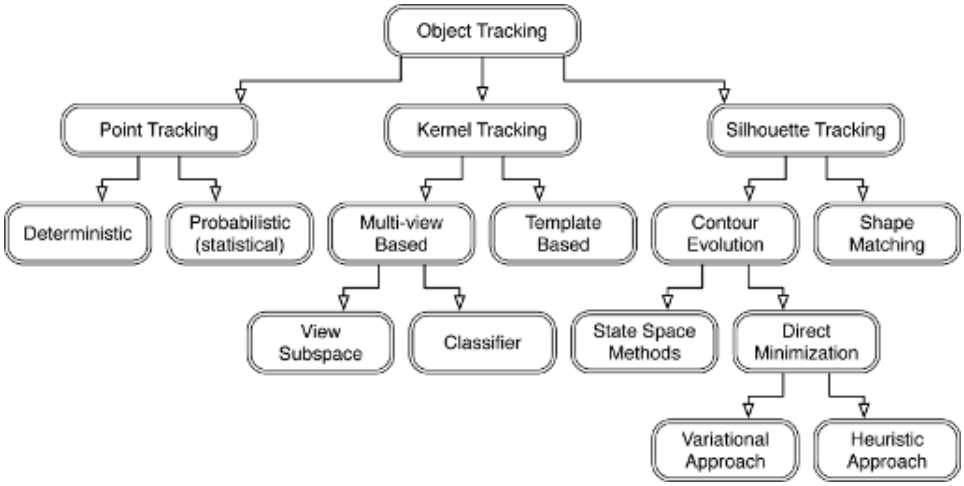
Tento výpis metod klasifikace není vyčerpávajícím výčtem, nýbrž jednoduchým přehledem. Zmíněna nebyla ani kategorie metod, které nevyžadují označené vzory, ale provádějí učení za chodu pozorováním objektů.

2.4 Sledování objektů

Po nalezení a klasifikaci objektu, následuje část sledování. Hlavní motivací pro sledování objektů v sekvenci snímků, je zachycení jejich pohybu v rámci sledované scény. Tato informace může pro některé aplikace počítačového vidění být značně užitečná, například kvůli kontrole, zda se sledovaný objekt neodchýlil od předem definované trasy.

Při sledování objektu je požadovaným cílem, v každém snímku video sekvence, nalézt novou polohu objektu. Pro nalezení nové polohy se dají využit například techniky, kdy objekt sledujeme využitím algoritmů speciálně určené pro sledování, které budou zmíněny níže, nebo postup, kdy se detekují a klasifikují objekty v každém snímku zvlášť a jen jsou vzájemně porovnávány s nalezenými objekty ze snímku předchozího. Oba tyto postupy mají své pro a proti. Například pokud bude k dispozici kvalitní model pozadí, detekce objektů popředí je jednoduchá a relativně nenáročná operace. Ale pokud objekt splyne s pozadím, například díky dlouhé nehybnosti, objekt se „ztratí“. Kombinace obou postupů by mohla přinést dobré výsledky.

Tato sekce bude zaměřená na popis algoritmů pro sledování. Základem pro tyto operace, je potřeba využít na reprezentaci objektu vhodnou formu, která se může liši pro každý případ a postup sledování. Když je potřeba sledovat jeden malý objekt, postačí na jeho reprezentaci jen jediný bod, protože jiný objekt nemůže ani se sledovaným přijít do styku. Proto je u vybírání té správné metody potřeba brát ohled i na podmínky, za kterých se bude objekty sledovat. Například pokud je jedinou součástí scény sledovaný objekt a pozadí je statické, nebo se pozadí mění v průběhu času a mohou nastat ko-



Obrázek 2.11: Rozdělení sledovacích metod. Převzato z publikace [23].

lize mezi sledovanými objekty, které nesmějí nijak narušit správnost sledování. Všechny požadavky je potřeba předem znát, aby byla vybrána ta nejvhodnější a nejrobustnější metoda pro dané podmínky. Na diagramu 2.4 jsou znázorněny metody sledování a jejich zařazení.

2.4.1 Optický tok

Optický tok je obecně vnímání vizuálního pohybu objektů vůči pozorovateli. V reálném prostředí tedy optický tok pozorovatelé určují z pozorování okolního světa, jako je určení směru pohybu jedoucího auta nebo „pohyb“ statických objektů, když se pohybuje pozorovatel.

V počítačovém vidění je optický tok metoda pro určení pohybu bodu [2]oproti předchozím snímkům. Optický tok se dá využívat také pro zjištění vzdálenosti objektů a to na základě informace o rychlosti pozorovatele a vektoru pohybu bodů, protože objekty blíže pozorovateli budou mít větší pohyb než vzdálenější objekty. Proto se metoda optického toku také využívá v robotice na řízení a automatickému vyhýbání překážek.

Optický tok staví na předpokladu, že pokud se intenzita hodnoty pixelu v aktuálním snímku změnila oproti předchozímu, znamená to pohyb. Pokud by se ale optický tok vypočítával pouze na úrovni pixelů, mohl by mít velice špatné výsledky na správné určení směru pohybu. Proto se bere v potaz i okolí pixelu. Dalším předpokladem je fakt, že okolí pixelu má stále stejnou intenzitu a jas barvy, i když se poloha může změnit. Navíc sousední body jsou většinou součástí stejné plochy a proto mohou mít také stejný směr a rychlost pohybu.

Pokud bude scéna obsahovat objekt, například kouli, která má jednolitou barvu a bude se pohybovat kolem své osy, metoda optického toku tento pohyb

[2]feature je v tomto případě bod a jeho okolí

nezachytí. Objekt se sice pohybuje, ale žádný z bodů snímku nemění svoji intenzitu a ani jas, a proto se zdá být objekt stále statický. Kdybychom ale začali pohybovat zdrojem světla, vyvolalo by to změny jasu jednotlivých bodů a tím pádem i pohyb.

Existují dvě kategorie optického toku, které se nazývají *řídký* a *hustý* optický tok. Na obrazech 2.4.3 a 2.4.3 jsou obě znázorněny.

Hustý optický tok

Do této kategorie patří například metody [24, 25]. Slovo hustý zde odráží skutečnost, že je optický tok počítán pro všechny body snímku. Někdy je také označována slovem *globální*.

Tento přístup umožňuje zjištění pohybu všech částí scény a díky tomu dostat kompletní přehled o dění na pozorované scéně. To se využívá například v navigačních systémech nebo robotice. Grafické znázornění může být barevnou mapou, kde se jednotlivé vektory pohybu obarví podle velikosti nebo směru.

Řídký optický tok

Na rozdíl od předchozího kategorie, se zde pracuje jen s určitou množinou bodů, které se metodou optického toku sledují. Nejrozšířenější je metoda [26].

Nejprve je potřeba získat množinu bodů, která bude vhodná pro sledování. Na to se využívají feature detektory popsané dříve. Body jsou na aktuálním snímku hledány v okolí původní polohy a to maximálně v určitém rozsahu. Proto se může stát, že díky vysoké rychlosti objektu, mohou být některé body při sledování ztraceny, nebo naopak u některých nesprávně určena nová poloha.

Optický tok a sledování objektů

Pro sledování objektů pomocí optického toku, je potřeba si nejenom uchovávat danou množinu bodů, podle kterých je objekt sledován, ale také informaci o struktuře objektu, jako obrys nebo ohraničující obdélník. Protože samostatné body nemají dostatek informací, o sledovaném objektu, na to, aby se mohl vypočítat nový obrys objektu.

2.4.2 Kalman filtr

Jedná se o pravděpodobnostní metodu, založenou na měření informací z dřívějších pozorování a výstupem je předpokládaná nová poloha sledovaného objektu. Protože kalman filtr je založen na měření více proměnných a bere v potaz i šum naměřených dat, je více přesný v predikcích než algoritmy, které by stavěly jen na jedné z těchto proměnných. Takto vytváří model pohybu objektu pro maximální přesnost predikce nové polohy.

$$X_t = K_t * Z_t + (1 - K_t) * X_{t-1} + \omega \tag{2.6}$$

Rovnice 2.6 obecně popisuje jak Kalman filtr provádí predikci aktuální polohy X_t. K_t je míra učení [3], Z_t je odhadnutá poloha a X_{t-1} označuje předchozí polohu. Výhodou Kalman filtru je používání šumu ω při predikci. Předpokládá se že šum má Gaussovu distribuci, i když v reálném prostředí to nemusí být vždy přesný odhad.

Postup kalman filtru se skládá ze dvou kroků:

1. Predikce – filtr na základě vytvořeného modelu odhadne novou polohu objektu.
2. Korekce – přesná naměřená hodnota polohy je filtru předána, aby mohl v dalším kroku zlepšit svoji predikci.

Kalman filtr je, podle původní definice, vhodný hlavně pro sledování objektů s lineárním pohybem, většina sledovaných objektů má ale nelineární pohyb a proto byly vyvinuty modifikace rozšiřující klasický Kalman filtr o podporu predikce nelineárních pohybů.

2.4.3 Mean-shift a Cam-shift

Při sledování objektů pomocí mean-shift algoritmu, je daný objekt reprezentovaný histogramem v barevném spektru. Jedná se o iterační metodu, která hledá objekt v novém snímku dokud nenajde oblast jejíž histogram má nejlepší shodu s histogramem reprezentující objekt, nebo do zadané možné odchylky. Díky této reprezentaci a možné odchylce při hledání, se tato metoda hodí pro sledování objektů, jejíž tvar se může měnit.

Cam-shift, na rozdíl od mean-shift, periodicky upravuje histogram reprezentující objekt a rozložení features. To umožňuje dynamicky upravovat velikost sledovaného objektu.

Předchozí tři kapitoly jsou popsány velice dobře ve článku [23], který obsahuje přehled a popis metod jak detekce, tak i sledování objektů.

[3] Kalman filtr je rekurzivní metoda a využívá vážený průměr pro hodnoty

Obrázek 2.12: Ukázka zobrazení hustého optického toku. Zdroj [27].

Obrázek 2.13: Sledování bodů pomocí optického toku. Zdroj [27].

Kapitola 3

Analýza a návrh

V této kapitole budou představeny již existující systémy a aplikace. Poté bude představeno zvolené implementační prostředí systému, který je výstupem této práce.

3.1 Publikované techniky

Histogram orientovaných gradientů (HOG)

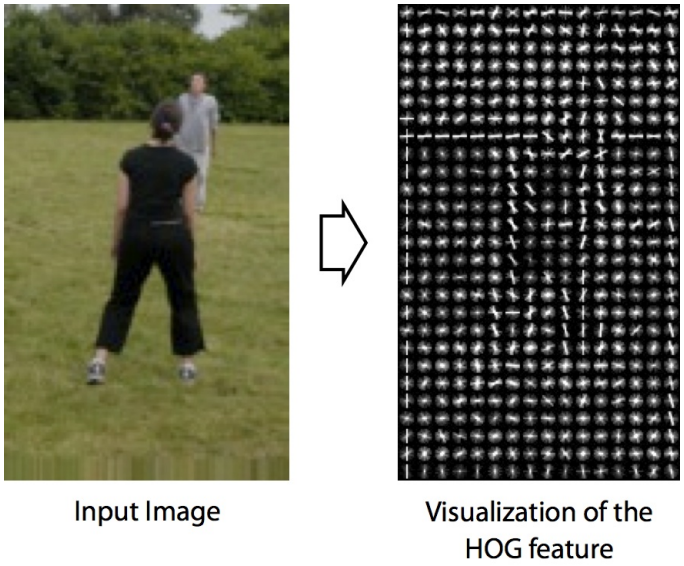
HOG je technika popisu objektu pomocí histogramů, které jsou využívány jako features. HOG se používá pro detekci a klasifikaci naučených objektů a v dnešní době je to jeden z nejpoužívanějších algoritmů při detekci lidí.

Detektor využívá HOG pro reprezentaci objektu jeho celkovou podobu, nikoli jen množinu features, u kterých by si dále potřeboval uchovávat informace o jejich zvájemném vztahu.

Využití HOG, pro účely detekce lidí, bylo prvně publikováno Dalalem a Triggsem [28]. Ve své verzi používají tedy HOG na popis objektu a pomocí této reprezentace naučí SVM klasifikátor rozpoznávat osoby, viz obraz 3.1.

Objekt, který je potřeba detekovat, je rozdělen na menší, překrývající se bloky, a pro každý tento blok je vypočítán histogram orientace. Spojením všech bloků se vytvoří model objektu. Všechny vstupní vzory, na kterých se SVM klasifikátor učí, musí mít stejné rozměry. V [28] používají vzory člověka s rozměry 64×128. Pro zajištění správného fungování je potřeba, aby hledaný objekt, ve vstupních datech, měl stejné, nebo větší rozměry, než na kterých byl detektor naučený.

Hledání člověka ve vstupním obrazu se provádí pomocí klouzavého okénka. Je tedy prohledáván postupně celý obraz na přítomnost objektu a oblasti, které mají dostatečnou shodu s vytvořeným modelem jsou označeny jako člověk.



Obrázek 3.1: Vizualizace HOG deskriptoru. Zdroj [29].

Sledování pomocí HOG deskriptorů

Protože HOG je jen určitá forma reprezentace oblasti, nemusí se používat jen při detekci objektů. Například v [30] je popsána metoda jak pomocí HOG deskriptorů sledovat objekty, navíc je tato metoda odolná i na překrytí objektů. Autor nejprve pro každý sledovaný objekt nalezne význačné body FAST detektorem [7] a poté pro každý vypočítal HOG deskriptor, jejichž pomocí je objekt sledován.

Další metody

Dále budou lehce představeny další existující projekty.

Multi-Target Tracking in Real-Time Surveillance Video[31]
: Tato metoda a implementace jsou zajímavé především výkonem. Umožňuje totiž detekci a sledování osob v reálném čase na videu ve vysokém rozlišení. Pro detekci byl autory použit HOG deskriptor a pro sledování metoda KLT [9]. Takových výsledků autoři dosáhli díky optimalizované, více vláknové implementaci.

Tracking-Learning-Detection [32]
: V této práci autor vyvinul metodu pro dlouhodobé sledování objektu. Objekt je reprezentován plochou, přesněji ohraničujícím obdélníkem, ve které se nachází. Při sledování objektu si tento algoritmus zároveň vytváří i model objektu. Díky tomu se může detekce postupem času i zlepšovat.

Continuous Energy Minimization for Multi-Target Tracking [33]
V této publikaci se autor zabývá sledováním objektů pomocí hledání optimální trajektorie objektů.

click2stream [34]
Click2stream je komerční aplikace, která nabízí klientům zpracování videa z IP kamer. Tento portál poskytuje služby jako teplotní mapy pohybu a počítání pohybujících se objektů.

Google Tango [35]
Tento projekt je zde zařazen hlavně díky své jedinečnosti. Zatím se jedná o prototyp telefonu, který by měl být schopný „vnímat“ okolí a pohyb stejným způsobem jako lidé. Telefon obsahuje spoustu senzorů snímajících pohyb zařízení a pomocí toho může vytvářet například 3D model okolního. Možnosti využití takového zařízení jsou obrovské, například usnadnit pohyb a navigaci zrakově handicapovaných lidem.

3.2 Implementační prostředí

Navrhovaný systém bude implementován využitím těchto prostředků:

3.2.1 OpenCV

Tato knihovna obsahuje implementace algoritmů počítačového vidění a strojového učení. Cílem vývoje knihovny OpenCV je poskytnout ucelený nástroj pro zpracování obrazu a tím zjednodušit vývoj nových projektů. OpenCV se dá považovat za takový standard v počítačovém vidění a knihovna je hodně rozšířená i mezi společnostmi jako Google a Microsoft [36]. Knihovna OpenCV je uvolněna ve formě otevřených zdrojových kódu, proto ji může kdokoli upravovat nebo přidávat nové algoritmy.

Knihovna OpenCV byla vybrána hlavně pro svou kvalitu implementace algoritmů a rozšířenosti. Navíc přímo podporuje vývoj aplikací v programovacím jazyce Python 3.2.2. Při implementaci byla použita verze OpenCV 2.4.8.

Možné alternativy

OpenCV obsahuje pravděpodobně největší kolekci optimalizovaných algoritmů pro potřeby počítačového vidění, ale existují i jiné možné alternativy. SimpleCV [37] není alternativa v úplném slova smyslu, protože v sobě zahrnuje OpenCV i další knihovny, ale zjednodušuje a sjednocuje jejich rozhraní.

Možnou alternativou je si používané algoritmy implementovat a nepoužívat hotové knihovny. Takové řešení je nejlepší hlavně případě, kdy je překážkou rozhraní algoritmů, které nedisponují potřebnými funkcemi.

3.2.2 Python

Python je interpretovaný objektový programovací jazyk [38]. Zdrojový kód je nejprve přeložen interpretem do spustitelné podoby a poté spuštěn. Proto pokud je dostupný tento interpret na nějaké platformě, například Windows nebo Linux, je také kód v Pythnu spustitelný na této platformě a není potřeba nijak zasahovat do samotné struktury zdrojového kódu programu. Díky tomu je Python jednoduše přenositelný. Jeho vývoj je také ve formě otevřených zdrojových kódů.

Přenositelnost, jednoduchost a efektivita vývoje byly hlavními faktory při rozhodnutí vybrat Python na implementaci systému. OpenCV ve verzi 2.4.8 podporuje pouze Python verze 2.x a proto pro implementaci byl použit Python verze 2.7.6 místo nejnovější verze 3.4.

Možné alternativy

Vybrat správný programovací jazyk pro implementaci nemusí být zcela jednoduchá záležitost. Protože stejná úloha implementovaná v jednom programovacím jazyce může běžet rychleji než v jiném.

Python je interpretovaný jazyk, který se překládá do spustitelné podoby až při spuštění programu. To může hrát roli při výkonnosti. Na rozdíl od toho jsou programovací jazyky C a C++ předkompilovány a výsledkem této operace je binární spustitelný soubor. Ve srovnání s Pythonem dosahuje C++ v rychlosti kódu vždy lepších výsledků [39, 40]. Proto využití jazyka C nebo C++ je dobrou volbou, hlavně pro systémy kde je důležitá výsledná rychlost.

3.3 Navrhované řešení

Pro realizaci navrhovaného systému byly vybrány tyto algoritmy:

Metoda odčítání pozadí pro segmentaci objektů popředí od pozadí,

Optický tok pro sledování objektů a

HOG a SVM pro klasifikaci pohybujících se objektů.

Pro segmenatci pohybujících objektů od pozadí se dá využit i optický tok, přesněji hustý optický tok 2.4.1, ale v porovnání s odčítáním pozadí, je více náchylný na změny osvětlení a šum scény. Některé tyto nedostatky mohou být odstraněny použitím dalších metod jako hranový detektor, který je odolný na změny osvětlení [41].

Kapitola 4

Realizace

V této kapitole bude popsána implementace a použité funkce knihovny OpenCV. Výsledná implementace má tuto souborovou strukturu:

main.py – Jedná se o hlavní soubor pomocí kterého se celá aplikace spouští. Nastavují se zde potřebné počáteční proměnné.

moan_common.py – Jsou zde nadefinovány pomocné proměnné, funkce a třídy, které se využívají v ostatních částech kódu.

moan_tracker.py – Třída implementující sledování objektů.

moan_detector.py – Obsahuje algoritmus pro detekci a klasifikaci objektů.

moan_heatmap – Obsahuje třídu, která se stará o ukládání a vykreslování teplotní mapy.

4.1 Detekce a klasifikace objektů

Pro nalezení objektů byla použita dříve popsaná metoda odčítání pozadí. OpenCV nabízí přes rozhraní `cv2.BackgroundSubtractorMOG()`. Jedná se o třídu kompletně implementující potřebné algoritmy a je založena na postupu popsaném v [18]. Je tedy využito více Gaussovských křivek pro reprezentaci možných hodnot pixelu. Konkrétně byly použity tři a oproti použití pěti se průměrná doba zpracování jednoho snímku snížila přibližně 1.3×.

Protože vstupní data ani model pozadí nejsou vždy dokonalé, je nejprve potřeba ze vstupních snímků odstranit šum, pro jeho odstranění se používá metoda `cv2.GaussianBlur()`, který vstupní snímek rozostří. OpenCV nabízí mnoho metod na rozostření obrazu, ale šum z reálného prostředí má podobnou distribuci jako Gaussova. Dalšího snížení doby zpracování jednoho snímku se docílilo zmenšením rozměrů vstupního obrazu a to zase přibližně 1.3×.

Obrázek 4.1: Chybná klasifikace objektů může být i když jsou plně viditelné. Údajné postavy jsou ohraničeny.

V každém iteraci systému je načten jeden snímek, z videa nebo přímo z webkamery, odečteno pozadí, prahován a výstupem je binární obraz s vyznačenými oblastmi pohybu. Tento obraz je dále zpracován pomocí funkce `cv2.findContours()`, který vrací seznam všech nalezených objektů ve formě obrysu. Znázorněno na obrazech B.

Klasifikace objektů pomocí HOG deskriptoru `cv2.HOGDescriptor()`, který v OpenCV nabízí již naučený klasifikátor na rozpoznávání osob. Tento klasifikátor je ale naučený na vzorech o rozměrech 64×128 a proto všechny oblasti musejí být nejprve přeškálovány, aby osoba v oblasti měla minimálně tyto rozměry. Tento krok si ale vyžaduje předem známý rozměr osob, které se ve vstupních datech nacházejí. I přes dobrou kalibraci rozměrů, nedosahuje klasifikace dobrých výsledků a může se stát, že jen jeden objekt z celého videa bude klasifikován. Navíc je klasifikace velice časově náročná, a její náročnost se zvyšuje s počtem nalezených objektů v aktuálním snímků. Příklady špatné klasifikace na obrazu 4.1 a v příloze B.

4.2 Sledování objektů

Sledování objektů bylo prvně zamýšleno provádět tak, že by pro každý sledovaný objekt byly nalezeny klíčové body, tedy features, a ty pomocí optického toku sledovat. Navíc OpenCV disponuje hotovým řešením, jak features detektoru (například `cv2.ORB`, `cv2.FAST`, `cv2.SURF`), tak i k hledání optického toku bodů na následující snímek `cv2.calcOpticalFlowPyrLK()`. Optický tok pro množinu bodů funguje spolehlivě, je ale potřeba pomocí dalších heuristik odstraňovat „zbloudilé“ body. Proto byly odstraněny vždy body, které měli větší vzdálenost od svého předka než hodnota mediánu všech vzdáleností.

Obrázek 4.2: Úspěšné sledování objektu.

Po nalezení bodů na novém snímku, se vypočítala možná oblast, kde by se objekt mohl nacházet, a pro zpřesnění polohy byl použit CAM-shift `cv2.CamShift()`. Pro CAM-shift je ale potřeba předzpracovat prohledávanou oblast, což také zpomalovalo běh programu.

Výsledkem byl velice špatný koncept sledování. Proto byla implementován jednoduchý algoritmus, který sleduje objekty jen na základě jejich polohy. V každém novém snímku jsou nalezeny objekty popředí, které jsou porovnávány s objekty nalezenými na předchozím snímku, na základě jejich středů, jak znázorňuje algoritmus 4.1.

4.1 Sledování objektů na základě polohy

1. Nalezni objekty na snímku t a vlož do množiny Q_t.
2. Nalezni středy objektů Q_t.
3. Každý objekt ze množiny Q_{t-1}, jehož střed se nalézá v nějakém objektu z Q_t, je stejný objekt.
4. Pro všechny objekty z Q_{t-1}, které nebyly takto nalezeny, najdi nejbližšího souseda z Q_t, a ty jsou stejný objekt.

Výhodou tohoto přístupu je nízká náročnost na výkon. Dokonce sledování objektu není nijak špatné a algoritmus dokáže sledovat objekt po celou dobu jeho pohybu. Problém nastává u částečného a plného překrytí jinými objekty. Protože u objektů není sledován žádný jiný feature než poloha, není po oddělení možné rozeznat který objekt je který. Pro potřeby tvorby teplotních map je ale tento postup dostačující. Příklad sledování pomocí polohy je znázorněn na obrazu 4.2 a B je ukázka kolize dvou objektů.

4.3 Teplotní mapa

Teplotní mapa je implementována jako dvourozměrné pole, kdy se pozice odpovídající poloze objektu inkrementuje. Inkrementace se provádí buď o konstantu, nebo o hodnotu o kterou se objekt posunul z předešlého snímku. Takto uchovaná informace se musí obarvit podle barevného schéma. Na to je využitá funkce z OpenCV `cv2.applyColorMap()` a barevné schéma `COLORMAP_JET`. Oba způsoby vykreslování jsou znázorněny na obrazech B.

Problém může nastat u dlouhých videozáznamů, neboť datový typ prvků pole je 32bitový float a může dojít k jeho přetečení.

4.4 Zhodnocení a testování bakalářské práce

Požadavky na realizaci byly odlišné od výsledné implementace systému. Bylo požadováno aby systém uměl rozeznat osoby od ostatních objektů, ale kvůli nedokonalému sledování bylo od tohoto cíle upuštěno. Výsledkem praktické části je spíše takový proof-of-concept [4]systém, který představuje některé techniky počítačového vidění.

Problém u sledování optickým tokem, jehož vyřešením by se mohla zlepšit jeho úspěšnost, je výpočet aktuální polohy objektu, která závisí hlavně na sledovaných bodech.

Zlepšení výsledků klasifikace by se mohlo docílit naučením vlastního klasifikátoru na vzorech, které odpovídají objektům na konkrétních vstupních datech. Ale to by znamenalo zlepšení jen pro tyto konkrétní data, nikoli v obecném měřítku.

Testování systému bylo prováděno na videozáznamech nalezených u jiných projektů na počítačové vidění. Na stránkách [42] je k dispozici velká kolekce zdrojů.

4.4.1 Další vývoj

Pro další rozvoj systému by bylo potřeba vyladit existující implementaci a to zejména sledování objektů. Další možností může být parametrizovat některé operace, aby mohl být systém lépe nastaven na konkrétní vstupní data.

Protože s mnoha věcmi se při návrhu systému nepočítalo, obsahuje kód spoustu duplicitních částí, proto by před dalším vývojem bylo potřeba také přepracovat projekt, aby struktura více odpovídala aktuálnímu konceptu.

Je také potřeba vyřešit ukládání teplotních map, aby výsledek mohl být tvořen pro záznamy dlouhé i několik dní. Možným řešením by mohla být normalizace dat, ale je potřeba dbát na validitu uložených dat.

[4]Představení konceptu a existence řešení

Závěr

Cílem této práce bylo navrhnout a implementovat systém pro analýzu pohybu videa a vhodnou formou reprezentovat naměřené údaje. Nejprve byla vypracována rešerše existujících postupů a možných řešení na základě publikovaných článků, podle kterých byl navrhnut ideální stav systém. Následně se prováděli experimenty s navrhovanými algoritmy a jejich zhodnocení. Bohužel nebylo možné všechny plně demonstrovat.

Nakonec se nepodařilo implementovat systém pokrývající v plném rozsahu zadané požadavky. Byly ale alespoň poskytnuty informace o technikách počítačového vidění a přehled existujících postupů v této oblasti.

Pro výrazné zlepšení by bylo potřeba navrhnout jiné postupy než zvolené při implementaci. Počítačové vidění je náročný obor, a proto navrhnutí systému s vysokou úspěšností, je nezbytené se plně seznámit s danou problematikou.

Literatura

[1] Discover Real Immersive Interactive Computing | Kinect for Windows. Dostupné z: http://www.microsoft.com/en-us/kinectforwindows/

[2] Griswold, A.: How Retailers Track Shoppers In Heat Maps - Business Insider. Dostupné z: http://www.businessinsider.com/how-retailers-track-shoppers-in-heat-maps-2014-1

[3] Hama-Girl: Mew. Dostupné z: http://hama-girl.deviantart.com/art/Mew-308535128

[4] Canny, J.: A Computational Approach to Edge Detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, ročník PAMI-8, č. 6, nov 1986: s. 679–698, ISSN 0162-8828, doi:10.1109/tpami.1986.4767851. Dostupné z: http://dx.doi.org/10.1109/tpami.1986.4767851

[5] Sobel, I.; Feldman, G.: A 3x3 Isotropic Gradient Operator for Image Processing, 1968, never published but presented at a talk at the Stanford Artificial Project.

[6] Kass, M.; Witkin, A.; Terzopoulos, D.: Snakes: Active contour models. *INTERNATIONAL JOURNAL OF COMPUTER VISION*, ročník 1, č. 4, 1988: s. 321–331.

[7] Rosten, E.; Drummond, T.: Machine Learning for High-speed Corner Detection. In *Proceedings of the 9th European Conference on Computer Vision - Volume Part I*, ECCV'06, Berlin, Heidelberg: Springer-Verlag, 2006, ISBN 3-540-33832-2, 978-3-540-33832-1, s. 430–443, doi:10.1007/11744023_34. Dostupné z: http://dx.doi.org/10.1007/11744023_34

[8] Rublee, E.; Rabaud, V.; Konolige, K.; aj.: ORB: An efficient alternative to SIFT or SURF. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, Nov 2011, ISSN 1550-5499, s. 2564–2571, doi:10.1109/ICCV.2011.6126544.

[9] Shi, J.; Tomasi, C.: Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, Jun 1994, ISSN 1063-6919, s. 593–600, doi: 10.1109/CVPR.1994.323794.

[10] Kdejsme.cz | Četnost příjmení nebo jména v České republice. Dostupné z: http://www.kdejsme.cz/prijmeni/Dvořák/hustota/

[11] Disney wikia: Olaf. Dostupné z: http://disney.wikia.com/wiki/File:Olaf_transparent.png

[12] A Threshold Selection Method from Gray-Level Histograms. *Systems, Man and Cybernetics, IEEE Transactions on*, ročník 9, č. 1, Jan 1979: s. 62–66, ISSN 0018-9472, doi:10.1109/TSMC.1979.4310076.

[13] MacQueen, J.: Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, Berkeley, Calif.: University of California Press, 1967, s. 281–297. Dostupné z: http://projecteuclid.org/euclid.bsmsp/1200512992

[14] Davies, E.: *Computer and Machine Vision: Theory, Algorithms, Practicalities*. Academic Press, Elsevier, 2012, ISBN 9780123869081. Dostupné z: http://books.google.cz/books?id=AhVjXf2yKtkC

[15] OpenCV dev team: How to Use Background Subtraction Methods - OpenCV 3.0.0-dev documentation. Dostupné z: http://docs.opencv.org/trunk/doc/tutorials/video/background_subtraction/background_subtraction.html

[16] Wren, C.; Azarbayejani, A.; Darrell, T.; aj.: Pfinder: real-time tracking of the human body. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, ročník 19, č. 7, Jul 1997: s. 780–785, ISSN 0162-8828, doi:10.1109/34.598236.

[17] Stauffer, C.; Grimson, W. E. L.: Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, ročník 2, 1999, ISSN 1063-6919, s. –252 Vol. 2, doi:10.1109/CVPR.1999.784637.

[18] Kaewtrakulpong, P.; Bowden, R.: An improved adaptive background mixture model for real-time tracking with shadow detection. In *Proceedings of 2nd European Workshop on Advanced Video Based Surveillance Systems*, ročník 5308, 2001.

[19] Achkar, F.: Hysteresis-based selective Gaussian-Mixture model for real-time background update and object detection, 2006. Dostupné z: http://spectrum.library.concordia.ca/9285/

[20] Bouwmans, T.; El Baf, F.; Vachon, B.: Background Modeling using Mixture of Gaussians for Foreground Detection - A Survey. *Recent Patents on Computer Science*, ročník 1, č. 3, Nov 2008: s. 219–237. Dostupné z: http://hal.archives-ouvertes.fr/hal-00338206/en/

[21] Toyama, K.; Krumm, J.; Brumitt, B.; aj.: Wallflower: principles and practice of background maintenance. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, ročník 1, 1999, s. 255–261 vol.1, doi:10.1109/ICCV.1999.791228.

[22] Belongie, S.; Malik, J.; Puzicha, J.: Shape matching and object recognition using shape contexts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, ročník 24, č. 4, Apr 2002: s. 509–522, ISSN 0162-8828, doi:10.1109/34.993558.

[23] Yilmaz, A.; Javed, O.; Shah, M.: Object Tracking: A Survey. *ACM Comput. Surv.*, ročník 38, č. 4, Dec 2006, ISSN 0360-0300, doi: 10.1145/1177352.1177355. Dostupné z: http://doi.acm.org/10.1145/1177352.1177355

[24] Horn, B. K. P.; Schunck, B. G.: Determining Optical Flow. *ARTIFICAL INTELLIGENCE*, ročník 17, 1981: s. 185–203.

[25] Farnebäck, G.: Two-frame Motion Estimation Based on Polynomial Expansion. In *Proceedings of the 13th Scandinavian Conference on Image Analysis*, SCIA'03, Berlin, Heidelberg: Springer-Verlag, 2003, ISBN 3-540-40601-8, s. 363–370. Dostupné z: http://dl.acm.org/citation.cfm?id=1763974.1764031

[26] Lucas, B. D.; Kanade, T.: An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1981, s. 674–679. Dostupné z: http://dl.acm.org/citation.cfm?id=1623264.1623280

[27] Optical Flow – OpenCV-Python Tutorials 1 documentation. Dostupné z: opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_video/py_lucas_kanade/py_lucas_kanade.html

[28] Dalal, N.; Triggs, B.: Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, ročník 1, June 2005, ISSN 1063-6919, s. 886–893 vol. 1, doi:10.1109/CVPR.2005.177.

[29] Object detection. Dostupné z: http://www.cs.cornell.edu/courses/CS4670/2012fa/projects/p5/index.html

[30] Garate, C.; Bilinski, P.; Bremond, F.: Crowd Event Recognition using HOG Tracker. In *Twelfth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (PETS-Winter)*, Snowbird, UT, United States: IEEE, Prosinec 2009, s. 1–6, doi:10.1109/PETS-WINTER.2009.5399727. Dostupné z: http://hal.inria.fr/inria-00515197

[31] Benfold, B.; Reid, I.: Stable Multi-Target Tracking in Real-Time Surveillance Video. In *CVPR*, June 2011, s. 3457–3464.

[32] Kalal, Z.; Mikolajczyk, K.; Matas, J.: Tracking-Learning-Detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, ročník 34, č. 7, jul 2012: s. 1409–1422, ISSN 0162-8828, doi:10.1109/TPAMI.2011.239. Dostupné z: http://dx.doi.org/10.1109/TPAMI.2011.239

[33] Milan, A.; Roth, S.; Schindler, K.: Continuous Energy Minimization for Multitarget Tracking. *IEEE TPAMI*, ročník 36, č. 1, 2014: s. 58–72, ISSN 0162-8828, doi:10.1109/TPAMI.2013.103.

[34] Click2Stream | If cameras are eyes. We give them brains. Dostupné z: http://www.click2stream.com

[35] ATAP Project Tango – Google. Dostupné z: https://www.google.com/atap/projecttango/

[36] OpenCV. Dostupné z: http://opencv.org/

[37] SimpleCV. Dostupné z: http://www.simplecv.org/

[38] Python Software Foundation: Welcome to Python.org. Dostupné z: www.python.org

[39] onlyjob: Perl, Python, Ruby, PHP, C, C++, Lua, tcl, javascript and Java benchmark/comparison. Dostupné z: http://onlyjob.blogspot.cz/2011/03/perl5-python-ruby-php-c-c-lua-tcl.html

[40] mandelbrot benchmark | Computer Language Benchmarks Game. Dostupné z: http://benchmarksgame.alioth.debian.org/u32/performance.php?test=mandelbrot&sort=fullcpu

[41] SuganyaDevi, K.; Malmurugan, N.; Sivakumar, R.: EFFICIENT FOREGROUND EXTRACTION BASED ON OPTICAL FLOW AND SMED FOR ROAD TRAFFIC ANALYSIS. In *International Journal of Cyber-Security and Digital Forensics (IJCSDF)*, Nov 2012, s. 177–182.

[42] CV Datasets on the web. Dostupné z: http://www.cvpapers.com/datasets.html

Příloha **A**

Seznam použitých zkratek

2D Dvojrozměrný prostor

3D Trojrozměrný prostor

RGB Red-Green-Blue (Červená-Zelená-Modrá). Výsledná barva je složena z intenzit těchto tří barev.

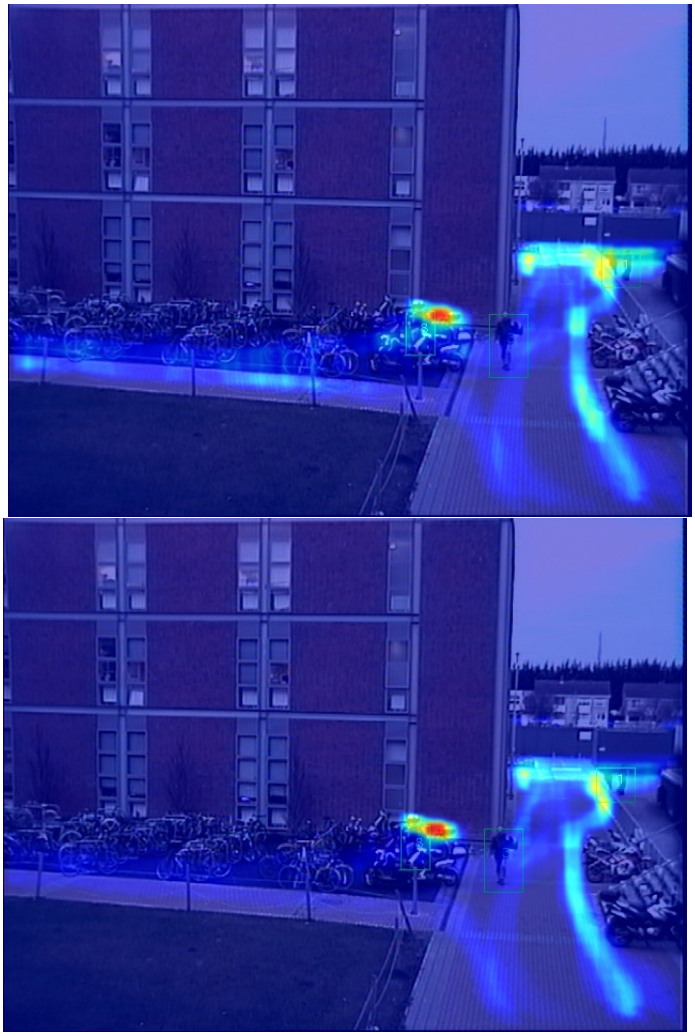
fps Frames per second

ms Milisekunda. 1 sekunda = 1000 milisekund

Příloha B

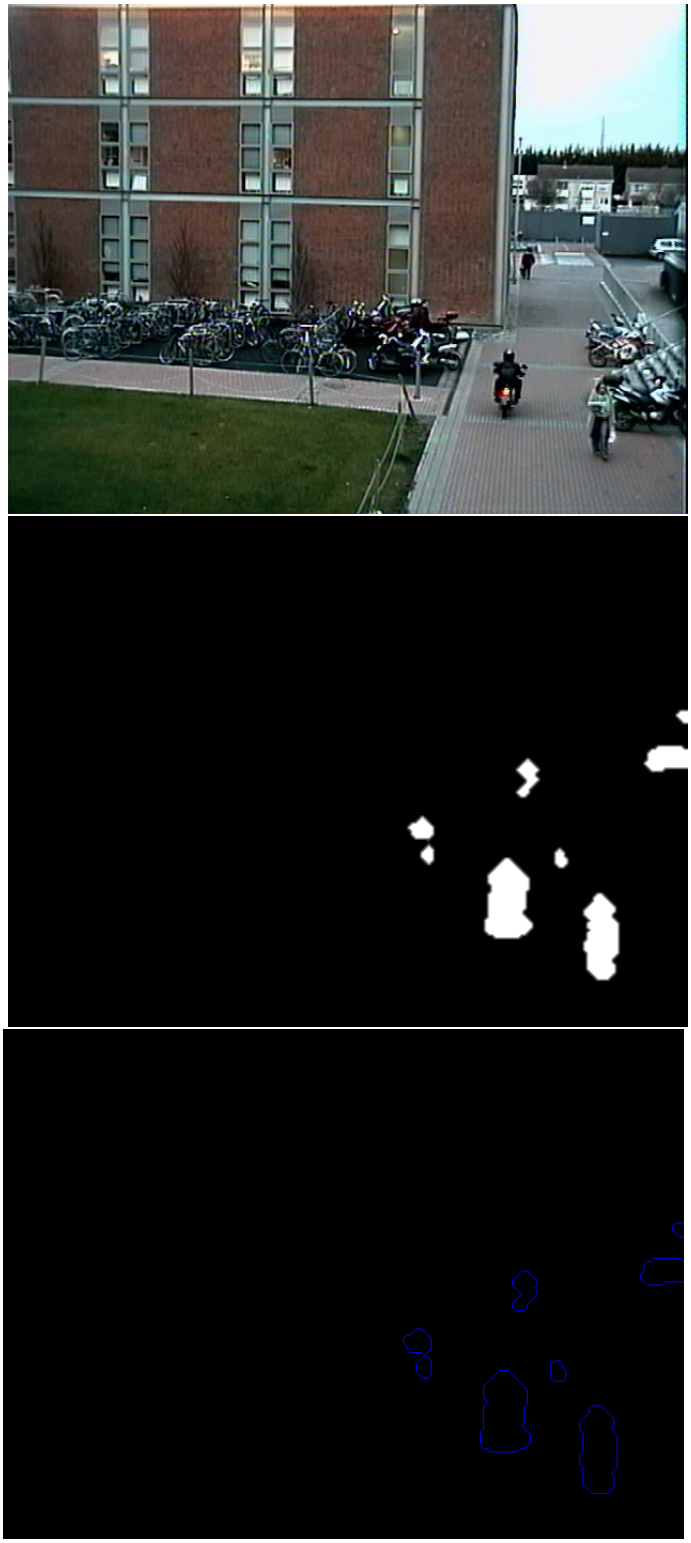
Snímky z aplikace

Obrázek B.1: Jedna postava na levém obrazu nebyla detekována, ale na pravém již ano.



Obrázek B.2: První obraz je teplotní mapa podle pohybu objektu a druhý přičítáním konstanty.

Obrázek B.3: Po kolizi dvou objektů algoritmus nesprávně přiřadí původní ID.



Obrázek B.4: vstupní obraz, popředí scény, nalezené obrysy

Příloha C

Uživatelská příručka

Seznam parametrů:

−v | −−verbose – Pokud je tento parametr použit, zobrazí se doba za jakou je zpracován jeden snímek a aktuální fps.

−h | −−help – Zobrazí nápovědů.

−s | −−source – Slouží pro zadání vstupního videozáznamu.

−t | −−type – Typ tvorby teplotní mapy. Dvě možné hodnoty 0 a 1.

−e | −−experimental – Zapne experimentální funkce klasifikace a sledování objektů pomocí optického toku.

−o | −−output – Soubor do kterého se má uložit výsledná teplotní mapa.

−−width – Šířka vstupního videa. Slouží pro kalibraci při použití kamery.

−−height – Výška vstupního videa. Slouží pro kalibraci při použití kamery.

Příloha D

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `samples` ..... testovací videozáznamy
- `src`
  - `impl` ..... zdrojové kódy implementace
    - `main.py`
    - `moan_common.py`
    - `moan_detector.py`
    - `moan_heatmap.py`
    - `moan_tracker.py`
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
    - `images`
    - `BP_Cvacho_Ondrej_2014.tex`
    - `csn690.bst`
    - `cvut-logo-bw.pdf`
    - `FITthesis.cls`
    - `mybibliographyfile.bib`
- `text` ..... text práce
  - `BP_Cvacho_Ondrej_2014.pdf` ..... text práce ve formátu PDF