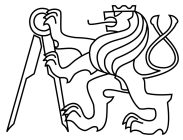
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Detekce a sledování pohybu osob na základě záznamu z kamerového systému
Student:	Bc Michal Řapek
Vedoucí:	doc. RNDr. Ing. Marcel Jiřina, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce zimního semestru 2017/18

Pokyny pro vypracování

1) Seznamte se s úlohou detekce a sledování osob v kamerovém záznamu z více kamer, které jsou umístěny nad prostorem, kde se osoby pohybují.
2) Proveďte rešerši metod, které se zabývají touto úlohou, a dále nastudujte metody zpracování obrazu, které jsou využitelné pro danou úlohu.
3) Navrhněte postup a dílčí metody pro detekci a sledování osob v kamerových záznamech. Zohledněte specifika snímacího procesu.
4) Navržené metody implementujte v programovacím jazyku Java s využitím volně dostupných knihoven.
5) Implementované metody ověřte na naměřených reálných datech, vyhodnoťte dosaženou přesnost a navrhněte možná zlepšení.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 22. února 2016

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra ... SOFTWAROVÉHO INŽENÝRSTVÍ

Diplomová práce

Detekce a sledování pohybu osob na základě záznamu z kamerového systému

Bc. Michal Řapek

Vedoucí práce: doc. RNDr. Ing. Marcel Jiřina, Ph.D.

16. února 2017

Poděkování

Rád bych poděkoval Marcelu Jiřinovi za dohled nad prací a cenné rady. Také jsem velmi vděčný mé rodině za poskytnutou podporu.

Prohlášení

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

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

V Praze dne 16. února 2017 .....

Odkaz na tuto práci

Řapek, Michal. *Detekce a sledování pohybu osob na základě záznamu z kamerového systému.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Úkolem této práce je implementace knihovny funkcí pro detekci a sledování osob v supermarketu. Za tímto účelem práce obsahuje souhrn nejpoužívanějších metod v podobných případech, tedy pro detekci osob v hloubkové mapě při pohledu shora. V druhé části obsahuje práce popis implementace a výsledky testování.

Klíčová slova Supermarket, HOG, SVM, Detekce osob, Sledování osob, Kinect, RGB-D, Hloubková mapa

Abstract

The task of this work is to implement a library of functions for people detection and tracking. For this purpose the thesis contains a summary of the most commonly used methods in similar cases, ie for detection of persons in depth map when viewed from above. The second part contains a description of implementation and results of testing.

Keywords Supermarket, HOG, SVM, Person detection, Person tracking, Kinect, RGB-D, Depth map

Obsah

Seznam obrázků

Seznam tabulek

Úvod

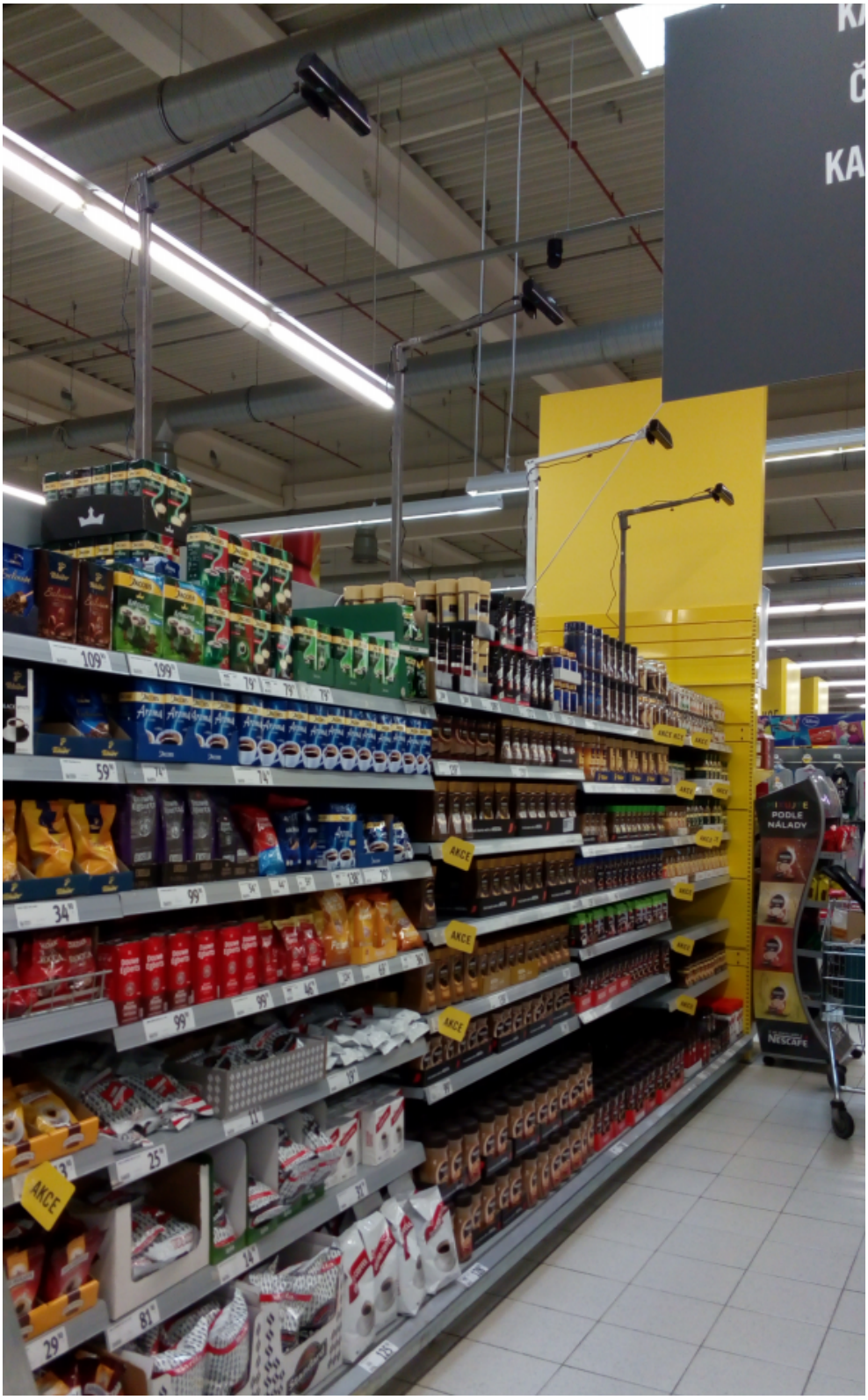
Mým úkolem v této práci je implementovat knihovnu funkcí pro detekci a sledování osob v sekvenci snímku. Tato knihovna by měla poté nahradit aktuální způsob detekce a sledování osob v systému, který je použit v supermarketu pro sledování a zaznamenávání pohybu zákazníků, za účelem sběru dat jejich průchodů a vytvoření grafického modelu chování návštěvníka.
Samotná detekce osob bude probíhat nad snímky hloubkové mapy pořízené kamerou, která snímá danou oblast pohledem dolů a mírně šikmo jak je vidět na obrázku 0.1. Tato kamera pořizuje jak snímky hloubkové mapy, tak i snímky barevné. Ty můžeme vidět na obrázcích 1.2 a 1.3.

Výstup požadovaný pro jednotlivé snímky je ve formě pozice nalezené osoby ve snímku a unikátního identifikátoru dané osoby. Tento identifikátor pak slouží ke sledování osob mezi jednotlivými snímky. Výsledné průchody jsou pak použity k vytvoření modelu chování.

Nedostatek aktuálního řešení lze spatřit v tom, že dochází k situacím, kdy jedné sledované osobě bylo přiděleno několik různých identifikátorů. Hlavním cílem mnou implementované metody by mělo být vyřešení tohoto problému a ke ztrátě a re-identifikaci označené osoby by tedy mělo docházet pouze v krajních případech.

Písemnou část práce jsem rozdělil na část teoretickou, která je v kapitole 1 a část praktickou, která je v kapitole 2 a 3. V teoretické části jsem se věnoval analýze daného problému, aktuálního řešení a způsobů použitých při řešení podobných problémů. Z této analýzy jsem poté vycházel při návrhu vlastního řešení. Tento návrh je také součástí teoretické části 1.4.

V praktické části jsem popsal průběh detekce a sledování osob spolu s implementací jednotlivých metod. Tato kapitola dále popisuje průběh implementace, který obsahuje změny v implementaci oproti původnímu návrhu tak, aby výsledné řešení vyhovělo v co možná největší míře daným požadavkům.



Obrázek 0.1: Ukázka umístění kamer nad regály

Tyto změny byly provedeny na základě testování během implementace, které je popsané v další kapitole. V závěru je popsáno provedené testování implementovaného řešení a vyhodnocení dosažených výsledků.

Pro srovnání s aktuálním řešením jsem otestoval aktuální řešení stejným způsobem na stejných datech.

Kapitola 1

Analýza a návrh

V této části nejdříve podrobněji popíšu daný problém, nastavení systému, pro který hodlám dané řešení implementovat a poté aktuálně implementované řešení. V dalších částech se pak budu zabývat aktuálně nejpoužívanějšími metodami, nad zadaným problémem a z nich odvozeného návrhu vlastního řešení.

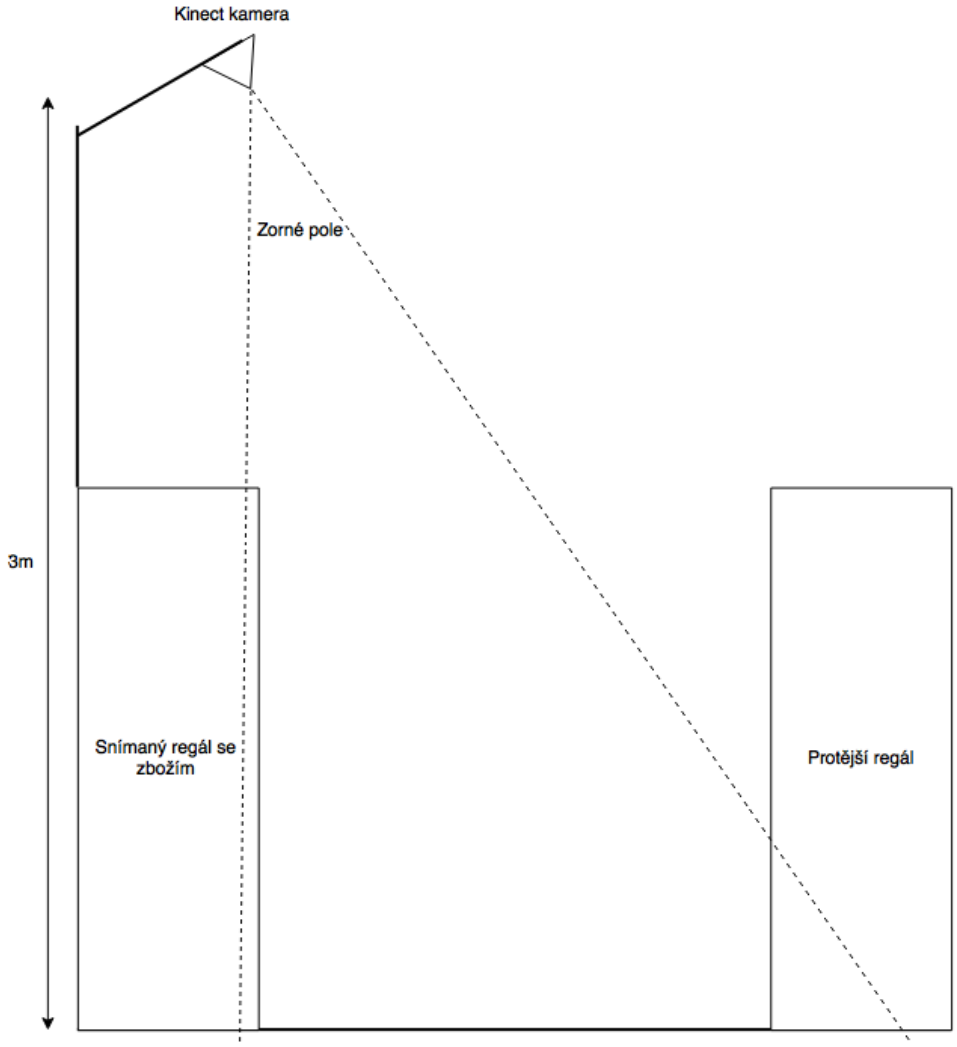
1.1 Nastavení systémů

Úspěšnost i způsob detekce závisí na několika faktorech. Jde o vzdálenost, na jakou se detekce provádí, prostředí ve kterém se detekce provádí, úhel pohledu kamery, nad jejímiž snímky se detekce provádí, ale i typu kamery a tedy výsledného snímku, který máme k detekci k dispozici. V této části bych chtěl popsat právě to, v jakém prostředí detekce bude probíhat. Jaký formát snímku budu mít k dispozici a některé další faktory, které jsem jmenoval.

V mém případě je kamera umístěna nad regálem v supermarketu tak, že snímá i regál pod sebou. Přesněji kamera je umístěna ve výšce přibližně tří metrů a svým zorným polem pokrývá oblast mírně přesahující jeden regál. Toto umístění je zachyceno na obrázku 1.1

Důvodem pro snímání regálu pod kamerou je využívání těchto dat při analýze přístupu ke zboží. Detekci zda zákazník sáhl pro nějaké zboží do regálu, provádí jiná část systému.

Každá kamera je připojena k jednomu počítači, během snímání a vyhodnocování nekomunikuje s ostatními. Kameru spolu s jedním počítačem budu dále referovat jako modul. Výsledná kompozice těchto modulů poté pokrývá celou uličku v supermarketu, jak můžeme vidět na obrázku 0.1. Jak bylo řečeno, moduly mezi sebou nekomunikují a výsledné propojení nasnímaných dat probíhá až nad tabulkou průchodů, která je výstupem z každého modulu. Mým úkolem je část softwaru pro program běžící na jednom modulu, který budu detekovat a sledovat osoby na snímcích pro daný modul.



Obrázek 1.1: Schéma umístění kamery

A bude poskytovat data o průchodech jednotlivých osob pro vytvoření zmíněné tabulky.

Další důležitou roli pro úspěšnost detekce hraje typ pořízeného snímku. V mém případě je snímek pořizován kamerou Kinect, která je typu RGB-D, neboli kromě klasického barevného obrazu ještě poskytuje hloubkovou mapu. Zaměřím se právě na hloubkovou mapu, jelikož bylo požadováno zadáním provádět detekci nad tímto typem snímku.
Hloubková mapa je obraz nebo kanál obrazu, který obsahuje informaci o vzdálenosti objektu od kamery. Tato hloubková mapa ještě před tím než jí dostane k dispozici vyvíjená část pro detekci upravena tak, aby hodnota pixelu neurčovala vzdálenost objektu ke kameře, ale výšku, ve které se nachází od podlahy. Jednotlivé pixely pak obsahují výšku objektu v daném místě, tento přepočet je udělán na základě znalosti výšky, ve které je kamera umístěna.
Další úpravou, která je provedená se snímkem je, že spodní část této hloubkové mapy, která zachycuje regál pod kamerou je odříznuta a ukládá se samostatně, jelikož jak bylo řečeno výše je zpracovávána jinou částí programu. Pořízený RGB snímek můžeme vidět na obrázku 1.2 a k němu odpovídající hloubkovou mapu již po úpravě na obrázku 1.3.

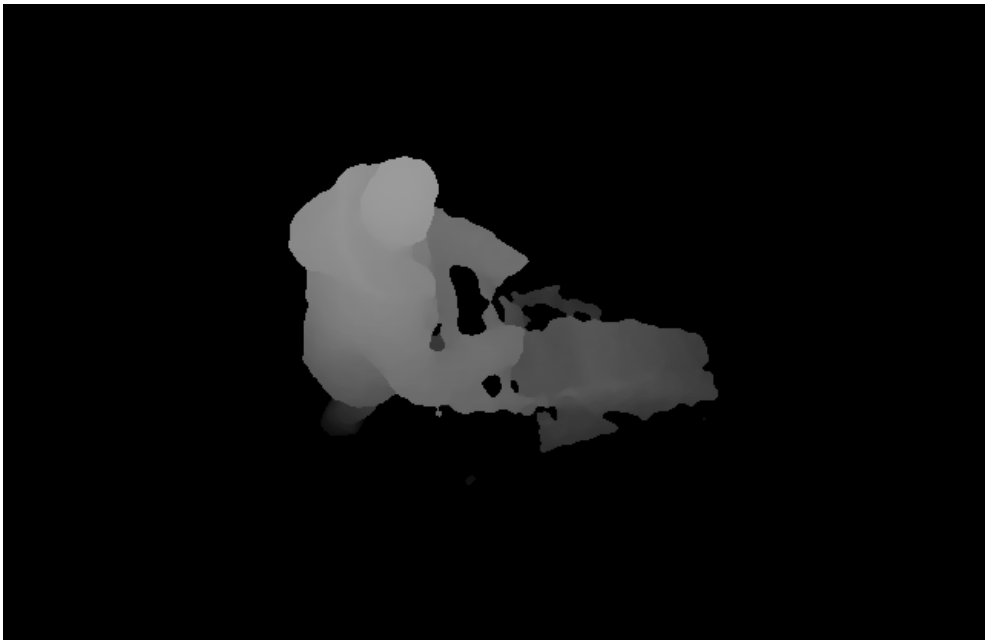
Kamera snímá 30 snímků za sekundu a ukládá je ve formě png obrázků, nad touto sekvencí budu provádět sledování osob. Ze sekvence jsou odstraněny snímky, které neobsahují žádné pohybující se předměty či osoby.
Pro procházení snímků ve správném pořadí slouží jméno snímku. To je ve formě časové značky pořízení snímku MMDD_HHMMSS_MSS. Například snímek pořízený sedmnáctý den desátého měsíce v osm hodin, pět minut, čtyřicet sekund a dvě stě třicet milisekund bude mít jméno 1017_080540_230. Toto jméno zároveň slouží ke spárování barevného snímku se snímkem hloubkové mapy.

Dalším faktorem pro úspěšnost detekce je kvalita snímků. Na snímcích 1.4 a 1.5 můžeme vidět, že hloubková mapa ne vždy obsahuje veškerá očekávaná data. Poškození hloubkové mapy ve spodní části snímku není až tak časté a vážné, tím pádem nemá takový vliv na detekci. Na druhou stranu k častému poškození dochází při horní části snímku. Když se osoba pohybuje u protějšího regálu, často se stává, že je výsledný snímek zdeformovaný, jak je vidět na snímku 1.4. Anebo na hloubkové mapě úplně chybí.

1.2 Analýza

V této části jsem se zaměřil na analýzu řešení problémů, které jsou podobné tomu mému. Tedy takové, kde jde o detekci kamerou směřující ze shora dolů a to jak na barevných snímcích, hloubkové mapě či kombinováním obou.

Obrázek 1.2: RGB snímek pořízený umístěným senzorem



Obrázek 1.3: Hloubková mapa k snímku 1.2

Obrázek 1.4: Poškozená hloubková mapa k snímku 1.5

Obrázek 1.5: RGB snímek, pro který byla pořízena poškozená hloubková mapa

V článku [1] se zabývají detekcí za účelem počítání osob. Pro detekování pohybujících se objektů používají metodu odečítání pozadí, tu aplikují na hloubkovou mapu.
Tato metoda se běžně využívá v bezpečnostních systémech k detekování objektů. Myšlenka této metody je popsána v článku [2] následovně. Aktuální snímek se odečte od výchozího snímku pozadí, tento výchozí snímek se získá ze statického pozadí v průběhu časového úseku. Výsledkem odečtení těchto dvou snímků je poté snímek, který obsahuje nové anebo pohybující se objekty.
Tato metoda se dá využít i na barevných snímcích, jak je popsáno v článku [3]. Dle článku [2] je vyžadován robustní model pozadí, od kterého se bude odčítat, aby byla metoda spolehlivá. A i přesto je tato metoda citlivá na náhlou změnu osvětlení a pohybující se pozadí. Proto je lepší použití hloubkové mapy, jelikož na rozdíl od barevného snímku, na hloubkovou mapu nemá pohybující se pozadí a změna osvětlení žádný nebo minimální vliv.

Poté co pomocí této metody získají snímek pouze s objekty zájmu. Jsou tyto shluky analyzovány na počet osob pomocí U-disparity. Poté pomocí hledání lokálních maxim jsou označeny hlavy těchto osob. Potvrzení, že jde opravdu o osobu je na základě velikosti shluku, který se poté rozdělí na příslušné osoby. Takto detekovaným osobám je poté v dalším kroku přidělena odpovídajíc barva z barevného snímku na základě získaných pozic. Sledování těchto osob na sekvenci snímků je pak prováděno na základě porovnání pozice shluků mezi snímky a získané barvy. Pro urychlení výběru nejbližších shluků využívají Kalmanův filtr k predikci pohybu a umístění shluku v dalším snímku. Autoři udávají přesnost počítání osob v reálném čase 93%.

Pro moji práci bylo specifikováno využití pouze hloubkové mapy, nebudu tedy používat barvu jako dodatečnou informaci pro sledování osob mezi snímky. Nebudu využívat ani metodu odčítání pozadí. Jak je vidět na obrázku 1.3, snímky hloubkové mapy, nad kterými bude vyvíjená část provádět detekci, byly již upraveny. Ze snímku je vidět, že pozadí již bylo odstraněno. Je pravděpodobné, že toho bylo docíleno právě metodou odečítání pozadí, během před-zpracování snímku, kdy se hodnoty pixelů změní ze vzdálenosti od kamery na vzdálenost od země.

Jelikož se metoda odečítání pozadí běžně používá při řešení těchto problémů, můžeme se s ní setkat i v článku [3]. Zde je také použita na hloubkovou mapu. Důvody proč využít hloubkovou mapu na místo barevného snímku již byly zmíněny výše. Sledování je v tomto článku pak prováděno tak, že shluky které zůstaly po odstranění pozadí, jsou srovnány se shluky na dalším snímku. Spojení osob mezi snímky pak probíhá na základě tohoto srovnání, kde se pospojují shluky s největším přesahem. V tomto případě je použita pouze hloubková mapa, takže se nevyužívá žádná dodatečná informace ke spojení dvou shluků, jako tomu bylo v případě barvy v předchozím článku.

Metoda odečítání pozadí je použita i v článku [4], kde přicházejí s metodou pro více kamerový systém, ten spočívá ve spojování obrazů z různých kamer v překryvu. Před samotným spojováním a zpracováním snímku, používají předzpracování snímku, aby eliminovali paralax efekt, který zkresluje objekty u kraje snímku. Za tímto účelem mapují snímek hloubkové mapy ze souřadnic kamery do souřadnic světa. Je uvedeno, že tato rekonstrukce je výpočetně náročná pro svoji komplexnost a proto není možné systém používat v reálném čase.

Na výsledný snímek poté použijí metodu odečítání pozadí a tím docílí, že snímek bude obsahovat nové nebo pohybující se objekty. Na zkoumaném snímku máme po použití metody odečtení pozadí pouze pohybující se a nové objekty. Pomocí grafové segmentace jsou v tomto obrázku označeni kandidáti. Tito kandidáti jsou poté testováni proti hemielipsoid modelu hlavy. Hemielipsoid model slouží k popisu 3D vzhledu hlavy, ten se používá k vyhodnocení podobností mezi kandidátní oblastí a ideální hlavou. Hlava je při pohledu shora nejlepším vodítkem pro detekci osoby.

Dle článku [5] je grafová segmentace metoda, která k jednotlivým vrcholům přistupuje jako k vrcholům grafu. Mezi sousedními vrcholy jsou vytvořeny hrany a tyto propojené vrcholy se poté porovnávají. Na základě velikosti podobností se tyto vrcholy dělí do segmentů.

Vzhledem k podobnosti některých předmětů s tvarem hlavy, rozšířili detekci ještě o metodu, která má za úkol potvrdit, že jde opravdu o člověka. Za tímto účelem vytvoří snímek nejkratších vzdáleností od hlavy dané osoby k ostatním pixelům takzvanou "Geodesic Distance Map". Z tohoto snímku ještě odstraní ruce a ponechají pouze ramena.

Pro klasifikaci takto vzniklého snímku využívají lineární SVM (Support vector machine). Jak se můžeme dočíst v článku [6] "Introduction to Support Vector Machines", jde o metodu strojového učení, která se používá k řešení binární klasifikace. SVM model reprezentuje příklady jako body v prostoru, které jsou jasně odděleny do dvou skupin nadrovinou.

Klasifikace poté probíhá mapováním nového případu. Do tohoto prostoru na základě toho, na kterou stranu nadroviny je nový případ mapován, je predikována jeho kategorie. V tomto případě zda jde o osobu či ne. Jako vstup používají HOG (Histogram Orientovaných Gradientů) spolu s CoG (Comparison of Granules). Samotné sledování osoby pak probíhá pomocí opětovné detekce v jednotlivých snímcích.

Původně jsem myslel, že bych použil postup z tohoto článku, ale při dalších konzultacích jsem dostal podrobnější informace o tom, jak daný systém vypadá. Jak bylo řečeno v části 1.1, systém je rozdělen do nespolupracujících modulů, což vyloučilo použití spojování obrazů.

Na druhou stranu mě to nasměrovalo k použití SVM pro klasifikaci zda jde o člověka.

SVM je spolu s HOG často používáno v počítačovém vidění. Ne pouze jako nástroj pro dodatečné potvrzení, že daný kandidát je opravdu osoba, jako tomu je v článku [4], ale i k samotné detekci jako je tomu v článku [7].

Kde se osoba detekuje právě pomocí SVM na základě HOG vlastností pro hlavu a ramena. Detekce v tomto případě probíhá nad hloubkovou mapou, ta je nejdříve před-zpracována, ale jde pouze o odstranění šumu, aby se urychlil a zlepšil vlastní algoritmus. Použití HOG deskriptoru pro detekci pomocí SVM lze použít i nad barevným snímkem, jak je vidět v článku [8]. I když v tomto případě jde o detekci osob při pohledu z boku.

SVM se používá i v článku [9]. Kde se nejprve naleznou kandidáti, kde by se mohla vyskytovat hlava. Prvním krokem k nalezení těchto kandidátů je hledání lokálního maxima. Jak je, ale uvedeno nalezená maxima se nemusí shodovat se skutečným středem hlavy. Z tohoto důvodu se na tyto maxima aplikuje gradientní algoritmus. Výsledkem je seznam konečných kandidátů na hlavu, kde se ještě eliminují duplicity. Rozhodnutí zda kandidát je hlava, je učiněno SVM klasifikátorem, pro něj je v tomto článku vstupem HDD (Histogram of Depth Difference). Sledování osoby dále probíhá pomocí detekce v jednotlivých snímcích a následném spojování těchto osob s osobami ze snímku minulého na základě vzdálenosti mezi těmito osobami.

Prošel jsem ještě několik dalších článků, které se zabývaly stejným problémem, tedy detekcí osob při pohledu shora. Ty také používaly SVM klasifikátor s HOG či HDD. Jelikož jsem se zaměřil na analýzu způsobů detekce, kdy kamera snímá scénu ze shora dolů. Nenašel jsem jinou metodu, která by byla úspěšnější. Důvodem proto může být fakt, že použití SVM klasifikátoru je momentálně metoda označovaná termínem "Stat of art"což označuje metodu "na nejvyšší úrovni všeobecného rozvoje, dosažené v dané oblasti v určitém čase".

Shrnutí

Nejpoužívanější a momentálně nejúspěšnější metodou pro detekci osob při pohledu shora je použití SVM klasifikátoru k rozhodnutí zda jde o osobu. Jako hlavním identifikátorem pro klasifikaci osoby, na snímcích při pohledu shora, je hlava dané osoby jakožto nejlépe viditelná část. Pro sledování detekovaných osob mezi jednotlivými snímky, se nejčastěji používá predikce polohy a následné spojení nejbližších osob či překryv shluku symbolizující jednu osobu. Kde shluky s největším překryvem mezi jednotlivými snímky jsou označeny za tutéž osobu.

1.3 Aktuální řešení

Výsledek mé práce, pokud dosáhne lepších výsledků, by měl nahradit aktuální řešení pro detekování a sledování osob v nastavených podmínkách.

Aktuální řešení přistupuje k této úloze následovně. Využívá pouze snímků hloubkové mapy, stejně jako je předepsáno pro moje řešení. Detekce osoby na snímku pak probíhá následovně.

Prvním krokem je nalezení objektů na snímku. Zde se ke snímku přistupuje jako k matici výšek a hledání objektů probíhá tak, že se prochází řádek za řádkem a nenulové sousedící pixely se spojují do objektů. Tyto objekty si poté drží pozici všech svých pixelů. Po tomto kroku tedy máme detekovány objekty v podobě shluků pixelů.

Dalším krokem je určit zda daný shluk je osoba nebo zda daný shluk není tvořen více osobami. Tato kontrola probíhá tak, že se prochází shluk tvořící tento objekt a hledají se v něm struktury podobné hlavě. Pokud je nalezeno více hlav, je objekt rozdělen v polovině vzdálenosti mezi středy hlav.

Identifikace hlavy probíhá následovně. Vezme se okolí nejvyššího bodu v objektu, přičemž rozdíl okolních výšek pixelů musí být menší než 20. Těmto okolním pixelům se vytvoří kontura a spočítá se kulatost této kontury. Kromě kontury se v okolí opíše kruh, přičemž poloměr tohoto kruhu nesmí být větší než určitý práh.

Poté co proběhne identifikace a rozdělení objektů, provede se přiřazení těchto nových objektů k těm z minulého snímku na základě prostorového překryvu. Toto přiřazení provádí tři důležité kroky. Rozdělení objektu, pokud se k němu přiřadilo více objektů z minulého snímku. Vytvoření nového zákazníka jestli6e se nový objekt nepřiřadil k žádnému existujícímu z minulého snímku. Tento zákazník vznikne pouze, pokud je na tomto objektu nalezena hlava. A nakonec smazání zákazníků z minulého snímku, ke kterým nepřiřadil žádný z nových objektů.

Takto funguje aktuální řešení. Jak jsem již zmiňoval, toto řešení trpí na výpadky sledování. To znamená, že nesleduje jednu osobu pod tím stejným identifikátorem po celou dobu jejího pobytu ve scéně.

1.4 Návrh

V této části popíši svůj návrh řešení, který je založen převážně na článcích [7], [9] a proč jsem se pro daný postup rozhodl.

Dále pak v jednotlivých podkapitolách popíši, jak fungují zvolené algoritmy a metody.

Na základě úspěšnosti detekce publikované v článcích [7], [9] a také proto, že použití hloubkové mapy bylo upřednostněno zadáním, jsem se rozhodl, že své řešení založím pravě na těchto dvou článcích. Jak jsem zmínil výše na obrázku 1.3 je vidět, že není potřeba na hloubkovou mapu již aplikovat metody pro odstranění šumu či metodu odečítání pozadí pro odstranění pozadí.

Pro průběh detekce jsem se nechal inspirovat článkem [9] to znamená, že v prvním kroku naleznu kandidáty pro detekci hlavy. Za tímto účelem jsem se rozhodl postupovat stejně a použít metodu pro hledání lokálních maxim. Vzhledem k tomu, že snímek, na kterém bude probíhat detekce, je již upraven a obsahuje jen předměty ve scéně jako jsou osoby, nákupní košíky a další předměty, které se mohou pohybovat uličkou. Přišlo mi efektivní hledat lokální maxima a tak pro tyto objekty najít nejvyšší body.

Pro redukci počtu maxim v případě minimální šance, že jde o hlavu, půjde nastavit spodní mez na ukládané maximum. Během implementačního testování jsem pro tuto mez používal nastavení mezi sto dvaceti až sto čtyřiceti centimetry, abych tak redukoval maxima právě na nákupních košících a dalších. Tato spodní mez je také z důvodu, že v konečném použití nás nebudou zajímat malé děti.

Tato mez však přináší možnost špatné detekce v případě, že se daná osoba bude sklánět pod danou mez. Abych předešel této chybě, bude v průběhu algoritmu použita metoda, která bude brát v potaz historii detekovaných hlav a na jejím základě predikovat pravděpodobné pozice hlav v daném snímku. Tyto pozice budou přidány k seznamu maxim před dalším krokem detekce. To by mělo zaručit, že se provede klasifikace pomocí SVM klasifikátoru i na těchto pozicích a měla by se minimalizovat šance, že z důvodu této úpravě nebudeme detekovat některou z osob v této krajní situaci.

Jak zmiňuje článek [9] lokální maxima se s největší pravděpodobností liší od skutečného středu hlavy, který potřebujeme pro další kroky. Z tohoto důvodu se pokusíme maximum přesunout co nejblíže středu pomocí Mean-Shift algoritmu, který je také zmíněn za tímto účelem v tomto článku.

Po těchto krocích budeme mít pole kandidátních hlav. S největší pravděpodobností, se některá původní lokální maxima přesunou pomocí Mean-Shift algoritmu do stejného místa nebo jeho okolí. Proto ještě před samotnou detekcí eliminujeme tyto násobné kandidáty a ponecháme pouze jednoho.

V klasifikační části detekce jsem se inspiroval článkem [7] a rozhodnutí zda jde o hlavu nebo nebudu provádět pomocí SVM klasifikátoru nad HOG. Pro SVM jsem se rozhodl z několika důvodů.

Za prvé v tomto článku prezentovali 97% úspěšnost. A jak jsem se zmínil výše, jde momentálně o nejpoužívanější metodu klasifikace v počítačovém vidění.

Další variantou by bylo testování proti hemielipsoid modelu hlavy jako v článku [4], ale varianta této detekce je použita v aktuálním řešení. A dalším důvodem proč jsem tuto metodu zavrhl, je fakt, že v pozici z které kamera snímá danou scénu, nemáme hlavu přímo z vrchu, ale mírně z boku, také velikost se v každé části snímku mírně liší. S čímž předpokládám, že si spíše poradí SVM klasifikátor.

Další důležitou součástí řešení je sledování detekovaných osob přes sekvenci snímků. Sledování budu provádět pomocí detekce v jednotlivých snímcích. Neboli v každém snímku detekuji dané osoby, a těm poté přiřadím číselný identifikátor na základě vzdálenosti jednotlivých osob od osob na snímcích předchozích. Při spojování těchto osob bude využita predikce pozice z minulého snímku tak, že osoba v daném snímku dostane přidělený identifikátor nejblíže predikované osoby ze snímku minulého. Díky tomu si budeme moci pamatovat osoby i několik snímků zpět. V případě, že na jednom snímku osobu špatně detekujeme, budeme ji moc detekovat pod správným identifikátorem na snímcích následujících.

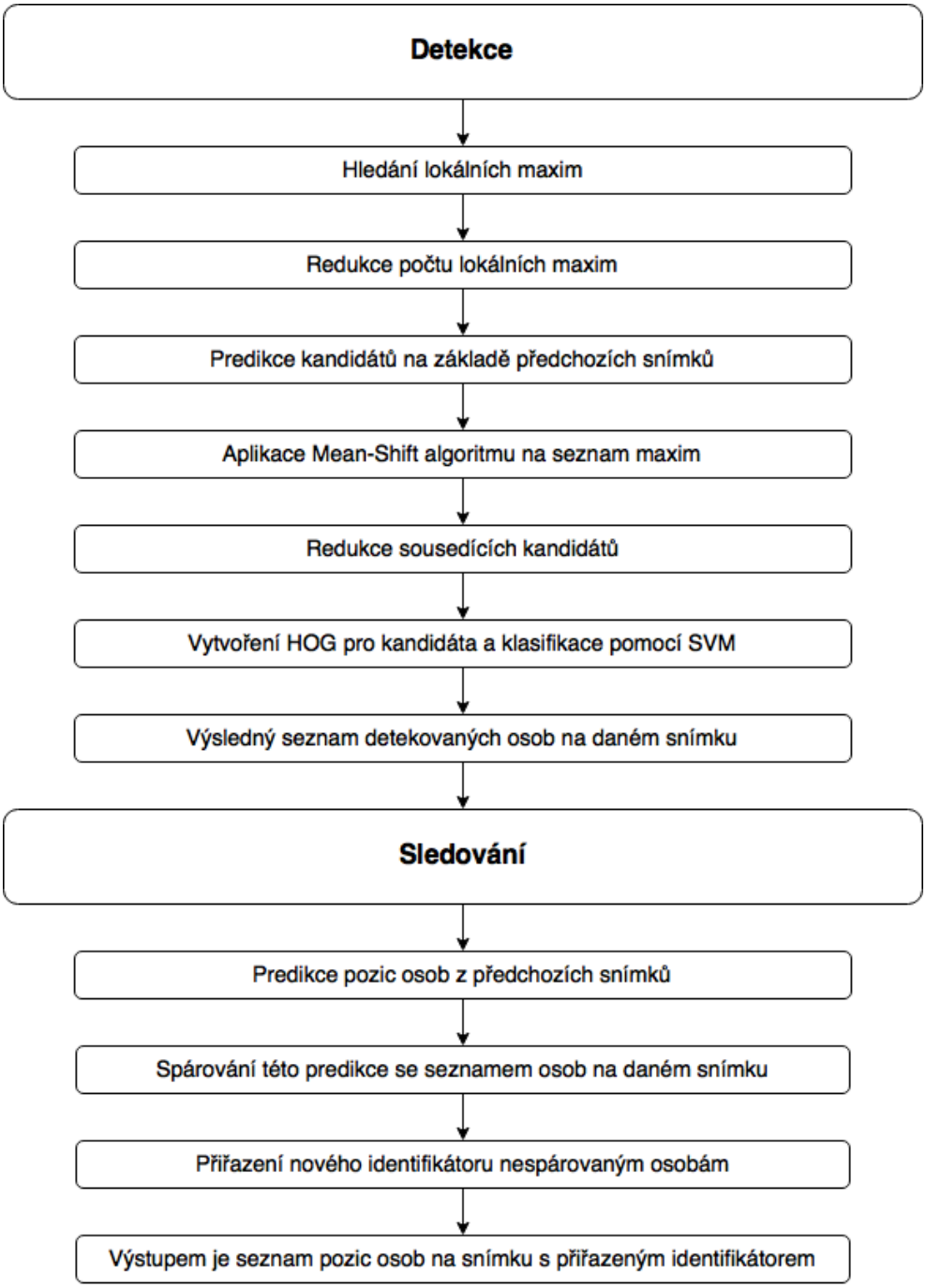
Predikce bude fungovat následovně. Nejprve si pro každou osobu budu pamatovat, pozici v posledním snímku, čítač před kolika snímky byla naposledy spatřena a pohybový vektor, který bude udávat směr a rychlost pohybu. Poté se predikovaná pozice pro následující snímky vypočítá přičtením pohybového vektoru k poslední známé pozici. Pohybový vektor se před přičtením k aktuální pozici ještě vynásobí čítačem pro počet snímků, před kterými byla daná osoba naposled spatřena. To by mělo pomoci ke správnému spojení osob v případě výpadku detekce na některém ze snímků.

Na diagramu 1.6 můžete vidět navrhovaný průběh algoritmu celé detekce, včetně přiřazení identifikátoru osobám pro jeden snímek sekvence. Na grafu následujícím 1.7 pak můžete vidět průběh algoritmu, který byl upraven na základě výsledků během implementačního testování.

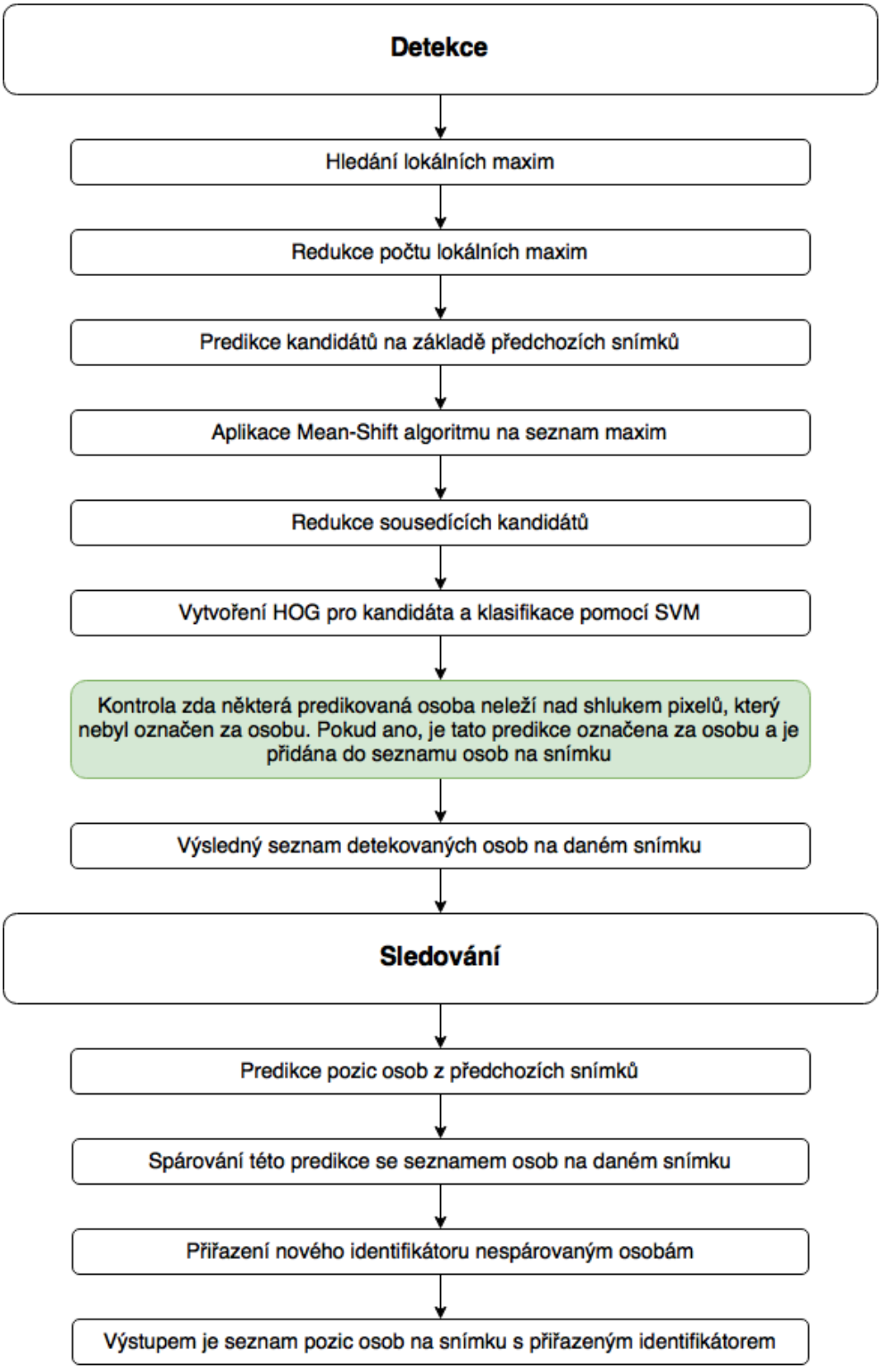
1.5 Použité metody

1.5.1 Mean-shift

Na základě článků [10] a [11]. Mean-Shift je algoritmus pro hledání lokálního maxima hustoty v daném vzorku dat. Jde o iterativní gradientní algoritmus. Pro zvolený bod se vypočítá gradient ze vzorků v okolí, které je definované zvoleným kernelem. Tento bod se poté v každém kroku posouvá do oblasti s větší hustotou a to dokud nezačne konvergovat. Pomocí Kernel funkce se určuje váha okolních bodů pro nový odhad střední hodnoty.



Obrázek 1.6: Navrhnutý průběh detekce a sledování



Obrázek 1.7: Implementovaný průběh detekce a sledování

Mezi nejběžnější kernel funkce patří

Flat kernel

$$k(x) = \begin{cases} 1 & \text{if } x \leq \lambda \\ 0 & \text{if } x < \lambda \end{cases}$$

Gaussian kernel

$$k(x) = \mathrm{e}^{-\frac{\|x\|^2}{2\sigma^2}}$$

Vážená střední hodnota hustoty v okolí určeném kernel funkcí je

$$m(x) = \frac{\sum_{x_i \in N(x)} K(x_i - x) x_i}{\sum_{x_i \in N(x)} K(x_i - x)}$$

kde $N(x)$ je množina bodů sousedících s x pro něž platí $K(x)! = 0$. Rozdíl $m(x) - x$ se nazývá Mean-Shift, algoritmus poté nastaví $x = m(x)$, a opakuje odhad dokud $m(x)$ nekonverguje.

1.5.2 Histogram Orientovaných Gradientů

Dalším použitým algoritmem je HOG [12]. HOG je deskriptor vlastností používaný v počítačovém vidění a zpracování obrazu za účelem detekování objektu. Tato technika počítá výšky orientace gradientu v lokální části obrazu. Je podobná HOH (Histogram Orientovaných Hran), ale liší se v tom, že počítá na husté mřížce rovnoměrně rozmístěných buněk a používá normalizaci překrývajících se lokálních kontrastů pro zlepšení přesnosti.

1.5.2.1 Výpočet Gradientů

Prvním krokem bývá před-zpracování snímku pro zajištění normalizace barev a gama hodnot. Jak poukázali Dalal a Triggs tento krok může být vynechán, jelikož následná deskriptorová normalizace dosáhne stejného výsledku. Místo toho se v prvním kroku počítá rovnou hodnota gradientu. Nejběžnější metodou je použití 1-D centrované diskrétní derivační masky aplikované v horizontálním a vertikálním směru. Dalal a Triggs testovali použití jiných složitějších masek jako 3x3 Sobel mask, ale zjistili, že tyto masky fungují hůře pro detekci lidí. Stejně jako při testování Gaussovského vyhlazování aplikovaného před derivační maskou.

1.5.2.2 Orientace Gradientu

Druhým krokem je rozdělit snímek na buňky. Nyní pro každou buňku určíme histogram jejích gradientů. Každý pixel má váhu pro orientovaný histogram kanálu, založenou na hodnotě získané při výpočtu gradientu. Kanály histogramu jsou v rozsahu 0° - 180° nebo 0° - 360° podle toho zda je gradient se znaménkem nebo bez znaménka.

Dalal a Triggs zjistili, že bezznaménkový gradient ve spojení s devíti kanálovým histogramem fungoval nejlépe v jejich pokusech s detekcí osob. Každý gradient patří k jednomu kanálu podle svojí orientace, výsledný HOG pro buňku je součtem gradientů v jednotlivých kanálech.

1.5.2.3 Bloky deskriptoru

Vzhledem k změně kontrastu, musí být velikost gradientu lokálně normalizovaná, za tímto účelem spojujeme buňky do bloků. Tyto bloky se většinou překrývají, což znamená, že jedna buňka přispívá více jak jednou do výsledného deskriptoru. Existují dvě hlavní geometrie bloku obdélníková R-HOG a kruhová C-HOG. R-HOG bloky jsou obecně čtvercové mřížky reprezentované třemi parametry. Počtem buněk v bloku, počtem pixelů v buňce a počtem kanálů histogramu buňky. V pokusech Dalala a Triggse s detekcí osob, byly zjištěny jako optimální parametry 2x2 buňky na blok, 8x8 pixelů na buňku s 9 kanálovým histogramem.

1.5.2.4 Normalizace Bloku

Dalal a Triggs zkoumali čtyři různé metody pro normalizaci bloku. Máme nenormalizovaný vektor v, který obsahuje všechny histogramy daného bloku. Potom je k norma pro $k = 1, 2$ a e je nějaká malá konstanta, normalizační vektor může být jeden z následujících.
L2-norm:

$$f = \frac{v}{\sqrt{\|v\|_2^2 + e^2}}$$

L2-hys: Je stejná jako L2-norm pouze omezuje velikost v na $0, 2$. L1-norm:

$$f = \frac{v}{\|v\|_1 + e}$$

L1-sqrt:

$$f = \sqrt{\frac{v}{\|v\|_1 + e}}$$

V testech Dalal a Triggs zjistili, že L2-hys, L2-norm a L1-sqrt poskytují podobný výkon, ale L1-norm poskytuje méně spolehlivý výkon. Nicméně všechny čtyři metody vykazují velké zlepšení proti nenormalizovaným datům.

1.5.2.5 Výsledný vektor

Posledním krokem je spojení normalizovaných histogramů, neboli deskriptorů, jednotlivých bloků do výsledného vektoru. Tento vektor má vysokou dimenzi což ústí ve velké paměťové nároky a náročné matematické operace.

1.5.3 Klasifikace pomocí lineárního SVM

SVM je metoda strojového učení, která se používá ke klasifikaci a regresní analýze. Vstupem mého lineárního SVM bude HOG vektor zmíněný výše. SVM je binární klasifikátor, to se hodí pro náš účel, jelikož pro vstupní data budeme chtít vědět, zda je na obrázku hlava či ne.

1.5.3.1 SVM model

Předtím než můžeme začít SVM používat k predikci, musíme vytvořit model, na jehož základě se bude predikce provádět. Tento model je vytvořen na základě tréninkových dat, jelikož jde o učení s učitelem, jsou tréninková data označena kategorií, do které patří. SVM reprezentuje tréninková data jako body v prostoru, ty se snaží oddělit takovou nadrovinou, která má největší vzdálenost k nejbližšímu bodu tréninkových dat. Výsledkem je rozhodovací funkce $f(x) = w\Delta x + b$

Jak bylo řečeno SVM je metoda strojového učení s učitelem z tohoto důvodu budu potřebovat množinu tréninkových dat, u které pro jednotlivé obrázky definuji kategorii, do které patří. Za tímto účelem jsem si udělat jednoduchou aplikaci, která dovoluje procházet obrázky, označit jednotlivé výřezy, zda obsahují hlavu či ne a uložit pro natrénování modelu.

Kapitola 2

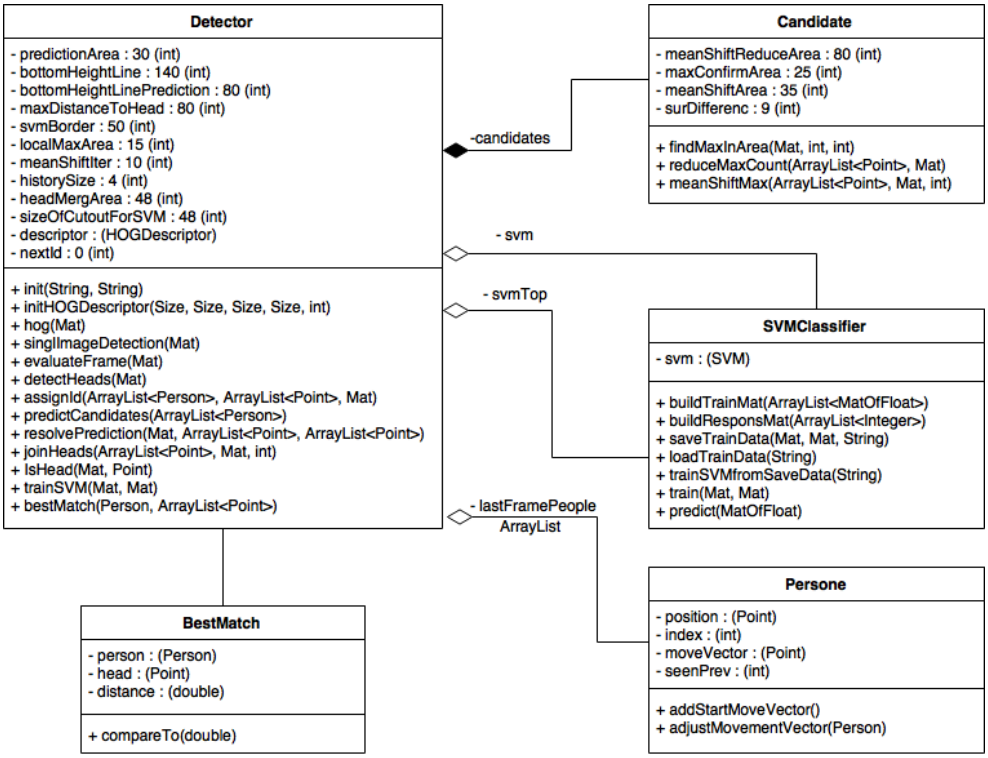
Realizace

Vyvíjený program bude použit jako knihovna rozšiřující funkcionalitu programu, který je napsaný v Jave. Z tohoto důvodu jsem své řešení programoval také v Jave. Pro práci s obrazem jsem si vybral OpenCV knihovnu, která je distribuovaná jako open source. A také obsahuje implementaci SVM, kterou využívám ke klasifikaci. Navíc je již používána v programu, s kterým se má vytvořená knihovna integrovat.

Problém jsem rozdělil do tří tříd Detector, SVMClassifier a Candidate. Důvod byl následující.
Třída Detector zaobaluje celou detekci a průběh algoritmu. Třída Candidate poskytuje třídě Detector metody k nalezení kandidátů. A třída SVMClassifier se stará pouze o způsob klasifikace. Toto rozdělení by mělo usnadnit například budoucí nahrazení dílčích částí, klasifikace a hledání kandidátů. Diagram tříd můžeme vidět na obrázku 2.1

Detekce

V první fázi jsem implementoval detekci osob pro jednotlivé nezávislé snímky. To znamená bez použití predikce pozic osob z předchozích snímků.
Podle návrhu jsem jako první začal implementovat hledání lokálních maxim. Pro hledání lokálních maxim bylo potřeba nalézt optimální velikost oblasti, ve které se lokální maximum bude hledat. Na velikosti této oblasti závisí počet lokálních maxim a také v případě, že daná oblast je příliš veliká, může dojít k tomu, že pro dvě blízko se pohybující osoby se detekuje maximum pouze na jedné a tak přijdeme o jednu osobu. V druhém případě kdy je daná oblast příliš malá, dostaneme velké množství maxim, které by zpomalovalo průběh algoritmu. Velké množství maxim také znamená, že jsou rozprostřena po celém povrchu daného objektu a to může mít za následek, že Mean-Shift nedostatečně posune krajní maxima a způsobit tak problém s detekcí

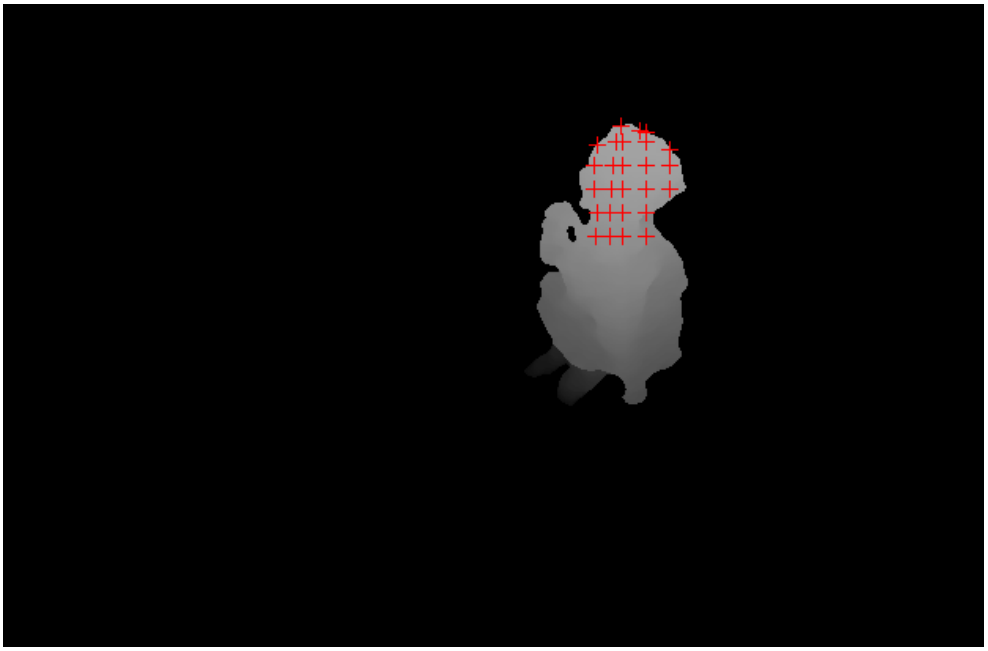


Obrázek 2.1: Diagram tříd

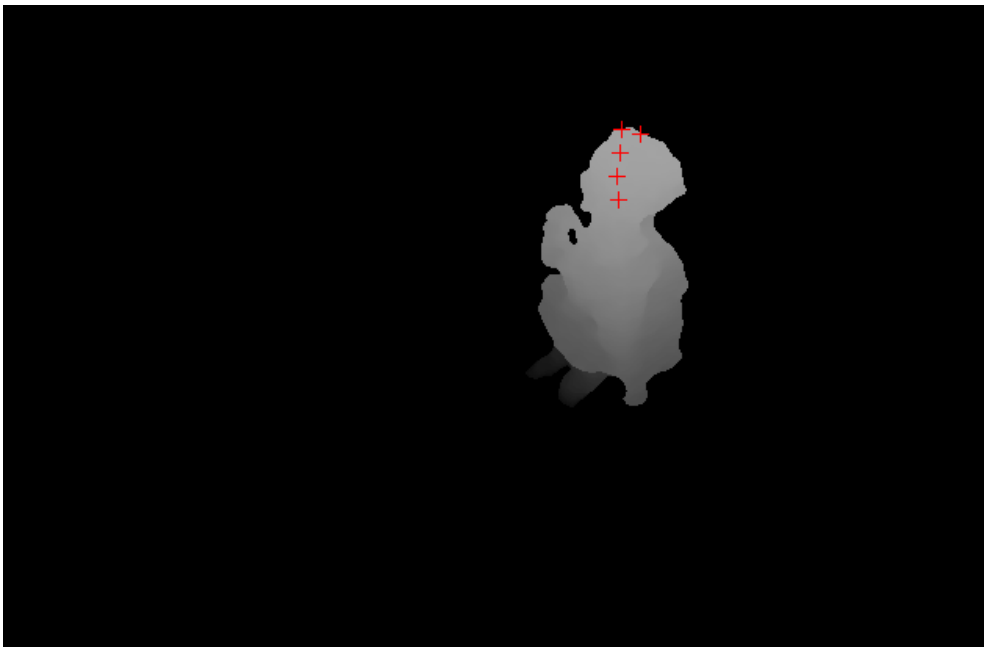
Po otestování několika velikostí této oblasti jsem se rozhodl, že přidám metodu, která bude redukovat počet lokálních maxim. Na obrázku 2.2 je vidět maxima před přidáním redukce, v tomto případě jich je 26. Na obrázku 2.3 pak můžeme vidět maxima pro ten samý snímek, ale po aplikování redukce jich zůstalo pouze 5.

Dalším krokem po implementaci získání maxim byl krok, přesunutí těchto maxim blíže ke středu hlavy. Za tímto účelem jsem na základě analýzy, zvolil metodu Mean-Shift. Tuto metodu jsem ne-implementoval, jelikož jsem použil Mean-Shift, který je součástí knihovny openCV. Pro tuto metodu bylo znovu potřeba nastavit okolí, ve kterém se Mean-Shift provádí. Při zvolení příliš velké oblasti se bod přesune ke středu celého shluku a to není přesně to, čeho chceme dosáhnout. Z toho důvodu jsem se rozhodl použít oblast o trochu menší než je velikost oblasti, ve které poté chci provádět detekci pomocí SVM klasifikátoru. Tato oblast je tak velká, aby obsáhla hlavu osoby.

V některých případech stále docházelo k tomu, že se bod přesunul jiným směrem nebo nedostatečně. Větší počet iterací pomohl některým bodům přesunout se blíže, ale ne dost efektivně.



Obrázek 2.2: Výsledek hledání maxim



Obrázek 2.3: Výsledek redukce maxim z obrázku 2.2

Obrázek 2.4: Výsledek aplikování mean-shiftu na naxima z obrázku 2.3

Jak bylo zmíněno v článku [4], zde před použitím SVM klasifikátoru odstranili paže dané osoby. Implementoval jsem podobnou část, která před použitím Mean-Shiftu odstraní v blízkém okolí testovaného maxima vše co je pod zadaný výškový rozdíl od původního maxima. Po přidání této části již byl výsledek Mean-Shiftu dostatečně uspokojivý. Na obrázku 2.4 je vidět výsledek Mean-Shiftu pro maxima z obrázku 2.3.

Dodatečně jsem přidal metodu, která ještě odstraní kandidátní body, které jsou v těsné blízkosti. To pro obrázek 2.4 znamená, že zbude pouze jeden kandidát, pro kterého se bude provádět klasifikace pomocí SVM.

Tímto jsem měl naimplementovanou část pro získání kandidátů na osobu v jednom nezávislém snímku. Jako další jsem implementoval SVM klasifikátor. Jelikož jsem použil SVM, které je součástí openCV, tak většina práce byla s naimplementováním jednoduchého nástroje, který mi umožní natrénovat model pro můj SVM klasifikátor.

Tato aplikace prochází předložené obrázky, používá dříve implementované části hledání maxim a Mean-Shift, aby nalezla kandidáty. A poté se pro každého kandidáta zeptá, zda jde o hlavu. Zadanou odpověď uloží společně s HOG dané oblasti.
Jelikož aktuální implementace openCV v Jave mi neumožnila nahrát uložený

model pro SVM. Musel jsem implementovat vlastní ukládání tréninkových dat a jejich následovné načtení a použití k opětovnému natrénování modelu. Rozšiřující funkcionalitou je, možnost vložení snímků, které obsahují objekty, které chceme použít k natrénování negativní odpovědi. Pro tyto snímky se automaticky doplní, že daný výřez neobsahuje hlavu.

Dosavadní výsledek detekce na nenavazujících snímcích jsem podrobil dlouhému testování. A samotná úspěšnost detekce osob na těchto snímcích závisela samozřejmě na kvalitě modelu pro SVM klasifikátor. Výsledky od těchto modelů se pohybovaly s úspěšností mezi 82 až 93 procenty. Během tohoto testování jsem na základě výsledku přidal například zmíněnou funkcionalitu, která umožnila přidat do modelu negativní příklady rychle a ve větším objemu.

Toto testování probíhalo nad tisícovkou obrázků vybraných z dostupných testovacích dat. Při výběru těchto snímků jsem se snažil rovnoměrně pokrýt všechny možné situace, kde se může na snímku vyskytovat osoba a zároveň se případ bude lišit od ostatních. Dalšími snímky byly snímky obsahující další prvky, ve výšce hlavy, které se mohou vyskytovat na snímku. Jedná se převážně o zboží, které může být v některých případech špatně rozpoznáno SVM klasifikátorem, a další jako jsou zvednuté ruce nad úroveň hlavy nebo jako je balónek na obrázku 2.5. Balón je podobný vršku hlavy, a i když se mi povedlo několikrát natrénovat takový model, který ho nedetekoval, vtom to snímku. Při testování na celé sekvenci, ho stejně v některých snímcích SVM klasifikátor shledal hlavou.

Sledování

Po dokončení detekce v jednotlivých snímcích jsem začal s vytvářením sledování osob přes sekvenci snímků. Podle návrhu jsem přidal část, která bude predikovat pozici osob na daném snímku z pozic osob na snímcích minulých. Pro uchování a práci s osobami z minulých snímků jsem si vytvořil pomocnou třídu Person. V této třídě jsem si pro každou detekovanou osobu ukládal poslední známou pozici, počet snímků před kolika byla daná osoba naposledy spatřena a pohybový vektor.

Po každém snímku se tento vektor upravuje na základě toho, o kolik se kterým směrem osoba posunula oproti minulé pozici. Toto však bylo nedostačující, jelikož tato úprava byla příliš ovlivněna faktem, že bod který definuje osobu, není umísťován na stejném místě. To mělo za následek, že v případě, kdy se daná osoba pohybovala příliš pomalu, mohl se tento bod v dalším snímku vyskytnout například v protisměru původního pohybu. To v některých případech způsobovalo, že daná osoba byla predikována na špatné pozici. Z tohoto důvodu jsem udělal úpravu, která tuto změnu pohybového vektoru koriguje, jak přesně tato korekce funguje, naleznete v kapitole 2.5.1.

Obrázek 2.5: Testovací snímek s balónem

Tyto predikované body jsem přidal k seznamu, který je poté kontrolován SVM klasifikátorem. Jak jsem uváděl dříve, například pro balón jako je na obrázku 2.5, se mi stávalo, že v některých snímcích byl predikován jako hlava. Ale samozřejmě SVM neposkytuje odpověď se stoprocentní jistotou, proto se mi stávalo, že v sekvenci na některých snímcích, nebyla hlava detekována ani tam kde by měla být.

Jak jsem zmiňoval v úvodu podobný problém má aktuální řešení. Za účelem zlepšení robustnosti vlastního sledování jsem se rozhodl přihlédnout k predikci s větším významem. Zachoval jsem část, která přidá predikované body mezi kandidáty pro SVM. Ale poté co se provede klasifikace pomocí SVM. Tak ještě znovu vezmu tyto predikované kandidáty a na každého se podívám, zda je v jeho okolí detekována osoba, pokud ne je tento kandidát označen za osobu. Tato úprava znatelně zlepšila schopnost sledování osoby pod jedním id, stále však měla výpadky na krajích snímků. Pro zlepšení v této části jsem přidal jednu iteraci mean-shiftu, která se spustí pro predikované body, tím se zlepší umístění predikovaného bodu. Tato úprava řešila i ztrátu sledované osoby v případě, že se pohybovala po horním okraji snímku a mizela tak hlava dané osoby ze snímku.

Pro vylepšení samotné detekce těchto osob, které se pohybují již od samého začátku na snímku tak, že není vidět jejich hlava, jsem přidal druhý SVM klasifikátor. Ten to druhý klasifikátor jsem natrénoval tak, aby akceptoval také ramena pro klasifikaci, že jde o osobu. Poté pro každý bod, pro který se provádí klasifikace, se nejdříve zkontroluje, které SVM se použije. V případě, že kandidátní bod leží v horních padesáti pixelech, použije se SVM klasifikátor, který akceptuje i ramena v opačném případě se použije původní klasifikátor, který detekuje hlavu. Tuto hranici padesáti pixelů je ovšem možné změnit pomocí konfiguračního souboru.

2.1 Detector

Třída Detector zaobaluje průběh celé detekce v několika metodách. Jinými slovy slouží jako přístupový bod k detekci, jelikož obsahuje metodu, která řídí celý průběh detekce. V obrázku 2.6 je vidět průběh detekce v jednom snímku.

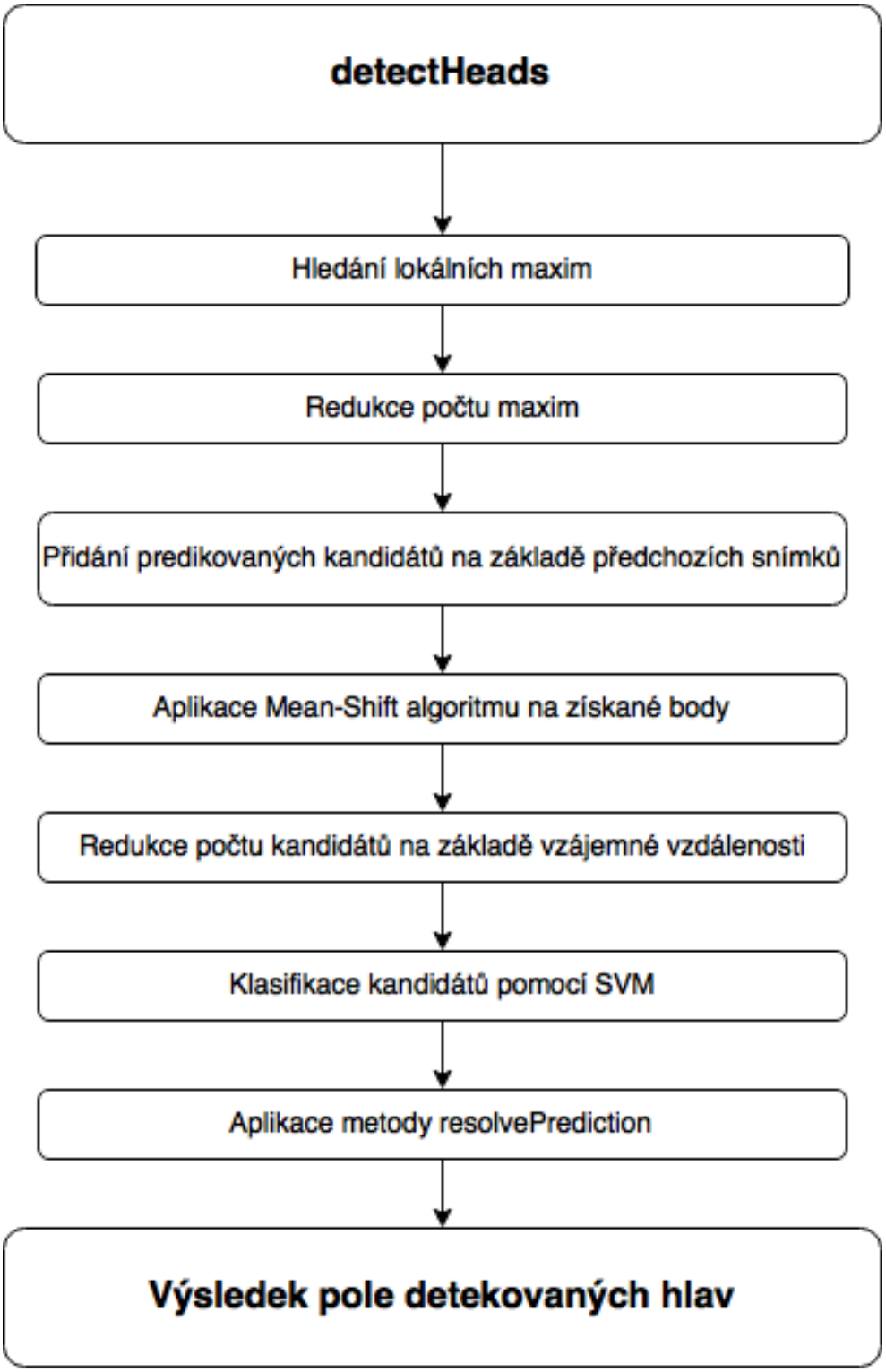
O vyhodnocení jednoho snímku se stará metoda evaluateFrame. Jak je vidět na obrázku 1.7, vyhodnoceni jednoho snímku můžeme rozdělit na dvě části. Detekci osob ve snímku a spojení detekovaných osob s osobami na předchozím snímku. Za tímto účelem třída Detector obsahuje dvě metody detectHeads 2.1.1 a assigneId 2.1.2, kde detectHeads se stará o detekci osob v daném snímku a assigneId se stará o spárování nalezených osob s osobami na snímcích předchozích a na základě toho jim přiřadí číselný identifikátor. Zbylá část potom pouze upraví data do dohodnuté výstupní podoby, neboli jak bylo řečeno výše do HashMapy, kde klíč je identifikátor a hodnota je bod symbolizující osobu.

Jak bylo řečeno v návrhu, ke klasifikaci bude použit SVM klasifikátor nad histogramem orientovaných gradientů. Za účelem získání histogramu pro SVM klasifikátor je součástí této třídy HOGDescriptor. Jedná se o strukturu, která je součástí OpenCV knihovny a umožňuje mi vygenerovat vector histogramu, bez nutnosti měnit formát obrázku.

2.1.1 detectHeads

Metoda detectHeads využívá metody ze zbylých dvou tříd, pro vytvoření seznamu osob v daném snímku. Tento snímek je jediným vstupem metody.

Jak tato detekce funguje, můžeme vidět na obrázku 1.7. Vidíme, že prvním krokem je nalezení všech možných míst na snímku, kde by se mohla vyskytovat osoba. Za tímto účelem se využívá metoda třídy Candidate findMaxInArea, jak přesně tato metoda funguje, je popsáno v 2.2.1.



Obrázek 2.6: Průběh metody detectHeads

Výsledkem této metody je prvotní seznam kandidátů na osoby v daném snímku. Počet kandidátů závisí na parametru metody findMaxInArea pro velikost oblasti, ve které se hledá lokální maximu. Čím menší oblast tím více kandidátů dostaneme pro jednu osobu a zároveň se zvětšuje šance, že detekujeme dvě osoby blízko u sebe. Po testování s několika velikostmi oblasti, se mi osvědčily hodnoty od <12,18>, tyto hodnoty jsou v pixelech. Pro další testování jsem šel středem a použil hodnotu 15. Tento prvotní seznam při tomto nastavení obsahuje velké množství kandidátů, které minimalizuji metodou reduceMaxCount, její fungování je popsáno v 2.2.2. Po těchto dvou krocích mám list kandidátů, který je dále potřeba upravit tak, abych ho mohl využít pro klasifikaci pomocí SVM.

Když jsem přestal testovat detekci v jednotlivých nenavazujících snímcích a zaměřil se na sekvenci navazujících snímků. Pro zvýšení úspěšnosti detekce jsem přidal metodu predictCandidates, tato metoda je součástí třídy Detector. Jak přesně funguje, naleznete v části 2.1.3. Jak její název napovídá, jejím výstupem je seznam kandidátních pozic pro osoby v daném snímku na základě snímků předchozích. Tyto kandidátní pozice se poté přidají k seznamu kandidátů vytvořeného z předchozích dvou metod.

Poté je dalším krokem úprava tohoto seznamu kandidátů tak, aby se dal úspěšně použít pro klasifikaci pomocí SVM klasifikátoru. Prvním krokem této úpravy je použití metody meanShiftMax. Tato metoda je součástí třídy Candidate a jak přesně vypadá, naleznete v 2.2.3. Vstupem je předchozí seznam kandidátů a výstupem je seznam těchto kandidátů přesunutých do centra hustoty. V mém případě to znamená do středu hlavy, alespoň v případě, kdy jde o kandidáta v blízkosti hlavy osoby. Toto je první část úpravy. Dalším krokem, je potom další minimalizace počtu kandidátů, jelikož v důsledku předchozí metody teď mnoho kandidátů sdílí stejnou hlavu. K tomuto účelu je použita metoda joinHeads jejímž výstupem je seznam kandidátů pro SVM klasifikátor. Tato metoda je součástí třídy Detector a popsána je v části 2.1.4.

V tuto chvíli tedy mám redukovaný seznam kandidátů a dalším krokem jak už bylo řečeno je otestovat kandidáty tohoto seznamu v SVM klasifikátoru. Za tímto účelem je zde metoda IsHead, ta je součástí Detectoru a jak už z názvu vyplývá, pro daný bod vrací, zda jde o hlavu.

Tímto krokem mi končila původní implementace podle návrhu. Když jsem, ale začal testovat detekci na sekvenci navazujících snímků, potýkal jsem se s nedostatečnou robustností, co se týče sledování jedné osoby. To znamená, že pokud se v několika po sobě jdoucích snímcích opakovala situace, že SVM klasifikátor špatně vyhodnotil předložený bod a osoba nebyla v daném snímku detekována, tak algoritmus na sledování měl problém udržet pro jednu osobu ten samý identifikátor.

Po inspiraci aktuálním řešením, jsem se rozhodl také více používat historii sledování a přidal jsem metodu resolvePrediction.
Tato metoda je důkladněji popsaná v části 2.1.5. Z pohledu vyhodnocení snímku tato metoda vrátí seznam bodů, který obsahuje predikované body, které na základě historie sledování byly vyhodnoceny jako osoby. Tento list se poté přidá k seznamu osob získaného vyhodnocením předchozího seznamu kandidátů pomocí SVM klasifikátoru. A vrátí se jako výsledek metody detectHeads.

2.1.2 assigneId

Jak již bylo řečeno výše, dalším krokem po metodě detectHeads je metoda assigneId. Vstupem této metody je seznam osob z předchozího snímku a seznam osob, který jsem dostal pro aktuální snímek pomocí metody detectHeads. Úkolem této metody je přiřadit identifikátor osobám na aktuálním snímku pokud možno tak, aby se pod stejným identifikátor daná osoba sledovala po celou dobu jejího pohybu v nepřetržité sekvenci snímků.

Toto přiřazení probíhá následovně. Vezmou se osoby z předchozích snímků a predikuje se jejich pozice na aktuálním snímku. Tyto body se poté pospojují na základě vzdálenosti. Důvodem proč použiji body predikované je, že v případě blízko sebe se pohybujících osob by aktuální a minulá pozice mohly poškodit systém přiřazení identifikátoru na základě vzdálenosti. Když by došlo k prohození jejich identifikátorů.

Pro úspěšné spárování osob mezi novým a starým snímkem metoda assigneId začne metodou bestMatch. Tato metoda se pustí pro každou osobu z minulého snímku a jejím výstupem je pole osob z aktuálního snímku seřazených podle vzdálenosti k danému bodu.
Nabízí se myšlenka, že generování seřazených seznamů pro všechny body by mohlo mít negativní dopad na rychlost algoritmu, ale vzhledem k velikosti scény by neměl být počet osob tak velký, aby toto zpomalení bylo nepřijatelné obzvlášť, když se nepočítá s během algoritmu v reálném čase.

Poté co tedy získáme tyto seřazené seznamy pro všechny předešlé body. Pustí se metoda assigneId do spárování osob z minulého snímku s osobami ze snímku aktuálního. K tomuto párování jsem použil dvořící rituál. Abych zajistil, že všechny osoby na aktuálním snímku budou spárované s nejbližší osobou ze snímku předešlého. Pro zajištění, že se mi nespárují osoby, které na sebe pouze zbudou, například jedna osoba na jedné straně vyjde a na druhé straně přijde osoba nová. Za tímto účelem je zde nastavená maximální vzdálenost, na kterou lze dvě osoby spárovat. To zabrání této situace, která by jinak rozbila sledování osob pod jedním identifikátorem.

Pro případ, že v některém snímku z jakéhokoliv důvodu některou osobu nedetekujeme a dojdeme tak na konec seznamu kandidátů pro spárování s danou osobou. Tak se tato metoda postará o to, aby tato nespárovaná osoba byla zařazena do seznamu osob z tohoto snímku s o jeden zvýšeným počtem snímků, kdy byla naposledy spatřena. Takže pokud se v určitém počtu snímků tato osoba znovu ukáže na místě, kde předpokládáme její výskyt, je jí přiřazen správný identifikátor. Tento seznam osob, si udržuje třída Detector interně. Je to seznam osob, které se vyskytly na posledních n snímcích. Samozřejmě pokud daná osoba přesáhne limit pro zachování v seznamu tak v případě, že nebyla detekována, již se nepokračuje v jejím uchovávání. Nové osoby jsou do tohoto seznamu zařazeny s novým unikátním identifikátorem.

2.1.3 predictCandidates

Jak jméno funkce napovídá, jejím úkolem je předpovědět pozici kandidátů na daném snímku, na základě pozic osob ve snímcích předchozích. Kolik snímků do historie se má osoba udržovat v paměti, se dá nastavit proměnnou historySize.
Vstupem metody je seznam lidí z předchozích snímků. Seznam je ve formě pole s použitím pomocné třídy Person. Jak vypadá třída Person, naleznete v 2.5.1

Metoda pak funguje velice jednoduše. Každá osoba si uchovává svou aktuální pozici na předchozím snímku, počet snímků kdy byla viděna naposledy a pohybový vektor, který se upravuje na základě pohybu osoby v jednotlivých snímcích. Metoda poté vezme tyto hodnoty a na základě nich vypočte novou pozici v dalším snímku. A to tak, že k aktuální pozici přičte pohybový vektor násobený počtem snímků, kdy byla osoba spatřena naposledy. V případě, že není nová pozice mimo hranice snímku, přidá se do seznamu pro výstup.

Počet snímků, pro který je osoba udržována v paměti v případě nebyla detekována jsem po přidání metody resolvePrediction nastavil na hodnotu tři. Jelikož s použitím metody resolvePrediction se řešení stalo velice stabilním co se týče schopnosti udržet sledování jedné osoby pod jedním identifikátorem po celou dobu pohybu osoby po sekvenci snímků.

K výpadku dochází pouze ve dvou situacích, hloubková mapa byla poškozena a ani metoda resolvePrediction nebyla schopna danou osobu zachytit. Anebo se dvě osoby přiblížily na takovou vzdálenost, že se spojily v jednu.

2.1.4 joinHeads

Tato metoda slouží k redukci počtu kandidátu v dané oblasti. Vstupy jsou pole kandidátů v mém případě kandidátů na hlavu, takže pole hlav. Snímek, pro který se provádí sloučení a velikost oblasti, ve které se body slučují.

Redukce funguje následovně pro každého kandidáta. Vypočítám vzdálenost k dalším kandidátům. V případě, že vzdálenost mezi kandidáty je pod poskytnutou mez, tak se porovná, který kandidát je vyšší, neboli daný pixel obsahuje větší hodnotu a tento kandidát se ponechá. V případě, že je kandidát odstraněn již se pro něj další srovnání neprovádí.

Po několika testech jsem přidal ještě vylepšení, které odstraní další kandidáty v okolí bodu a to v okolí o 60% větším. Tyto odstraňované body jsou omezeny podmínkou na velikost výškového rozdílu mezi body. Důvodem této úpravy bylo odstranit kandidáty z ramen a zad, jejichž vyhodnocení obsahovalo část hlavy a tak občas došlo k tomu, že je SVM vyhodnotilo jako hlavu.

Jako nejlepší hodnoty pro výškový rozdíl, které jsem testováním získal z dat, která jsem měl k dispozici, se jevila hranice pro rozdíl výšek jako 17 centimetrů. To znamená, že všechny body, které jsou v této oblasti a jejich hodnota je maximálně o 17 centimetrů menší než testovaného bodu, jsou odstraněny. Toto vylepšení pak odstraní kandidátní body z ramen osoby a tak se vyhne falešně pozitivní hlavě. Zároveň rozdíl ve výškách zaručuje, že pouze v malém procentu případů přijdeme o bod, který byl na skutečné hlavě.

Výstupem je potom redukovaný seznam kandidátů pro SVM klasifikátor. Tato metoda funguje jiným způsobem než metoda 2.2.2 reduceMaxCount a na rozdíl od té lze použít i poté co seznam kandidátů byl vyhodnocen pomocí SVM klasifikátoru, z tohoto důvodu jsem se rozhodl, ji přidat do třídy Detector a ne do třídy Candidate.

2.1.5 resolvePrediction

resolvePrediction je metoda, která je za robustností sledování osob v záznamu. Jejím úkolem je vzít seznam kandidátů, který byl vytvořen na základě předchozích snímků a zjistit zda se na predikovaných pozicích nachází osoby.

Prvním krokem je zjistit, které z predikovaných pozic nebyly vyhodnoceny SVM klasifikátorem jako osoba. Ostatní s tohoto seznamu odstraním. Jde pouze o odstranění predikovaných bodů, které jsou v okolí již detekovaných osob. Velikost okolí jsem nastavil na 40 pixelů. Zkoušel jsem i větší oblast, ale čím větší oblast tím větší šance, že se zbavíme jedné osoby v těsné blízkosti druhé.
Na druhou stranu s nastavením na 40 pixelů jsem nezaznamenal potíže a v průběhu testů vycházela predikce bodu v případě již nalezených osob do blízkosti tohoto bodu, takže v případě, že by se objevil problém se ztrátou osob v důsledku velikosti této oblasti, dá se hodnota této hranice změnit pomocí konfiguračního souboru spolu s ostatními hodnotami.

Po tomto kroku mi zůstanou pouze body na pozicích, kde jsem nedetekoval osobu, ale na základě předchozích snímků bych ji zde očekával.
Nad touto množinou bodů dojde k rozhodnutí, zda na dané pozici je osoba či ne. Na základě toho jestli se na dané pozici nachází nějaký objekt. K tomu dojde tak, že se vezme hodnota pixelu na této pozici a porovná se, se spodní mezí na výšku pro vyhodnocení této predikce. Tuto mez jsem při testování nastavil na 80 centimetrů. Při použití na testovacích datech se nikdo pod tuto mez nesklonil. Jelikož tato hodnota se může měnit v závislosti na požadavcích sledování, lze nastavit také pomocí konfiguračního souboru. Výstupem je tedy seznam osob, které budou přidány k seznamu z detekce pomocí SVM klasifikátoru. Díky tomu dojde jen velmi zřídka či vůbec ke ztrátě osoby ve snímku.

Na druhou stranu to sebou ovšem přináší i jednu nevýhodu. A tou je případ, kdy pro nějakého kandidáta SVM klasifikátor rozhodne, že se jedná o hlavu, i když ve skutečnosti není. V takovém případě mi tato metoda bude stále sledovat tohoto falešného kandidáta, pokud na predikované pozici bude nějaký objekt.

2.2 Candidate

Tato třída obsahuje metody, které se starají o nalezení kandidátů ve snímku pro klasifikaci pomocí SVM.

2.2.1 findMaxInArea

Jak již název napovídá, úkolem této metody je nalézt maxima. To je prvním krokem v navrhnuté detekci. Lokální maxima se hledají v oblasti dané velikosti. K jejich hledání používám metodu z openCV knihovny minMaxLoc().
Metoda začne v levém horním rohu obrázku a prochází snímek v sousedících čtvercích dané velikosti. V každém tomto čtverci se pak najdou lokální maxima pomocí zmíněné metody z openCV. Výstupem minMaxLoc je struktura, která obsahuje hodnotu nalezeného minima a maxima společně s jejich pozicí v daném čtverci.
Metoda findMaxInArea má ještě jeden parametr a tím je spodní mez pro nalezené maximum. Pokud nalezené maximum je rovno nebo větší této mezi je toto maximum vloženo do pole pro výstup v opačném případě se ignoruje. Před vložením se ještě upraví souřadnice nalezeného maxima ze souřadnic v nalezeném čtverci na souřadnice pro daný snímek.

2.2.2 reduceMaxCount

Výsledkem metody findMaxInArea je pole maxim pro daný snímek. Toto pole obsahuje velké množství maxim přes objekty na snímku. Úkolem této metody je snížit počet těchto maxim na přijatelnější množství.

Toho docílí tak, že vezme jedno maximum a použije ho jako střed pro vytvoření oblasti pro hledání maxima nového. Toto hledání probíhá stejně jako ve findMaxInArea za pomoci minMaxLoc. Redukce je u konce, když se projdou všechna poskytnutá maxima. Taková maxima, která jsou v oblasti nově nalezeného maxima, se rovnou odstraní.

2.2.3 meanShiftMax

Úkolem této metody je přesunout nalezená maxima do středu hustoty. Tím by se ve většině případů měl bod daného maxima přesunout do středu hlavy dané osoby. Jak funguje mean-shift je vysvětleno v části 1.5.1. Metoda neimplementuje samotný mean-shift, ale využívá mean-shift z openCV knihovny. Vstupem metody je pole kandidátních bodů, nad kterými se provede mean-shift, daný snímek a počet iterací mean-shiftu.

Prvním krokem pro každý bod je úprava okolí daného bodu, po testování jsem totiž došel k tomu, že při odstranění okolí, které je o konstantu menší než velikost daného bodu, je potom konvergence ke středu shluku rychlejší a přesnější. Tuto úpravu provádím pouze v blízkém okolí daného bodu. Samotný mean-shift poté probíhá v oblasti o něco menší. Oblast pro úpravu jsem zvolil osmdesát pixelů od daného bodu. Oblast pro samotný mean-shift je potom třicet pět pixelů od daného bodu. V případě větší vzdálenosti pro mean-shift se již bod nepřesune do středu hlavy, ale do středu celého shluku, což je nežádoucí proto tak to veliká oblast. Předpokladem, že se nalezení kandidáti přesunou do středu hlavy, je fakt, že kandidáti nalezení pomocí findMaxInArea jsou na nejvyšším místě daného shluku, což je ve většině případů pravě hlava.

Jak bylo řečeno výše, metoda využívá metodu meanShift, která je součástí openCV. Při každé iteraci se bod přesune směrem ke středu shluku v mém případě většinou hlavy.

Metoda meanShiftMax je využívána v metodě 2.1.1 detectHeads, zde jsem zvolil počet iterací deset. Důvodem proč nemít pouze jednu iteraci je, aby se ke středu hlavy dostaly i vzdálenější body. Na druhou stranu nesmím zvolit tolik iterací, že by se nám všechny body přesunuly na jedno místo. To by byl problém v případě osob, které jsou blízko sebe. Zároveň se mi v případě, že se přesunou špatným směrem, nevzdálí příliš daleko. Mně se na testovacích datech dobře osvědčilo právě deset iterací, ale na větším vzorku dat se později může dojít k jinému závěru proto jsem tuto hodnotu udělal nestavitelnou pomocí konfiguračního souboru. Posledním krokem je pak vložení nové pozice daného bodu do výstupního pole bodů.

2.3 SVMClassifier

Tato třída obsahuje pouze jednu metodu použitou při samotné detekci a tou je metoda predict. Jejím vstupem je histogram orientovaných gradientů testovaného kandidáta. Výstupem je pak rozhodnutí zda jde o hlavu či ne, respektive zda jde o osobu či ne.

Dalšími metodami jsou metody použité k uložení a natrénování SVM klasifikátoru. Pro natrénování SVM klasifikátoru to jsou metody, které vytvoří pole vstupních dat a pole ohodnocení těchto dat, zda jde o hlavu či ne. Bohužel aktuální wrapper openCV knihovny nemá naimplementovanou metodu pro načtení natrénovaného SVM modelu. A proto pro natrénování modelu ukládám právě tyto dvě pole.
Detector potom načte soubor s těmito daty a natrénuje na nich SVM klasifikátor, k tomuto slouží metoda trainSVMfromSaveData.

2.4 Řídící proměnné

Tato sekce má za úkol pouze shrnout všechny proměnné, které upravují detekci a jakým způsobem ji ovlivňují. Jejich nastavení budu řešit až v sekci testování.

Detector

predictionArea Udává velikost oblasti okolo predikovaného bodu. V případě, že se v této oblasti nenachází jiná osoba, je tento bod označen za osobu pokud, splňuje další podmínky, které byly vysvětleny výše.

bottomHeightLine Tato proměnná udává spodní mez pro hledání maxim.

bottomHeightLinePrediction Udává spodní mez pro odsouhlasení predikovaného bodu jako osoby.

maxDistanceToHead Maximální vzdálenost mezi osobou aktuální a z minulého snímku pro přiřazení identifikátoru.

svmBorder Vzdálenost od vrchního okraje snímku, která tvoří předěl mezi použitým SVM klasifikátorem.

localMaxArea Velikost oblasti, ve které se hledají lokální maxima.

meanShiftIter Počet iterací Mean-Shiftu.

historySize Počet snímků, na kterých osobu nedetekujeme, ale stále si ji pamatujeme.

headMergArea Velikost oblasti, ve které se dvě osoby spojí v jednu.

sizeOfCutoutForSVM Velikost výřezu kolem kandidátního bodu pro SVM klasifikátor, neměla by se měnit beze změny HOG deskriptoru.

Candidate

meanShiftReduceArea Velikost oblasti kolem maxima, ve které se provádí úprava před mean-shiftem.

surDifferenc Velikost rozdílu okolí, při jehož překročení se provádí úprava okolí před mean-shiftem.

maxConfirmArea Velikost oblasti, ve které se kolem lokálního maxima provádí druhé hledání.

meanShiftArea Velikost výřezu, ve kterém se provádí mean-shift.

2.5 Pomocné struktury

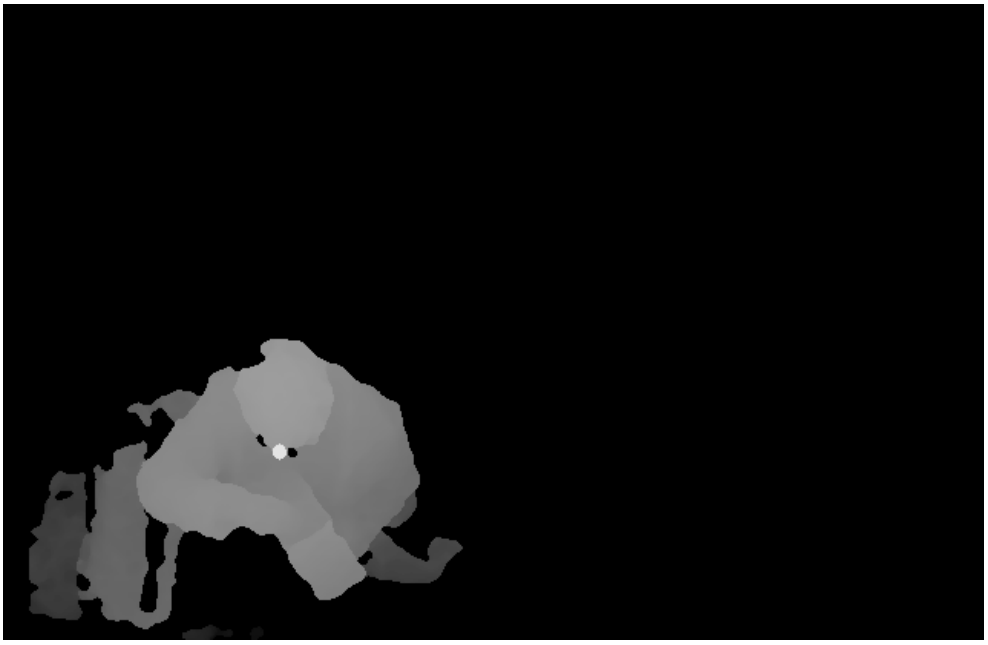
Ve třídě Detector jsem ještě vytvořil dvě pomocné třídy. A to třídu Person, která slouží Detectoru k uchování informací o dané osobě spojené s jejím sledováním přes sekvenci snímků. A třídu BestMatch, která slouží k uchování informací pro párování za účelem přiřazení identifikátoru osobě.

2.5.1 Person

Jak bylo řečeno výše jednou částí při sledování osob na snímcích, je predikce jejich pozice na snímku dalším. A jak jsem zmínil, za tímto účelem si uchováváme počet snímků, před kolika jsme naposledy viděli danou osobu a pohybový vektor této osoby. Další informací, kterou tato třída drží je samozřejmě pozice dané osoby na posledním snímku, kde byla detekována a identifikátor této osoby přidělený během detekce.

Při vytvoření třídy se pohybový vektor nastaví na výchozí hodnotu, která predikuje pohyb do snímku. Toto prvotní nastavení je z důvodu, kdy predikce probíhá na přičtení pohybového vektoru k pozici, jako byla v předchozím snímku. Nastavení této prvotní hodnoty není tak veliké, aby poškodilo predikci v případě, že daná osoba vstoupí do snímku pomalu nebo stojí na rozmezí dvou snímku. Ale zároveň zlepšuje výsledky v opačných případech.

Důvodem k tomuto, je způsob jakým se pohybový vektor upravuje. Během první implementace jsem jednoduše počítal pohybový vektor mezi každými dvěma snímky znovu. Tento postup by nejspíš fungoval dostatečně dobře v případě, kdy bod, kterým označujeme danou osobu, by byl vždy na stejném místě dané osoby.



Obrázek 2.7: Hloubková mapa s defektem

To bohužel, ale neplatí a hned z několika důvodu prvním je kvalita snímku jak jsem již zmiňoval v sekci 1.1 a můžeme vidět na snímku 1.4. Tyto výpadky můžou mít za následek posunutí detekovaného bodu i opačným směrem v případě pomalu pohybující se osoby nebo defektu v hloubkové mapě jako na snímku 2.7.

Ovšem to může mýt katastrofální důsledek, jelikož v dalším snímku bude daná osoba predikována opačným směrem. Následkem toho bude osoba označena novým identifikátorem, v nejhorším případě může být původní identifikátor přiřazen osobě jiné. Vzhledem k snímkovací frekvenci 30 snímků za vteřinu je posun osoby mezi jednotlivými snímky ve většině případů malý. Tím pádem by k této nejhorší variantě nemělo doházet tak často.

Abych tomuto předešel, zvolil jsem jinou metodu. Pohybový vektor se stále vypočítá na základě dvou posledních snímků, ale místo toho, aby se tato hodnota uložila, porovná se s původním pohybovým vektorem a v případě příliš velké změny je tento nový pohybový vektor zmenšen o konstantu. Tato úprava se děje jak pro nárůst, tak pro pokles rychlosti pohybu.
Zkoušel jsem i procentuální zvětšení a zmenšení, ale měl jsem lepší výsledky s touto konstantní úpravou, jelikož dokáže lépe reagovat na opravdovou změnu pohybu dané osoby. A zároveň ho neovlivní šum při detekci.

2.6 Pomocné nástroje

Jak jsem v předcházejícím textu zmiňoval, detekce je řízena značným množstvím hodnot, které jí nějakým způsobem upravují. O těchto hodnotách a mnou získaném nastavení budu mluvit až v další části testování. Ale abych vůbec mohl nějaké testování celé detekce a jednotlivých nastavení daných hodnot provádět musel jsem si udělat pomocné aplikace. Jelikož mnou dodané řešení bude pouze knihovna funkcí.

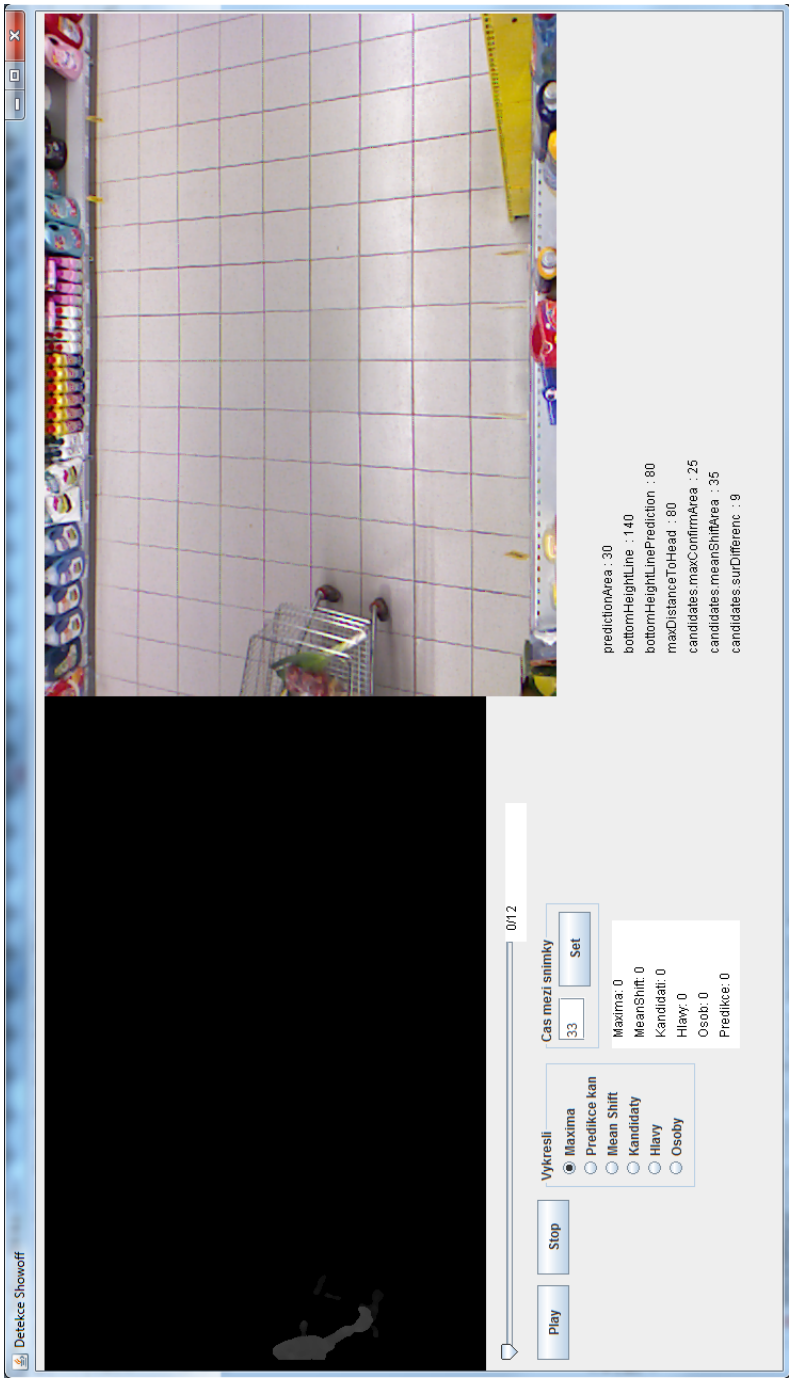
Abych mohl začít provádět nějaké testování, potřeboval jsem natrénovat SVM klasifikátor. V případě SVM jde o metodu strojového učení s učitelem, tedy pro každý vzorek je nutné určit, zda jde o hlavu či ne. Za tímto účelem jsem si vytvořil jednoduchou aplikaci, která mi dovolí vytvořit tréninková data pro SVM klasifikátor. Jak bylo řečeno v sekci 2.3, aktuální openCV neumožňuje uložit natrénovaný model, a proto tato pomocná aplikace ukládá tato data potřebná pro natrénování SVM klasifikátoru. Jelikož jde o velmi zdlouhavý proces ohodnotit tak velké množství snímků, tak poté co jsem již měl za sebou trénování velkého množství modelů a věděl jsem, které snímky jak označit přidal jsem metodu logování, která zaznamenává pozici daného tréninkového výřezu na snímku a jeho ohodnocení. Tato úprava byla nutná později, když jsem testoval různé nastavení HOG desriptoru a potřeboval jsem tak přetrénovat daný model.

Tato aplikace je součástí přiloženého disku, i když není součástí výsledku mé práce a hodnocení.

Další pomocnou aplikaci, kterou jsem potřeboval, je samozřejmě něco, co by mi umožnilo testovat jednotlivé metody dané knihovny. Proto jsem vytvořil aplikaci, která mi umožnila testovat jednotlivé úpravy v nastavení proměnných i způsobu provádění detekce a sledování.

Stejně jako předchozí aplikace ani tato není součástí výsledku mé práce.

Pro zadané snímky provede detekci a její výsledky si uloží do paměti. Umožňuje tedy pohybovat se oběma směry videa, zastavit video a pro jednotlivé snímky zobrazit výsledky dílčích kroků detekce. To znamená, že můžeme zobrazit nalezená maxima, jejich pozici po použití mean-shiftu, výsledné kandidáty po provedení sloučení, ale i výsledky, predikce a zadaná osoba byla detekována pomocí SVM klasifikátoru, anebo pomocí heuristiky nad predikovanými body. Tuto aplikaci můžete vidět na obrázku 2.8.



Obrázek 2.8: Testovací aplikace

Kapitola 3

Testování

V této části popíši testování řešení. A to ve dvou sekcích. První sekce bude o testování během implementace k nalezení nejlepšího nastavení proměnných pro detekci i sledování. A druhá o testování úspěšnosti řešení.

3.1 Testovací data

Ještě než se pustím k samotnému testování. Něco málo k testovacím datům. Pro samotný vývoj jsem dostal data nasnímaná za jeden den. Ta jsou rozdělená do hodin od 7 hodin do 22 hodin. Přibližně obsahující 122 000 snímků. Dostupné jsem měl jak snímky hloubkové mapy, tak snímky barevné. Důvodem pro barevné snímky je, abych byl schopen určit, co přesně se na snímcích nachází neboť jak je vidět na obrázcích 1.5 a 1.4, ne vždy je to tak snadné.

Druhou sadu dat jsem dostal k otestování výsledku k porovnání se stávajícím řešením. Jde znovu o data z jednoho dne tentokrát od 7 hodin do 14 hodin, Data obsahují přibližně 77 000 snímků. K těmto datům byla navíc přidána anotace v podobě JSONS souborů. Anotace obsahuje označené osoby na snímku pomocí rovnoběžníku, který obklopuje danou osobu. Bohužel tato anotace je udělána na základě barevných snímků, jelikož anotuje osoby i na snímcích, kde na hloubkové mapě není nic zachyceno. K tomu jak jsem s tím naložil až v sekci Závěrečné testování 3.3

3.2 Implementační testování

Během implementace jsem měl k testování dostupnou pouze první sadu dat. Z nich jsem vybral přibližně tisíc snímků, které jsem použil k natrénování prvního modelu pro SVM klasifikátor. Ohodnocení a následné vytvoření modelu jsem prováděl mnohokrát opakovaně, jelikož pokaždé jsem zjistil trochu víc o tom jak SVM bude reagovat na dané ohodnocení snímku.

Tyto natrénované modely jsem využíval k testování pro nalezení nejlepšího nastavení řídících proměnných.

Detector

predictionArea Tuto oblast jsem nastavil na 30 pixelů to je o něco menší než velikost oblasti pro slučování hlav, důvodem bylo, aby v případě, že se dvě osoby přiblíží až tak blízko, predikce si s tím stále dokáže poradit a jedna z osob se neztratí.

bottomHeightLine Slouží k omezení detekce, jelikož ve výsledku nás až tolik nezajímají děti a také k ignorování nákupních košíků. Já jsem zvolil velikost 140 centimetrů.

bottomHeightLinePrediction Ovlivňuje zejména, jak dlouho budeme detekovat osobu při opouštění snímku, například pokud ve snímku zůstává noha dané osoby. Já jsem zvolil 80 centimetrů, což pro testovací data bylo dostačující, jelikož se mi nikdo neskláněl pod danou úroveň, když byl ve snímku.

maxDistanceToHead Pro tuto hodnotu jsem zvolil 150 pixelů což je dostačující, abych nespojil příchozí a odchozí osobu a zároveň pokrývá historii snímků, kterou jsem nastavil na 3 snímky.

svmBorder Zde jsem neprováděl žádné testování velikost 50 pixelů, je to o něco více než, když by daná osoba byla polovinou hlavy mimo snímek.

localMaxArea Jako nejlepší hodnota se mi jevila velikost 15 pixelů, pro větší oblast bylo pochopitelně nalezeno méně maxim a to mělo za následek, že při redukci jejich počtu se vytvářeli příliš oddělené seskupení a tím dávali více možností SVM klasifikátoru udělat chybu při detekci.

meanShiftIter Počet iterací Mean-Shiftu, jsem nastavil na 10, při tolika iteracích se většina bodů z hlavy osoby přesune do jednoho místa a to přibližně ke středu.

historySize Tuto proměnnou jsem nastavil na 3 snímky, i když se mi nestalo na testovacích datech, že by osoba nebyla detekována více než v jednom snímku. A to díky metodě 2.1.5 resolvePrediction.

headMergArea Tuto hodnotu jsem nastavil stejně jako tu následující, pokud jsou dva body ve vzdálenosti menší, než 48 pixelů spojí se v jednu hlavu.

sizeOfCutoutForSVM Já jsem zvolil velikost 48 pixelů, tato velikost kandidátního bodu je dostatečná k pokrytí celé hlavy.

Candidate

meanShiftReduceArea Tuto oblast jsem nastavil na 80 pixelů, důvodem pro to bylo. Aby další iterace mean-shiftu nebyly ovlivněné.

surDifferenc Tento rozdíl jsem nastavil na 9 centimetrů cokoliv s větším rozdílem je odstraněno.

maxConfirmArea Velikost této oblasti, ve které se provádí hledání lokálních maxim pro redukci jejich počtu, jsem nastavil na 25 pixelů. Důvodem proto bylo docílit redukce sousedních maxim.

meanShiftArea Tato oblast nesmí být příliš velká, aby nebyl mean-shift ovlivněn i okolím, které nás nezajímá. Jelikož počítáme s tím, že kandidátní body jsou v okolí hlavy, zvolil jsem velikost 35 pixelů. Ta zaručuje malé posuny tím správným směrem.

3.3 Závěrečné testování

Toto testování jsem prováděl na druhé sadě dat, jak pro vytvořené řešení, tak pro řešení původní. Model pro SVM, který rozhoduje, zda daný kandidát je osoba jsem vytvořil z prvního datasetu na přibližně deseti tisících snímcích.

Jelikož anotace pro druhou sadu dat byla prováděna pravděpodobně na barevných snímcích, jsou osoby anotované i na poškozených snímcích. Poškození snímků je různě závažné od chybějící časti jako ve snímku 1.4, toto poškození by nemělo narušit sledování dané osoby. A to kvůli tomu, že daná osoba při vstupu do snímku nebyla poškozena, k poškození došlo až v průběhu jejího průchodu, i přesto však zabírá dostatečně velkou souvislou část.

Další formu poškození můžeme vidět na obrázku 3.1. Na tomto obrázku jsou ve skutečnosti dvě osoby, druhá osoba u horního okraje snímku však byla znatelně poškozena.

Hloubková mapa však může být poškozena i na tolik, že neobsahuje žádnou osobu, i když na barevném snímku se osoba vyskytuje. Pro získání lepší informace o úspěšnosti testování jsem snímky obsahující poškození ve formě druhých dvou případů odebral z testovací množiny.

Při odstraňování těchto snímků jsem si všiml, že k poškození dochází pouze v horní části snímku, tedy když osoba prochází u protějšího regálu.

Jak jsem říkal osoba je anotována ve chvíli, kdy se nějaká její část dostane do snímku, tím pádem pro snímek 3.2 to již znamenáno, že obsahuje osobu. Jelikož mé i původní řešení provádí detekci osoby na základě hlavy, nastavil jsem oblast od krajů snímku, ve které se nebude provádět vyhodnocení. Abych tak eliminoval negativní dopad těchto snímků na výsledek úspěšnosti detekce, když ještě ve snímku není podle čeho danou osobu detekovat.

Obrázek 3.1: Chybějící osoba v hloubkové mapě

Obrázek 3.2: Anotace osob

Tabulka 3.1: Výsledky testování mého řešení

Hodina	Správně detekováno	Špatně detekováno	Úspěšnost
7	495	50	90,8%
8	1515	240	86,3%
9	2260	427	84,1%
10	7769	745	91,2%
11	10135	802	92,6%
12	12989	883	93,6%
13	19783	1701	92%
14	7686	879	89,7%
Celkem	62632	5727	91,6%

Tabulka 3.2: Výsledky testování původního řešení

Hodina	Správně detekováno	Špatně detekováno	Úspěšnost
7	485	60	88,9%
8	1613	175	90,2%
9	2653	200	92,9%
10	7565	2665	73,9%
11	9846	3529	73,6%
12	12769	2440	83,9%
13	16766	9222	64,5%
14	8378	1778	82,4%
Celkem	60075	20069	74,9%

Vyhodnocování pak probíhá následovně. Proběhne detekce osob ve snímku a výsledek se porovná proti anotaci a spáruje se detekcí přidělený identifikátor s identifikátorem uvedeným v anotaci. Za chybu se pak počítá, pokud se změní identifikátor, přidělený k identifikátoru anotované osoby. Pokud detekovaná osoba nespadá do oblasti vymezující anotovanou osobu. Anebo případ, kdy anotovaná osoba nebyla vůbec detekována.
Za správné je považováno, pouze pokud detekovaná osoba spadá do oblasti pro anotovanou osobu. A daná anotovaná osoba ještě nebyla spojena s identifikátorem detekované osoby anebo identifikátor detekované osoby byl již spojen s danou anotovanou osobou. Výsledky mé detekce pro testovací data jsou zachyceny v tabulce 3.1.

Stejným způsobem jsem prováděl detekci na původním řešení, výsledky najdeme v tabulce 3.2

Z mnou naměřených výsledků vyplívá, že původní řešení často identifikuje osobu, i když se na dané pozici žádná nenachází.

3.4 Detekované problémy

Součástí přiloženého disku je video, které obsahuje detekci pro testovací data z druhé sady, přesně pro desátou hodinu. Na tomto videu můžeme vidět, kde nastávají problémy.

Většina chyb v detekci je způsobena špatným vyhodnocením kandidáta SVM klasifikátorem. Jak můžeme vidět v 1:57 na přiloženém videu. Ve chvíli, kdy se daná osoba skloní a vytvoří větší masu, detekuje SVM klasifikátor více osob. Domnívám se, že toto je způsobeno nepříliš kvalitním SVM modelem.

Bohužel lepší model by vyžadoval větší počet tréninkových dat, to bohužel se zmíněnou chybou v openCV, není příliš možné. Jelikož tréninková data ze zmíněných přibližně deseti tisíc snímků mají velikost 150MB. Druhým faktorem je doba, která je poté potřeba k znovu natrénování modelu.

Jednou z možností je nahradit použité SVM a využít knihovnu s jinou implementací. Další možností by bylo použití sekundární kontroly, která by potvrzovala rozhodnutí SVM. Popřípadě zkusit vytvořit robustnější model. Bohužel v případě robustnějšího modelu si nejsem jistý, jak velké zvětšení počtu trénovacích obrázků, by to vyžadovalo. Pro první model jsem použil tisíc snímků v porovnání s aktuálním modelem, který byl natrénován na desetinásobném počtu snímků, si tento model nevedl o mnoho hůře.

Další chybu, kterou můžeme na videu vidět v čase 3:33, ve chvíli, kdy se hlavy těchto dvou osob přiblíží do těsné blízkosti, jsou body detekované na hlavě pravé osoby přesunuty mean-shiftem k hlavě druhé osoby, jelikož je daná osoba vyšší. To má za následek, že levou osobu detekujeme příliš blízko osoby pravé a jejich hlavy se poté sloučí v jednu.

Zkoušel jsem změnit nastavení proměnných, které toto ovlivňují, ale bohužel takové nastavení, které by zamezilo této chybě, mělo za následek, chybu ve více běžných případech. A to pak způsobí pokles v úspěšnosti detekce, navíc předpokládám, že k přiblížení na této úrovni nebude docházet tak často, aby to způsobovalo nějaký vážný problém.

Do budoucna bych navrhoval přidat nějaký druh další heuristiky, který by tento případ řešil.

Dalším možným budoucím vylepšením by bylo přidání druhotného zpracování. Jak jsem zmiňoval, moje řešení není schopné detekovat osobu do té doby, než se do snímku dostane její hlava. Pro vylepšení detekce v těchto snímcích by se dala použít forma druhotného zpracování. Kdy tak jako se predikuje pozice osoby na dalším snímku. Tak by se prošly snímky s detekovanými osobami v opačném pořadí a na základě predikce by osoba měla být detekována i na těchto snímcích.

Závěr

V průběhu vypracování této práce jsem prošel a seznámil se s metodami detekce a sledování osob ve videozáznamu. Analyzoval jsem aktuální řešení problémů podobných tomu, který jsem řešil já. A na základě této analýzy jsem přišel s návrhem řešení zadaného problému.

Podle návrhu jsem poté provedl implementaci, kterou jsem podrobil dlouhému testování, za účelem dosáhnutí co možná nejlepších výsledků. Toho jsem dosáhl úpravami původního návrhu i správným nastavením proměnných, které řídí průběh detekce a sledování.

Řešení se ukázalo velmi robustní pro udržení stejného identifikátoru pro danou osobu v jedné sekvenci snímku. I když občas dochází, ke ztrátě dané osoby převážně v důsledku poškození hloubkové mapy. Anebo z důvodu špatného vyhodnocení kandidáta SVM klasifikátorem.

Dodatečně jsem vytvořil aplikace pro prezentaci výsledků a pro natrénování SVM klasifikátoru, které mi pomohly odstranit některé s chyb plynoucích ze špatného nastavení.

Výsledná implementace dosáhla celkové úspěšnosti pro detekci a sledování 91%, což si myslím, že s přihlédnutím ke kvalitě snímků hloubkové mapy je celkem slušný výsledek.

Literatura

[1] Wateosot, C.; Suvonvorn, N.: Top-view Based People Counting Using Mixture of Depth and Color Information. 2013.

[2] Thanarat Horprasert, D. H. a. L. S. D.: A Robust Background Subtraction and Shadow Detection. Dostupné z: https://www.researchgate.net/profile/Larry_Davis3/publication/228929364_A_robust_background_subtraction_and_shadow_detection/links/09e4150c7748b18c85000000.pdf

[3] Javier Lorenzo-Navarro, M. C.-S.; Hernandez-Sosa, D.: On the Use of Simple Geometric Descriptors Provided by RGB-D Sensors for Re-Identification. 2013. Dostupné z: http://www.mdpi.com/1424-8220/13/7/8222

[4] Tseng, T.-E.; Liu, A.-S.; Hsiao, P.-H.; aj.: Real-time people detection and tracking for indoor surveillance using multiple top-view depth cameras. *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, 2014. Dostupné z: http://ieeexplore.ieee.org/document/6943136/

[5] Felzenszwalb, P. F.; Huttenlocher, D. P.: Efficient Graph-Based Image Segmentation. Dostupné z: http://cs.brown.edu/~pff/papers/seg-ijcv.pdf

[6] *Support Vector Machines*. Dostupné z: http://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html

[7] Tia, Q.; Bo Zhou, W.-h. Z.; Wei, Y.; aj.: Real-time people detection and tracking for indoor surveillance using multiple top-view depth cameras. 2013. Dostupné z: http://www.jsoftware.us/vol8/jsw0809-19.pdf

[8] Rishabh, I.; Satish, A.: Human Detection in RGB Images. Dostupné z: http://www.ics.uci.edu/~dramanan/teaching/ics273a_winter08/projects/irishabh_HumanDetector.pdf

[9] Rauter, M.: Reliable Human Detection and Tracking in Top-View Depth Images. *Computer Vision and Pattern Recognition Workshops (CVPRW), 2013 IEEE Conference on*, 2013. Dostupné z: http://ieeexplore.ieee.org/document/6595924/

[10] Cheng, Y.: Mean Shift, Mode Seeking, and Clustering. Dostupné z: http://home.ku.edu.tr/mehyilmaz/public_html/mean-shift/00400568.pdf

[11] Comaniciu, D.; Meer, P.: Mean shift: a robust approach toward feature space analysis. Dostupné z: http://ieeexplore.ieee.org/abstract/document/1000236/

[12] Dalal, N.; Triggs, B.: Histograms of Oriented Gradients for Human Detection. Dostupné z: http://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf

Příloha **A**

Seznam použitých zkratek

SVM Support Vector Machine

HOG Histogram of Oriented Gradients

HDD Histogram of Depth Difference

CoG Comparison of Granules

HOH Histogram Orientovaných Hran

Příloha **B**

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `ukazka_detekce.mp4` ..... záznam ukázky detekce
- `src`
  - `impl` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
- `dist` ..... výsledná knihovna funkcí pro detekci osob
- `misc` ..... dodatečné soubory
  - `src` ..... zdrojové kódy pomocných aplikací
    - `test_app` ..... zdrojové kódy testovací aplikace
    - `train_app` ..... zdrojové kódy aplikace pro trenování SVM
  - `svm_model` . data pro vytvoření modelů SVM, použitá během testování