Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Bakalářská práce

Odhadování a predikce počtu lidí v menze na základě analýzy obrazu z kamer

Plzeň 2012 Radek Šmolík

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 8. května 2012

Radek Šmolík

Abstract

Estimation and prediction of number of people in the university cafeteria based on analysis of camera pictures.

The aim of this work is to create a program that handles images from cameras located on the premises of the university canteen. The program will then be able to estimate the current number of people inside. Results of the program should be used to obtain information about the movement of the people in the canteen.

The work shows the possibilities of processing images by different methods and describes basic methods of working with pictures, you can find more details about described methods and also other methods in the cited literature.

My solution is based on using thresholding method. Objects are separated from the background and then human figures are recognized. People are marked in the picture and the number of people is stored in the data field. Chart uses these values and shows information about the number of people in the canteen during the day.

The program provides well results in case that people are not gathered in large groups, it works better for separated human figures. Large groups are splitted automatically in attempt to recognize people according their expected size, but results are not exact, as human shapes are overlaped.

Obsah

1 Úvod

Cílem této práce je vytvořit program, který zpracuje snímky z kamer, umístěných v prostorách univerzitní menzy tak, že je schopen odhadnout aktuální počet strávníků.

Výsledky programu slouží k získání představy o pohybu strávníků v prostorách menzy. Bude možné určit, v jaký den a v jakou denní dobu menzu navštěvuje nejvíce/nejméně lidí a podle výsledků si zvolit, kdy je nejvhodnější menzu navštívit, aniž by bylo nutné čekat ve frontách na výdej jídel.

K vytvoření programu je použit jazyk JAVA. Program nejprve získává snímky z kamer, které ukládá na disk. Na snímky se aplikují metody pro zpracování a úpravu digitálního obrazu, aby byla konečná identifikace lidských postav v obraze co nejpřesnější. Výsledky programu jsou k dispozici ve formě grafu.

2 Zpracování obrazu

Obraz lze chápat jako obrazovou funkci $f(x, y)$, kde (x, y) jsou souřadnice pixelu. Hodnota $f(x, y)$ udává informaci o barvě v daném bodě, která může nabývat různých hodnot v závislosti na bitové hloubce obrazu. Bitová hloubka obrazu udává množství barev, které jsou k dispozici pro každý pixel. Např. pro 1 bitovou hloubku jsou k dispozici dvě možné hodnoty – bílá a černá. Se vzrůstajícím počtem bitů roste jejich množství.

Pro barevné obrazy se používají barevné modely. Pro RGB model je každý bod obrazu tvořen třemi hodnotami (barevnými kanály). Tzn., že při 8 bitové hloubce na každý kanál RGB je potřeba $8 * 3 = 24$bitů.

Příklady bitových hloubek pro 3 kanály RGB (tab. 2.1):

Bitů na kanál	Barev na kanál	Bitů na pixel	Barev pixelu
4	$16(2^4)$	$3 * 4 = 12$	$4096(2^{12})$
8	$256(2^8)$	$3 * 8 = 24$	$16777216(2^{24})$
16	$65536(2^{16})$	$3 * 16 = 48$	2^{48}

Tabulka 2.1: Bitové hloubky obrazu (RGB)

Barevné modely (RGB, CMYK, HSV a další) reprezentují barvu pixelu:

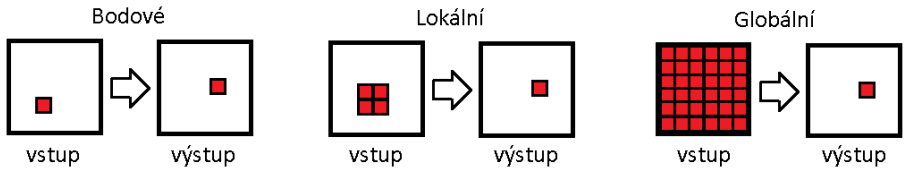
- RGB – hodnota se skládá ze tří složek, kanálů (R – červená, G – zelená, B – modrá barva)
- CMYK – čtyři složky (C – azurová, M – purpurová, Y – žlutá, K – černá barva)
- HSV – tři složky (H – odstín barvy, S – sytost barvy, V – hodnota jasu)

2.1 Předzpracování obrazu

Cílem předzpracování je připravit obraz tak, aby byl požadovaný výsledek zpracování co nejlepší.

Metody se dělí do několika skupin viz obr. 2.1, v závislosti na okolí aktuálního bodu (pixelu):

- bodové – bod je určen na základě jednoho bodu původního obrazu.
- lokální – bod je závislý na malém okolí bodu původního obrazu.
- globální – výsledný bod se určuje na základě celého původního obrazu.



Obrázek 2.1: Možnosti předzpracování obrazu

2.1.1 Filtrace

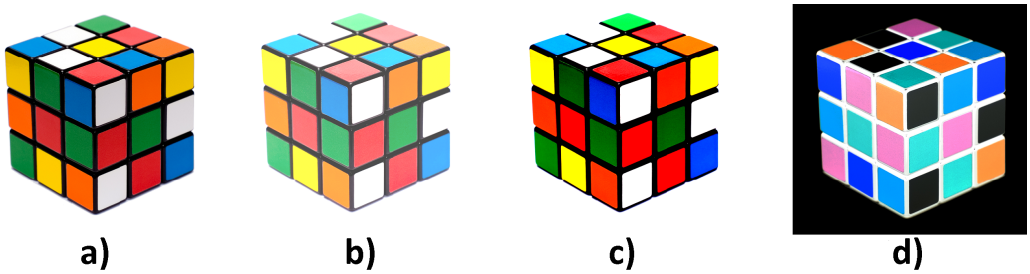
Kromě samotné informace o obrazu se může v obraze vyskytovat falešná informace nahodilého původu [5]. Obraz vypadá jako zrnitý. Tato informace se nazývá šum a je nežádoucí. Šum je přidán do obrazu během jeho snímání. Je většinou způsoben snímacím zařízením nebo přítomností prachu na scéně. Jeho přítomnost ztěžuje další zpracovávání obrazu a často je nutné ho potlačit. K tomu se používají různé filtry. Filtrace, neboli vyhlazování, pracuje vždy s větším množstvím snímků nebo s okolím bodu.

Metoda průměrování přes více snímků je založena na porovnávání pixelů jednotlivých obrázků a jako výsledná hodnota se vezme jejich průměrná nebo nejčastěji se vyskytující hodnota. Druhá metoda pracující s okolím bodu, zpracovává sousední hodnoty pixelů, které mohou mít předem zvolenou váhu a určuje výslednou hodnotu pixelu jako průměr nebo medián okolí. Filtrace s sebou nese několik nevýhod. Při filtraci více snímků roste výpočetní náročnost.

Filtrací se také ztrácí část informace z obrazu, dochází ke snížení ostrosti hran a obraz se jeví jako rozmazaný. Podrobnější informace o těchto metodách a dalších způsobech filtrování viz [2].

2.1.2 Jasové transformace

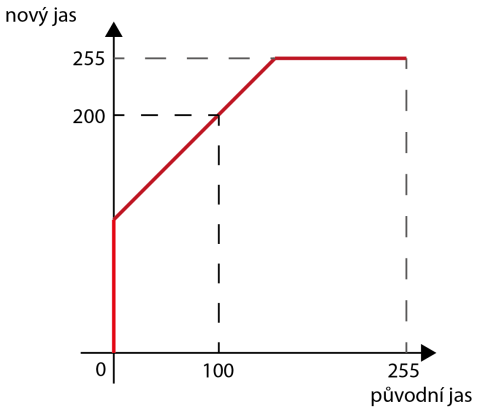
Pokud je obraz nerovnoměrně nebo nevhodně osvětlen, je možné využít jasové transformace. Hodnota jasu každého bodu se nahradí novou hodnotou, v závislosti na prováděné operaci (viz obr. 2.2).



Obrázek 2.2: Jasové transformace: a) původní obraz [8], b) úprava jasu, c) úprava kontrastu, d) negativ obrazu.

Při transformaci může dojít ke ztrátě informace, např. při extrémním zvýšení jasu se ztratí část obrazových dat a zpět je již nelze získat.

Na obr. 2.3 je vidět, jak transformace jasu probíhá. Např. původní hodnota jasu *100* se nahradí větší hodnotou *200*.



Obrázek 2.3: Funkce jasové transformace: zvýšení jasu

2.1.3 Geometrické transformace

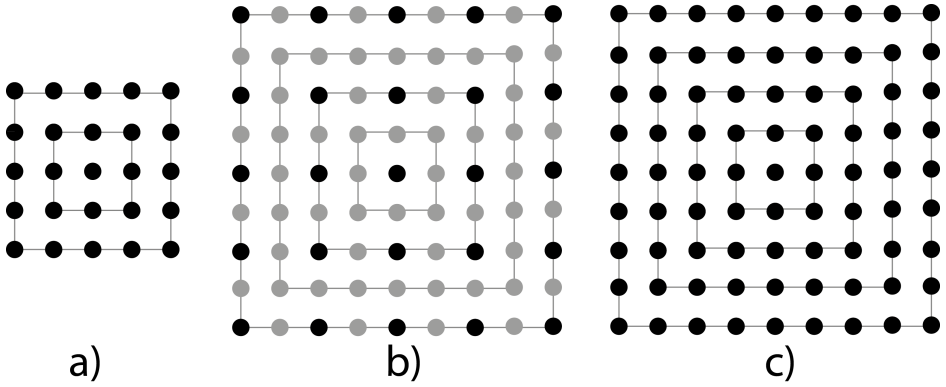
Mnohdy je potřeba obraz vhodně pootočit, zmenšit, posunout, apod. K tomu slouží geometrické transformace, které mohou být ztrátové. Např. změna velikosti obrazu se provádí interpolací obrazu a existují různé stupně. (Pro 1.stupeň interpolace se při zvětšování obrázku přidává mezi každé dva sousední pixely jeden nový (obr. 2.4), tzn. že z původně jednoho pixelu se udělají čtyři. Pro 2. stupeň se pak přidávají dva nové pixely, apod.)

Nové pixely se počítají na základě zvolených metod interpolace:

- Nejbližší soused – nový bod je dopočítán podle nejbližšího bodu původního obrazu.
- Bilineární interpolace – nový bod je dopočítán podle 2 nejbližších bodů.
- Bikubická interpolace – nový bod je dopočítán podle 4 nejbližších bodů.

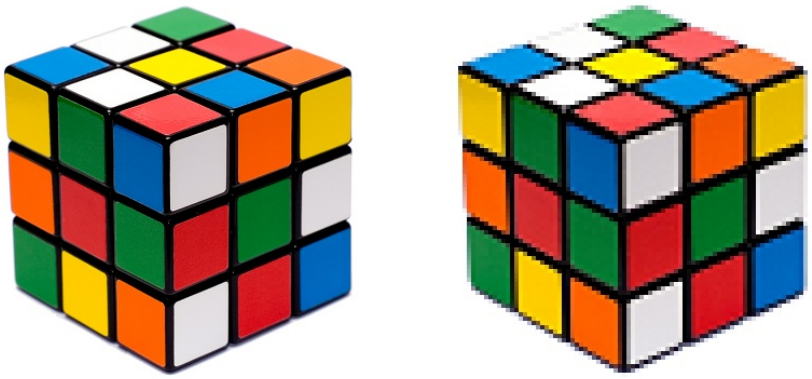
Více o interpolaci obrazu [6].

Příklad interpolace 1.stupně pro zvětšení obrazu viz obr. 2.4. Mezi každé dva sousední body se přidá jeden nový – na obr. 2.4b) šedé body.



Obrázek 2.4: Interpolace 1.stupně, zvětšení: a) původní obraz, b) postup zvětšení, c) výsledek

Zmenšením se ztrácí informace a dojde ke ztrátě detailů a rozostření obrazu. Na obr. 2.5 je vidět ztráta informace obrazu po jeho zmenšení.



Obrázek 2.5: Ztráta informace obrazu

2.2 Segmentace

Cílem segmentace je nalezení objektů v obraze. Obraz je rozdělen na části, kde každá část představuje jeden objekt obrazu. Nevýhodou segmentace obrazu je nutnost částečně znát obsah zobrazované scény, aby vůbec bylo možné najít jednotlivé objekty. Proto se místo jednotlivých objektů v obraze hledají jen souvislé oblasti s podobnými vlastnostmi a ty se dále zpracovávají.

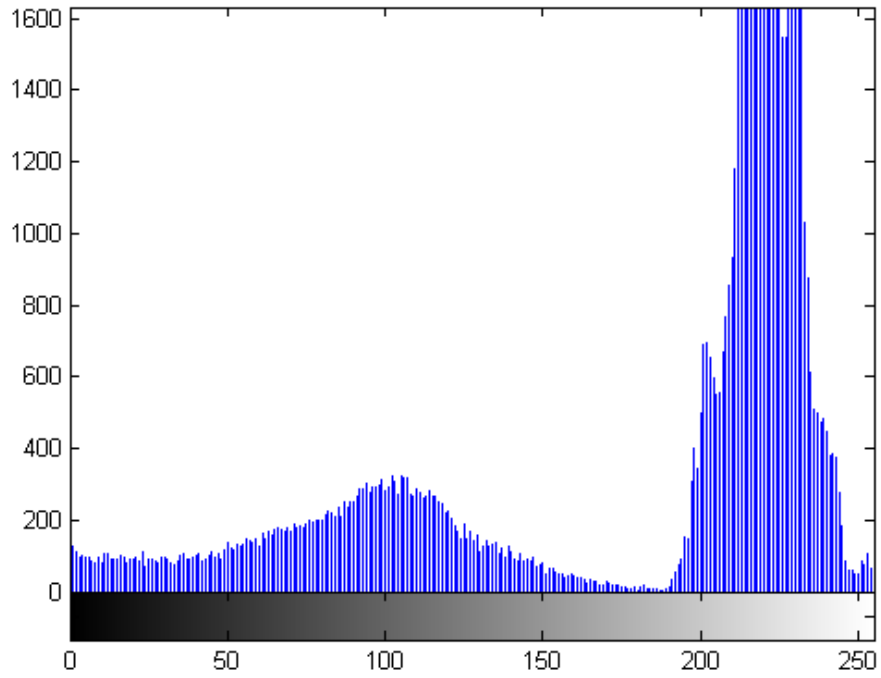
Hlavní skupiny segmentace:

- Prahování
- Segmentace založená na detekci hran
- Segmentace založená na hledání oblastí

2.2.1 Prahování

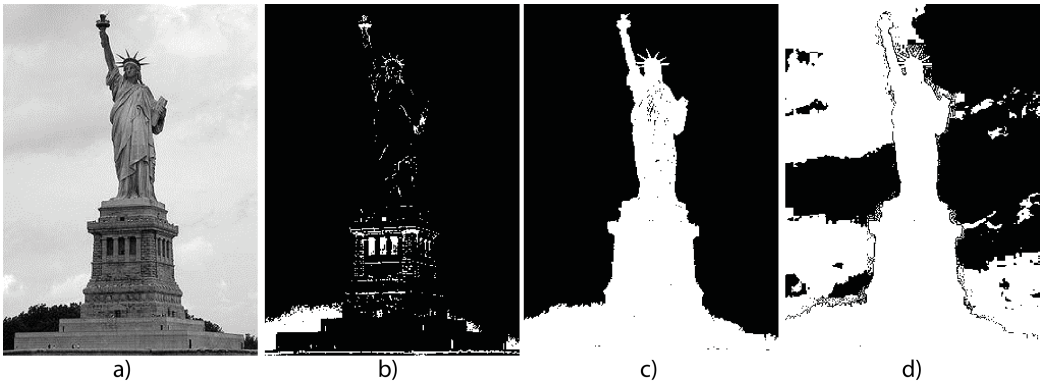
Jedná se o nejjednodušší metodu segmentace. Cílem této metody je oddělit objekty od pozadí. Využívá se histogramu (obr. 2.6), který udává, kolik pixelů z obrazu má jaký jas. Oddělení objektů od pozadí je určeno volbou prahu v histogramu (obr. 2.7). Pixely ležící pod prahem dostanou hodnotu 0, tj. jedná se o pozadí, pixely nad prahem dostanou 1, jde o objekty. Z toho je jasné, že vznikne obraz pouze se dvěma hodnotami pixelů, tedy obraz binární. Tato metoda nemá příliš význam pro objekty, které mají jasem velice blízko k pozadí. Zde není možné určit takový práh, který by objekty dokázal rozumně segmentovat.

Je několik způsobů prahování. Globální prahování využívá jeden práh pro celý obraz. Dalším prahování je tzv. adaptivní prahování, které používá více prahů pro různé oblasti obrazu. V případě, že nechceme jen binární obraz, lze volit větší počet prahů. Pak např. pro dva prahy dostaneme obraz se třemi hodnotami pixelů, apod.



Obrázek 2.6: Histogram pro obr. 2.7 a)

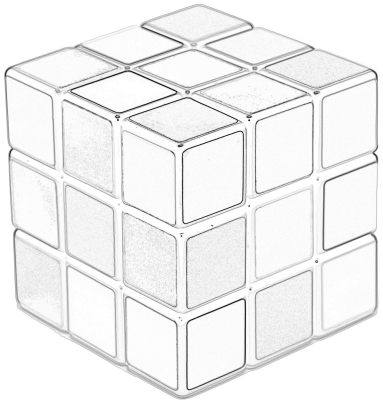
V některých případech je užitečné používat poloprahování. Všechny pixely pozadí mají stejnou hodnotu (např. bílá nebo černá barva), zatímco pixely objektů si zachovají své původní hodnoty.



Obrázek 2.7: Volba prahu: a) původní snímek [9], b) nízký, b) vhodný, c) vysoký práh.

2.2.2 Detekce hran

Hrana v obraze je místo, kde dochází k výrazné změně intenzity jasu (obr. 2.8). Pro její nalezení se používají gradientní metody, využívající hranové operátory [2]. Hranové operátory lze použít pro detekci čar a bodů v obraze.



Obrázek 2.8: Nalezení hran v obraze 2.2 a)

2.2.3 Narůstání oblastí

Metoda se dá uplatnit v obraze, kde je přítomný šum a obtížně se hledají hranice objektů. Využívá vlastnosti homogenity, tj. rozdělí obraz do co největších souvislých oblastí, které jsou homogenní. Většinou se jako kritérium homogenity považuje jasová složka nebo barva.

Hledání oblastí probíhá tak, že v prvním kroku se považuje každý bod obrazu za samostatnou oblast. Okolo bodu se hledají sousední oblasti, které splňují kritérium homogenity. Pokud podmínku splňují, tyto oblasti se spojí a vystupují jako jedna oblast. Takto se pokračuje, dokud nenalezneme všechny maximální oblasti. Původní rozdělení obrazu nemusí být vždy v poměru jeden bod je jedna oblast, ale lze zvolit počáteční segmentaci obrazu na oblasti o velikosti 2x2, 4x4 pixelů apod. Tímto se dá dobře eliminovat přítomnost šumu.

2.2.4 Srovnávání se vzorem

Pro aplikaci této metody musíme předem přesně znát, jak hledaný objekt vypadá. Vytvořený vzor objektu hledáme v obraze. Testuje se míra souhlasu obrazu se vzorem. Test se provádí pro každý bod obrazu, proto je metoda velmi časově náročná. Další komplikací je možnost natočení, či zkreslení objektu v obraze. V tomto případě by bylo nutné testovat míru souhlasu pro všechny možnosti geometrické transformace. To má za následek další zvýšení časové náročnosti. Šum přítomný v obraze ovlivňuje míru souhlasu se vzorem. To lze částečně vyřešit použitím filtrů.

Více o metodách segmentace [2].

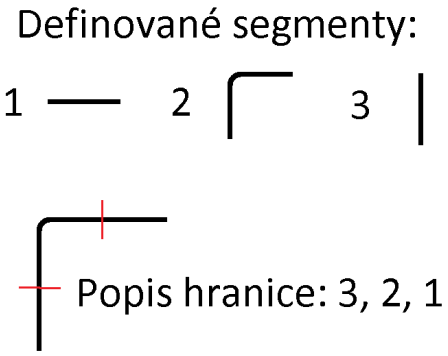
3 Popis a identifikace objektů

Objekt lze popsat oblastí nebo hranicí v obraze. Jsou dva druhy popisu objektu. Z prvního lze zpětně rekonstruovat tvar objektu, tzn. popis zachovává informace. Druhý způsob popíše objekt, ale zpětně již nelze získat informaci o jeho tvaru.

3.1 Hranice objektu

3.1.1 Popis posloupností segmentů

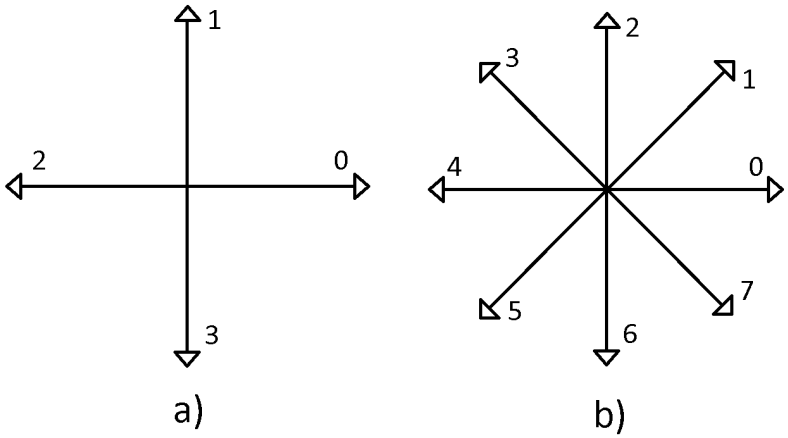
Hranice objektů se nahrazují různými typy segmentů, které se musí předem definovat a očíslovat (obr. 3.1). Výslednou hranici reprezentuje posloupnost čísel segmentů.



Obrázek 3.1: Segmenty hranice

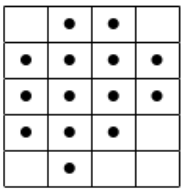
3.1.2 Freemanovy řetězové kódy

Popisuje objekty na základě směru hranice (obr. 3.2). Pro každý bod hranice určí změnu oproti předchozímu a popíše hranici odpovídajícím číslem. Popis lze volit pro tzv. 4-okolí nebo 8-okolí. Podle typu okolí je dosažena přesnost popisu hranice. Popis je nezávislý na otočení objektu.



Obrázek 3.2: Typy okolí: a) 4-okolí, b) 8-okolí

Příklad popisu obrázku pro 8-okolí (obr. 3.3):



Obrázek 3.3: Příklad 8-okolí

Freemanův kód: 076553221

Geometrické popisy

Využívají se částečně Freemanovy řetězové kódy. U metody *přímost hranice* se určuje poměr mezi celkovým počtem bodů v hranici a počtem bodů, ve kterých hranice není přímá, tzn. kde mění směr. Metoda *délka hranice* přiřazuje délku o hodnotě 1 horizontálním a vertikálním posuvům, pro diagonální posuv je dána hodnota odmocniny 2. Takto se určí délka hranice, která je vlastně obvodem objektu.

3.2 Popisy jednoduchých objektů

Metoda má využití pouze pro velmi jednoduché tvary objektů. Popisuje se celá oblast objektu. Využívají se k tomu např. následující vlastnosti:

- Podlouhlost: poměr mezi délkou a šířkou nejmenšího opsaného obdélníku
- Výstřednost: poměr dvou nejdelších na sebe kolmých tětiv
- Velikost: počet bodů v objektu
- Nekompaktnost:
$$\frac{delka\_hranice^2}{velikost}$$
- Eulerovo číslo: *počet souvislých oblastí - počet děr* (obr. 3.4)
- Výška, šířka

Obrázek 3.4: Eulerovo číslo: 1 souvislá oblast, 2 díry.

4 Klasifikace objektů

Klasifikace je posledním krokem při zpracování obrazu. Klasifikací se rozumí zařazování nalezených objektů do tříd, které jsou předem známy.

4.1 Příznakové rozpoznávání

Metoda je založena na využití příznaků. Hodnoty jednotlivých příznaků vytvoří vektor, kterým je pak obraz charakterizován.

Způsob zařazení objektů do tříd se provádí pomocí následujících klasifikátorů:

4.1.1 Metoda nejbližšího souseda

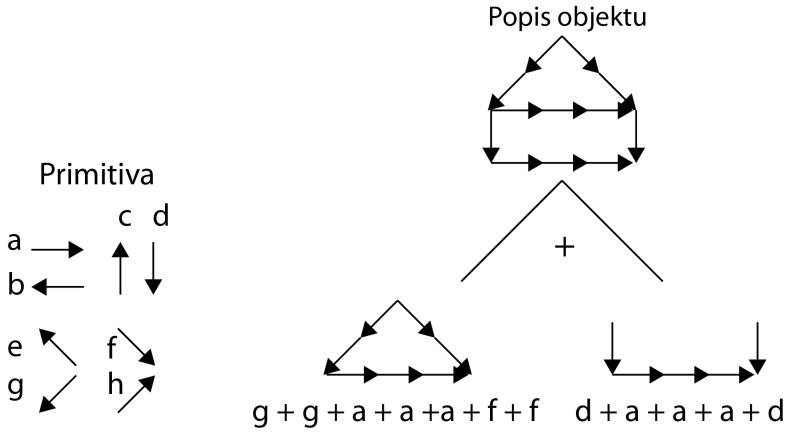
Třídění objektů probíhá tak, že se určuje vzdálenost od všech objektů, které jsou již zařazeny a objektu, který se zařazuje. Třída nejbližšího objektu je přiřazena novému objektu. Pro hledání vzdálenosti se používá např. Euklidovská metrika [7] a je počítána z příznakových vektorů objektů, tj. jak jsou si objekty podobné.

4.1.2 Metoda minimální vzdálenosti

Jednotlivé třídy zde zastupuje pouze jeden hlavní objekt. Ten má souřadnice těžiště všech objektů dané třídy. Přiřazení tříd probíhá podobně jako u předchozí metody nejbližší soused, tj. na základě minimální vzdálenosti od hlavního a vkládaného objektu.

4.2 Strukturální rozpoznávání

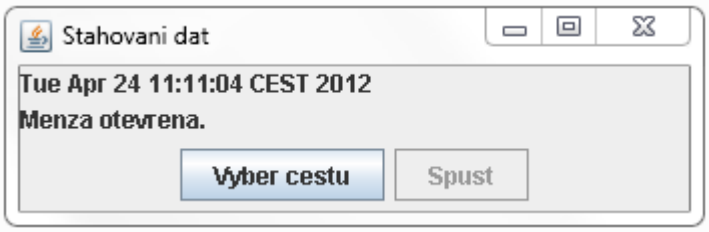
Objekt se popíše jeho elementárními vlastnostmi, kterým se říká primitiva. Primitiva mají mezi sebou vzájemný vztah (obr. 4.1).



Obrázek 4.1: Strukturální rozpoznávání: primitiva

5 Praktická část

Důležitou částí k napsání programu jsou zdrojová data. Ty lze získat z univerzitních kamer, umístěných v prostorách menzy. Pro sběr dat jsem vytvořil aplikaci (obr. 5.1), napsanou v jazyce JAVA, která sbírá data ze všech kamer menzy.



Obrázek 5.1: Aplikace pro stahování obrazu kamer

5.1 Stahování dat

Data jsou získávána ze tří kamer, umístěných v prostorách budovy univerzitní menzy. Jejich obraz je dostupný na webových stránkách [http://skm.zcu.cz], odkud je aplikace stahuje. Obrazy kamer se vždy mění ve stanoveném intervalu, při kterém se ukládají na disk. Kamery fungují pouze v provozní dobu menzy, tj. 4,5 hodiny denně. Testovací data jsem sbíral ze všech kamer po dobu jednoho týdne.

5.1.1 Struktura programu

Program musí po spuštění nepřetržitě stahovat snímky z kamer. Ty ale běží pouze v pracovní dny a v provozní době menzy. Pozoroval jsem, že často některá kamera neběží a neposkytuje potřebný obraz nebo není dostupný na webových stránkách z důvodu výpadku internetového připojení.

Celý program je založen na systémovém čase (třída `Hodiny.java`). Program kontroluje čas a během otevírací doby stahuje (`StahniObrazek.java`) data z kamer. Stahování probíhá v intervalu změny snímků kamer. Mimo provozní dobu menzy program pouze čeká na její otevření.

Aby neprobíhalo zbytečně stahování dat i přes víkendy, program kontroluje systémový datum a získává data pouze v pracovní dny.

Před samotným stahováním snímků se provádí test dostupnosti URL (`ExistenceURL.java`). To řeší problémy výpadků stahování, které mohou vzniknout nedostupností webových stránek, nefunkčního internetového připojení nebo nedostupnosti některé z kamer. V tomto případě se program stále snaží v intervalu obnovy snímků stahovat další data.

Pokud jsou splněny všechny podmínky v programu, tj. menza je otevřená, je pracovní den a jsou dostupná data z kamer, dojde k jejich uložení na disk (`UlozObrazek.java`).

V uživatelském rozhraní (`OknoOvladani.java`) je uživatel informován o aktuálním čase a datu (obr. 5.1). Zobrazuje se dostupnost kamer a informace, zda je aktuálně menza otevřena.

Paměťové nároky

Velikost jednoho uloženého snímku se pohybuje od *30* do *50 kB*. Pro snímky jednoho dne ze tří kamer ($3 * 1607 = 4821$) je potřebné místo na disku kolem *240 MB*.

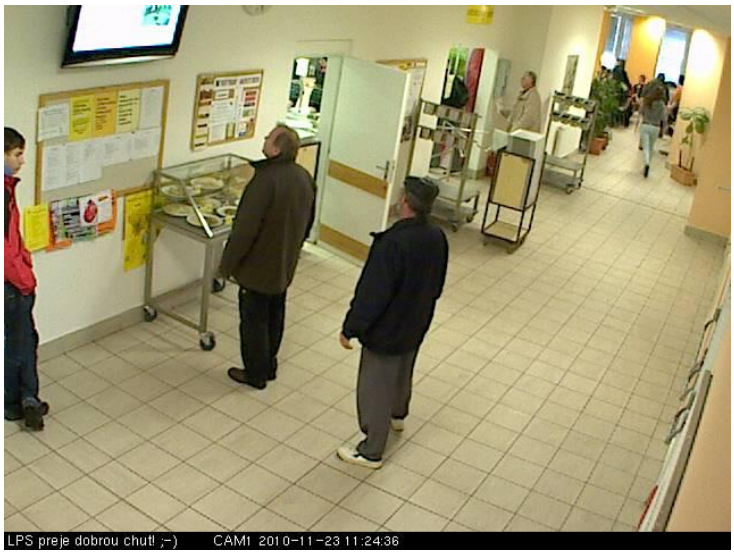
Dalším úkolem bylo vhodně navrhnout algoritmus, který na získaných snímcích identifikuje osoby. K tomu je potřeba jednotlivé snímky předem předzpracovat. Jelikož zpracování obrazu je závislé na mnoha faktorech, např. osvětlení snímané scény, šumu v obraze, bylo nutné navrhnout takový algoritmus, který by co možná nejvíce eliminoval tyto nežádoucí vlastnosti.

5.2 Hledání objektů

Prvním krokem při hledání lidských postav je oddělení pozadí od popředí snímku, tj. lidských postav. Z toho plyne, že je vhodné začít upravovat snímky metodou prahování (třída `Prahovani.java`), která přesně toto splňuje. Aby bylo možné použít prahování, je nejdříve nutné převést barevný snímek na šedotónový (`PrevodDoCB.java`).

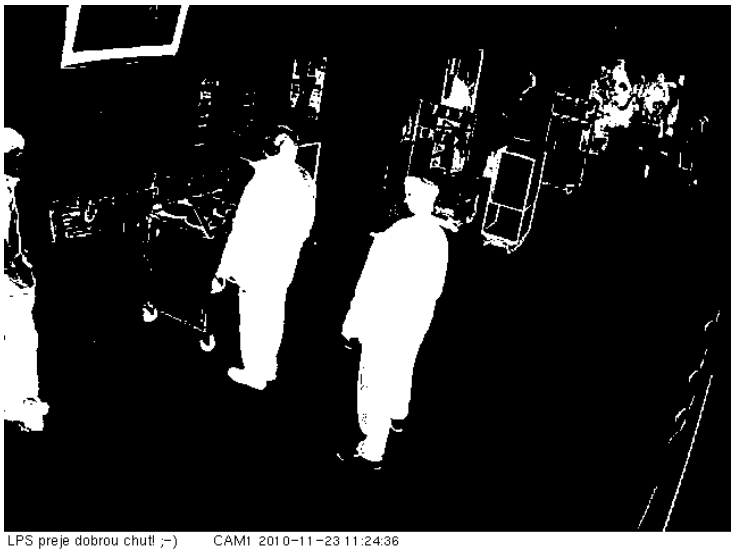
Ztratí se tak informace o barvě, ale to dalšímu zpracování nevadí, neboť použitý algoritmus je navržen tak, aby pracoval pouze s binárním obrazem. Hledané objekty (postavy) v obraze se odlišují od pozadí (jsou tmavší), tj. ideální pro využití metody prahování. Proto není nutné dále využívat informaci o barvě a stačí pouze binární obraz (obr. 5.3). Bílá – objekty, černá – pozadí.

Příklady předzpracování budu ukazovat na následujícím obr. 5.2 z první kamery, nicméně metody fungují i pro ostatní kamery.



Obrázek 5.2: Vzorový obrázek

Jak je vidět na obrázcích 5.2 a 5.3, došlo k oddělení pozadí od popředí a zůstaly zvýrazněny pouze objekty.



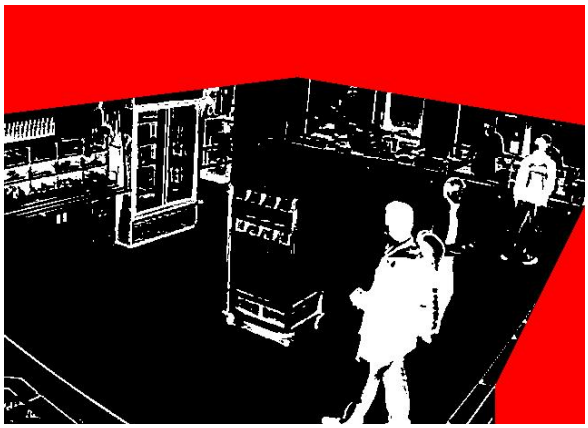
Obrázek 5.3: Obrázek 5.2 po prahování

Bohužel mezi objekty se nenalézají jen lidské postavy. Jsou zde např. různé stojany, část televizoru a spousta dalšího šumu.

Část objektů by se mohla odstranit vhodnější volbou prahu při prahování, ale musíme brát ohled na to, že nejsou všechny snímky stejně osvětleny. Tzn., že bychom se mohli v jednom snímku zbavit šumu a nežádoucích objektů, ale již v dalším snímku by se mohlo stát, že díky špatnému osvětlení ztratíme i některé lidské postavy. Proto je nutné zvolit hodnotu prahu takovou, aby prahování bylo efektivní pro všechny snímky a neztráceli jsme informace o lidských postavách. Vice informací o vhodné volbě prahu v kapitole 5.4.

Nežádoucích objektů se kompletně po prahování nezbavíme. Proto je nutné další zpracování obrazu. Uvážíme-li, že se postavy pohybují vždy jen v určité části snímku, je možné místa, kde se osoby nemohou vyskytnou, dále vynechat ze zpracování (`OriznutiObrazu.java`). Pro každou kameru se jedná o jiná místa, viz následující ukázky obr. 5.4, 5.5, 5.6. Místa jsou označena červenou barvou. V této části programu se odstraní i lišta v dolní části snímku, informující o aktuálním datu a čase.

Obrázek 5.4: Oříznutí 1.kamera



Obrázek 5.5: Oříznutí 2.kamera

Obrázek 5.6: Oříznutí 3.kamera

Odstranění těchto oblastí eliminuje část nežádoucích objektů v obraze a tím se usnadní identifikace postav.

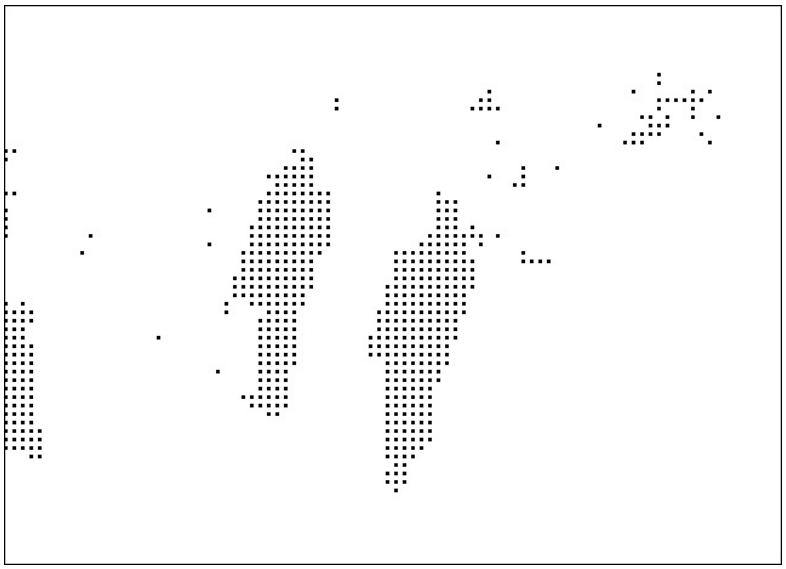
5.3 Identifikace nalezených objektů

Aktuálně jsou k dispozici snímky z kamer, které jsou upraveny tak, že by měly obsahovat pouze požadované objekty, tj. lidské postavy. Jak je vidět např. na obr. 5.4, jsou na snímku vidět i další objekty, které jsou nadbytečné.

Proto je nutné navrhnout takový algoritmus, který je schopen rozeznat lidskou postavu od ostatních objektů v obraze. Uvážíme-li, že lidská postava se ve snímcích jeví jako plný, větší, podlouhlý a vysoký objekt, usnadní se tím jeho odlišení od ostatních. Bohužel objekty v zadních částí snímaného prostoru se jeví jako menší, proto bude obtížnější je identifikovat, viz pravý horní roh obr. 5.4.

Aby bylo možné splnit zadání práce, je potřeba dostat ze snímku údaj o počtu vyskytujících se osob. Nestačí tedy jen vizuální informace. Proto je celý snímek prohledáván a hledají se objekty již zmiňovaných vlastností (souvislá, vysoká oblast), (`OznacPostavu.java`). Prohledávání snímku je řešeno dvěma cykly *for*. Snímek je tedy matice bodů, která se prohledává [1]. Aby procházení snímku netrvalo příliš dlouho, zvolil jsem zpracovávání pixelů ze snímku s určitým rozestupem. Tím myslím, že nezpracovávám pixel po pixelu a řádku po řádce, ale vždy každý *x-tý* pixel a každou *x-tou* řádku (viz obr. 5.7). Tím se částečně eliminují i chyby oblastí, které nemusí být nutně v celé řádce souvislé, např. při nevhodné volbě hodnoty prahu, může vzniknou v souvislé oblasti prázdné místo.

Body na obr. 5.7 vznikly tak, že při prohledávání snímku po řádcích se nalezla v řadě souvislá oblast bodů. Geometricky si to lze představit, že v řádce je úsečka. Úsečka vznikne tak, že vezmu pixel v obraze a jeho následujících x pixelů, na základě zvoleného rozlišení.



Obrázek 5.7: Rozlišení s hodnotou 7 pro obr. 5.2

Pokud tyto pixely tvoří plnou čáru, na místě prvního pixelu vytvořím bod. Aby se eliminovaly chyby po prahování, je zde možnost tolerovat určitou chybu, kdy nalezená úsečka může být přerušena. Více v kapitole 5.4.

Tímto postupem nalezneme v obraze body, jejichž počet v řádce symbolizuje šířku objektu. Na obr. 5.7 je stále vidět přítomný šum, tj. osamocené body nebo malé skupinky bodů. Tyto body se odstraní velice snadno a to tak, že pokud se najde ve snímku jakýkoliv bod, který nemá alespoň jeden sousední bod, je smazán (`OznacPostavu.java`).

Dalším krokem je zjištění pro každou skupinu bodů, zda se opravdu jedná o lidskou postavu, nikoli např. chladničku apod. Zjistí se šířka a výška každé skupiny podle počtu bodů v řádce a ve sloupci. Pokud zjištěné hodnoty odpovídají nastaveným parametrům pro postavu (kapitola 5.4), je oblast prohlášena za postavu. Během zjišťování šířky a výšky oblasti se zjišťují krajní body oblasti, které v případě prohlášení oblasti za postavu slouží k vizuálnímu označení oblasti do původního snímku (obr. 5.8, 5.9, 5.14).

Obrázek 5.8: Označení postav: 1.kamera

Obrázek 5.9: Označení postav: 2.kamera

5.3.1 Překryv postav

Přesnou identifikaci jednotlivých osob ztěžuje velmi častý překryv postav. Doposud popsaný algoritmus identifikuje shluk postav pouze jako jedinou, proto je třeba ho dále upravit.

Využil jsem k tomu znalosti přibližné velikosti postav vzhledem k pozici na snímku. Tzn., že bližší postava ke kameře je na snímku větší, nežli postava v dáli. Objeví-li se v blízkosti kamery objekt velikosti, který neodpovídá předpokládaným rozměrům lidské postavy, buď'to širší nebo vyšší, bude tento objekt rozdělen na části, opět v závislosti na poloze od kamery (`OznacPostavu.java`). Rozdělení širšího objektu je vidět na obr. 5.10, kde rozdělení symbolizuje bílá svislá čára. Dělení vysokých objektů viz obr. 5.11.

Obrázek 5.10: Rozdělení širokého objektu

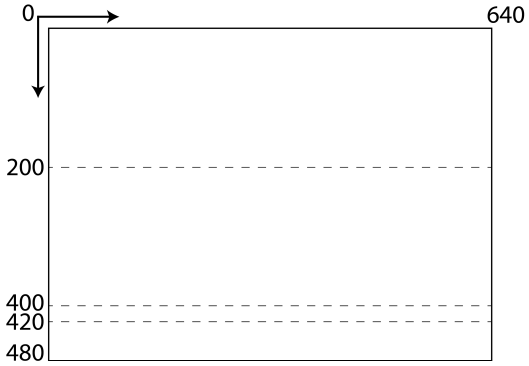
Obrázek 5.11: Rozdělení vysokého objektu

Pseudokód algoritmu pro rozdělování objektů:

```
IF 100px < siroky_objekt < 440px THEN
     IF spodek_postavy < 200px THEN
          pocet_lidi + 2
          rozdel obraz na 3 casti
     ELSE IF spodek_postavy < 400px THEN
          IF siroky_objekt > 200px THEN
               pocet_lidi + 2
               rozdel obraz na 3 casti
          ELSE
               pocet_lidi + 1
               rozdel obraz na 2 casti
          ENDIF
     ENDIF
ELSEIF siroky_objekt >= 440 THEN
     pocet_osob = plna_menza
ENDIF

IF spodek_postavy > 420px AND vysoky_objekt > 280px THEN
     pocet_osob + 1
     rozdel obraz na 2 casti
ENDIF
```

Zvolené hraniční hodnoty pixelů se jeví po prozkoušení různého nastavení jako rozumné. Pro lepší úspěšnost algoritmu by bylo vhodné rozdělit obraz do více částí. Rozdělení obrazu pro lepší představu na obr. 5.12.



Obrázek 5.12: Rozdělení obrazu na části

Plná menza

Problém je v případě plně zabraného prostoru strávníky. Vznikne jeden velký objekt, který lze jen těžko separovat na jednotlivé postavy. Proto se v tomto případě určí počet osob na snímku pevnou hodnotou a to konkrétně číslem *18*. Tuto hodnotu jsem zvolil po sledování množství zaplněných snímků a jeví se jako dobrý kompromis. Přes obrázek se nakreslí kříž, jako symbol plné menzy.

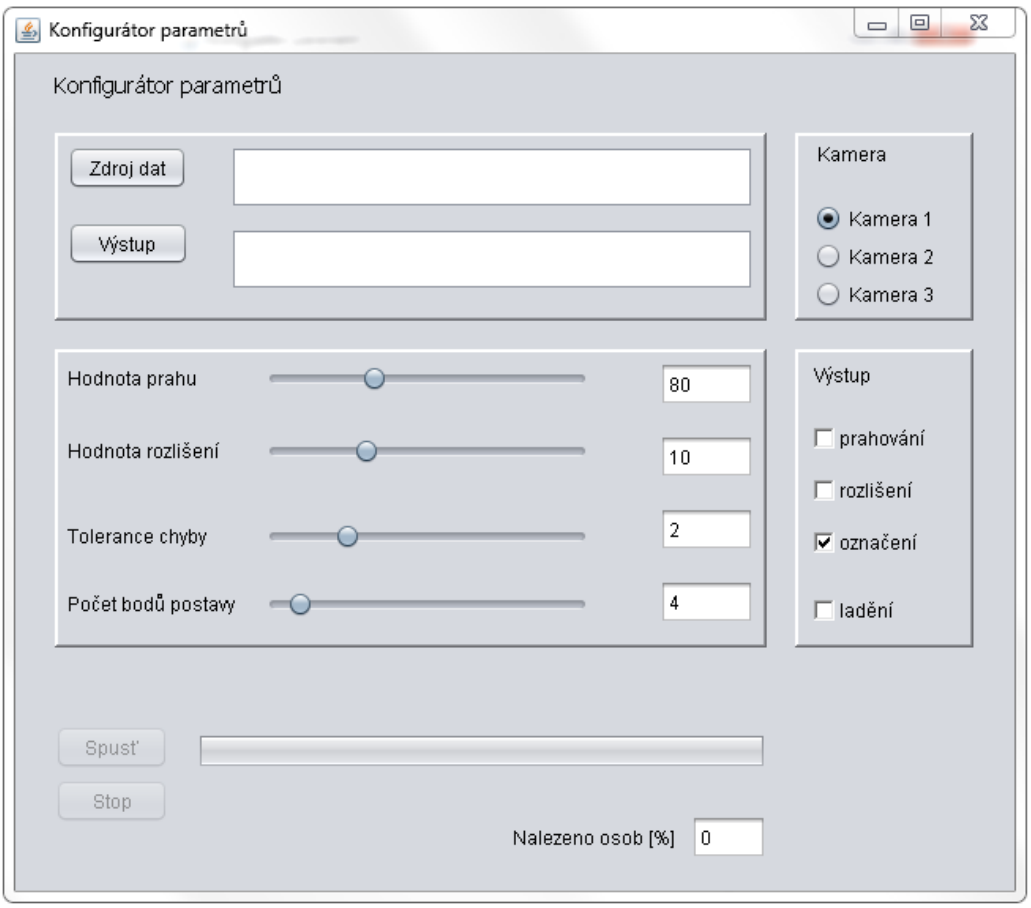
Bohužel snímky z kamer jsou v nízkém rozlišení, proto oblasti v zadních částech snímku je velice obtížné zpracovávat, proto může docházet k falešným detekcím lidských postav nebo k nenalezení vzdálených postav.

5.4 Ladění programu

Jelikož je algoritmus závislý na několika parametrech:

- Volba hodnoty prahu pro prahování obrazu
- Rozlišení pro prohledávání obrazu
- Volba minimální velikosti postavy

Vytvořil jsem aplikaci (`KonfigUI.java`) pro ladění těchto parametrů (obr. 5.13). Cílem je nalezení optimálního nastavení jednotlivých parametrů, aby byly výsledky identifikace lidských postav co možná nejlepší.



Obrázek 5.13: Konfigurátor parametrů

Aplikace je schopná nastavovat všechny potřebné parametry, které jsou použité při zpracování snímku. Pro rychlejší ladění algoritmu je nutné předem připravit snímky, na kterých je potřeba manuálně určit množství lidských postav.

Číslo, udávající počet postav na snímku se přidá za název snímku:

`hh-mm-ss_x.jpg`

- `hh-mm-ss` – formát názvu snímku (hodiny-minuty-sekundy)
- `x` – číslo udávající počet osob na snímku

Aplikace po dokončení zpracování všech vybraných dat vyhodnotí množství identifikovaných postav. To se počítá na základě správného počtu lidských postav na snímku a počtu postav nalezených programem. Hodnoty nastavených parametrů a výsledek hledání se ukládají do XML souboru *parametry.xml*, který nalezneme ve zvolené složce pro výstup. Více o XML v [3]. Tento výstup slouží k představě, s jakými parametry nastavení má program jakou úspěšnost. Pouze tato informace ale nestačí, je potřeba navíc provádět vizuální kontrolu na snímcích, zda nedochází k chybnému označování jiných objektů.

5.5 Časy výpočtu

Celkový počet snímků z jedné kamery za provozní dobu menzy je *1607*. Na těchto datech se provádí analýza průběhu jednoho dne chování strávníků v menze. Celkový čas výpočtu se pohybuje kolem *250* sekund, tj. kolem *4* minut . Testováno na *Intel Core i3 (2.53 GHz), 4GB RAM, Windows 7 Home Premium*. Na zpracování jednoho snímku tak připadá přibližně *156* milisekund.

5.6 Výsledky

Počet nalezených osob se počítá, jak je již napsáno v kap. 5.4, z připravených snímků, kde je předem určen přítomný počet lidí. Následující tab. 5.1, 5.2, 5.3 ukazují příklady výstupu programu při různém nastavení parametrů.

Práh	Rozlišení	Chyba	Postava	Poč. dat	Osob [%]
80	10	2	4	46	64
118	10	2	4	46	28
55	10	2	4	46	70
80	7	2	4	46	39
80	13	2	4	46	73
80	10	2	17	46	47
80	10	2	10	46	69
80	10	2	2	46	69

Tabulka 5.1: Nastavení parametrů, kamera č.1

Práh	Rozlišení	Chyba	Postava	Poč. dat	Osob [%]
80	10	2	4	54	42
118	10	2	4	54	45
55	10	2	4	54	71
80	7	2	4	54	24
80	13	2	4	54	62
80	10	2	17	54	42
80	10	2	10	54	63
80	10	2	2	54	38

Tabulka 5.2: Nastavení parametrů, kamera č.2

Hodnoty z tabulky nejsou moc vypovídající o úspěšnosti detekce. Proto je nutná vizuální kontrola označených snímků, zda nedochází s danými parametry k falešným detekcím a neoznačují se jiné objekty, než lidské postavy. Jako optimální nastavení (pro kameru č.1), které má dobrou úspěšnost a dochází k malému počtu falešných detekcí, je nastavení parametrů viz 1.řádka tab. 5.1.

Práh	Rozlišení	Chyba	Postava	Poč. dat	Osob [%]
80	10	2	4	42	38
118	10	2	4	42	49
55	10	2	4	42	49
64	11	2	5	42	33
70	13	2	5	42	31
70	11	2	5	42	21
64	11	2	10	42	51
64	11	2	2	42	17

Tabulka 5.3: Nastavení parametrů, kamera č.3

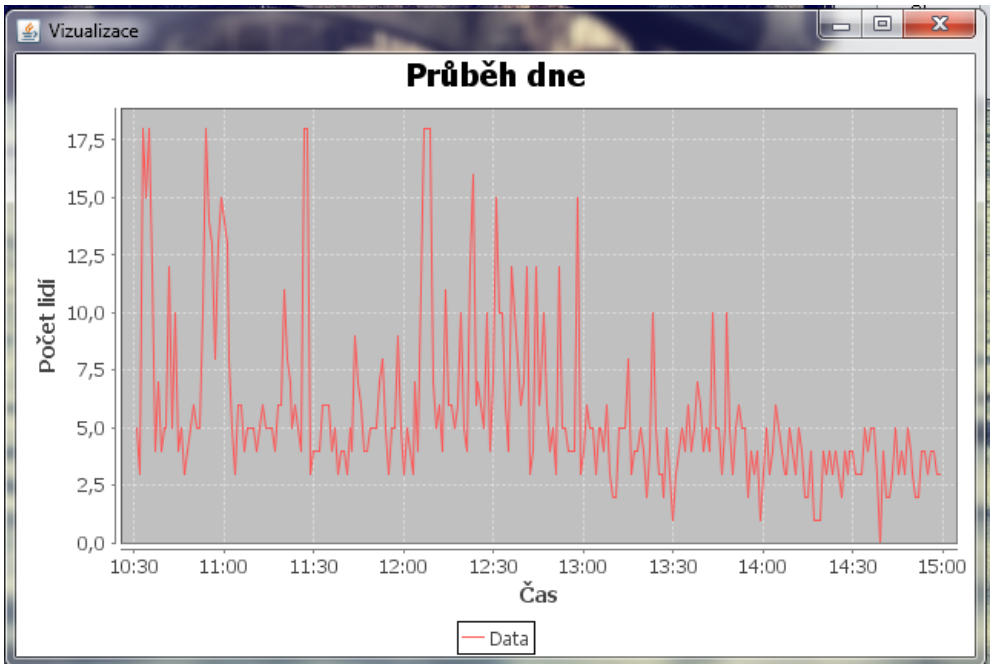
To samé se dá říci i pro druhou kameru, kde testování probíhalo stejným způsobem a konečné nastavení parametrů se shoduje s parametry první kamery. Pouze pro třetí kameru je potřeba nastavit parametry na hodnoty 4.řádky tab. 5.3 a to z důvodu přítomných stojanů a mrazáku v obraze (obr. 5.14), které by s parametry prvních dvou kamer způsobovaly časté falešné detekce.

Obrázek 5.14: Označení postav: 3.kamera

Hodnoty z tabulek 5.1, 5.2, 5.3 jsou ale mírně zavádějící, protože pro ladění algoritmu byly vybrány vhodné snímky, na kterých docházelo k malému překryvu osob, aby bylo dobře vidět, jak algoritmus pracuje a šel lépe odladit. Proto jsou výsledná čísla pro všechny druhy snímků nižší, než uvádí tabulka.

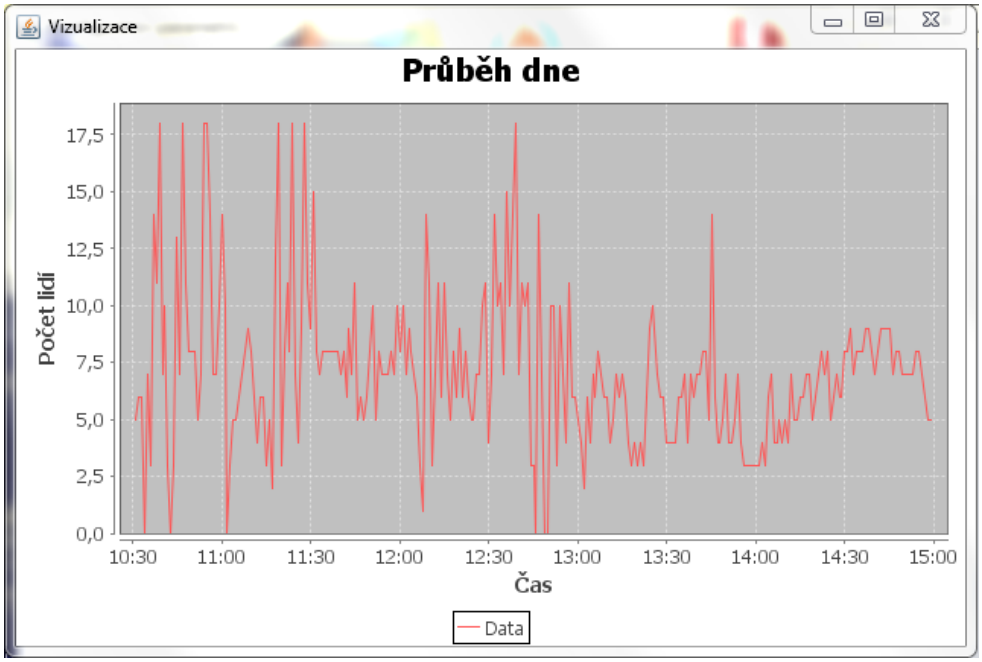
5.7 Vizualizace výsledku

Program v průběhu analýzy jednotlivých snímků ukládá hodnoty o počtu nalezených osob do grafu (`Vizualizace.java`). Je zde použita knihovna *JFreeChart*, která vytváří a zobrazuje grafy. Více o knihovně [4]. Příklad průběhu jednoho dne viz obr. pro jednotlivé kamery 5.15, 5.16, 5.17.

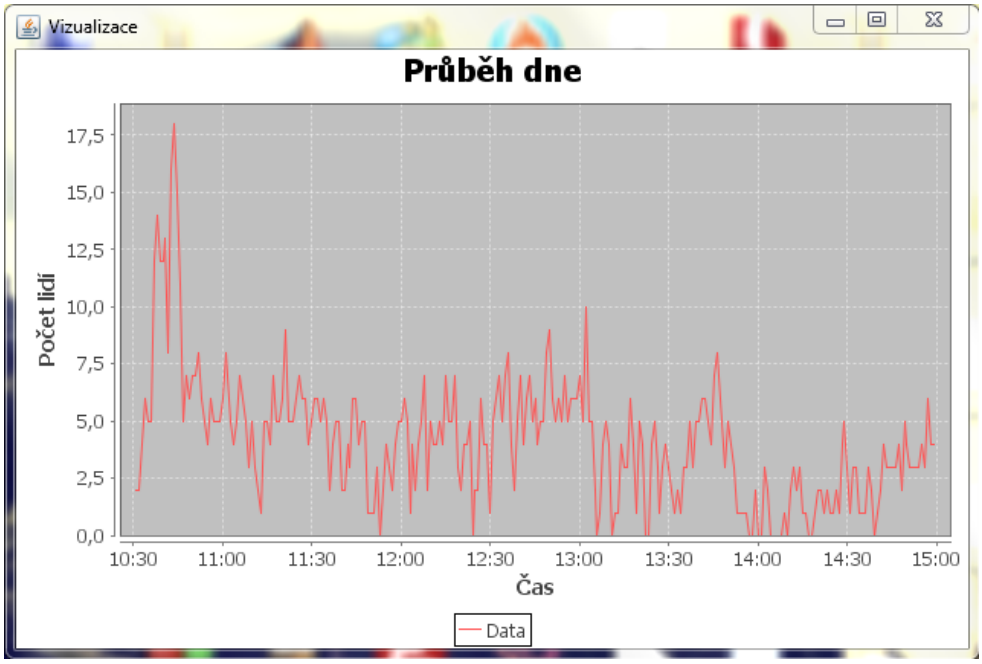


Obrázek 5.15: Vizualizace průběhu dne, 1.kamera

Z výstupu je vidět, že v absolutní většině případů je vyšší návštěvnost z počátku otevření menzy. To samé platí o časech, kdy končí vyučovací hodiny a studenti jdou do menzy na oběd. Ke konci otevírací doby menzy návštěvnost klesá a nedosahuje již takových extrémů, jako v dřívějších časech.



Obrázek 5.16: Vizualizace průběhu dne, 2.kamera



Obrázek 5.17: Vizualizace průběhu dne, 3.kamera

5.8 Závěr

Cílem práce bylo vytvořit si představu o chování strávníků v prostorách univerzitní menzy v průběhu dne. Vytvořený program je schopen vizualizovat průběhy jednotlivých dnů formou grafu, ze kterého lze vidět, jak provoz menzy probíhá.

Program označuje v každém snímku nalezené osoby. Z toho lze snadno vidět, jak je algoritmus úspěšný. Pokud uživatel vhodně zvolí jeho parametry, je identifikace lidských postav velmi úspěšná pro oddělené objekty (lidské postavy). Z důvodu nižšího rozlišení dostupných snímků z kamer, je možnost identifikace vzdálených postav složitá, a proto se stává, že postavy nejsou nalezeny nebo naopak jsou nalezeny jiné objekty a prohlášeny za lidskou postavu.

Vzájemné překryvy postav na snímku se program snaží částečně eliminovat na základě velikosti objektu a jeho pozici na snímků (vůči vzdálenosti od kamery). Díky tomu může z jednoho nalezeného objektu identifikovat více postav. Pochopitelně postavy, které stojí z velké části své plochy za sebou, není možné rozeznat a program je identifikuje jako jedinou. V případě zaplnění celého prostoru větším počtem osob a jejich překryvu, vznikne jeden velký objekt a program prohlásí menzu za plně obsazenou a o konkrétní zjišťování počtu lidských postav se nesnaží, neboť by to nebylo s tímto algoritmem možné.

Program je schopen odhadovat množství lidí v univerzitní menze. Zatím není schopen provádět predikci chování strávníků na základě již získaných dat. To je jedna z věcí, kterou bych při pokračování v této práci dokončil. Jako další zlepšení by bylo dobré lépe navrhnout identifikaci překryvu osob nebo alespoň provést lepší odladění stávajícího způsobu identifikace na základě vzdálenosti postav od kamer.

Seznam obrázků

Seznam tabulek

Literatura

[1] Bogdan Kiszka, *1001 tipů a triků pro jazyk JAVA*, Computer press, 2009.

[2] Jiří Lažanský *Umělá inteligence*, Academia, 2001.

[3] Pavel Herout, *Java a XML*, Koop, 2007, ISBN: 978-80-7262-307-4.

[4] David Gilbert, *JFreeChart*, 2011, [cit. 6.5.2012]. Dostupné z WWW: `http://www.jfree.org/jfreechart`.

[5] Ondřej Neff, *Co je to šum*, 2010, [cit. 6.5.2012]. Dostupné z WWW: `http://www.digineff.cz/cojeto/ruzne/sum.html`.

[6] IDL Online Help, *Interpolation methods*, 2007, [cit. 6.5.2012]. Dostupné z WWW: `http://idlastro.gsfc.nasa.gov/idl\_html\_help/Interpolation\_Methods.html`.

[7] Eric Weisstein, *Euclidean Metric*, 2012, [cit. 6.5.2012]. Dostupné z WWW: `http://mathworld.wolfram.com/EuclideanMetric.html`.

[8] Rubiks_cube_by_keqs.jpg, [cit. 6.5.2012]. Dostupné z WWW: `http://upload.wikimedia.org/wikipedia/commons/b/bb/Rubiks_cube_by_keqs.jpg`.

[9] socha-svobody-3.jpg, [cit. 6.5.2012]. Dostupné z WWW: `http://www.usa24.cz/wp-content/gallery/socha-svobody/socha-svobody-3.jpg`.

A Seznam příloh

A.1 Nápověda k aplikaci pro stahování dat

A.2 Nápověda k aplikaci pro nastavování parametrů

A.3 Nápověda pro spuštění programu

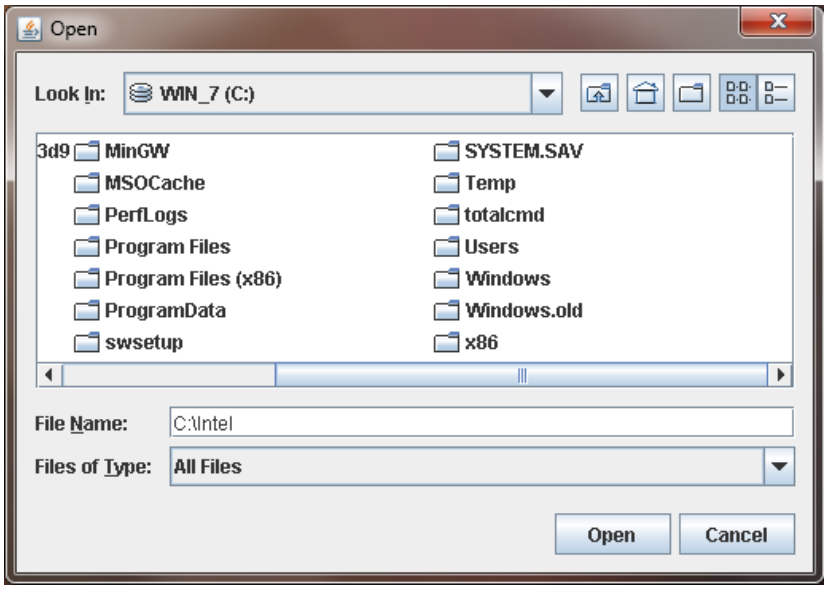
A.1 Nápověda k aplikaci pro stahování dat

Přeložení programu

Program je napsán v jazyce JAVA s využitím vývojového prostředí *Eclipse*. Obsahuje soubor *build.xml*, který zdrojové soubory přeloží a je umístěn ve složce projektu programu. Přeložení se provede příkazem `ant distjar` v příkazovém řádku (je nutné mít nainstalován *Ant*, potřebný pro překlad) nebo se soubor spustí přímo v prostředí *Eclipse*. Přeložením vznikne spustitelný soubor s názvem *StahovaniDat.jar* v podadresáři *jar*. Pro úspěšný překlad je nutné, aby byla zachována adresářová struktura projektu.

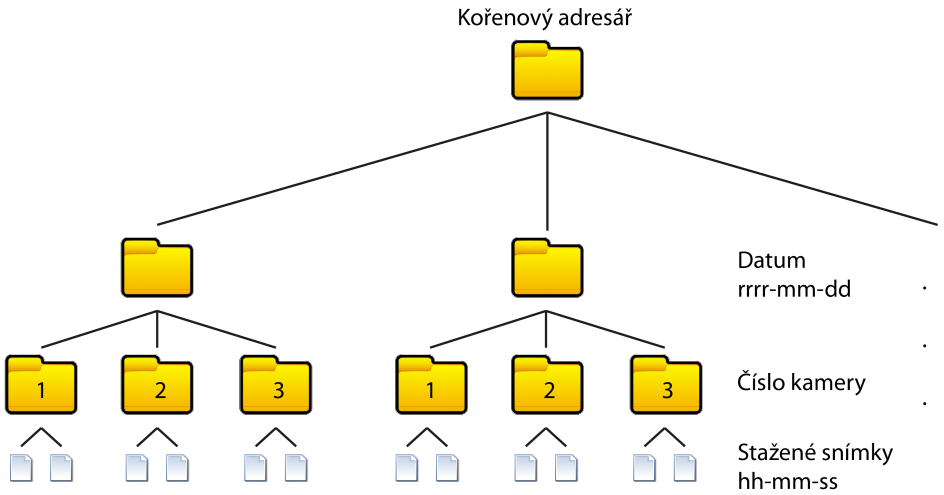
Spuštění programu

Aplikaci pro stahování dat spustíme souborem *StahovaniDat.jar*. V okně (obr. 5.1) je aktivní pouze jedno tlačítko *Vyber cestu*. Tlačítkem vybereme cestu, kam se budou ukládat stahované snímky z kamer (obr. 5.18).



Obrázek 5.18: Výběr cesty pro ukládání

Po vybrání cesty je aktivní tlačítko *Spust*, které spustí stahování dat do zvoleného adresáře s následující strukturou (obr. 5.19):



Obrázek 5.19: Adresářová struktura pro ukládání dat

- *rrrr–mm–dd* – formát: rok–měsíc–den
- *hh–mm–ss* – formát: hodina–minuta–sekunda

A.2 Nápověda k aplikaci pro nastavování parametrů

Přeložení programu

Program je napsán v jazyce JAVA s využitím vývojového prostředí *NetBeans*. Obsahuje soubor *build.xml*, který zdrojové soubory přeloží a je umístěn ve složce projektu programu. (Využívá souboru *nbproject/build-impl.xml*, bez něhož překlad neproběhne.) Přeložení se provede příkazem `ant jar` v příkazovém řádku, (je nutné mít nainstalován *Ant*, potřebný pro překlad). Spustitelný soubor je také možno automaticky generovat z *NetBeans*. Přeložením programu vznikne spustitelný soubor s názvem *ZpracovaniDat.jar* v podadresáři *dist*. Pro úspěšný překlad je nutné, aby byla zachována adresářová struktura projektu.

Spuštění programu

Aplikaci pro nastavení parametrů spustíme souborem *ZpracovaniDat.jar*. Otevře se okno (obr.5.13), kde je několik možností nastavení:

- Tlačítko *Zdroj dat* – cesta ke zpracovávaným snímkům
- Tlačítko *Výstup* – cesta k uložení výsledků programu
- Tlačítko *Spust'* – spustí zpracovávání snímků
- Tlačítko *Stop* – okamžité ukončení zpracovávání
- Přepínač *Kamera* – data z jaké kamery jsou využívána
- Posuvník *Hodnota prahu* – nastavení hodnoty prahu pro prahování
- Posuvník *Hodnota rozlišení* – nastavení rozlišení při hledání objektů

- Posuvník *Tolerance chyby* – kolikrát může být přerušen hledaný objekt
- Posuvník *Počet bodů postavy* – minimální velikost postavy
- Přepínač *Prahování* – uložení prahovaných snímků
- Přepínač *Rozlišení* – uložení snímků po hledání objektů
- Přepínač *Označení* – uložení snímků s označenými osobami
- Přepínač *Ladění* – pro ladění algoritmu, nutné mít připraveny snímky a znát počet postav

Ukazatel ve spodní části okna ukazuje aktuální množství zpracovaných dat.

A.3 Nápověda pro spuštění programu

Program spustíme souborem *ZpracovaniDat.jar*. Nastavení parametrů je již vhodně přednastaveno, proto je potřeba pouze vybrat cestu k datům, výstupní cestu a spustit program tlačítkem *Spusť*