VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

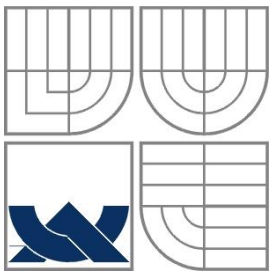
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

AUTOMATICKÉ POČÍTÁNÍ LIDÍ Z PANORAMATICKÉ FOTOGRAFIE

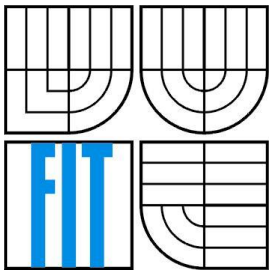
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE ONDŘEJ BLUCHA
AUTHOR

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

AUTOMATICKÉ POČÍTÁNÍ LIDÍ Z PANORAMATICKÉ FOTOGRAFIE

AUTOMATIC PEOPLE COUNTING FROM PANORAMIC PHOTOGRAPHY

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

ONDŘEJ BLUCHA

VEDOUCÍ PRÁCE
SUPERVISOR

ING. MARTIN VEĽAS

BRNO 2014

Abstrakt

Tato bakalářská práce se zabývá automatickým počítáním lidí z panoramatické fotografie. Toto je velmi užitečné pro počítání velkého počtu lidí, například na stadionech nebo na koncertech. Skládá se ze dvou částí. První částí je spojování fotografií, ke které byly použity příznakově založené metody. Druhou částí je počítáni osob pomocí detekce obličejů, ke které byl použit detektor Viola-Jones. Pomocí testování bylo zvoleno ideální nastavení parametrů použitých metod pro daný problém.

Klíčová slova

panorama, spojování fotografií, detekce obličejů, Viola-Jones, kaskáda klasifikátorů, AdaBoost, lokální příznaky, RANSAC, promítání na válec, blending

Abstract

This bachelor thesis deals with automatic people counting from panoramic photography. This is very useful for counting large number of people, such as on the stadium or on the concerts. It consists of the two parts. The first one is image stitching, which process the images by the feature-based methods. The second part is people counting using face detection, where were used Viola-Jones detector. The ideal setting of parameters for used methods was experimentally selected.

Keywords

panorama, photo stitching, face detection, Viola-Jones, cascade of classifiers, AdaBoost, local features, RANSAC, cylindrical projection, blending

Citace

Blucha Ondřej: Automatické počítání lidí z panoramatické fotografie, bakalářská práce, Brno, FIT VUT v Brně, 2014

Automatické počítání lidí z panoramatické fotografie

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Martina Veľase. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Ondřej Blucha
19. května 2014

Poděkování

Rád bych poděkoval svému vedoucímu bakalářské práce Ing. Martinu Vaľasovi za odbornou pomoc, cenné rady a věnovaný čas při řešení mé bakalářské práce.

Obsah

1 Úvod

V dnešní době je počítačové vidění na velkém vzestupu. Vzniká stále více aplikací, které jsou na něm založené. Tyto aplikace jsou implementovány do všemožné elektroniky od stolních počítačů přes mobilní telefony až po různé senzory v automobilech. Vývoj nových algoritmů se také ještě nezastavil. I přestože vzniká stále více aplikací, ne vždy spojují více různorodých činností dohromady. Problém například nastává, když potřebujeme určit počet lidí, které máme vyfotografované na více fotografiích. Pokud bychom měli pouze jednoduchou aplikaci na počítání lidí, pak bychom museli na různých fotografiích hledat stejné osoby a ty potom z celkového počtu osob odečíst. Tento problém jsem se rozhodl řešit pomocí této bakalářské práce, jejíž součástí bude i aplikace. Tato aplikace bude fungovat tak, že nejprve spojí několik fotografií dohromady do jedné panoramatické a pak z ní spočte osoby.

Využití takovéto aplikace je široké. Například může sloužit pro automatické určování počtu lidí na stadionech, na koncertech nebo kdekoliv jinde, kde je potřeba určit počet osob, ale všichni se nám do jednoho snímku nevejdou nebo by kvalita vyfotografovaných osob na fotografii byla nedostačující. Samozřejmě aplikaci půjde využít pouze pro detekci osob nebo pouze pro spojování fotografií.

Práce se skládá ze dvou základních problémů - spojování fotografií a detekce osob. Ke spojování fotografií použiji příznakově založené metody. Pro detekci osob nastuduji dva detektory, prvním bude Viola-Jones a druhým bude detektor založený na histogramech orientovaných gradientů (HOG). Na základě teoretické části vyberu vhodnější detektor a ten posléze naimplementuji. Dále se pokusím najít ideální nastavení parametrů použitých metod.

Technická správa obsahuje celkem 8 kapitol. V následujících dvou kapitolách uvedu teorii k detekci osob a spojování fotografií a budou popsány všechny části z již zmíněných metod. V další kapitole vzpomenu nejmodernější technologie na poli detekce osob a spojování fotografií. V páté kapitole bude zmíněn návrh aplikace. Následující kapitola bude popisovat implementaci programové části práce. V předposlední kapitole tuto aplikaci otestuji a na základě výsledků provedu vyhodnocení. V poslední kapitole provedu shrnutí celé práce.

2 Detekce osob

V této kapitole se budu zabývat detekcí osob, zejména pak jejich tváří. Pro detekci osob existuje několik možných detektorů. V následujících podkapitolách budou popsány zřejmě dva nejznámější detektory, a to Viola-Jones (2.1) a detektor založený na histogramech orientovaných gradientů (HOG) (2.2).

2.1 Viola-Jones

V roce 2001 Paul Viola a Michael J. Jones (1) představili detektor obličejů, který byl schopen zpracovat obrázky velmi rychle a zároveň dosahoval vysoké úspěšnosti detekce. Mezi jeho další pozitivní vlastnosti patří značná nezávislost na osvětlení a velikosti sledovaného objektu.

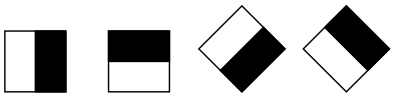
Vychází ze tří hlavních poznatků. Zaprvé byla zavedena nová reprezentace obrazu zvaná integrální obraz (2.1.2), který způsobuje, že jsou příznaky v detektoru počítány velmi rychle. Druhým je jednoduchý a efektivní klasifikátor, který používá algoritmus AdaBoost (2.1.4), který vybere malý počet kritických příznaků z velkého množství potenciálních příznaků. Třetí je kaskáda klasifikátorů (2.1.3), která umožňuje rychle zahodit výřezy s pozadím, zatímco více času stráví na výpočtech výřezů, kde je potenciální obličej. Tento detektor pracuje s obrázky ve stupních šedi. (1) (2)

2.1.1 Haarovy příznaky

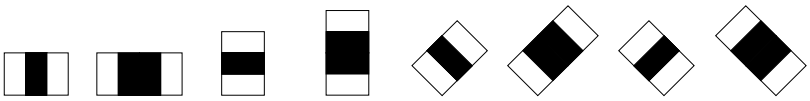
Při detekci obličeje se klasifikují obrázky na základě hodnoty jednoduchého příznaku. Použití příznaků je mnohem efektivnější a rychlejší než používat přímo pixely. Snahou detektoru Viola-Jones je získat velkou řadu jednoduchých příznaků s minimálními výpočetními nároky. Takovým typem příznaků jsou příznaky založené na principu podobnému definici Haarových příznaků (tzv. Haar-like features). Tyto příznaky používají změny kontrastu hodnot mezi sousedními obdélníkovými skupinami pixelů. Rozdíl kontrastu mezi skupinami pixelů je použit k určení relativně světlých a tmavých míst.

Používáme tři typy příznaků. Pokud spolu sousedí dva obdélníky, jedná se o hranový příznak (viz obrázek 2.1) a hodnota příznaku se počítá jako rozdíl mezi oběma obdélníkovými oblastmi. Pokud spolu sousedí tří obdélníky, jedná se o čárový příznak (viz obrázek 2.2) a hodnota příznaku se počítá jako součet vnějších obdélníků, od kterých se odečte vnitřní obdélník. Poslední používanou možností je, že jeden obdélník je obsažen ve druhém. Pak se jedna o příznaky středové (viz obrázek 2.3) a hodnota příznaku se počítá jako rozdíl vnějšího a vnitřního obdélníku.

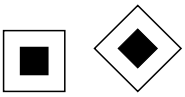
Jednotlivé příznaky jsou použity na celý vstupní obraz, přičemž zároveň dochází ke změně velikostí jednotlivých příznaků (tj. velikosti jednotlivých obdélníků) z velikosti 1×1 až na velikost odpovídající vstupnímu obrazu. Za předpokladu, že základní rozlišení detektoru je 24×24, dostaneme přibližně 160 000 hodnot příznaků. (1) (2) (3)



Obrázek 2.1: Hranové příznaky (4)



Obrázek 2.2: Čárové příznaky (4)



Obrázek 2.3: Středové příznaky (4)

2.1.2 Integrální obraz

Pro výpočet Haarových příznaků z běžného obrazu, bychom museli pro každý obdélníkový příznak počítat sumu intenzit pixelů daného příznaku. Proto byl zaveden tzv. integrální obraz, pomocí kterého lze spočítat obdélníkové příznaky velmi rychle. Každý bod tohoto obrazu je počítán jako součet všech bodů, které se nacházejí nad a zároveň nalevo od zadaného bodu. To nám definuje vzorec:

$$i_{int}(x,y) = \sum_{x'\leq x, y'\leq y} i(x',y') \tag{2.1}$$

kde $i_{int}(x,y)$ je hodnota integrálního obrazu a $i(x',y')$ je hodnota původního obrazu.

Výpočet hodnot integrálního obrazu se provádí podle následujících vzorců:

$$s(x,y) = s(x,y-1) + i\,(x,y) \tag{2.2}$$

$$i_{int}(x,y) = i_{int}(x-1,y) + s(x,y) \tag{2.3}$$

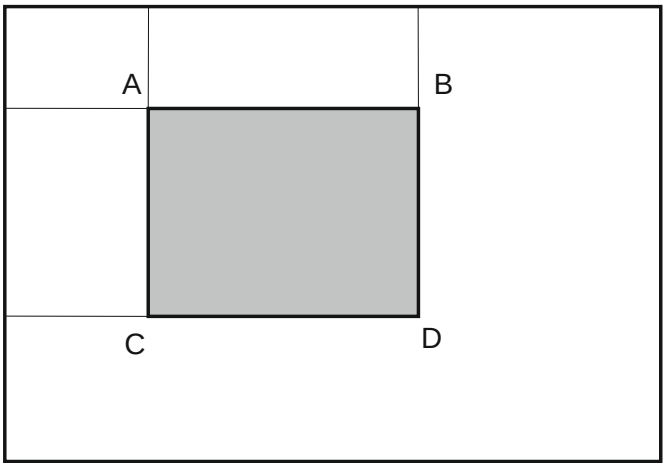
kde $s(x,y)$ je kumulovaný součet hodnot v řádku obrazu, za podmínek, že $s(x,-1) = 0$ a $i_{int}(-1,y) = 0$.

Celý integrální obraz lze spočítat pouze jedním průchodem originálního obrazu.

Při použití integrálního obrazu může být každý obdélník počítán pouze pomocí čtyř referencí do paměti, což nám umožní počítat libovolně velký obdélník konstantní rychlostí. Pokud máme příznak o dvou sousedních obdélnících, stačí nám jen 6 referencí do paměti a pokud máme příznak o třech sousedních obdélnících, stačí nám jen 8 referencí do paměti. (1) (2) (3)

Hodnota obdélníkového příznaku S z obrázku 2.4 bude vypočítána pomocí vzorce:

$$S = i_{int}(A) - i_{int}(B) - i_{int}(C) + i_{int}(D). \tag{2.4}$$



Obrázek 2.4: Ukázka výpočtu hodnot obdélníkových příznaku z integrálního obrazu (1)

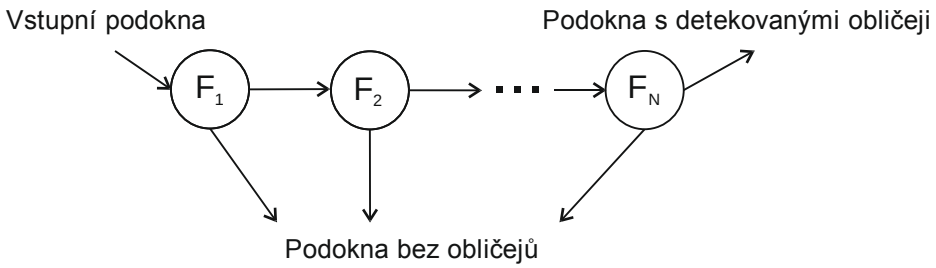
V předchozí části jsme si ukázali, jak se pomocí integrálního obrazu vypočtou obdélníkové příznaky, které jsou rovnoběžné s osou x. Často pracujeme s příznaky, které nejsou rovnoběžné s osou x. V praxi se nejčastěji používají příznaky natočené o 45°, jak můžeme vidět na obrázcích 2.1, 2.2 a 2.3. Při použití takovýchto příznaků potřebujeme tzv. rotovaný integrální obraz. Rotovaný integrální obraz je spočítán pomocí dvou průchodů obrazem. První průchod je prováděn zleva doprava a shora dolů a druhý průchod je prováděn zprava doleva a zdola nahoru. Pro výpočet obdélníkového příznaku nám opět stačí jen čtyři reference do paměti. (4)

2.1.3 Kaskáda klasifikátorů

Snahou detektoru je snížení průměrné doby, kterou detektor stráví prohledáváním každého podokna. Toho bylo dosaženo kaskádovým zapojením klasifikátorů. Bylo využito poznatku, že podoken, která neobsahují hledaný objekt, je mnohem více než podoken, která obsahují hledaný objekt. Proto jsou podokna bez hledaného objektu detekována rychleji.

Kaskáda klasifikátorů je degenerovaný binární rozhodovací strom. V každém stupni kaskády se rozhoduje, zda dané podokno obsahuje hledaný objekt nebo ne. Jsou tak detekovány téměř všechny podokna s hledanými objekty, zatímco zamítnutá je pouze část podoken bez hledaného objektu. U zamítnutého podokna je ukončena klasifikace. Podokno, které není zamítnuté, postupuje do dalšího stupně kaskády. To se opakuje tak dlouho, až se dojde k poslednímu stupni kaskády. Ten, když úspěšně projde, tak je dané podokno označené za podokno s hledaným objektem. Jedná se o sekvenční průchod. Kaskáda klasifikátorů je zobrazena na obrázku 2.5.

Každý stupeň kaskády je složen ze slabých klasifikátorů. Tyto slabé klasifikátory nedokážou spolehlivě detekovat hledaný objekt, ten dokáže detekovat až silný klasifikátor složený ze slabých klasifikátorů. Každý stupeň kaskády obsahuje více klasifikátorů než předchozí stupeň, čili dokáže detekovat lépe. (1) (2) (3)



Obrázek 2.5: Kaskáda klasifikátorů (1)

2.1.4 AdaBoost

Aby bylo možné detekovat objekty, musí být nejprve natrénován kaskádový klasifikátor. Pro podokno o velikosti 24×24 pixelů se vypočítá 160 000 Haarových příznaků, což je mnohem více než počet pixelů. I přesto, že každý příznak může být vypočten velmi efektivně, výpočet všech příznaků je velmi náročný. Experimentováním bylo dokázáno, že jen s pomocí velmi malého počtu Haarových příznaků může být vytvořen efektivní klasifikátor (1). Kvalitní klasifikátor by měl tyto příznaky najít.

Detektor Viola-Jones používá jako klasifikační algoritmus AdaBoost (Adaptive Boosting), který v roce 1995 představili Y. Freund a R. Schapire a který vychází z metody strojového učení zvané boosting. Používá se jak na výběr příznaků, tak na trénování klasifikátorů. Základním principem je lineární kombinace několika slabých klasifikátorů, někdy též označovaných jako slabí žáci, do jednoho silného klasifikátoru (silný žák), který je úspěšnější než jakýkoliv ze slabých klasifikátorů. Jedinou podmínkou slabého klasifikátoru je, aby jeho chyba byla menší než 50%. AdaBoost se od základního boostingu liší tím, že po každém kole učení je vypočtena nová váha slabého klasifikátoru podle velikosti jeho chyby.

K natrénování klasifikátoru potřebujeme jako vstup dvě sady obrázků. První sadou jsou pozitivní obrázky čili obrázky s hledaným objektem a druhou sadou jsou negativní obrázky čili obrázky, na kterých se nenachází hledaný objekt.

Na začátku jsou všechny váhy D_t nastaveny rovnoměrně. V každé smyčce algoritmu se pak vybere slabý klasifikátor s nejmenší váženou chybou klasifikace ε_t při daných váhách D_t. Musí se zkontrolovat, jestli chyba klasifikátoru ε_t není větší než 50%, pokud je, tak se daný algoritmus ukončí. Dále se vypočte váha α_t slabého klasifikátoru podle velikosti jeho chyby. V každém cyklu se musí nakonec aktualizovat jednotlivé váhy D_t trénovací množiny. Váhy u dobře klasifikovaných dat klesají, naopak u špatně klasifikovaných dat stoupají. Díky tomuto se nevybere v každém kroku stejný slabý klasifikátor. Váha vzorku lze přímo převést na výsledek klasifikátoru. Zvyšování počtu klasifikátorů může vést k přetrénování, ovšem na většině dat nemá AdaBoost tendenci se přetrénovat. Pseudokód algoritmu je popsán v algoritmus 1. (2) (3) (5)

Algoritmus 1: AdaBoost *(5)*

Vstup: $(x_1, y_1), \dots, (x_m, y_m)$, kde $x_i \in X$ – hodnota příznaku, $y_1 \in \{-1,1\}$ – třída odpovídající příznaku i

Inicializace vah $D_1(i) = \frac{1}{m}$

for $t = 1, \dots, T$ **do**

Vyber slabý klasifikátor s nejmenší váženou chybou

$$\varepsilon_t = \sum_{i=1}^{m} D_t(i)\left[y_i \neq h_j(x_i)\right]$$

if $\varepsilon_t \geq \frac{1}{2}$ **then** stop

Nastav $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$

Aktualizuj váhy:

$$D_{t+1}(i) = \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t}$$

kde Z_t je normalizační faktor

end

Výstup:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

2.2 HOG (Histograms of oriented gradients)

Dalším velmi používaným typem detektoru je detektor založený na histogramech orientovaných gradientů, který poprvé v roce 2005 představili Navneet Dalal a Bill Triggs (6). Hlavní myšlenkou tohoto detektoru je, že hledaný objekt může být popsán pomocí intenzity gradientů nebo směrnicemi hran. Mezi pozitivní vlastnosti detektoru HOG patří především odolnost vůči změně osvětlení, změně kontrastu, šumu, změně měřítka, velikosti, avšak není odolný vůči rotaci. Lze pracovat jak s fotografiemi ve stupních šedi, tak i s barevnými fotografiemi.

V prvním kroku se nejprve musí detekovat hrany v obraze. To se nejčastěji realizuje pomocí první derivace ve směru x a y. Před samotným výpočtem se nejprve provede Gaussovo rozostření. Derivace se provádí pomocí konvoluce vstupního obrazu s vhodným konvolučním jádrem. Jako vhodná konvoluční jádra byly experimentováním zjištěny matice:

$$D_x = [1 \quad 0 \quad -1] \tag{2.5}$$

$$D_y = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \tag{2.6}$$

Autoři dále zkoušeli použít k detekci hran i jiné masky jako například Sobelův operátor (2.7) a (2.8) nebo diagonální masku. Experimentováním však zjistili, že použitím první derivace ve směru x a y je dosahováno nejlepších výsledků.

$$D_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{2.7}$$

$$D_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \tag{2.8}$$

Dále autoři zjistili, že pokud před výpočtem derivace provedeme rozmazání vstupního obrazu pomocí Gaussova filtru, dosáhneme lepších výsledků. To je dáno především tím, že Gaussův filtr dokáže zamaskovat šum a některé nepodstatné detaily.

Dále se okno obrázku rozdělí na menší stejně velké oblasti, tzv. buňky. Buňky mohou být buď čtvercové (R-HOG - 2.6 vlevo), nebo kruhové (C-HOG 2.6 vpravo), většinou se však používají čtvercové buňky. Pro každou buňku se ze všech pixelů v buňce vypočítají lokální jednorozměrné histogramy orientovaných gradientů. Výsledná velikost úhlu gradientu může být buď v rozsahu 0° – 180°, nebo 0° – 360°. Velikosti gradientů obsažených v jednotlivých buňkách se rozdělí do několika tříd, ze kterých se následně vytvoří histogram pro každou buňku. Nejčastěji se rozsah úhlů dělí do 9 tříd, pokud používáme rozsah 0° – 180°, pak má každá třída šířku 20°.

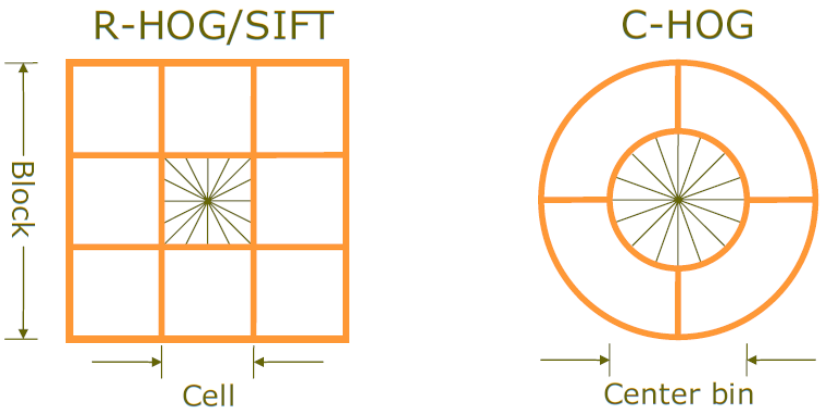
Pro zvýšení odolnosti na změně osvětlení, kontrastu mezi jednotlivými buňkami se používá normalizace. Normalizace se provádí nad částečně překrývajícími se bloky, které se při použití čtvercových buněk skládají nejčastěji z 3×3 buněk (obrázek 2.6). K normalizaci lze použít jednu ze čtyř odlišných normalizačních metod: L2-norm (2.9), L1-norm (2.9), L1-sqrt (2.10) a dále L2-hys, která je L1-norm následována clippingem (omezení maximální hodnoty v na 0,2) a renormalizace. Experimentováním bylo zjištěno, že poslední tři metody mají podobný výkon, zatímco první je o něco méně výkonná. (6)

$$\text{L2} - \text{norm: } f = \frac{v}{\sqrt{\|v\|_2^2 + e^2}} \tag{2.9}$$

$$\text{L1} - \text{norm: } f = \frac{v}{\|v\|_2^2 + e} \tag{2.10}$$

$$\text{L1} - \text{sqrt: } f = \sqrt{\frac{v}{\|v\|_1 + e}} \tag{2.11}$$

kde v je nenormalizovaný vektor obsahující všechny histogramy v daném bloku, $\|v\|_k$ je jeho k-norm pro $k = 1{,}2$ a e je malá konstanta (přesná hodnota je nedůležitá). (6)



Obrázek 2.6: Zobrazení jednoho bloku a buněk. Vlevo R-HOG, vpravo C-HOG (6)

2.2.1 SVM (Support vector machines)

K natrénování klasifikátoru založenému na HOG se nejčastěji používá algoritmus SVM (support vector machines - algoritmus podpůrných vektorů). Jedná se o metodu strojového učení, která spadá do kategorie učení s učitelem. Cílem algoritmu je najít takovou nadrovinu, která optimálně rozděluje dvě třídy trénovacích dat, tj. nadrovinu, která má maximální vzdálenost nejbližších bodů k nadrovině. K popisu nadroviny stačí pouze body, které jsou nejblíže nadrovině, tyto body nazýváme podpůrné vektory (suppot vector). Výhodou tohoto algoritmu je, že dokáže oddělit i nelineárně oddělitelná data (např. kružnici) lineární funkcí. Pro nelineárně oddělitelná data se používají jádrové transformace (kernel transformations), pomocí kterých je možné mapovat vstup do prostoru o více dimenzích. Zjednodušeně lze říci, že každý n-dimenzionální vektor lze převést na n+1-dimenzionální vektor, přidáním dalšího atributu závislého na původních n atributech. Pomocí toho lze oddělit nelineárně ohraničená data nadrovinou, tím že posuneme jednu třídu dat podél osy nové dimenze. Obecný popis lineárního klasifikátoru pak má tvar:

$$\boldsymbol{f(x) = w^T x + b} \tag{2.12}$$

kde w je váhový vektor a b je posunutí (bias).

Jestliže $f(x) \geq 0$ předpokládáme, že se jedná o pozitivní klasifikaci a jestliže $f(x) < 0$ předpokládáme, že se jedná o negativní klasifikaci.

V algoritmu SVM se optimální lineární oddělovač hledá pomocí metody kvadratického programování. Předpokládejme, že jsou příklady x_i s klasifikací $y = \pm 1$ a cílem je najít optimální oddělovače. Problém lze převést na hledání hodnot parametrů α_i, které maximalizují výraz:

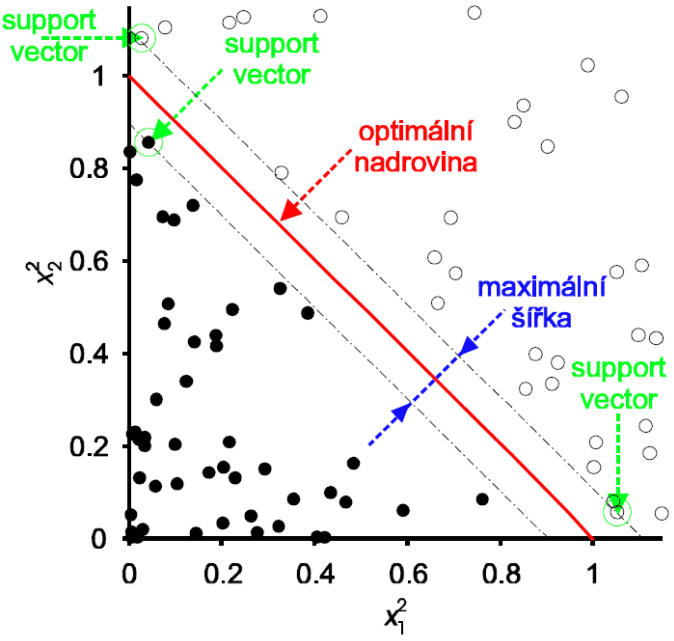
$$\sum_t \alpha_i - \frac{1}{2}\sum_{t,j} \alpha_i \alpha_j y_t y_j (x_i, x_j) \tag{2.13}$$

kde x_i je vstupní příklad, y_i je klasifikace vstupního příkladu $y = \{-1, +1\}$ a $\alpha_i$je váhový koeficient. Přičemž platí omezení:

$$a_i \geq 0 \text{ a} \tag{2.14}$$

$$\sum_i \alpha_i y_i = 0. \tag{2.15}$$

Na obrázku 2.7 je zobrazeno použití lineárního SVM na rozdělení dvou tříd objektů ve 2D prostoru. Zeleně jsou okroužkovány podpůrné vektory (support vector), červené přímka zobrazuje optimální nadrovinu a modrá úsečka zobrazuje onu maximální vzdálenost nejbližších bodů (podpůrných vektorů) k nadrovině.



Obrázek 2.7: Ukázka použití lineárního SVM na rozdělení dvou tříd objektů ve 2D prostoru (7)

Kromě lineárního jádra se často používá polynomiální jádro, RBF (Radial basis function) nebo sigmoidní jádro, které dokážou lépe oddělit oblasti, které nejsou lineárně oddělitelné.

Pokud používáme SVM k natrénování detektoru HOG, tak jedna třída bude obsahovat pozitivní obrázky a druhá třída bude obsahovat negativní obrázky. Hledaná nadrovina nám určuje, kterým směrem od nadroviny leží hledané objekty a kterým směrem od nadroviny leží pozadí obrázku. (7) (8) (9)

3 Spojování fotografií

V této kapitole se budu zabývat skládáním jednotlivých fotografií do výsledného celku. Cílem je najít překrývající se části několika fotografií a tím vytvořit výslednou panoramatickou fotografii.

Pro spojování fotografií se používají dvě skupiny metod, první jsou přímé metody, které přímo minimalizují rozdíly mezi pixely, a druhé jsou metody založené na příznacích, které extrahují řídkou sadu příznaků a potom hledají navzájem podobné příznaky. Příznakově založené metody mají výhodu v tom, že jsou odolnější proti pohybu scény a jsou potenciálně rychlejší. Avšak jejich největší výhodou je schopnost rozpoznat panoramatickou fotografii, to znamená, že automaticky najde sousední fotografie z neuspořádané sady fotografií, což tuto metodu děla ideální pro plně automatické spojování fotografií přijatých od uživatele. (10)

Následující podkapitoly se budou zabývat výhradně příznakově založenými metodami. U příznakově založených metod se nejprve musí detekovat klíčové body, ze kterých se pak vypočte deskriptor (3.1). Následně se musí nalézt shodné deskriptory (3.2). Poté se odhadne homografie (3.3.2) mezi fotografiemi za pomocí RANSAC (3.3.3). Nakonec se fotografie promítne na válec/kouli (3.3.4) a provede se blending (3.4) (6).

3.1 Detekce klíčových bodů a výpočet deskriptorů

Aby šlo spojit více fotografií do jedné, musíme nejprve u všech fotografií detekovat klíčové body, ze kterých se poté vypočte deskriptor. Klíčové body jsou body, které jsou něčím zajímavé, například rohy, osamocené body, konce čar atd. K detekci klíčových bodů a výpočtu deskriptorů slouží několik algoritmů.

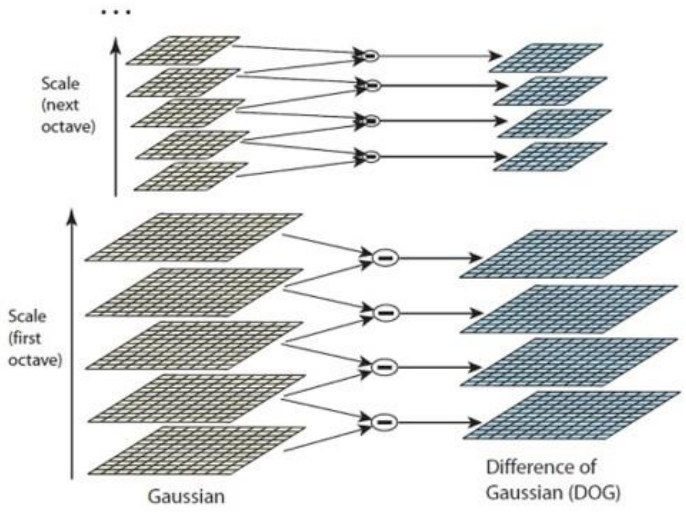
3.1.1 SIFT

SIFT (Scale Invariant Feature Transform) je algoritmus sloužící k detekci a popisu klíčových bodů, který vytvořil v roce 1999 David G. Lowe (11). Jak již jeho název napovídá, jeho hlavní vlastností je, že jeho klíčové body jsou invariantní vůči měřítku. Mezi jeho další pozitivní vlastnosti patří invariance vůči natočení, osvětlení a šumu fotografie a částečná invariance vůči a prostorové změně úhlu pohledu.

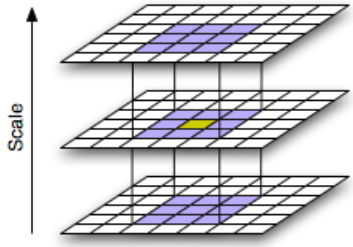
Cena extrakce klíčových bodů je minimalizována tím, že je zavedena kaskádová filtrace, v níž jsou nákladnější operace prováděny pouze v případě, že daný bod prošel předchozím stupněm kaskády. Zde jsou uvedeny hlavní fáze výpočtu.

V první fázi výpočtu se musí vytvořit scale-space reprezentace k detekci extrémů. Nejprve se použije difference-of-Gaussian (DoG). DoG nejdříve provede několikrát za sebou na stejném obrázku Gaussovo rozostření, čímž vznikne několik obrázků různě rozostřených. Toto Gaussovo rozostření se provede na obrázku v několika různých rozlišeních. Vždy dva sousední obrázky z jednoho měřítka se

od sebe odečtou, což je aproximace druhé derivace obrázku. Tím získáme rozdílové obrázky (obrázek 3.1). Z rozdílových obrázků vybereme lokání extrémy, což jsou pixely s hodnotou větší nebo menší než všechny jeho sousední pixely, a to včetně pixelů v 3×3 okolí na stejném místě v sousedních obrázcích ve scale-space (obrázek 3.2). Tyto lokální extrémy považujeme za potenciálně klíčové body.



Obrázek 3.1: Konstrukce scale-space pomoci DoG (11)



Obrázek 3.2: Určování lokálních maxim (11)

Ve druhé fázi je potřeba odstranit potenciálně klíčové body, který jsou nestabilní. To jsou potenciálně klíčové body, ve kterých je nedostatečný kontrast nebo leží podél hran.

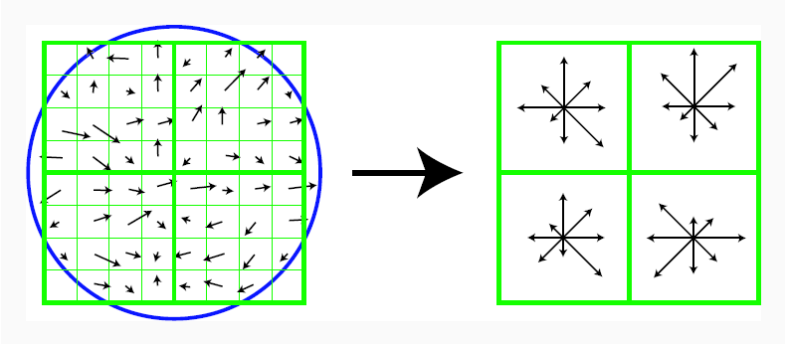
V další fázi se přiřadí všem zbylým klíčovým bodům jejich velikost $m(x, y)$ a orientace $\theta(x, y)$, která se vypočte podle těchto vzorců:

$$m(x,y) = \sqrt{\big(L(x+1,y) - L(x-1,y)\big)^2 + \big(L(x,y+1) - L(x,y-1)\big)^2} \tag{3.1}$$

$$\theta(x,y) = \tan^{-1}\frac{L(x+1,y) - L(x-1,y)}{L(x,y+1) - L(x,y-1)} \tag{3.2}$$

kde $L(x, y)$ je hodnota pixelu z Gaussově rozmazaného obrázku. Od této chvíle je každý klíčový bod invariantní vůči rotaci a měřítku.

V poslední fázi výpočtu se pro každý klíčový bod vypočte jeho deskriptor. Deskriptor je oproti klíčovému bodu více nezávislý na osvětlení a prostorové změně úhlu pohledu. Deskriptor se počítá z gradientů o určité velikosti a orientaci, které se nacházejí v daném okolí klíčového bodu. Podle měřítka klíčového bodu se vybere obrázek s požadovaným Gaussovským rozostřením. Aby bylo dosaženo invariance orientace, jsou souřadnice deskriptoru a orientace gradientu otočeny vzhledem k orientaci klíčového bodu. Pro větší účinnost jsou gradienty na všech úrovních pyramidy předpočítány. V každé podoblasti jsou gradienty složeny do orientovaného histogramu s délkou vektoru odpovídající součtu velikosti gradientu v dané podoblasti. Nejčastěji se používají 4×4 deskriptory počítané z pole 16×16 vzorku. Pole gradientů a pole deskriptorů je zobrazeno na obrázku 3.3. (11)



Obrázek 3.3: Výpočet deskriptoru (2×2 pole deskriptorů počítaných z pole 8×8 vzorků) (11)

3.1.2 SURF

Metoda SIFT byla velmi účinná, ale její rychlost nebyla nejvyšší, a proto se ji snažilo mnoho lidí vylepšit. Jedna z úspěšných vylepšení metody SIFT je metoda SURF (Speeded Up Robust Features), kterou v roce 2006 vytvořil Herbert Bay (12). Tato metoda zachovává všechny předchozí vlastnosti jako invarianci vůči měřítku, natočení, osvětlení a šumu fotografie a částečnou invarianci vůči a prostorové změně úhlu pohledu a navíc v porovnání s metodou SIFT je mnohem rychlejší. Největší změnou oproti SIFT je použití Hessovy matice k detekci klíčových bodů, a to především pro jeho rychlost výpočtu a přesnost.

Před samotným použitím detektoru se pro obrázek musí vypočítat jeho integrální obraz (viz kapitola 2.1.2 Integrální obraz).

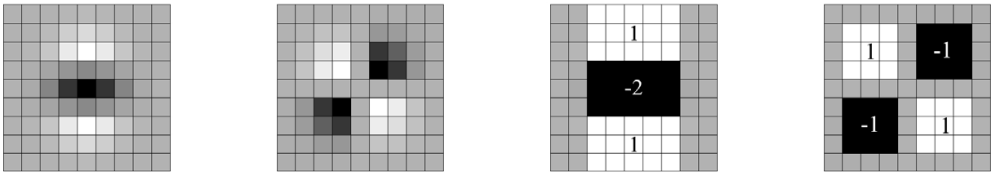
Hessova matice $\mathcal{H}(\mathbf{x}, \sigma)$ v bodě $\mathbf{x} = (x, y)$ a v měřítku σ je definována následovně:

$$\mathcal{H}(\mathrm{x}, \sigma) = \begin{bmatrix} L_{xx}(\mathrm{x}, \sigma) & L_{xy}(\mathrm{x}, \sigma) \\ L_{xy}(\mathrm{x}, \sigma) & L_{yy}(\mathrm{x}, \sigma) \end{bmatrix}, \tag{3.3}$$

kde $L_{xx}(\mathbf{x}, \sigma)$ je konvoluce druhé derivace Gaussovy funkce $\frac{\partial^2}{\partial x^2} g(\sigma)$ s obrázkem I v bodě $\mathbf{x}$ a podobně pro $L_{xy}(\mathbf{x}, \sigma)$ a $L_{yy}(\mathbf{x}, \sigma)$.

Detektor SIFT se snaží o co nejpřesnější aproximaci druhé derivace obrázku (přesněji Laplaciánu), experimentováním však bylo zjištěno, že i při velmi nepřesné aproximaci lze dosáhnout

kvalitních výsledků, a proto se autoři detektoru SURF rozhodli použít aproximaci pomocí obdélníkových filtrů. Obdélníkové filtry o velikosti 9×9 (poslední dva zprava na obrázku 3.4) jsou aproximací druhé derivace s $\sigma = 1{,}2$ a reprezentuje nemenší měřítko.



Obrázek 3.4: Obrázky nalevo znázorňují Gaussovu druhou parciální derivaci ve směru y a směru xy, obrázky napravo znázorňují aproximaci pomocí obdélníkových filtrů (šedé oblasti mají nulovou hodnotu). (12)

Díky použití obdélníkových filtrů a integrálního obrazu se nemusí opakovaně aplikovat stejný filtr na výstup již dříve filtrované vrstvy, ale místo toho se mohou s konstantní rychlostí tyto filtry libovolné velikosti použít na originální obrázek. 9×9 filtr je považován za výchozí měřítkovou vrstvu, na kterou se bude odkazovat jako na měřítko $s = 1{,}2$. Následující vrstvy jsou získány filtrací obrázku s postupně se zvyšující maskou. Konkrétně se jedná o filtry o velikosti 9×9, 15×15, 21×21, 27×27, atd. Se zvětšujícím se měřítkem se krok mezi následujícími velikostmi filtru zvětšuje na dvojnásobek (z 6 na 12, z 12 na 24). Klíčové body se, stejně jako u metody SIFT, naleznou v lokálních extrémech, čili v bodech, které jsou větší nebo menší než všech 26 sousedních pixelů.

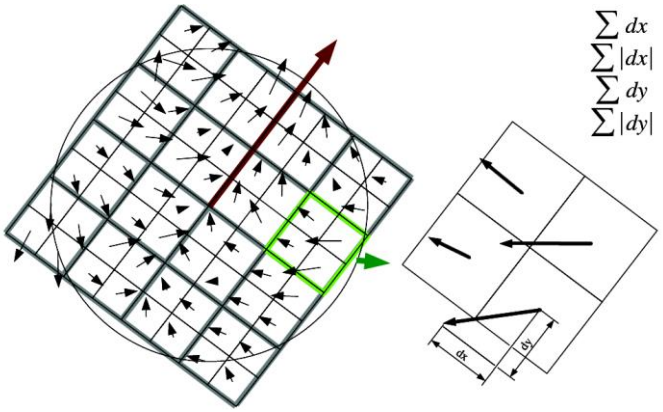
Aby byla zajištěna invariance vůči rotaci, je potřeba každému klíčovému bodu přiřadit jeho orientaci. Za tímto účelem se nejprve vypočtou odezvy Haarových vlnek (viz kapitola 2.1.1 Haarovy příznaky) ve směru x a y, a to v kruhu o poloměru $6s$ okolo klíčového bodu, v měřítku s, ve kterém byl klíčový bod detekován. Pro rychlejší filtrování se znovu používá integrální obraz. Dominantní orientace je odhadnuta jako součet všech odezev v posuvném okénku orientací, kde velikost jednoho okénka byla experimentálně určena na úhel $\frac{\pi}{3}$. Horizontální a vertikální odezvy uvnitř okna se sečtou a tím vzniká nový vektor. Klíčový bod pak získává orientaci po nejdelším vektoru.

Analogicky jako u metody SIFT, se v poslední fázi výpočtu pro každý klíčový bod vypočte jeho deskriptor. Nejprve se musí vytvořit čtvercová oblast se středem v klíčovém bodě a orientací podle orientace klíčového bodu. Velikost této oblasti většinou bývá $20s$. Tato oblast je rozdělena na menší pravidelné podoblasti o velikost 4×4. Pro každou podoblast se vybere několik pravidelně rozmístěných bodu, na kterých se vypočte odezva Haarových vlnek. V každé podoblasti se sečtou odezvy Haarových vlnek d_x a d_y a dále se sečtou i absolutní hodnoty odezev $|d_x|$ a $|d_y|$. A tak má každá podoblast vektor:

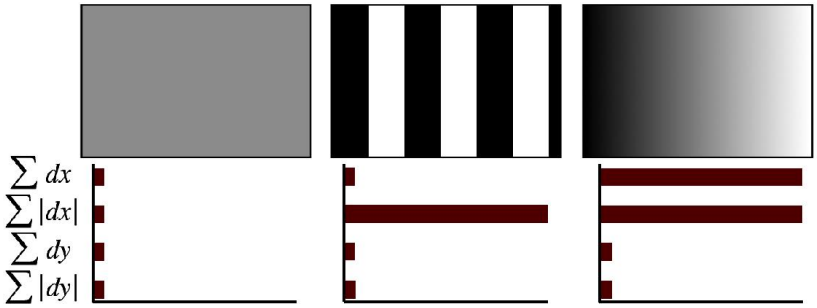
$$v = \left(\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|\right). \tag{3.4}$$

Výsledný vektor deskriptoru je počítán ze všech 16 podoblastí, tím pádem vzniká 64dimenzionální vektor.

Obrázek 3.5 zobrazuje, jak se vytváří deskriptor. Jak budou vypadat odezvy Haarových vlnek d_x a d_y a absolutní hodnoty odezev $|d_x|$ a $|d_y|$ pro 3 různé podoblasti je zobrazeno na obrázku 3.6. (12)



Obrázek 3.5: Tvorba deskriptoru SURF (12)



Obrázek 3.6: Na obrázku je vidět jak vypadají odezvy Haarových vlnek d_x a d_y a absolutní hodnoty odezev $|d_x|$ a $|d_y|$ na třech ruzných podoblastech. (12)

3.2 Nalezení shodných deskriptorů

Po nalezení všech deskriptorů je potřeba najít shodné deskriptory v různých fotografiích. Pro nalezení shodných deskriptorů se nejčastěji používají algoritmy Brure Force a FLANN.

Brute force pro každý deskriptor hledá nejlepší shodu, a to tak, že ke každému deskriptoru prozkoumá všechny deskriptory z ostatních obrázků. Výhodou tohoto řešení je, že najde opravdu nejlepší shodu, ale zato je časově náročný, a proto není vhodný pro databáze s velkým počtem obrázků.

FLANN (fast approximate nearest neighbors) pro hledání korespondenci mezi deskriptory automaticky vybírá mezi dvěma algoritmy, a to mezi náhodným k-d stromem a hierarchickým k-means. Tato metoda je rychlejší než Brute Force, ale jeho nevýhodou je, že nenachází shodné deskriptory s takovou přesností. (13)

3.3 Geometrické transformace

Aby se daly vstupní fotografie spojit k sobě, potřebujeme s nimi různě manipulovat, tak aby byly překrývající se oblasti nad sebou, a k tomu potřebujme znát geometrické transformace.

3.3.1 Základní geometrické transformace ve 2D prostoru

Mezi základní lineární geometrické transformace patří posunutí, otáčení, změna měřítka a zkosení, které budou níže popsány. Všechny základní transformace jsou zobrazeny na 3.7.

Aby šlo se všemi základními transformacemi pracovat jednotně, byly zavedeny tzv. homogenní souřadnice bodu. Umožňují nám vyjádřit všechny druhy základních transformací jednou transformační maticí a aplikovat je násobením matic a vektorů. Homogenní souřadnice bodu ve 2D s kartézskými souřadnicemi $[x, y]$ je uspořádaná trojice $[X, Y, w]$, pro kterou platí $X = xw, Y = yw$. Bod je svými homogenními souřadnicemi jednoznačně určen. Souřadnice w se nazývá váha bodu. (14)

Posunutí – posouvá obrazu vektorem posunutí $T(d_x, d_y)$. Nové souřadnice jsou dány vztahem (3.5) a transformační matice T má tvar (3.6). (14)

$$x' = x + d_x, \qquad y' = y + d_y \tag{3.5}$$

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ d_x & d_y & 1 \end{bmatrix} \tag{3.6}$$

Otáčení – otočí obraz o úhel α se středem otáčení v počátku souřadného systému. Nové souřadnice jsou dány vztahem (3.7) a transformační matice R má tvar (3.8). (14)

$$x' = x \cdot \cos\alpha - y \cdot \sin\alpha, \qquad y' = x \cdot \sin\alpha + y \cdot \cos\alpha \tag{3.7}$$

$$R = \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

Změna měřítka – změní měřítka obrazu s faktory změny měřítka S_x a S_y. Nové souřadnice jsou dány vztahem (3.9) a transformační matice S má tvar (3.10). Je-li faktor změny měřítka $S_{x,y} > 1$, dochází ke zvětšení, je-li $0 < S_{x,y} < 1$, dochází ke zmenšení a je-li $S_{x,y} < 0$, dochází k převrácení (zrcadlení). (14)

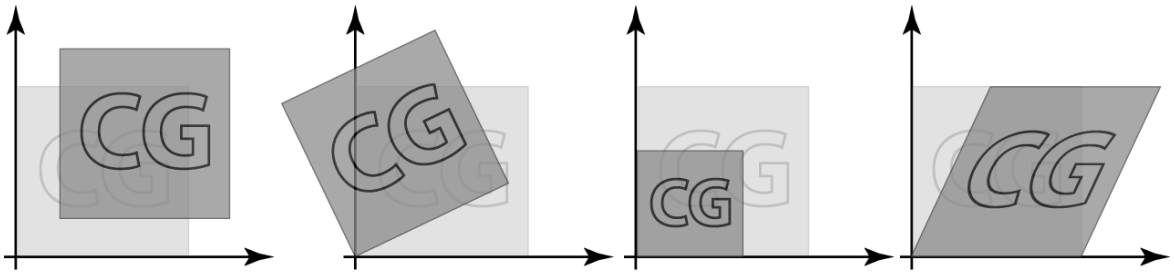
$$x' = x \cdot S_x, \qquad y' = y \cdot S_y \tag{3.9}$$

$$S = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.10}$$

Zkosení – zkosí obraz s faktory zkosení S_{hx} a S_{hy}. Nové souřadnice jsou dány vztahem (3.11) a transformační matice S_H má tvar (3.12). (14)

$$x' = x + S_{hx} \cdot y, \qquad y' = y + S_{hy} \cdot x \tag{3.11}$$

$$S_H = \begin{bmatrix} 1 & S_{hx} & 0 \\ S_{hy} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.12}$$



Obrázek 3.7: Zobrazení základních lineárních geometrických transformace. Zleva posunutí, otáčení, změna měřítka a zkosení (15)

V praxi se většinou setkáváme s afinní transformacemi, které se skládají z několika základních transformací. Afinní transformace jsou takové transformace, při kterých se zachovává kolinearita a dělicí poměr. Afinní transformace se pak dá vyjádřit jednou transformační maticí. Tuto afinní transformační matici získáme násobením dílčích základních transformačních matic, avšak při násobení matic musíme dodržet přesné pořadí. (14)

3.3.2 Homografie

Fotografie vytváříme ve 3D prostoru, tím pádem nám afinní transformace nebudou stačit, a proto se k transformaci obrázků používá homografie. Homografie je chápána jako mapování bodů a vyjadřuje transformaci mezi obrázky. K výpočtu homografie potřebujeme znát minimálně 4 sobě odpovídající body.

Homografie je reprezentována maticí H o rozměrech 3×3 v homogenních souřadnicích. Každou dvojici shodných bodů p' a p lze zapsat jako:

$$wp' = H \cdot p \tag{3.13}$$

kde w je homogenní souřadnice měřítka. Když do rovnice dosadíme, dostaneme:

$$\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{3.14}$$

Po roznásobení matic, dosazením w do prvních dvou rovnic a několika matematických úpravách dostáváme dvě lineární rovnice:

$$h_{00}x + h_{01}y + h_{02} - h_{20}x'x - h_{21}x'y - h_{22}x' = 0 \tag{3.15}$$

$$h_{10}x + h_{11}y + h_{12} - h_{20}y'x - h_{21}y'y - h_{22}y' = 0 \tag{3.16}$$

Složením rovnic pro n bodů vzniká matice A jejíž hodnost je $2n$ a tvoří lineární homogenní soustavu rovnic pro 9 neznámých. Řešením je nulový prostor této matice.

$$A \cdot h = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1'x_1 & -x_1'y_1 & -x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1'x_1 & -y_1'y_1 & -y_1' \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2'x_2 & -x_2'y_2 & -x_2' \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -y_2'x_2 & -y_2'y_2 & -y_2' \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_n'x_n & -x_n'y_n & -x_n' \\ 0 & 0 & 0 & x_n & y_n & 1 & -y_n'x_n & -y_n'y_n & -y_n' \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.17}$$

V praxi je však často nulový prostor matice A prázdný a vektor h se hledá pomocí metody nejmenších čtverců jako

$$h = \text{argmin}\|Ah\|^2 \tag{3.18}$$

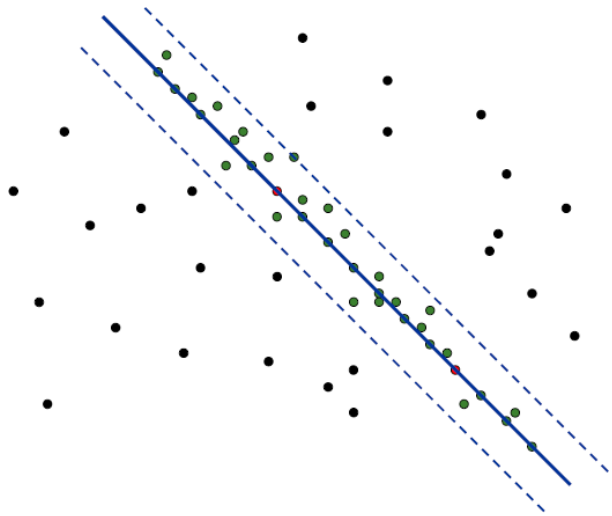
kde $\|X\|$ je euklidovská norma $\sqrt{\sum X_1^2}$. Nalezení vektoru se pak provádí pomocí SVD (Singular Value Decomposition). (16) (17)

3.3.3 RANSAC

Při tvorbě homografie nastává problém v tom, že ne všechny korespondence mezi body jsou správně určené. K výběru správných korespondencí nám slouží algoritmus RANSAC (RANdom SAmple Consensus), který již v roce 1981 publikovali M. Fischler a R. C. Bolles (14). RANSAC využívá předpokladu, že potenciálně spojované obrázky obsahují sadu bodů, které náleží hledané homografii, tzv. inliers, ale mimo to také sadu bodů, které jsou vzdálené od hledané homografie a nejspíš jsou spatně určené, tzv. outliers. Je vhodný pro data s velkým počtem vzorků.

Obrázek 3.8 znázorňuje použití RANSAC pro detekci přímky ve zdrojových datech. Na obrázku je znázorněna nejlepší iterace.

Při použití metody RANCAS k určení homografie se nejprve náhodně vyberou 4 páry korespondujících bodů. Z těchto bodů se vypočte homografie. Všechny korespondující body z prvního obrázku se transformují pomocí vypočtené homografie, a tím dostaneme body z druhého obrázku. Jestliže tyto body jsou vzdálené od korespondujících bodů v druhém obrázku v určité mezi (např. 1-3 pixely), jsou tyto body označeny jako inliers, jinak jsou označeny jako outliers. Tento postup se opakuje v cyklu tak dlouho, dokud počet inliersů nepřekročí daný práh nebo neprovedeme požadovaný počet iterací. Jako správná homografie je zvolena homografie s největším počtem inliers.



Obrázek 3.8: Nalezení přímky pomocí RANSAC. Červené body jsou prvotně zvolené body, zelené body jsou inliers a černé body jsou outliers

Pseudokód obecného algoritmu RANCAS se popsán v algoritmus 2.

Algoritmus 2: RANSAC *(19)*

Vstup: $X = \{x_1, x_2, \dots, x_n\}$ – množina korespondujících bodů o počtu n

for $k = 1 \dots K$ **do**

Náhodně vyber m položek z X a vypočítej model

Zjisti kolik bodu z X vyhovuje předdefinované toleranci e (jsou inliers)

if inliers $>$ práh t **then**

Přepočítej model použitím identifikovaných inliers

break

end

end

Výstup: model s největším počtem inliers nebo chyba

Počet iterací K je zvolen dostatečně velký, tak aby zajistil, že pravděpodobnost p (většinou nastavena na 0,99) vybere alespoň jednu sadu vzorků, která neobsahuje outliers. Nechť u reprezentuje pravděpodobnost, že všechny body jsou inliers a $v = 1 - u$ pravděpodobnost, že obsahuje i outliers. Pravděpodobnost, že algoritmus nikdy nevybere m bodů, které jsou inliers je $1 - p$.

$$1 - p = (1 - u^m)^K \tag{3.19}$$

Po úpravě rovnice zjistíme, že minimální počet iterací K potřebných k nalezení m bodů, které neobsahují outliers je:

$$K = \frac{\log(1-p)}{\log(1-u^m)} \tag{3.20}$$

Algoritmus RANSAC existuje v mnoha různých modifikacích. (19) (20)

3.3.4 Projekce

Poté co jsme vypočítali homografii a víme, jak a které fotografie k sobě patří, musíme se rozhodnout, jakým způsobem se provede spojení fotografií, to znamená vybrat mapovací projekci. Nejčastěji se používá projekce na plochu, válec nebo kouli, avšak každý povrch lze použít k projekci jako například krychle, šestiboký hranol atd. (10)

Projekce na rovinu – jde o základní typ projekce. Jeden obrázek je zvolen jako referenční a ostatní obrázky jsou deformovány tak, aby měly stejnou souřadnicovou soustavu s referenčním obrázkem. Jde stále o perspektivní projekci, a proto rovná přímka zůstává rovná. Tato projekce je použitelná pouze pro malý počet spojovaných fotek. Již při překonání zorného pole 90° je panoramatická fotografie značně zdeformována. Maximální zorné pole, které lze s touto projekcí dosáhnou, je 180°. Tato projekce se v praxi téměř nepoužívá. (10)

Projekce na válec – tato projekce simuluje projekci na plášť válce, jako kdyby pozorovatel stál uprostřed válce. Je vhodná pro tvorbu panoramatických fotografií, kdy fotograf stojí na jednom místě a fotografie tvoří otáčením kolem své osy. S touto projekcí lze již dosáhnout úplně 360° panoramatické fotografie. Zde se již nejedná o perspektivní projekci, a proto dochází k deformaci přímek. Dochází zde k transformaci souřadnicového systému u všech obrázků. Tento druh projekce je nejpoužívanější. (10)

Projekce na kouli – při použití této projekce se, na rozdíl od projekce na válec, musí navíc vyfotografovat vrchlík a podstava (povrch nad a pod bodem, ze kterého jsou fotografie pořizovány). Výsledek této projekce je někdy označován jako tzv. „malá planetka“. Stejně jako u projekce na válec, se nejedná o perspektivní projekcí a dochází ke změně souřadnicového systému u všech obrázků. (10)

3.4 Blending

Před tímto bodem již známe všechny geometrické transformace, které bude potřeba na spojení fotografii. V ideálním případě by všechny pixely v překrývající obrázcích měly stejnou intenzitu, avšak ve skutečnosti to tak většinou není. Nejčastějšími důvody jsou změna expozice, vinětace (pokles intenzity na okrajích fotografie), efekt paralaxy (způsobený nežádoucím pohybem z optického středu) a všechny chyby registrace, které jsou způsobeny chybným modelováním kamery, velkým zkreslením atd. A proto se používá blending, který se snaží rozdílnou intenzitu minimalizovat. Dále blending nastavuje šev, tak aby minimalizoval viditelnost přechodů mezi fotografiemi.

Aby bylo možné sloučit informace z více obrázků dohromady, je každému obrázku přidána váhová funkce $w(x, y) = w(x)w(y)$, kde $w(x)$ se lineárně mění od 1, ve středu obrázku, do 0, na okrajích obrázku. Obyčejný blending provádí vážený součet intenzit pro každou překrývající se oblast pomocí těchto vážených funkci. Při malé chybě registrace to však může způsobit rozmazání vysokofrekvenčního detailu.

Aby k tomu nedocházelo, používá se multi-band blending, který v roce 1983 představili Peter J. Burt a Edward H. Adelson (21). Myšlenka multi-band blendingu je míchání nízkých frekvencí přes velký prostorový rozsah a vysokých frekvencí přes malý rozsah. To může být provedeno v několika frekvenčních pásmech pomocí Laplaceovy pyramidy. (22) (10)

4 State of the art

V dnešní době se v oblasti rozpoznávání osob již téměř výhradně používají dva detektory, a to Viola-Jones a HOG (Histograms of oriented gradients). Tyto detektory jsou velmi rychlé a dokonce dokážou pracovat v reálném čase, a tak jsou často implementovány i v levnější elektronice jako jsou mobilní telefony nebo kompaktní fotoaparáty. Oba detektory také spojuje to, že mají vyučovaný klasifikátor. Ručně vytvořené klasifikátory se dnes již téměř nepoužívají, protože dosahují horších výsledků než vyučované klasifikátory. (23) Již před rokem 2001 (rok, kdy byl představen detektor Viola-Jones) existoval velmi kvalitní naučený detektor od Schneiderman-Kanade, ovšem tento detektor zpracovával obrázek přes noc a učil se několik měsíců. (23)

Pro detekci obličejů je stále nejlepší detektor Viola-Jones. Tento detektor je poměrně mladý a v průběhu let se příliš neměnil. Jedním z vylepšení je, že začal používat Haarovy příznaky, které jsou natočené o 45°. (4) Dále vzniklo několik variant AdaBoost. Nejpoužívanější jsou Real AdaBoost a Gentle AdaBoost.

Detektor založený na HOG je také poměrně mladý a příliš se neměnil. Změnami prošlo především SVM, kde se začala používat jiná než lineární jádra, pro rozpoznávání obrázků především RBF. Tento detektor se častěji používá k detekci chodců, dopravních prostředků, zvířat nebo předmětů, které lze lépe popsat pomocí gradientu.

V oblasti tvorby panoramatických fotografií existují přímé metody a příznakově založené metody. V současnosti se však téměř výhradně používají příznakově založené metody, a to především pro jejich větší odolnost proti pohybu scény, potenciálně vyšší rychlost a zejména pak schopnost rozpoznat panoramatickou fotografii. Kvalita a rychlost příznakově založených metod je závislá především na detektoru klíčových bodů a tvorbě deskriptorů. K detekci klíčových bodů a tvorbě deskriptorů byla dříve velmi populární metoda SIFT, která je však nyní již často nahrazována metodou SURF, která v podstatě vychází z metody SITF. Pro hledání shodných deskriptorů se většinou volí mezi metodami Brute force a FLANN, a to podle toho, jestli je pro nás důležitější rychlost nebo přesnost. Pro výpočet homografie se většinou používají vylepšené metody RANSAC, jako například Lokálně optimalizovaný RANSAC (LO-RANSAC) nebo PROSAC (PROgressive SAmple Consensus). (24) A jako blending se dnes nejčastěji používá multi-band blending. (25)

Z důvodů použití příznakově založených metod, není potřeba vybírat sousední fotografie, a tím pádem mohou vznikat aplikace pro tvorbu panoramatických fotografií s velmi jednoduchým ovládáním, a to i na mobilních telefonech nebo fotoaparátech.

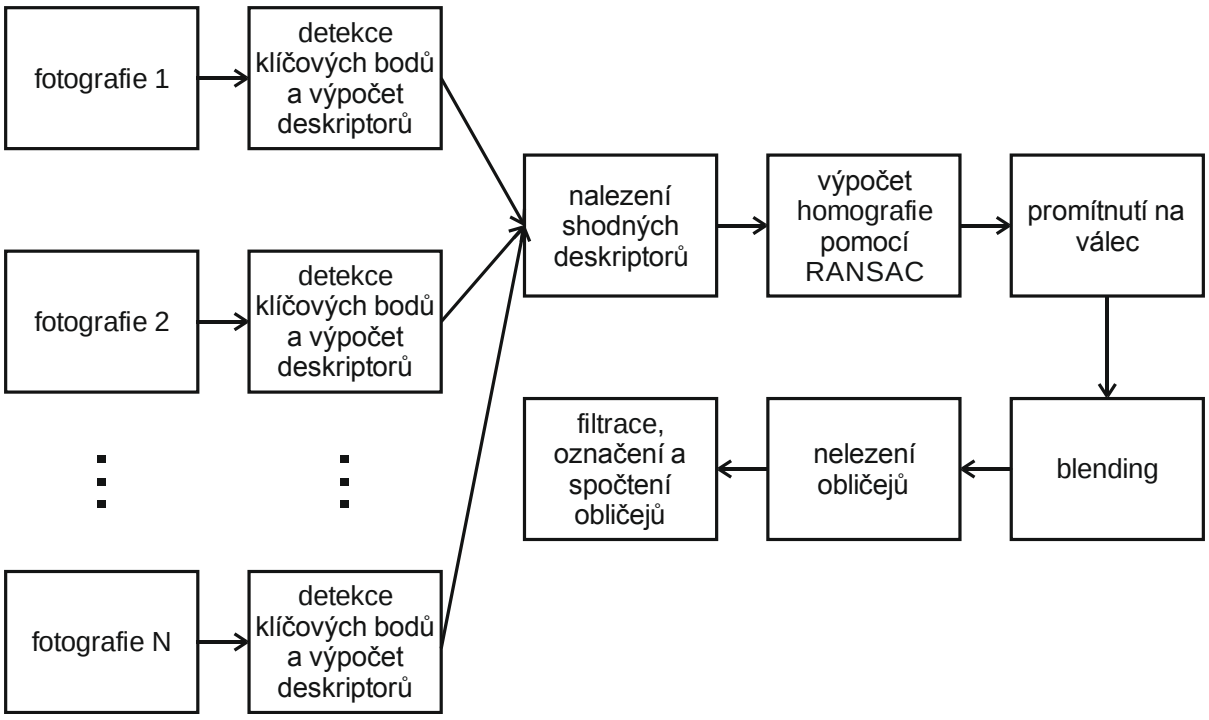
Žádná aplikace, která by primárně počítala lidi tak, že by nejprve spojila fotografie, nebyla nalezena.

5 Návrh

V této kapitole se budu zabývat návrhem programu a všemi jeho dílčími částmi. Při návrhu jsem vycházel z předchozích teoretických kapitol, které se zabývaly problematikou počítačového vidění.

Při návrhu aplikace byl brán v potaz problém počítání osob na velkých plochách, kdy nejsme schopni sejmout celou požadovanou plochu s lidmi do jednoho záběru nebo pokud rozlišení fotoaparátu není dostatečně velké k tomu, aby na výsledné fotografii byl detektor schopen rozpoznat požadované obličeje. Standartní řešení by k tomuto problému přistupovalo tak, že by nejprve rozpoznalo obličeje na všech fotografiích, poté by se snažilo najít stejné obličeje na různých fotografiích a ty poté odečetlo z celkového počtu nalezených osob. Toto řešení není úplně ideální a ani by nebylo příliš rychlé, zejména je pak problém nalézt dva stejné obličeje na fotografiích, kde jsou obličeje tvořeny malým počtem pixelů, což při fotografování na velkou dálku je velmi pravděpodobné. Z těchto důvodu jsem rozhodl, že při počítání osob na velkých plochách nejprve spojím všechny vstupní fotografie dohromady a z jedné výsledné panoramatické fotografie spočtu všechny obličeje. Touto technikou vyřeším problém hledání shodných obličejů pomocí jejich namapování při spojování fotografií.

Postup při návrhu aplikace je zobrazen na blokovém schématu na obrázku 5.1.



Obrázek 5.1: Blokové schéma návrhu aplikace

5.1 Spojování fotografií

V první části programu tedy budu spojovat vstupní fotografie. To provedu pomocí příznakově založených metod z důvodu již dříve zmíněných, jako jsou automatické nalezení sousedních fotografií nebo rychlost. Postup tedy bude probíhat v následující posloupnosti. Nejprve si ve všech fotografiích detekuji klíčové body, ze kterých vypočtu deskriptory. Poté zjistím, které deskriptory se shodují, za použití metody Brute Force, případně FLANN. Z těchto shodujících se deskriptorů vypočtu homografii za pomocí RANSAC. Pak obraz promítnu na válec, protože při tvorbě požadovaných fotek se dá předpokládat, že promítání na válec bude nejvhodnější. V poslední části spojování fotografií se provádí blendig. Kvalitní blending je pro rozpoznávání obličejů velmi důležitý, protože bez jeho použití by na obličejích mohly vznikat velké rozdíly jasu nebo by se obličej mohl rozdvojit a jejich detekce by byla velmi ztížená, proto budu používat některý z pokročilejších verzí blendingu.

Pomocí testování se pokusím nalézt ideální nastavení parametru, tak aby výsledná panoramatická fotografie byla co nejkvalitnější a doba vytvoření bylo co možná nejnižší.

5.2 Počítání osob

V druhé části programu budu počítat osoby na panoramatické fotografii. K tomu budu muset nejprve jednotlivě osoby rozpoznat. Protože při rozpoznávání velké skupiny osob většinou nejdou vidět celé postavy, budu se zabývat výhradně detekcí obličejů jednotlivých osob. Na základě poznatků z teoretické části o detektorech Viola-Jones a HOG jsem se rozhodl použít detektor Viola-Jones, který by měl být pro detekci obličejů vhodnější a zároveň rychlejší.

5.2.1 Detekce pomocí Viola-Jones

Při použití detektoru Viola-Jones potřebuji nejprve sadu pozitivních a sadu negativních obrázku, pomocí kterých se natrénuje kaskádový klasifikátor Haarových příznaků za použití metody AdaBoost.

U detekce pomocí metody Viola-Jones je potřeba předem si spočítat integrální obraz pro celou fotografií. V dalším kroku budu procházet vstupní fotografii pomocí menších oken o nejnižší velikosti 12×12, které se budou postupně posouvat a budou měnit jejich velikost vždy v určitém násobku. V každém okně spočítám Haarovy příznaky, které budu v kaskádě kontrolovat s natrénovanými Haarovými příznaky a při projití celé kaskády označím dané okno za pozitivní.

Tvář je většinou detekována několika různými sousedními okny, které jsou od sebe jen nepatrně posunuty a mají podobnou velikost, proto je potřeba tyto oblasti detekovat a označit jako jeden obličej. Pro snížení počtu nesprávných pozitivních detekcí přepokládám, že každý obličej je detekován minimálně N výřezy.

Za pomocí testování zjistím, jaké nastavení počtu sousedních oken a násobku změny měřítka je tento detektor ideální.

5.2.2 Filtrace a označení obličejů

Výsledný počet detekovaných osob často obsahuje i nesprávně určené pozitivní a negativní detekce, které je potřeba co nejvíce minimalizovat. Pokud se jedná o fotografie tribuny nebo davu lidí na určitém místě, dá se předpokládat, že všechny snímané osoby budou zhruba ve stejné vzdálenosti, a tím pádem budou mít zhruba stejnou velikost. Při sledování fotografií s lidmi na tribunách jsem vypozoroval, že některé nesprávné pozitivní detekce jsou mnohonásobně větší nebo menší než správné pozitivní detekce. Proto jsem se rozhodl, že při počítání lidí na tribunách, z výsledného počtu osob, odstraním tyto příliš velké nebo malé detekce, čímž dojde k snížení počtu nesprávně určených pozitivních detekcí. Jako nejvhodnější se zdá být zvolena hranice nastavena na dvojnásobek průměrné hodnoty a polovinu průměrné hodnoty.

Dále jsem vypozoroval, že některé nesprávně určené pozitivní detekce částečně zasahují do správně určených pozitivních detekcí, případně že některé tváře jsou detekovány dvakrát. Toto jsem se rozhodl vyřešit tím, že nacházím dvojice obdélníků detekující obličeje, u nichž existuje průnik. Jeden z těchto dvou obdélníků odstraním, protože téměř se stoprocentní jistotou ho mohu označit jako nesprávný nebo nadbytečný.

V poslední části programu vyznačím do výstupní panoramatické fotografie nalezené tváře do obdélníčku a vytisknu jejich počet.

6 Implementace

V této kapitole budu popisovat implementační část programu. Tako kapitola vychází z kapitoly 5 Návrh.

6.1 Použité nástroje

K implementaci tohoto projektu jsem použil programovací jazyk C++ a knihovnu OpenCV (verze 2.4.6).

Knihovna OpenCV (Open Source Computer Vision Library) (26) je open source knihovna počítačového vidění a softwaru pro strojové učení. Knihovna je dostupná pod licencí BSD, a proto ji lze zdarma používat a upravovat jak pro akademické, tak pro komerční účely. Knihovna OpenCV je dostupná pro programovací jazyky C++, C, Python, Java a MATLAB a podporuje operační systémy Windows, Linux, Android a Mac OS. Obsahuje více než 2500 optimalizovaných algoritmů, které obsahují komplexní sadu algoritmů počítačového vidění a strojového učení. Velké množství mnou potřebovaných algoritmu je již v knihovně OpenCV implementováno a tím pádem mi ulehčí velké množství práce, a proto jsem se rozhodl využít tuto knihovnu. (26)

Jazyk C++ jsem se rozhodl využít mimo jiné proto, že je jedním z podporovaných jazyků knihovny OpenCV. Jeho výhodou oproti samotnému C je vyšší míra abstrakce a možnost využít objektově orientované knihovny.

Pro vývoj programu jsem použil vývojové prostředí Microsoft Visual Studio 2012, které jsem používal pod operačním systémem Windows 8.

6.2 Spojování fotografií

Před samotným prováděním spojování fotografií, musím nejprve načíst fotografie. Fotografie načítám do třídy `Mat`, která je součástí knihovny OpenCV a která je vhodná pro práci s fotografiemi.

Spojování fotografií jsem se rozhodl naimplementovat pomocí dvou způsobů. Prvním způsobem je naimplementovat si algoritmus sám a použít knihovnu OpenCV jen na dílčí úkony a druhým způsobem je použití přímo třídy z knihovny OpenCV určené ke spojování fotografií.

Ke spojování fotografií jsem vytvořil třídu `ImageStitching.cpp`. Použité metody pracují s obrázky v odstínech šedi, proto v konstruktoru třídy načítám obrázky, které pak převádím do odstínu šedi.

6.2.1 Vlastní metoda

Nejprve popíšu implementaci vlastního spojování fotografií. To jsem naimplementoval do metody `stitchOwn()`.

Při použití tohoto způsobu postupuji v jednotlivých krocích podle návrhu. V prvním kroku musím detekovat klíčové body. K detekci klíčových bodů používám třídu `FeatureDetector`, u které lze nastavit různé druhy příznakových detektorů jako např. SURF, SIFT nebo OBR. Detektor klíčových bodů, kterému byly předány obrázky, vrací seznam nalezených pozicí. V dalším bodě je počítán pro každý klíčový bod jeho deskriptor, k tomu je použita třída `DescriptorExtractor` a i zde si lze vybrat z několika typů extraktorů jako např. SURF, SIFT nebo OBR. Detektoru extraktorů, kterému byly předány obrázky a klíčové body, vrací seznam vypočtených deskriptorů. Dále jsou nalezeny shodné deskriptory, a to pomocí třídy `DescriptorMatcher`. I zde lze volit z několika způsobů hledání shodných deskriptorů, jako jsou Brute force a FLANN. Ze shodných deskriptorů jsou dále nalezeny shodné deskriptory s nejmenší vzdáleností. V další části se bude pracovat jen s „dobrými" shodami. Jako „dobré" shody jsou brány shody, které mají vzdálenost maximálně 3× větší, než je vzdálenost nejmenší shody. Tato hodnota byla zvolena experimentálně. Pro všechny „dobré" shody získám klíčové body pro jednotlivé obrázky. Dále již budu počítat homografii. K tomu používám funkci `findHomography()`, které pomocí parametrů nastavím výpočet homografie pomocí RANSACu a předám jí seznamy s klíčovými body. Po nalezení homografie použiji funkci `warpPerspective()`, která aplikuje na druhý obrázek geometrické transformace určené homografií, které druhý obrázek natočí tak, aby odpovídal prvnímu obrázku. Pak už jen připojím druhou, transformovanou fotografii k té první. Konečné geometrické transformace již provádím s původními barevnými fotografiemi.

Nevýhodou tohoto způsobu implementace je, že zde nejsou doimplementovány pokročilejší části spojování fotografií, jako je promítání na válec a blending.

6.2.2 Použití třídy z knihovny OpenCV

Druhou možností spojování fotografií je použití přímo třídy z knihovny OpenCV. Tento způsob mám naimplementován v metodě `stitch()`.

Ona třída z knihovny OpenCV, kterou využiji ke spojování fotografií, se jmenuje `Stitcher`, ze které si vytvořím instanci. Výhodou této třídy je, že nepoužívá jednu striktní implementaci, ale před samotným spojením fotografií mohu volat metody této třídy, kterými mohu měnit vlastnosti spojování fotografií, což je velmi pozitivní vlastnost pro budoucí testování. Pomocí metody `setFeaturesFinder()` lze nastavit, jaký detektor klíčových bodů bude použit a s jakým nastavením. Na výběr máme samozřejmě mimo jiné detektory SURF, OBR, atd. Pomocí metody `setFeaturesMatcher()` lze zase zvolit požadovaný algoritmus na hledání shodných deskriptorů. Projekci lze měnit pomocí metody `setWarper()`, na výběr mám projekci na rovinu, na válec, na kouli a mnohé další. Dále lze měnit např. blending pomoci `setBlender()` a mnoho dalších vlastností. Samotné spojování fotografií provádí metoda `stitch()`, které předám seznam fotografií a ona vrátí jednu výslednou panoramatickou fotografii s požadovaným nastavením. Této metodě lze zadat i barevné fotografie.

6.3 Detekce obličejů

V další části budu popisovat implementaci algoritmů na rozpoznávání obličejů. Tuto část mám naimplementovanou ve třídě `PeopleCounting`. I zde nejprve v konstruktoru třídy načtu barevnou fotografii, kterou pak převedu do odstínu šedi. Jak již bylo zmíněno v návrhu, k detekci obličejů použiji detektor Viola-Jones.

6.3.1 Implementace Viola-Jones

Prvním algoritmem pro detekci obličejů, který jsem implementoval, je detektor Viola-Jones. Tento algoritmus jsem naimplementoval do metody `violaJones()`.

Velkou výhodou implementace tohoto algoritmu s pomocí knihovny OpenCV pro detekci obličejů je, že není potřeba natrénovat vlastní kaskádový klasifikátor Haarových příznaků s obličeji, ale ten je již součástí knihovny OpenCV, a proto používám tento již předem natrénovaný klasifikátor obličejů. Kaskádový klasifikátor vytvářím jako instanci třídy `CascadeClassifier`.

Když mám natrénovaný klasifikátor, dále musím projít postupně oknem v několika měřítcích a v každém podokně určit na základě kaskádového klasifikátoru, zda se jedná o podokno s obličejem či ne. I toto je již v knihovně OpenCV naimplementováno. Nad kaskádovým klasifikátorem stačí zavolat metodu `detectMultiScale()`, které je pomocí parametrů předána vstupní fotografie a požadované nastavení a ona vrátí seznam obdélníků, které ohraničují detekované obličeje. U příslušné metody lze nastavit, kolikrát se má změnit velikost podokna v každém měřítku, kolik sousedních podoken musí detekovat potenciální obličej, aby byl detekován jako obličej, a minimální a maximální velikost obličeje v pixelech. Pomocí testování bych měl najít ideální nastavení těchto parametrů.

6.3.2 Filtrování a označení obličejů

První způsob filtrace nesprávně určených pozitivních detekcí je odstranění příliš velkých a příliš malých detekcí. To jsem naimplementoval do metody `terracesFaces()`. Implementace tohoto algoritmu spočívá v tom, že si nejprve spočítám průměrnou výšku a průměrnou šířku detekovaných potenciálních obličejů. Následně projdu seznam všech detekovaných potenciálních obličejů a odstraním ty obličeje, jejichž výška nebo šířka je dvakrát větší nebo menší než je průměrná výška nebo šířka.

Druhým způsobem filtrace je odstranění překrývajících se potenciálních obličejů. To jsem naimplementoval do metody `overlapFaces()`. Algoritmus je naimplementován tak, že nad každou dvojici obdélníků detekujících potenciální obličeje provede operaci průniku a pokud průnikem těchto dvou obdélníků vznikne nový obdélník o určité výšce a šířce, tyto dva obdélníky se překrývají a druhý obdélník je tedy nadbytečný, a proto jej odstraním.

Nakonec je zde ještě metoda `rectangledFaces()`. Tato metoda pomocí funkce z OpenCV `rectangle()` ohraničí všechny nalezené obličeje do obdélníků.

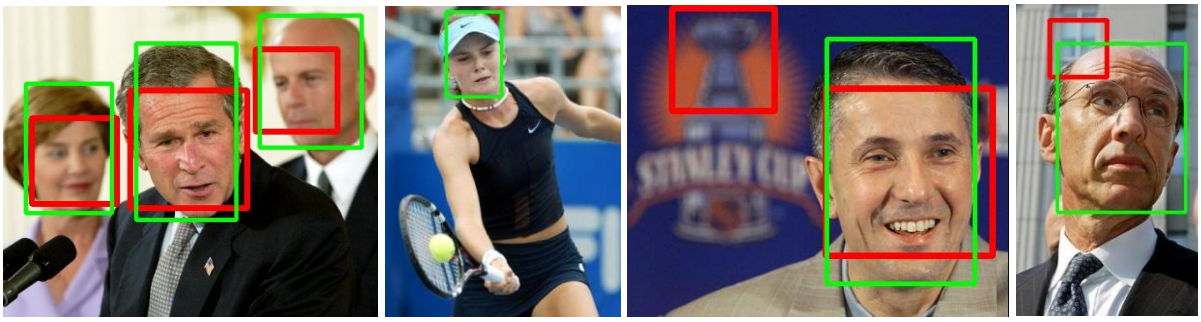
7 Testování a vyhodnocení výsledků

V této kapitole se budu nejprve zabývat testováním a vyhodnocením detekce obličejů a spojováním fotografií. V poslední, nejdůležitější podkapitole provedu testování a vyhodnocení počítání lidí ze skutečně vytvořené panoramatické fotografie.

7.1 Testování a vyhodnocení detekce obličejů

V první části testování budu testovat kvalitu a rychlost detekce obličejů a porovnám oba použité detektory. Testování budu provádět tak, že na sadě fotografií vyhledám pomocí použití detektorůu naimplementovaného v programu obličeje a ty pak porovnám se skutečnými obličeji, které se nacházejí na fotografii, a pro každou fotografii automaticky určím počet správných pozitivních detekcí (true positive - TP), nesprávně určených pozitivních detekcí (false positive - FP) a nesprávně určených negativních detekcí (false negative - FN).

K tomu jsem využil volně dostupnou databázi obrázků, kterou vytvořili v University of Massachusetts (27). Tato databáze obsahuje obrovský počet fotografií, které jsou nashromážděné z internetu. K samotnému testování jsem využil předpřipravené sady obrázků, které obsahují náhodně vybrané obrázky z databáze společně s anotacemi obličejů. Mnou používaná sada obrázků, nad kterou budu provádět experimenty, obsahuje 712 fotografií s 959 obličeji. Některé fotografie i s vyznačenými obličeji jsou zobrazeny na obrázku 7.1.



Obrázek 7.1: Ukázka obrázků použitých při testování. Červené obdélníky znázorňují detekce vypočtené programem a zelené obdélníky znázorňují referenční obličeje. Nalevo je obrázek se správnou detekcí, druhý zleva je obrázek s nesprávně určenou negativní detekcí, třetí zleva je obrázek s nesprávně určenou pozitivní detekcí a obrázek napravo obsahuje jak nesprávně určenou pozitivní detekci, tak nesprávně určenou negativní detekci. Obrázky použity z (27)

Pro testování detekce obličejů jsem vytvořil novou třídu s názvem `Testing`. Tato třída čte z vstupního textového souboru názvy fotografií, u kterých potom pomocí detektoru nalezne obličeje. V druhém textovém souboru jsou uloženy ke každé fotografii anotace obdélníků obsažených obličejů. Každý detekovaný obličej je porovnáván s každým anotovaným obličejem se snahou nalézt shodné

obličeje. Problém je, že oba obdélníky určující obličej nebudou téměř nikdy úplně shodné. Proto jako správnou detekci beru tu, jejichž průnik obdélníků je alespoň 40%. Toto procento se může jevit jako malé, ale problém je, že ony anotace určují celou hlavu osoby, kdežto detektor nachází pouze její obličej. Pro každou fotografii jsou spočítány správné pozitivní detekce, nesprávné pozitivní detekce a nesprávné negativní detekce a na konci testování je vypsán součet těchto hodnot pro všechny testované fotografie.

Protože testování je prováděno na různorodé směsici obrázků, kde se nemusí jednat pouze o fotografie lidí na tribunách, není pro tento případ vhodné používat metodu na odstranění příliš velkých nebo příliš malých potenciálních detekcí obličejů ani metodu na odstranění překrývajících se potenciálních detekcí.

Testování provedu postupně pro 3 různé hodnoty změny měřítka, a to 1,05 a 1,1 a 1,2. Změna měřítka udává, kolikrát bude zmenšena velikost okna v každém měřítku. Změnu měřítka budu provádět až do velikosti 12×12, tato hodnota byla zvolena proto, že v databázi jsou i velmi malé obličeje a pokud by tato velikost byla vyšší, byly by tyto obličeje diskriminovány. V každém měřítku budu testovat parametr, který udává počet sousedních oken, který detektor označí jako obličej, potřebný k detekci obličeje. Tento parametr budu nastavovat na hodnoty od 1 až po 20. Zjištěné hodnoty zapíši do tabulky a dopočtu k nim hodnoty precision a recall. Precision udává pravděpodobnost, že náhodně vybraná detekce je správná. Vypočítá se pomocí vzorce (28):

$$precision = \frac{true\ positive}{true\ positive + false\ positive} \tag{7.1}$$

Recall udává pravděpodobnost, že náhodně vybraný obličej je detekován správně. Vypočítá se pomocí vzorce (28):

$$recall = \frac{true\ positive}{true\ positive + false\ negative} \tag{7.2}$$

Pokud jsou vypočítány hodnoty precision a recall, je vhodné určit, jaký poměr těchto hodnot je nejvhodnější. K tomu se nejčastěji používá F-measure, což je harmonický průměr hodnoty precision a recall. Vzorec pro jeho výpočet vypadá takto (29):

$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{7.3}$$

Čím je hodnota F-measure blíže 1, tím je dané nastavení detektoru kvalitnější.

Nejprve provedu testování detektoru Viola-Jones pro hodnotu jednoho kroku změny měřítka nastavenou na 1,05. Při nastavení této hodnoty trvala detekce předpřipravené sady obrázků průměrně 144 s. Při změně počtu oken nutných k detekci se doba detekce nemění, ta je závislá pouze na změně hodnoty měřítka. Naměřené hodnoty jsou zaznamenány v tabulce 7.1, kde PO je počet sousedních oken nutných k detekci, DO jsou detekované obličeje, TP (true positive) jsou správné pozitivní detekce obličeje, FP (false positive) jsou nesprávné pozitivní detekce obličeje, FN (false negative) jsou nesprávné negativní detekce a F je F-measure.

PO	DO	TP	FP	FN	Precision	Recall	F
1	1322	917	405	42	0,6936	0,9562	0,8040
2	1097	907	190	52	0,8268	0,9458	0,8823
3	1012	899	113	60	0,8883	0,9374	0,9122
4	970	891	79	68	0,9186	0,9291	0,9238
5	934	885	49	74	0,9475	0,9228	0,9350
6	914	875	39	84	0,9573	0,9124	0,9343
10	864	851	13	108	0,9850	0,8874	0,9336
20	796	794	2	165	0,9975	0,8279	0,9048

Tabulka 7.1: Detekce pomocí Viola-Jones s krokem měřítka 1,05

Dále provedu další test s hodnotou změny měřítka 1,1. Pro toto měřítko trvala detekce všech obrázků průměrně 86 s. Naměřené hodnoty jsou zaznamenány v tabulce 7.2.

PO	PD	TP	FP	FN	Precision	Recall	F
1	1092	904	188	55	0,8278	0,9426	0,8815
2	966	883	83	76	0,9141	0,9208	0,9174
3	905	864	41	95	0,9547	0,9009	0,9270
4	878	855	23	104	0,9738	0,8916	0,9309
5	862	845	17	114	0,9803	0,8811	0,9281
6	849	836	13	123	0,9847	0,8717	0,9248
10	797	795	2	164	0,9975	0,8290	0,9055
20	674	674	0	285	1,0000	0,7028	0,8255

Tabulka 7.2: Detekce pomocí Viola-Jones s krokem měřítka 1,1

Poslední testování detekce s algoritmem Viola-Jones bylo provedeno s hodnotou změny měřítka 1,2. Zde byl průměrný čas detekce 66 s. Naměřené hodnoty jsou zaznamenány v tabulce 7.3.

PO	PD	TP	FP	FN	Precision	Recall	F
1	966	872	94	87	0,9027	0,9093	0,9060
2	887	854	33	105	0,9628	0,8905	0,9252

3	845	834	11	125	0,9870	0,8697	0,9246
4	818	813	5	146	0,9939	0,8478	0,9150
5	799	795	4	164	0,9950	0,8290	0,9044
6	775	772	3	187	0,9961	0,8050	0,8904
10	678	678	0	281	1,0000	0,7070	0,8283
20	376	376	0	583	1,0000	0,3921	0,5633

Tabulka 7.3: Detekce pomocí Viola-Jones s krokem měřítka 1,2

V tabulce 7.4 jsou uvedeny časy detekce pro jednotlivé kroky měřítka.

Změna měřítka	**Čas detekce [s]**
1,05	144
1,1	86
1,2	66

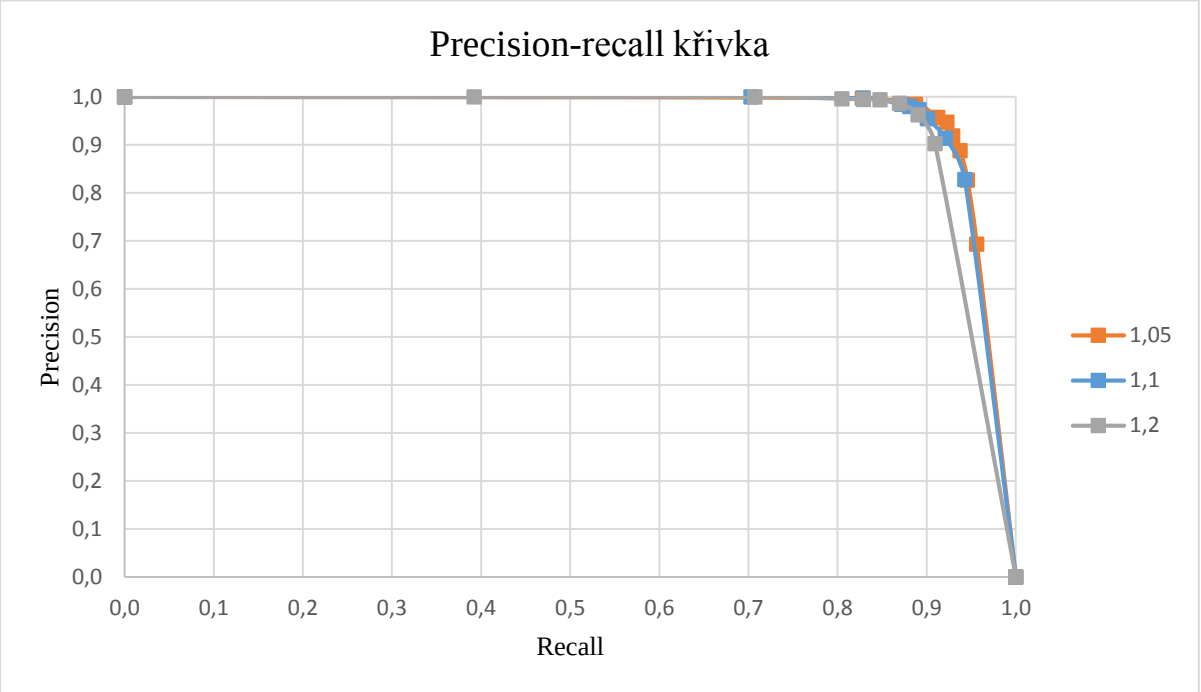
Tabulka 7.4: Časy detekce pro jednotlivé kroky měřítka.

Z naměřených výsledků jsem vytvořil precision-recall (obrázek 7.2) a ROC (obrázek 7.3) křivky, kde každá hodnota změny měřítka tvoří jednu křivku. U precision-recall křivky je osa x definována jako recall a osa y jako precision. U precision-recall křivky víme, že každá křivka začíná v bodě [0,1] a končí v bodě [1,0]. Čím blíže je tato křivka k bodu [1,1], tím ji lze považovat za méně chybovou. ROC (receiver operating characteristic) křivka je definována jako poměr nesprávně určených pozitivních detekcí k poměru správně určených pozitivních detekcí (recall). U ROC křivky víme, že každá křivka začíná v bodě [0,0] a končí v bodě [1,1]. Čím víc se tento druh křivky blíží bodu [0,1], tím ji lze považovat za méně chybovou. (28)

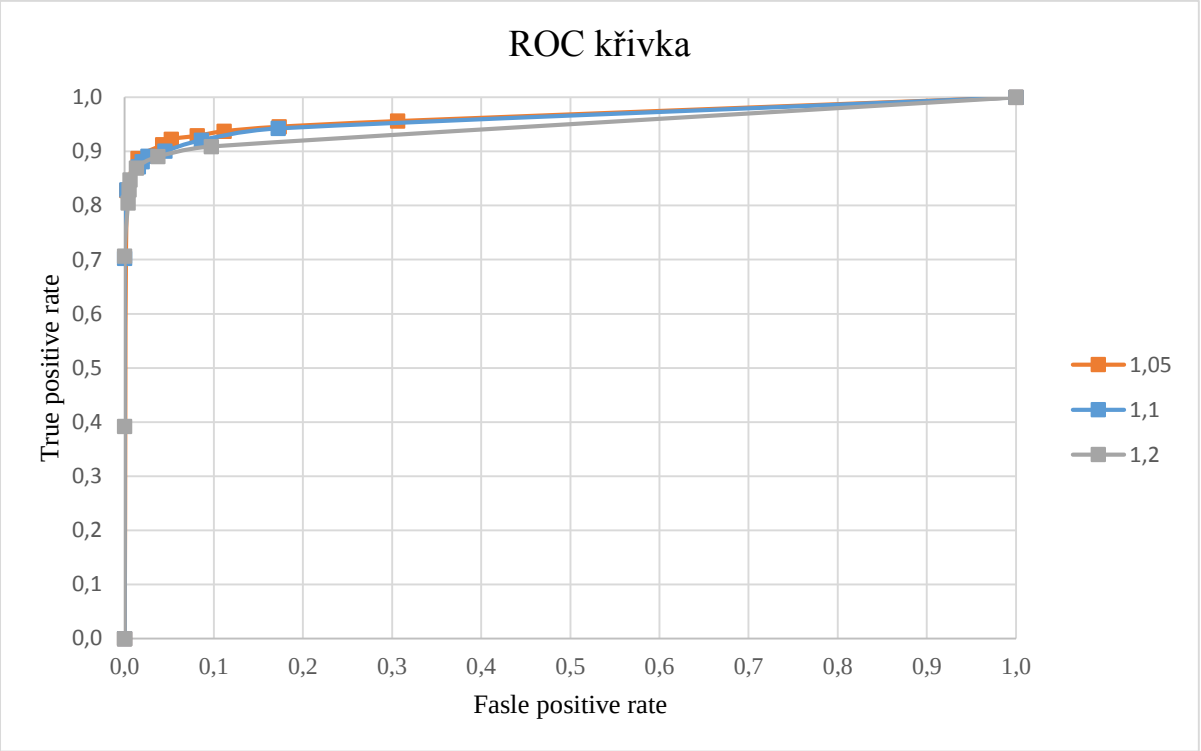
Z vytvořených grafů lze vyčíst, že nejlepších výsledků lze dosáhnout se změnou měřítkem 1,05, o něco horší výsledky jsou se změnou 1,1 a nejhorších výsledků dosáhneme použitím změny měřítka 1,2. Dá se předpokládat, že při ještě větší změně měřítka budou výsledky ještě horší. Ovšem při zmenšování změny měřítka výrazně roste čas, který je potřebný pro detekci. U změny měřítka 1,05 je tento čas 144 s, u změny měřítka 1,1 je to 86 s a u změny měřítka 1,2 je to 66 s. Proto použití menší změny měřítka nemusí být vždy pro daný problém nejvhodnější. Pokud bychom potřebovali použít detektor, pro nějakou aplikaci pracující v reálném čase, potom se jeví jako vhodnější použít změnu měřítka 1,1. Tato změna měřítka dosahuje jen nepatrně horší detekce, zato je výrazně rychlejší. Při použití změny měřítka 1,2 je již detekce více nepřesná a nedosahuje výrazného zrychlení, proto je tato změna měřítka vhodná jedině v případě, kdy potřebujeme opravdu velmi rychlou detekci a použití menší změny měřítka je nedostačující.

Pomocí hodnoty F-measure se dá zjistit, jaký počet oken potřebných pro detekci obličeje je pro danou změnu měřítka nejvhodnější. Pro změnu měřítka 1,05 lze dosáhnou nejlepších výsledků detekce obličeje použitím minimálně 5 oken, pro změnu měřítka 1,1 jsou to 4 okna a pro změnu měřítka 1,2 jsou to 2 okna.

Z dosažených výsledků jsem zjistil, že detektor Viola-Jones dosahuje nejlepší detekce, pokud je použita změna měřítka 1,05 a minimální počet oken nutný k detekci obličeje je nastaven na 5 nebo pokud je použita změna měřítka 1,1 a minimální počet oken nutný k detekci obličeje je nastaven na 4. Vhodné natavení parametrů vybíráme podle toho, jestli požadujeme tu nejpřesnější detekci nebo jestli nám stačí o něco málo horší detekce s výrazně nižším časem.



Obrázek 7.2: Graf znázorňující precision-recall křivku pro detektor Voila-Jones pro tři různé hodnoty změny měřítka



Obrázek 7.3: Graf znázorňující ROC křivku pro detektor Voila-Jones pro tři různé hodnoty změny měřítka

7.2 Testování a vyhodnocení spojování fotografií

V další části budu testovat kvalitu a rychlost spojování fotografií. U spojování fotografií je velkou nevýhodou, že automatické vyhodnocování testů je zde prakticky nemožné. Proto budu provádět testování tak, že dané nastavení provedu na několika fotografiích a z výsledku se pak pokusím metodou „od oka“ určit tu nejlepší. Jediným objektivně měřitelným parametrem je čas tvorby panoramatické fotografie.

K testování jsem používal sadu obrázků, která obsahovala jak moje vlastní fotografie, tak fotografie z (29).

Nejprve provedu testování spojování fotografií za pomocí metody z knihovny OpenCV. První budu zkoušet experimentovat s deskriptory. Jako první jsem zvolil deskriptor SURF. U tohoto deskriptoru jde nastavit hodnota prahu pro detekci klíčových bodů. Pouze příznaky, které mají hodnotu hessiánu větší než je tento práh jsou detekovány. To znamená, že čím vyšší nastavíme hodnotu prahu, tím méně klíčových bodů bude nalezeno. Testováním jsem ověřil fakt, že zvyšování hodnoty prahu má pozitivní vliv na rychlost spojování fotografií, např. jestliže je detektoru nastaven práh 100, pak spojení fotografií trvá 560 ms, pokud pro stejné dva obrázky zvolíme hodnotu prahu 500, pak spojení fotografií trvá jen 340 ms. Ovšem při přílišném zvětšení této hodnoty detektor najde tak málo bodů, že výpočet homografie již není možný. Na výslednou podobu panoramatické fotografie tento parametr nemá téměř

žádný vliv, dochází jen k minimálním změnám ve výsledné fotografii. Jako ideální nastavení tohoto parametru se jeví 500.

Druhým deskriptorem, který otestuji je deskriptor ORB. U tohoto detektoru lze nastavit hodnotu maximálního počtu zapamatovaných příznaků. Testování bylo dokázáno, že snižováním počtu příznaků klesá čas výpočtu. Pro spojení dvou fotografií dohromady je tento detektor o něco málo rychlejší než detektor SURF, ovšem při zvětšování počtu fotografií se detektor ORB stává pomalejším. Největší nevýhodou použití tohoto deskriptoru je, že i při nastavení velkého počtu maximálně zapamatovaných příznaků nemusí vždy docházet ke správnému spojení fotografií.

Dalším parametrem, který budu testovat, je blending. Porovnám mezi sebou bleding prováděný pomocí feather blender a multi-band blender. Porovnání provádím nad deskriptorem SURF. Testování bylo zjištěno, že feather blender bývá průměrně o 15 % rychlejší. Zato ale nedosahuje tak kvalitního blendingu jako multi-band blender. Porovnání obou blenderu je zobrazeno na obrázku 7.4. Na obrázku je patrné, že při použití father blendingu nejsou přechody úplně hladké a na fotografii se tvoří „mapy“. Toto by mohlo při rozpoznávání obličejů výrazně ztížit jejich detekci.

Obrázek 7.4: Porovnání dvou blenderů. Nalevo je výřez fotografie s použitím multi-band blendingi a napravo je výřez fotografie s použitím father blendingu. Zejména si pak povšimněte jasně viditelného přechodu na pravé fotografii označeného červenou elipsou.

Nakonec porovnám metodu na spojování fotografií z knihovny OpenCV s mnou naimplementovanou metodou. Mnou naimplementovaná metoda neobsahuje promítání na válec ani blending. Při spojení fotografií jde zřetelně vidět šev. Nevětší problém nastává, pokud fotografujeme nestálé objekty, jako například lidi. Zde může docházet k „rozpůlení“ obličeje, což může způsobit budoucí neschopnost detekovat obličej. Avšak tento detektor je až 3× rychlejší než detektor z knihovny, což je také důsledkem chybějícího promítání a blendingu. Porovnání obou metod je na obrázku 7.5.

Obrázek 7.5: Porovnání vlastní metody ke spojování fotografií (vlevo) s metodou z knihovny OpenCV (vpravo). U výřezu fotografie nalevo jde zřetelně vidět šev a dochází tam k půlení obličejů.

7.3 Testování a vyhodnocení počítání lidí z panoramatické fotografie

V této části testování se již budu zabývat tím, na co přesně byla aplikace navržena, a to určování počtu lidí z panoramatické fotografie. K tomu využiji poznatků z předchozích podkapitol.

K testování použiji vlastní sadu fotografií. Tato sada obsahuje fotografie zaplněné tribuny fotbalového stadiónu při zápase. Tribuna se nevešla do jednoho snímku a je tvořena vždy větším počtem snímků, což je ideální případ k testování. Ukázka výsledné panoramatické fotografie s vyznačenými obličeji je zobrazena na obrázcích 7.6 a 7.7.

Protože fotografie jsou snímané na velkou dálku a obličeje na nich jsou poměrně malé, budu pracovat s fotografiemi v plném rozlišení. Jedna fotografie má rozlišení 4928×3264 pixelů.

V předchozích podkapitolách jsem zjistil, že jako nejvhodnější se jeví použít ke spojování fotografií metodu z OpenCV s detektorem klíčových bodů SURF a s multi-band blendingem. K detekci obličejů se zase hodí použít detektor Viola-Jones se změnou měřítka 1,1 a minimálním počtem oken nutných k detekci obličeje nastavených na čtyři. Proto provedu testování reálných fotografií s tímto nastavením.

Testování jsem prováděl tak, že jsem spustil program s požadovanými fotografiemi a z výstupní panoramatické fotografie jsem ručně spočítal skutečný počet osob nacházejících se na fotografii, počet nesprávných potenciálních detekcí obličejů, počet nedetekovaných obličejů a tyto hodnoty jsem porovnal s výstupem programu udávající počet potenciálně nalezených osob. Toto jsem provedl pro dvě výstupní panoramatické fotografie (obrázky 7.6 a 7.7) a pro obě jsem ještě porovnal hodnoty bez použití filtrace nesprávně určených pozitivních detekcí obličejů a s použitím. Nutno podotknout, že při

opakovaném použití na stejných datech se mohou výsledky mírně lišit, což je způsobeno tím, že ke spojování fotografií se na výpočet homografie používá RANSAC, který je založen na náhodném výběru, a tím pádem může být výsledná panoramatická fotografie vždy trochu jiná. Výsledky testování jsou zobrazeny v tabulce7.6, kde PO je počet oken nutných k detekci, SO jsou skutečné obličeje na fotografii, DO jsou detekované obličeje, FP jsou nesprávně určené pozitivní detekce, FN jsou nesprávně určené negativní detekce a F je F-measure. Časy tvorby panoramatické fotografie, detekce obličejů a celkové časy jsou zobrazeny v tabulce 7.5.

Obrázek 7.6: Ukázka testovací fotografie č. 1. Výsledná panoramatická fotografie celé tribuny s vyznačenými obličeji. Složená z 11 fotografií. (Ve snížené kvalitě, ukázka v plné kvalitě je součástí CD.)

Obrázek 7.7: Ukázka testovací fotografie č. 2. Výsledná panoramatická fotografie prostředního sektoru tribuny s vyznačenými obličeji. Složená z 11 fotografií. (Ve snížené kvalitě, ukázka v plné kvalitě je CD.)

Obrázek	**Čas spojení [s]**	**Čas detekce [s]**	**Celkový čas [s]**
1	44	81	125
2	51	53	104

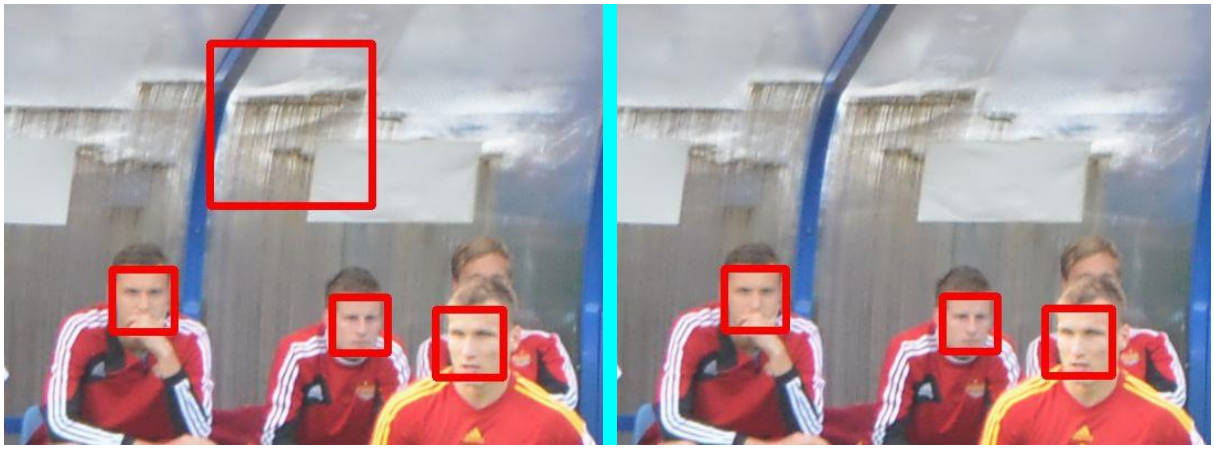
Tabulka 7.5: Čas výpočtu výstupní panoramatické fotografie s detekcí obličejů

Obr.	PO	Filtr	SO	DO	FP	FN	Prec.	Recall	F
1	4	bez	1613	1371	28	270	0,9796	0,8326	0,9001
1	4	s		1360	17	270	0,9875	0,8326	0,9035
1	2	bez		1508	117	222	0,9224	0,8624	0,8914
1	2	s		1459	68	222	0,9534	0,8624	0,9056
2	4	bez	288	258	11	41	0,9574	0,8576	0,9048
2	4	s		256	9	41	0,9648	0,8576	0,9081
2	2	bez		296	46	38	0,8446	0,8681	0,8562
2	2	s		281	31	38	0,8897	0,8681	0,8787

Tabulka 7.6: Porovnání výsledků výstupních panoramatických fotografií, kde PO je počet oken nutných k detekci, SO jsou skutečné obličeje na fotografii, DO jsou detekované obličeje, FP jsou nesprávně určené pozitivní detekce, FN jsou nesprávně určené negativní detekce a F je F- measure.

Z výsledků lze vidět, že aplikace dosahuje celkem kvalitních výsledků. Je ale patrné, že počet nesprávně detekovaných negativních detekcí je výrazně vyšší než počet nesprávně určených pozitivních detekcí, v důsledku toho je počet potenciálních detekcí obličejů dosti nižší než počet skutečných obličejů. Pokud bychom chtěli mít počet potenciálních detekcí obličejů shodný s počtem skutečných obličejů, pak by musel být počet nesprávných pozitivních detekcí shodný s počtem nesprávných negativních detekcí. Z tabulky 7.2 je patrné, že poměr nesprávných pozitivních a nesprávných negativních detekcí je si nejpodobnější při nastavení počtu oken nutných k detekci na 2. Proto jsem provedl ještě testování s touto hodnotou a vyznačil jsem výsledky do tabulky 7.6. Při nastavení této hodnoty opravdu dostáváme přesnější výsledky počtu detekovaných osob. Zato ale klesá hodnota F-measure. K dalšímu snížení chybných detekcí lze dospět pomocí použití filtrace nesprávně určených pozitivních detekcí obličejů. Ukázka některých odstraněných nesprávných pozitivních detekcí je zobrazena na obrázcích 7.8 a 7.9. V některých případech bylo dosaženo odstranění téměř 50 % nesprávně určených pozitivních detekcí a tím byla také zvýšena hodnota F-measure. V jednom případě je dokonce hodnota F-measure při použití dvou oken nutných k detekci po filtraci nesprávných pozitivních detekcí vyšší než při použití čtyř oken.

Obrázek 7.8: Porovnání dvou výřezů fotografií, nalevo bez použití odstranění extrémních a překrývajících se obdélníků, napravo s použitím.



Obrázek 7.9: Porovnání dvou výřezů fotografií, nalevo bez použití odstranění extrémních a překrývajících se obdélníků, napravo s použitím.

Při detekci velkého davu lidí však nastává problém, že ne všechny obličeje lidí jsou detekovatelné, například velký problém je při detekci lidí, kteří se nedívají před sebe, kteří mají něco před obličejem nebo kteří mají nasazenu kšiltovku příliš do tváře. U těchto osob se musíme smířit s tím, že tyto osoby nebudou detekovány ani při sebebenevolentnějším natavení detektoru. Ukázka téměř nedetekovatelných obličejů je na obrázku 7.10.

Obrázek 7.10: Ukázka téměř nedetekovatelných obličejů.

Z dosažených výsledků nastává dilema, zda zvolit detektor s nižší chybovostí nebo detektor, jehož výsledky se více blíží skutečným výsledkům. Pokud zvolíme druhou možnost, je problém v tom, že

výsledek není ani tak dán kvalitou detektoru jako spíš náhodou a v některých případech může dosahovat výrazně horších výsledku než první detektor. Proto jsem se rozhodl pro použití detektoru, který potřebuje čtyři sousední okna nutné k detekci obličeje.

8 Závěr

Cílem této práce bylo vytvořit program, který by byl schopný automaticky spočítat velké množství lidí, které se nevejde do jednoho snímku. To jsem vyřešil tak, že jsem spojil fotografie dohromady do jedné panoramatické a pak z ní spočetl obličeje. Výhodou tohoto řešení je, že není potřeba nacházet stejné obličeje na různých fotografiích, ale na jedné panoramatické fotografii. Mezi dostupnými aplikacemi jsem nenašel žádnou, která by se touto problematikou zabývala.

Práce je rozdělena na dvě základní části. První část je zaměřena na spojování fotografií. Tuto problematiku jsem řešil pomocí příznakově založených metod. Použil jsem dvě metody, a to vlastní metodu a metodu obsaženou v knihovně OpenCV. Výsledky testů jasně hovoří pro použití metody z OpenCV. Experimentováním s metodou z OpenCV jsem zjistil, že k detekci klíčových bodů je nejvhodnější použít deskriptor SURF s nastavením prahu pro detekci klíčových bodů na 500. Dále jsem prověřoval význam blendingu. Zjistil jsem, že při použití jednodušších blendingů není výsledná fotografie příliš kvalitní, jsou na ní vidět přechody a tvoří se na ni „mapy“, což by při pozdějším použití výrazně ztížilo detekci obličejů.

Druhou částí je počítání osob. Tuto část řeším tak, že pro každou osobu detekuji pouze její obličej a tyto obličeje pak spočítám. K detekci obličejů jsem se rozhodl použít detektor Viola-Jones. Tento detektor dosahuje nejlepších výsledků při nastavení změny kroku měřítka na 1,1 a počtu oken nutných k detekci na čtyři. Při tomto nastavení vychází hodnota na F-measure 0,9309. Pro snížení nesprávně určených pozitivních detekcí jsou z výsledného počtu detekovaných obličejů odstraněny obličeje, u nichž téměř určitě lze říct, že jsou nesprávné. Mezi odstraněné detekce patří detekce ohraničené příliš velkým nebo příliš malým obdélníkem a překrývající se detekce. Tato filtrace je však vhodná použít pouze pro detekci osob na tribunách nebo skupin lidí, které se nacházejí zhruba ve stejné vzdálenosti.

Celá aplikace byla nakonec otestována na reálných fotografiích, na kterých se nacházela zaplněná tribuna fotbalového stadiónu při zápase. Dosažené výsledky potvrdily, že tato aplikace je pro daný problém použitelná v praxi. Na kvalitu výsledku má však také vliv, v jaký okamžik je fotografie pořízena. Nejlepších výsledků je dosaženo, pokud jsou počítané osoby v relativním klidu a největší počet obličejů je natočeny jedním směrem.

Nyní se jedná o konzolovou aplikaci. Do budoucna by se tato aplikace dala ještě rozšířit o grafické uživatelské rozhraní. Dále by se daly, podobně jako u většiny programů na tvorbu panoramat, oříznout černé okraje u výsledné panoramatické fotografie, u toho by však hrozilo, že by došlo i k oříznutí obličeje a to by bylo nežádoucí.

Literatura

1. VIOLA, Paul a Michael JONES. *Robust Real-Time Face Detection.* [Online] Netherlands : Kluwer Academic Publishers, 2004. International Journal of Computer Vision 57.

2. PŘINOSIL, Jiří a Martin KROLIKOWSKI. *Využití detektoru Viola-Jones pro lokalizaci obličeje a očí v barevných obrazech.* [Online] Brno, Česká republika : Vysoké učení technické v Brně, 2008.

3. WILSON, Phillip Ian a John FERNANDES. *Facial feature detection using Haar classifiers.* [Online] 2006.

4. LIENHART, Rainer a Jochen MAYTD. *An Extended Set of Haar-like Features for Rapid Object Detection.* [Online] Santa Clara, CA 95052, USA : Intel Labs, Intel Corporation.

5. MATAS, Jiří a Jan ŠOCHMAN. *AdaBoost.* [Online] Prague : Czech Technical University.

6. DALAL, Navneet a Bill TRIGGS. *Histograms of Oriented Gradients for Human Detection.* Montbonnot : INRIA Rhône-Alpes, 2005.

7. ŽIŽKA, J. *Support vector machines (SVM).* [Online] Brno : Masarykova univerzita, 2004.

8. NG, Andrew. *The Simplified SMO Algorithm.* [Online] 2009.

9. POŠÍK, Petr. *Optimální rozdělující nadplocha. Support vector machine. Adaboost.* [Online] Prague : Czech Technical University in Prague, 2012.

10. SZELISKI, Richard. *Image Alignment and Stitching.* [Online] Redmond, WA 98052 : Microsoft Research, Microsoft Corporation, One Microsoft Way, 2005.

11. LOWE, David G. *Distinctive Image Features from Scale-Invariant Keypoints.* [Online] Vancouver, B.C., Canada : University of British Columbia, 2004.

12. BAY, Hubert, Andreas ESS, Tinne TUYTELAARS a Luc Van GOOL. *Speeded-Up Robust Features (SURF).* [Online] Zurich a Leuven : ETH Zurich a Katholieke Universiteit Leuven, 2006.

13. IWAMURA, Masakazu, Tomokazu SATO a Koichi KISE. *What Is the Most EfficientWay to Select Nearest Neighbor Candidates for Fast Approximate Nearest Neighbor Search?* [Online] Osaka : Graduate School of Engineering, Osaka Prefecture University, 2013.

14. KRŠEK, Přemysl. *Základy počítačové grafiky: Studijní opora IZG.* [Online] Brno : Fakulta informačních technologií Vysokého učení technického v Brně, 2006.

15. PELLACINI, Fabio. *Geometric transformations.* [Online] 2008.

16. SNAVELY, Noah. *Homographies and RANSAC.* [Online] Cambridge : MIT CSAIL, 2006.

17. BENEDA, Martin. Homografie a epipolární geometrie. *Trilobit.* 2010, 2.

18. FISCHLER, Martin a Robert C. BOLLES. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM.* Červen 1981, Sv. 24, 6, stránky 381-395.

19. NAVRÁTIL, Jan. *Transformace a RANSAC.* [Online] Brno : Vysoké učení technické v Brně.

20. DERPANIS, Konstantinos G. *Overview of the RANSAC Algorithm.* [Online] 2010.

21. BURT, Peter J. a Edward H. ADELSON. *A Multiresolution Spline With Application to.* [Online] místo neznámé : RCA David Sarnoff Research Center, 1983.
22. BROWN, M. a D. G. LOWE. *Recognising Panoramas.* [Online] Vancouver, Canada : University of British Columbia.
23. MATAS, Jiří. *Detekce objektů pomocí scanning window.* [Online] Prague : CzechTechnicalUniversity, 2010.
24. HAST, Anders, Johan Nysjö, Andrea Marchetti. *Optimal RANSAC - Towards a Repeatable Algorithm for Finding the Optimal Set.* [Online] Uppsala : Uppsala University.
25. HOIEM, Derec. *Image Stitching.* [Online] Urbana : University of Illinois, 2010.
26. *OpenCV.* [Online] [Citace: 2. květen 2014.] http://opencv.org/.
27. JAIN, Vidit a Eric LEARNED-MILLER. *FDDB: A Benchmark for Face Detection in Unconstrained Settings.* [Online] Amherst : University of Massachusetts, 2010. UM-CS-2010-009.
28. DAVIS, Jesse a Mark GOADRICH. *Parameter specifying how many neighbors each candidate rectangle should have to retain it.* [Online] Madison : Department of Computer Sciences and Department of Biostatistics and Medical Informatics, University of.
29. ROTHSCHILD, Hripscak. Agreement, the F-Measure, and Reliability in Information Retrieval. *Journal of the American Medical Informatics Association.* 2005, Sv. 12, 3.