VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

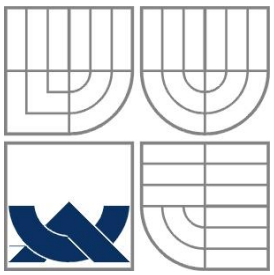
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

NA NĚCO UKAŽ A JÁ TI ŘEKNU, CO TO JE

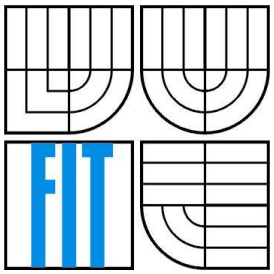
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE JAKUB DOHNAL
AUTHOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

NA NĚCO UKAŽ A JÁ TI ŘEKNU, CO TO JE

POINT AT SOMETHING AND I TELL YOU, WHAT IT IS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAKUB DOHNAL

VEDOUCÍ PRÁCE
SUPERVISOR

ING. VÍTĚZSLAV BERAN

BRNO 2009

Abstrakt

Tato práce se zabývá analýzou obrazu se zaměřením na detekci rukou a následné určení směru, kterým ruce ukazují. Práce je spojena s detekcí obličeje, detekcí barvy lidské kůže a trackováním objektů ve videu.

Abstract

This paper concerns video analysis, focusing on hand detection and following hand direction specification. The work includes topics such as face detection, skin-like color detection and tracking objects in a video sequence.

Klíčová slova

Zpracování obrazu, detekce barvy lidské kůže, detekce obličeje, trackování barevného objektu ve videu, CamShift, meanshift, haar-like rysy, openCV, openGL, barevné modely

Keywords

Image processing, skin-like color detection, face detection, color-based video tracking, CamShift, meanshift, haar-like features, openCV, openGL, color models

Citace

Jakub Dohnal: Na něco ukaž a já ti řeknu, co to je, bakalářská práce, Brno, FIT VUT v Brně, 2009

Na něco ukaž a já ti řeknu, co to je

Prohlášení

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

……
Jakub Dohnal
18. květen 2009

Poděkování

Rád bych poděkoval vedoucímu bakalářské práce, Ing. Vítězslavu Beranovi za poskytnutí odborných rad a technického zázemí a Mgr. Janě Skokanové za pomoc při získání testovacích dat.
Děkuji také své rodině a přátelům, kteří mě při práci podporovali.

Obsah

1 Úvod

Dnešní doba je plná automatizace. Stroje za nás vykonávají stále více úkonů, ke kterým potřebují vnímat své okolí. Počítačové vidění nám tak pomáhá v automobilech při parkování, v obchodech při sledování potenciálních zlodějů, na letišti při kontrole zavazadel, v medicíně, kriminalistice, ve zbrojním průmyslu, při prozkoumávání vesmíru a v mnoha, mnoha dalších oblastech lidské činnosti.

Lidé se již odedávna snaží zjednodušit ovládání přístrojů tak, aby bylo co nejintuitivnější a nejpřirozenější. Co může být intuitivnějšího a přirozenějšího než prosté ukázat a poručit. Práce „Na něco ukaž a já ti řeknu, co to je.“ je zaměřena na první část takového přístupu k ovládání.

2 Detekce kůže

Problematika detekce a analýzy pohybu rukou je spojena s detekcí kůže, detekcí obličeje, a trackováním pohybu rukou.

Vytvořením modelu lidské kůže se vědci zabývají už dlouhá léta. Bylo napsáno mnoho prací popisujících jednoduché i složitější způsoby jak detekovat lidskou kůži. Model lidské kůže můžeme využít v různých oblastech zpracování obrazu, mezi které patří určování obsahu obrázků a webových stánek, obsahově založená komprese obrázků, automatické úpravy kontrastu a pro tuto práci důležitá především detekce obličejů.

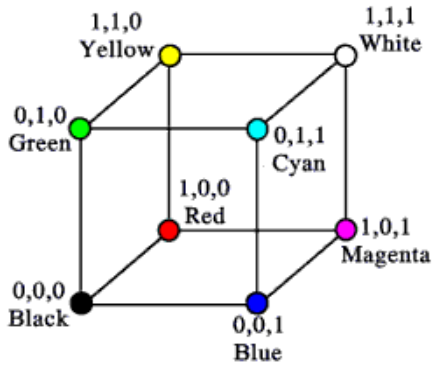
2.1 Barevné modely

Při detekci kůže a vlastně v celé oblasti počítačového vidění se využívá několika barevných prostorů. [1] Barevný model, někdy také barevný prostor, je abstraktní matematický model popisující způsob, kterým jsou základní barvy míchány do barvy výsledné. [2] Barvy jsou reprezentovány jako n-tice čísel, většinou trojice nebo čtveřice hodnot.

2.1.1 RGB model

Jedná se o model založený na červené (Red Channel), zelené (Green Channel) a modré (Blue Channel) barvě, jejichž *aditivním skládáním* získáváme další barvy.

Tento model je nejpoužívanějším modelem a je využíván při reprezentaci barev v různých knihovnách, programech, při tvorbě webových stránek apod.



Obrázek 1: Znázornění RGB modelu [11]

2.1.2 RGBA model

Jde o rozšířený RGB model o *složku průhlednosti* (Alpha Channel).

2.1.3 Normalizované RGB

Tento model také vychází z RGB modelu. Jeho barevné složky získáme z RGB pomocí následujících rovnic:

$$r = \frac{R}{R + G + B}$$

$$g = \frac{G}{R + G + B}$$

$$b = \frac{B}{R + G + B}$$

Normalizovanému RGB se také někdy říká rg model, protože třetí složku můžeme dopočítat z rovnice:

$$r + g + b = 1$$

Tento model svou normalizací ztrácí závislost mezi složkami r a g, která nese informaci o intenzitě jasu. [3] Proto bývá normalizované RGB využíváno v oblasti počítačového vidění ve scénách, které nejsou rovnoměrně osvíceny. Po normalizaci nám tedy zůstávají jen „čisté barvy".

2.1.4 Model HSV

[4] Model tvořený třemi složkami, *barevným tónem* (Hue), *sytostí* (Saturation) a *jasem* (Value). Je známý také jako HSB model, jas (Brightness). Je založen na intuitivním způsobu lidského vnímání barvy. [5] Intuitivnost modelu a oddělenost jasu od barvy mu přidaly na oblíbenosti v oblasti výzkumu detekce kůže.

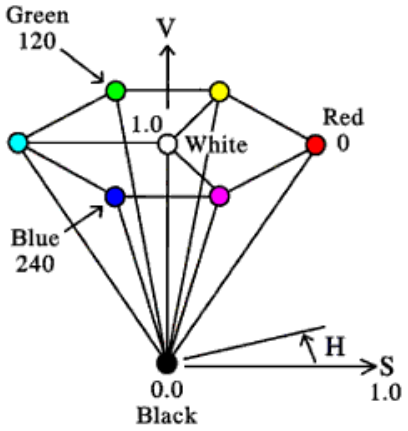
Složky jsou z RGB modelu odvozeny z rovnic:

$$H = \arccos \frac{\frac{1}{2}((R - G) + (R - B))}{\sqrt[2]{((R - G)^2 + (R - B)(G - B))}}$$

$$S = 1 - 3\,\frac{\min(R, G, B)}{R + G + B}$$

$$V = \frac{1}{3}(R + G + B)$$

Výhodou tohoto modelu je, že má extrahovanou složku jasu, která bývá při detekci kůže nepotřebná. Naopak nevýhodou je složitost převodu ze základního modelu RBG.



Obrázek 2: Znázornění HSV modelu [11]

2.2 Způsoby detekce kůže

Cílem detekce kůže je vytvořit taková pravidla, na základě kterých nám algoritmus, který má na vstupu obraz vrátí oblast, nebo oblasti obrazu, na kterých se nalézá pokožka.

2.2.1 Explicitně definovaná pravidla

Tato metoda určuje oblast kůže na základě pravidel, které určují hranici v barevném modelu. Příkladem takové metody je [6]:

```
% Barva kůže při rovnoměrném denním osvětlení
R > 95 AND G > 40 AND B > 20 AND
max{R, G, B} – min{R, G, B} > 15 AND
% složky RGB nesmí být blízko u sebe
% eliminace šedé
|R – G| > 15 AND
% také složky R a G nesmí být blízko u sebe
% jinak se nejedná o jemnou pleť
R > G AND R > B
% červená složka musí být nejvyšší složkou
OR
% Barva kůže pod zdrojem světla a boční osvětlení
R > 220 AND G > 210 AND B > 170 AND
|R – G| ≤ 15 AND
% složky R a G nesmí být blízko u sebe
R > B AND G > B
% složka B musí být nejmenší
```

[5] Výhodou takovýchto metod je jejich rychlost, naopak nevýhodou je problém nalezení vhodného barevného prostoru a pravidel, které jsou vytvořeny na základě pozorování.

2.2.2 Neparametrické modely rozložení kůže

[5] Hlavní myšlenkou tohoto druhu detekce kůže je nalezení rozložení barvy ze cvičného vzorku dat bez odvození explicitního modelu barvy kůže.

2.2.2.1 Normalizovaná vyhledávací tabulka

[5] Barevný prostor je rozdělen do několika skupin, kde každá skupina odpovídá určité oblasti barev. Každá skupina si uchovává informaci o tom, kolikrát se barva z této skupiny objevila na cvičných obrázcích kůže. Po ukončení „učení“ je tento histogram normalizován a tím převeden z histogramu na diskrétní pravděpodobnostní rozložení

$$\mathrm{P_{skin}(c)} = \frac{\mathrm{skin[c]}}{\mathrm{N}}$$

Kde *skin*[c] je hodnota sloupce histogramu, do kterého patří barva *c*. N je potom normalizační koeficient, který může odpovídat například [7] součtu hodnot sloupců histogramu nebo [8] maximální hodnotě histogramu. Hodnota $P_{skin}(c)$ odpovídá pravděpodobnosti toho, že barva *c* je barvou kůže.

2.2.2.2 Bayesův klasifikátor

[5] Z normalizované tabulky je zjištěna pravděpodobnost $P_{skin}(c)$, ta přesněji řečeno odpovídá pravděpodobnosti, že vidíme barvu c, přičemž víme, že c je barva kůže, tedy $P(c|skin)$. My však potřebujeme znát pravděpodobnost, že se jedná o kůži, při dané barvě *c*, tedy $P(skin|c)$.

Pro zjištění této pravděpodobnosti je využito Bayesova pravidla [9]:

$$P(\text{skin}|c) = \frac{P(c|\text{skin})P(\text{skin})}{P(c|\text{skin})P(\text{skin}) + P(c|\neg\text{skin})P(\neg\text{skin})}$$

Pravděpodobnosti $P(\text{skin})$ a $P(\neg\text{skin})$ můžeme určit z počtu vzorků s barvou kůže a ze vzorků bez kůže z cvičného souboru dat.

Potom nerovnost $P(\text{skin}|c) \geq \Theta$, kde Θ je hraniční hodnota, může být použita jako pravidlo pro určení kůže.

[5] Neparametrické metody mají dvě hlavní výhody. Jsou rychlé při cvičení a při užívání. Jejich nevýhodou je neschopnost interpolace chybějících dat a značná paměťová náročnost. Pro 8bit RGB potřebujeme pole o velikosti 2^{24} (2^8 x 2^8 x 2^8) prvků. Velikost pole však můžeme snížit hrubším vzorkováním barev např. 2^7 x 2^7 x 2^7, nebo 2^6 x 2^6 x 2^6. [7] Porovnáním výsledků při různých vzorkováních se ukázalo, že nejlepších výsledků se dosahuje při vzorkování 2^5 x 2^5 x 2^5.

2.2.3 Parametrické modely rozložení kůže

[5] Vysoké nároky na paměť u neparametrických modelů a jejich vysoká závislost na kvalitě obrázků cvičných vzorků vyvolala nutnost nalezení modelů, které jsou kompaktnější a schopnější zobecnit a interpolovat data získané z cvičných vzorků.

2.2.3.1 Single Gaussian

[5] Rozložení barvy kůže může být modelováno pomocí [10] normálního rozložení rovnicí [5]:

$$P(c|\text{skin}) = \frac{1}{2\pi\sqrt[2]{\sigma_s}} \times e^{-\frac{1}{2\sigma_s}(c-\mu_s)^T(c-\mu_s)}$$

Kde *c* je vektor barvy a μ a σ jsou parametry normálního rozložení (vektor střední hodnoty a rozptyl). Parametry jsou určeny z cvičného vzorku dat následujícím způsobem:

$$\mu_s = \frac{1}{n}\sum_{j=1}^{n} c_j$$

$$\sigma_s = \frac{1}{n-1}\sum_{j=1}^{n}(c_j - \mu_s)^T(c_j - \mu_s)$$

Kde *n* je celkový počet vzorků barvy kůže c_j. $P(c|\text{skin})$ je potom ukazatel podobnosti barvy *c* na barvu kůže.

2.2.3.2 Mixture of Gaussians

[5] Jedná se o zobecnění modelu Single Gaussian následující úpravou:

$$P(c|\text{skin}) = \sum_{i=1}^{k} \pi_i \cdot P_i(c|\text{skin})$$

k je počet sloučených prvků, π_i jsou slučovací parametry, pro které platí:

$$\sum_{i=1}^{k} \pi_i = 1$$

A $P_i(c|\text{skin})$ jsou pravděpodobnostní rozložení, přičemž každé má vlastní střední hodnotu a rozptyl.

3 Detekce obličeje

Detekce obličeje v obraze je v dnešní době ve velkém rozvoji. Za zmínění stojí například nová verze programu [12] Picasa od společnosti Google, která dokáže automaticky detekovat obličej a vyretušovat efekt tzv. „červených očí“, nebo některé digitální fotoaparáty, které naleznou obličeje ve fotografované scéně, automaticky nastaví optimální světelnost a zaostří na ně.

3.1 Rozdělení metod detekcí obličeje

Metod, které nám umožňují detekovat obličej je mnoho. Část z nich je shrnuta v tabulce níže (v tabulce jsou uvedeny i původní názvy metod v anglickém jazyce).

Approach *(přístup)*	**Representative Works** *(Práce reprezentující daný přístup)*
Knowledge-based *(znalostně založené metody)*	
	Multiresolution rule-based method [14]
Feature invariant *(rysově neměnné metody)*	
Facial Features *(rysy obličeje)*	Grouping of edges [15]
Textures *(textury)*	Space Gray-Level Dependance matrix of face pattern [16]
Skin Color *(barva kůže)*	Mixture of Gaussian [17]
Multiple Features *(vícenásobné rysy)*	Integration of skin color, size and shape [18]
Template Matching *(porovnání s šablonami)*	
Predefined face templates *(předefinované šablony obličeje)*	Shape template [19]
Deformable Templates *(deformovatelné šablony)*	Active Shape Model (ASM) [20]
Appearance-based *(vzhledově založené metody)*	
Eigenface	Eigenvector decomposition and clustering [21]
Distribution-based *(distribučně založené)*	Gaussian distribution and multilayer perceptron [22]
Neural Network *(neuronové sítě)*	Ensemble of neural network and arbitration schemes [23]
Support Vector Machine (SVM)	SVM with polynomial kernel [24]
Naive Bayes Classifier	Joint statistics of local appearance and position [25]
Hidden Markov Model (HMM)	Higher order statistics with (HMM) [26]
Information-Theoretical Approach	Kullback relative information [27]

Tabulka 1: Rozdělení metod pro detekci obličeje [13]

3.2 Haar-like features

Metoda Haar-like features může být zařazena mezi rysově neměnné metody. Tuto metodou zmiňuji podrobněji, protože patří mezi rychlé a oblíbené metody detekce obličeje. [28] Proces detekce může být mnohem výkonnější, pokud je založen na detekci rysů, v nichž je zakódována informace o detekovaném objektu, v našem případě obličeje.

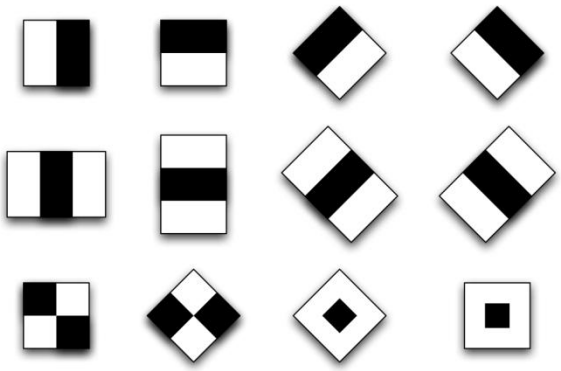
Z těchto důvodů jsem ji použil při implementaci programu.

3.2.1 Princip detektoru

[29] Tato metoda je založena na porovnání součtů intenzit v sousedících oblastech uvnitř detekčního okna. Obvykle se jedná o seskupení dvou a více sousedících oblastí. Takovému seskupení říkáme *Haar-like features* (Haar-like rysy).

Tyto rysy využívají faktu, že na objektech stejného typu můžeme najít společné vlastnosti. Příkladem takovéto společné vlastnosti při detekci obličeje může být tmavá oblast kolem očí v porovnání s čelem.

Rysy jsou uvnitř detekčního okna testovány a na základě výsledku je určeno, zda se uvnitř objekt nalézá, či nikoliv.



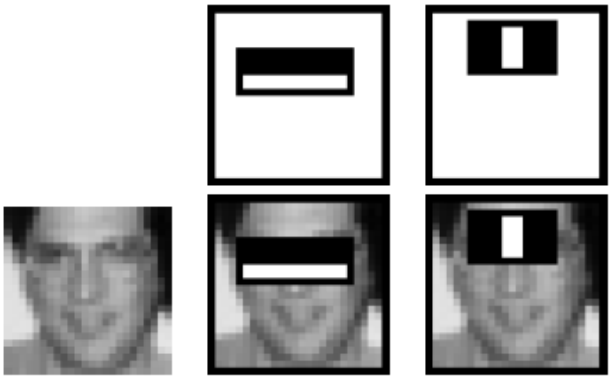
Obrázek 3: Základní tvary rysů [29]

[28] Tento detektor objektů byl původně navržen Paulem Violou a později vylepšen Rainerem Lienhartem.

3.2.2 Cvičení detektoru

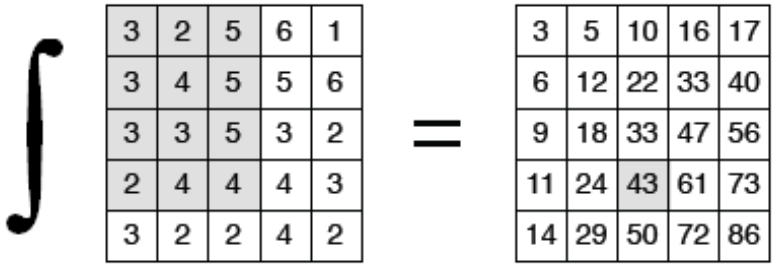
Nejdříve je klasifikátor (přesněji *kaskáda* zesílených *klasifikátorů* pracujících s haar-like rysy) cvičen na několika stovkách stejně velkých vzorků obsahujících daný objekt (např. obličej). Takovéto vzorky se nazývají *pozitivní vzorky*. Dále proběhne cvičení na *negativních vzorcích*, libovolných vzorcích (např. strom, auto, nebe a další vzorky bez obličeje), které mají opět stejnou velikost. Při cvičení je cílem nalézt nejlepší kombinaci haar-like rysů popisujících vzorek. [29] Pozitivní vzorky potom mají podobnou strukturu a odlišují se od negativních vzorků. Tomuto procesu říkáme *cvičení detektoru*.

[29] Hlavní výhodou této metody je, že se skládá především z operací součtů a operací porovnání, což činní tuto metodu rychlou. Její přesnost je vysoce závislá na kvalitě databáze, která byla použita při cvičení.



Obrázek 4: Ukázka použití rysů na cvičném obličeji [30]

Nevýhodou metody je nutnost počítat součet intenzit pro vyhodnocení každého rysu a vyvolává potřebu velkého množství vyhledávání v detekčním okně. Tento problém byl eliminován uvedením [30] *integrálního obrázku*. Algoritmus projde obrázek pouze jednou a uloží součty intenzit všech pixelů, které se nachází nad a vlevo od aktuálního pixelu.



Obrázek 5: Příklad integrálního obrázku

3.2.3 Slabý klasifikátor

[31] *Slabé klasifikátory* jsou sestrojeny z jednoho nebo několika málo Haar rysů s nacvičenými prahovými hodnotami. Tyto klasifikátory pak klasifikují detekční okno jako pozitivní (nalézá se v něm obličej), nebo negativní (obličej se v něm nenalézá) podle toho, zda je vyhodnocení rysu nad nebo pod příslušnou prahovou hodnotou.

Jediný slabý klasifikátor neposkytne dostatek informací pro určení obličeje v detekčním okně, pokud však zkombinujeme několik slabých klasifikátorů do jednoho silného klasifikátoru, získáme pro určení obličeje dostatek informací.

3.2.4 Silný klasifikátor

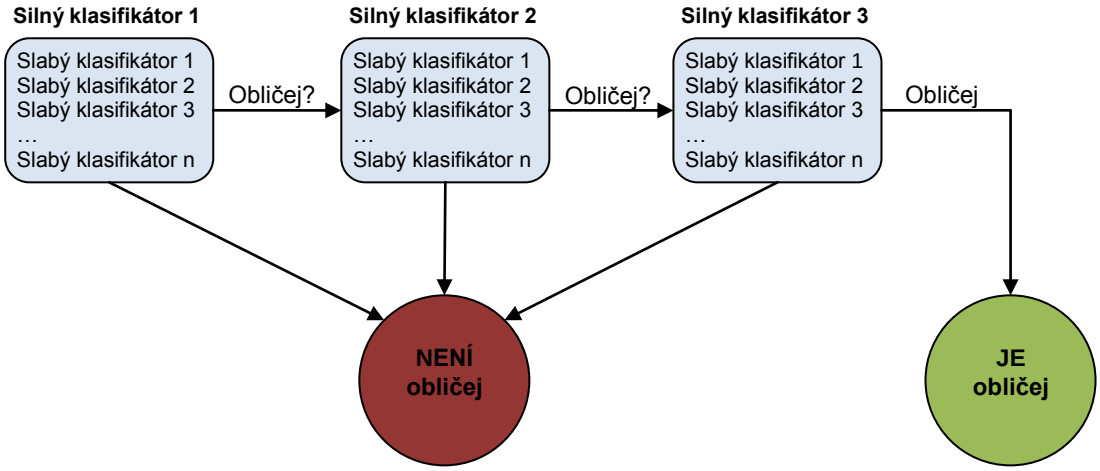
[31] *Silný klasifikátor* je kombinací několika slabých klasifikátorů. Každý slabý klasifikátor má přidělenou váhu, která závisí na jeho přesnosti při detekci. Při klasifikaci detekčního okna silným klasifikátorem jsou vyhodnoceny všechny slabé klasifikátory. Váhy slabých klasifikátorů jsou při klasifikaci sečteny a tento součet je porovnán s předdefinovanou hodnotou prahu pro určení, zda detekční okno obsahuje obličej, nebo ne.

Prahová hodnota musí být určen tak, abychom maximalizovali počet korektních detekcí a zároveň zachovali nízký počet chybných detekcí. Pokud je počet chybných detekcí příliš vysoký, musíme přidat do silného klasifikátoru více slabých klasifikátorů.

3.2.5 Kaskáda klasifikátorů

[31] Namísto vyhodnocení jediného velkého silného klasifikátoru, [30] Viola a Jones navrhli koncept *kaskády*. Jde o jednoduchou posloupnost po sobě jdoucích silných klasifikátorů, které mají vysokou úspěšnost pozitivní detekce a nízkou falešné detekce.

Pokud klasifikátor určí oblast v detekčním okně jako oblast bez obličeje, je toto okno okamžitě označeno jako negativní a zahozeno (viz. Obrázek 6: Příklad kaskády).



Obrázek 6: Příklad kaskády

Díky tomuto jsou všechny silné klasifikátory vyhodnoceny jen v případě, že je v detekčním okně detekován obličej. Protože většina detekčních oken obličej neobsahuje, všechny silné klasifikátory musí být vyhodnoceny jen zřídka.

4 Trackování

[32] *Trackování*, je proces lokalizace pohybujícího se objektu (případně objektů) ve video sekvenci. Algoritmus analyzuje snímky ze sekvence a určuje pozici objektu v aktuálním snímku.

Hlavním problémem trackování je propojení pozic objektu v po sobě jdoucích snímcích. Toto je problém především v případě, kdy se objekt pohybuje příliš rychle vzhledem k rychlosti, kterou je obraz snímán.

4.1 Algoritmy pro trackování

V oblasti trackování rozlišujeme dvě hlavní oblasti: *Target Representation and Localization* a *Filtering and Data Association*.

Algoritmy pro trackování musí být rychlé a efektivní. Aby mohlo být trackování využito v praxi, musí být zvolený algoritmus schopen fungovat v reálném čase a nesmí zabírat příliš velké množství výpočetních prostředků.

4.1.1 Target Representation and Localization

[32] Většinou se jedná o proces, který postupuje od spodu nahoru. Výpočetní náročnost těchto algoritmů bývá nízká.

Příklady tohoto typu algoritmů:

- **Blob tracking** – jedná se o trackování skvrn, např. blob detection, blokově založená korelace, optical flow
- **Kernel-based tracking** (Mean-shift tracking) – procedura iterativní lokalizace založené na maximalizaci podobnosti, např. Bhattacharyya coefficient
- **Contour tracking** – algoritmy jsou založeny na detekci hranic objektu, např. Condensation algorithm
- **Visual feature matching** - např. Registration

4.1.2 Filtering and Data Association

[32] Jedná se o proces seshora dolů, který zahrnuje začlenění dřívějších informací o scéně nebo o objektu. Řeší dynamiku objektu a vyhodnocení různých hypotéz. Výpočetní náročnost je mnohem vyšší než u 4.1.1.

Příklady tohoto typu algoritmů:

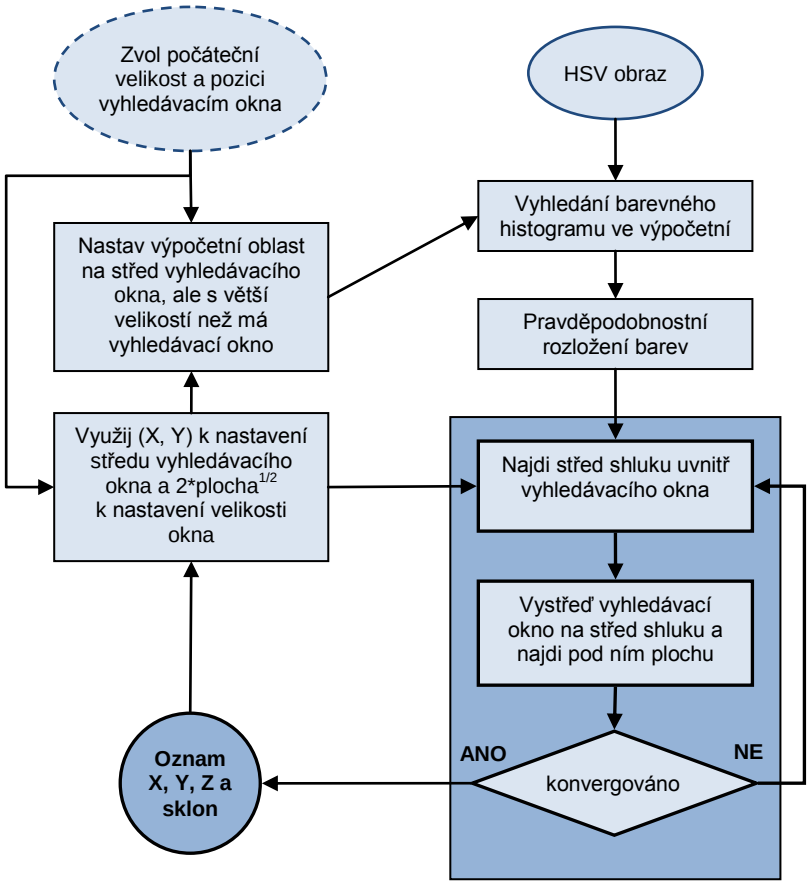
- **Kalman filter** – Kalmanův filtr, optimální rekurzivní Bayesin filter pro lineární funkce spojené s Gaussovským šumem.
- **Particle filter** (částicový filtr) – užitečný pro vzorkování základové stavově-prostorové rozložení nelineárních a negaussovských procesů.

4.2 CamShift

[33] Jedná se o úpravu *Mean Shift* algoritmu. Mean Shift algoritmus je robustní neparametrická technika pro nalezení modu pravděpodobnostního rozložení. Mean Shift nebyl původně určen pro použití jako trackovací algoritmus, ale je v této oblasti dobře využitelný.

Mean Shift algoritmus pracuje s pravděpodobnostním rozložením. Pro trackování barevných objektů musí být obraz reprezentován jako pravděpodobnostní rozložení trackované barvy. K tomuto využíváme histogramu. Rozložení barvy v obrázku z video sekvence se v průběhu času mění, proto byl Mean Shift upraven tak, aby se dynamicky přizpůsoboval pravděpodobnostnímu rozložení barvy, které trackuje. Takto upravený Mean Shift algoritmus, splňující zmíněné požadavky, se jmenuje *CamShift*. CamShift je zkratka pro „Continuously Adaptive Mean Shift Algorithm".

Při detekci obličeje trackuje CamShift X-ovou a Y-ovou pozici objektu a plochu s pravděpodobnostním rozložením barvy kůže reprezentující například obličej. Plocha je úměrná Z-ové souřadnici, tedy vzdálenosti od kamery. Trackován je také úhel naklonění.



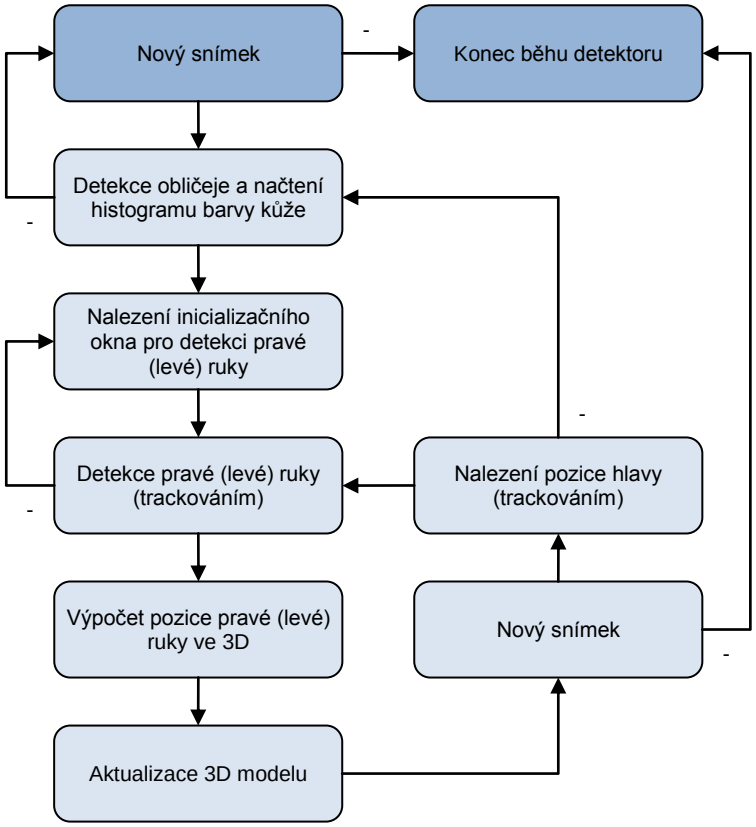
Obrázek 7: Blokové schéma průběhu trackování [33]

Obrázek 7: Blokové schéma průběhu trackování [33] zobrazuje princip trackování pomocí CamShiftu. Každý snímek videa je převeden na pravděpodobnostní rozložení barvy. Střed a velikost trackovaného objektu jsou nalezeny CamShif algoritmem pracujícím s tímto pravděpodobnostním rozložením barvy obrázku. Tmavě modrý obdélník na Obrázek 7 je algoritmus Mean Shift. Detekovaná velikost a pozice jsou oznámeny a použity pro vyhledávací okno v dalším cyklu trackování.

5 Implementace projektu

Tato kapitola se zabývá implementací jednotlivých částí projektu, jejichž teoretickému rozboru byly věnovány kapitoly předchozí. Projekt byl implementován v jazyce C++ s využitím knihoven OpenCV [34], OpenGL [35] a Glut [36], která je nadstavbou knihovny OpenGL.

Blokové schéma, kterým se implementace řídí, je znázorněno na Obrázek 8.



Obrázek 8: Blokové schéma detektoru

5.1 Detekce obličeje

Tato část v sobě spojuje hned dvě předchozí kapitoly: 2. kapitolu, zabývající se detekcí kůže a 3. kapitolu, zabývající se detekcí obličeje.

Tuto problematiku řeší ve zdrojovém kódu 2 třídy

- `FaceDetector`: detekce obličeje
- `SkinDetector`: detekce kůže

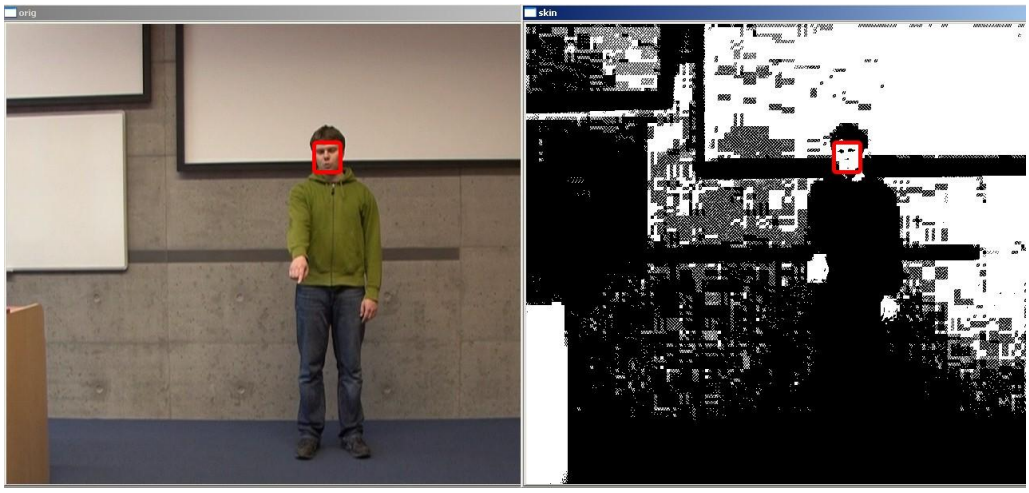
Třída `FaceDetector` detekuje obličej v obrázku. Její metoda `Detect(IplImage* imgOrig)` přijímá jako parametr obrázek, v němž se snaží nalézt obličej. Nejdříve se načte kaskáda Haar klasifikátoru viz kapitola 0, poté je obrázek zmenšen a převeden do odstínu šedé. Funkcí knihovny OpenCV [28] `cvHaarDetectObjects` jsou v obrázku nalezeny tvary odpovídající obličeji.

Při detekci obličeje však často docházelo k případům, kdy nebyl detekován obličej, ale například zeď nebo jiný objekt, viz Obrázek 9.

Obrázek 9: Chybná detekce obličeje

Tento problém vyvolal nutnost kontroly detekovaného obličeje. Protože detektor pracuje s obrázkem v odstínech šedi, byla pro ověření detekovaného obličeje použita kontrola pomocí barvy lidské kůže.

Kontrolu provádí třída `SkinDetector`, jejímuž konstruktoru je předán obrázek, ve kterém byl obličej nalezen a obdélník udávající oblast, v níž byl obličej detekován. Jedná se o jednoduchý detektor kůže z kapitoly 2.2.1. Detektor sám nepodává ideální výsledky, protože ve velké míře určuje barvu kůže i na místech, kde není, viz Obrázek 10. To však nevadí, protože ověřována je pouze oblast, v níž je detekován obličej. Tato oblast je na obrázku vyznačena červeným obdélníkem. Pokud poměr plochy oblasti a plochy s barvou kůže překročí zvolenou prahovou hodnotu, je detekce obličeje potvrzena.



Obrázek 10: Vstupní obrázek detektoru kůže a výstupní zobrazení ploch s barvou kůže (bílá barva)

Po detekování obličeje vytvoří třída `FaceDetector` barevný histogram, který je později využit pro detekci rukou. Tuto operaci provádí metoda `histCreator(IplImage* frame)`. Nejprve je obrázek převeden z RGB do HSV, poté je na základě hraničních hodnot zadaných uživatelem vytvořena maska a následně je z HSV oddělena složka Hue.

Histogram je vytvořen pomocí funkce knihovny OpenCV `cvCalcHist(&hue, hist, 0, mask)`, která je aplikována na obrázek `hue` v místě kde byl nalezen obličej. Při vytváření histogramu je na obrázek aplikována výše zmíněná maska.

5.2 Trackování

Pro trackování byl použit algoritmus CamShift. Jeho princip je popsán v kapitole 4.2. Jeho hlavní výhodou je snadný způsob trackování na základě barevného rozložení. Knihovna OpenCV obsahuje funkce, s nimiž je možné za pomocí tohoto algoritmu trackovat.

Trackováním se v programu zabývá třída `BodyTracker`. Kostruktor třídy přijímá jako parametr část těla (`type`), určující kterou trackujeme, , obrázek `frame`, v němž trackování provádíme, obdélník `r`, pro určení počáteční pozice trackované části těla a histogram `h`, který je uložen do `BodyTracker::hist`, která uchovává histogram po celou dobu trackování.

Proměnná `type` může nabývat několika hodnot:

- `HEAD` – pro detekci hlavy
- `ARML` – pro detekci levé ruky
- `ARMR` – pro detekci pravé ruky

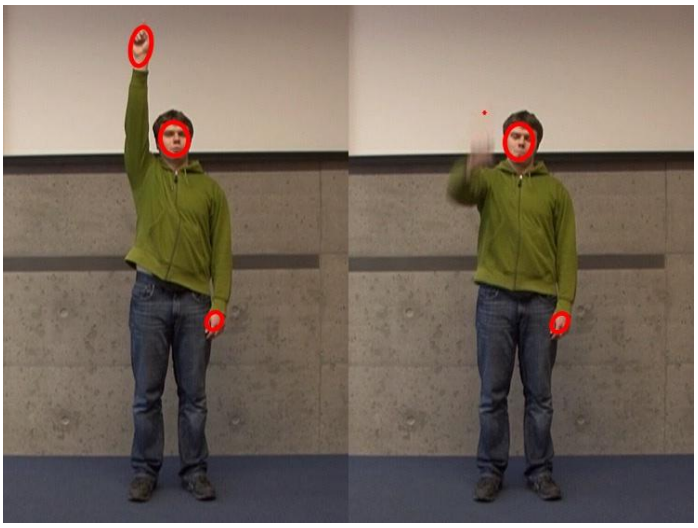
Při vytváření první instance třídy `BodyTracker` jsou vytvořeny také pomocné obrázky, které jsou při trackování dále využívány.

V každém snímku musíme spočítat, které pixely odpovídají barevnému rozložení histogramu `hist`. O to se stará metoda `prepBackProject`. Ta nejprve obrázek převede z RGB do HSV a oddělí složku Hue. Poté vytvoří masku `mask`, která je definována uživatelem zadanými parametry. Tyto parametry udávají hraniční hodnoty složek Saturation (`smin`) a Value (`vmin` a `vmax`) modelu HSV při detekci a jsou uložené jako členské proměnné třídy `Settings`. Metoda `prepBackProject` pak pomocí funkce knihovny OpenCV `cvCalcBackProject` vypočítá, které pixely odpovídají barevnému rozložení. Na závěr je aplikována maska `mask` a obrázek je připraven pro trackování (viz Obrázek 11).

Obrázek 11: (Zleva) originální obrázek, backprojection, po aplikování masky

Vlastní trackování je prováděno metodou `nextPosition`. Tato metoda nejprve ověří, zda nebyl trackovaný objekt v předchozím snímku ztracen → výška a šířka objektu jsou nulové (viz Obrázek 12). K tomuto obvykle dochází v případech, kdy se trackovaný objekt pohybuje příliš rychle vzhledem k rychlosti, kterou snímá kamera, nebo v případě, že barva kůže neodpovídá kvůli nerovnoměrnému osvícení pravděpodobnostnímu rozložení barvy kůže. V takovém případě musí být pro další trackování nastaveno nové počáteční vyhledávací okno.

Pokud takováto situace nastane, je zavolána metoda `getInitBox`, která vrátí právě takové okno. V případě, že je ztraceným objektem ruka, je podle pozice hlavy určena oblast, v níž se ruka nachází. Pokud je však ztraceným objektem obličej, musí být v obraze opět detekován.

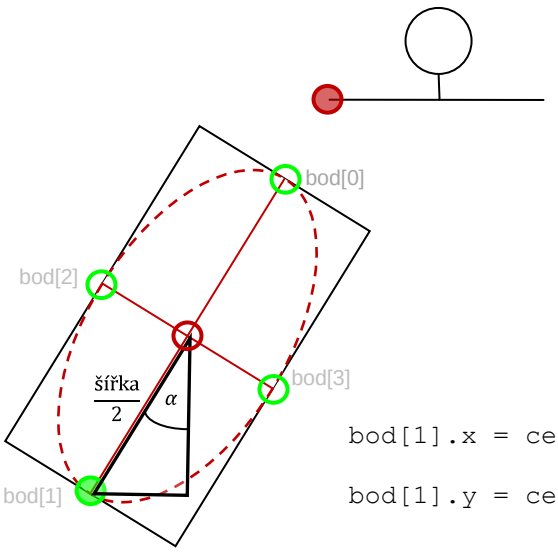


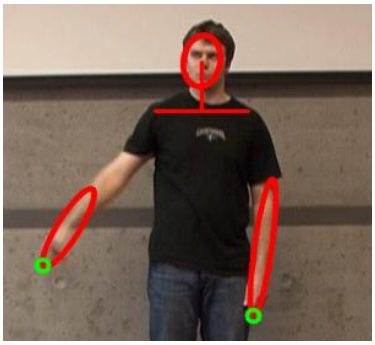
Obrázek 12: Vlevo ruka detekovaná, vpravo po ztracení

Metoda `nextPosition` v případě trackování levé nebo pravé ruky vymaskuje oblast, v níž se ruce nemohou nacházet. Pokud je trackován obličej, je vymaskována čtvercová oblast o straně velikosti šířky těla umístěná na pozici, kde byl obličej naposled detekován. Následně je zavolána funkce `cvCamShift`, která na základě pozice počátečního vyhledávacího okna nalezne novou pozici trackovaného objektu. Tato nová pozice je potom nastavena jako výchozí pozice vyhledávacího okna při následujícím trackování.

Funkce `cvCamShift` zjistí pozici objektu, ale tato pozice je určena středem, výškou, šířkou a nakloněním objektu. Pro určení směru, kterým osoba ukazuje, musíme zjistit pozici dlaně. Proto je nakonec metodou `nextPosition` zavolána metoda `calcPosition`, která ze zjištěné pozice objektu vypočítá souřadnice konce dlaně.

Při zjišťování konce dlaně jsou vypočteny souřadnice středů stran detekovaného obdélníka (na obrázku zelená prázdná kolečka). Střed objektu známe (červené prázdné kolečko) a šířku a výšku také. X-ové a Y-ové souřadnice bodů vypočítáme pomocí Pythagorovy věty. Z těchto bodů je jako konec dlaně vybrán ten bod, který má od ramene (červené plné kolečko) největší vzdálenost (zelené plné kolečko).





$$bod[1].x = center.x - \cos(\alpha) \times šířka \times 0{,}5$$

$$bod[1].y = center.y - \sin(\alpha) \times šířka \times 0{,}5$$

Obrázek 13: Znázornění výpočtu pozice dlaně

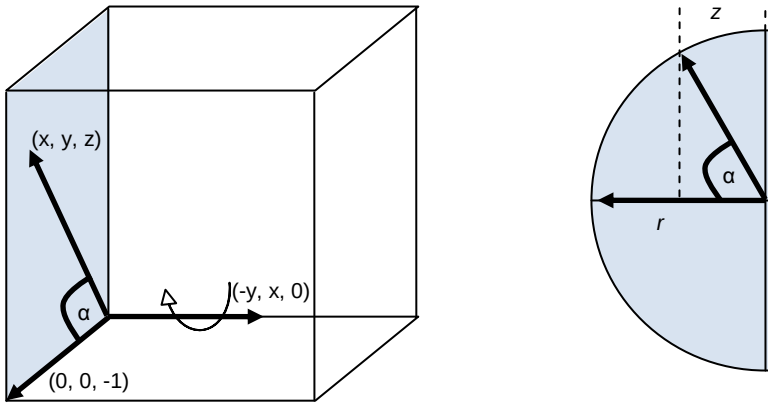
Poslední důležitou metodou třídy `BodyTracker` je `getRotation`. Tato metoda slouží k přepočítání souřadnic z 2D do 3D. Metoda vrací strukturu `RotationParams` nesoucí informaci o úhlu, o který je ruka vychýlena a o složkách vektoru, kolem kterého se ruka otočí o zmíněný úhel.

Nejdříve je dopočítána složka z pomocí rovnice koule:

$$x^2 + y^2 + z^2 = r^2 \Rightarrow z = \sqrt[2]{r^2 - x^2 - y^2}$$

Potom je násobením vektorů počáteční pozice ruky (0,0,-1) a detekované pozice ruky (x, y, z) vypočítán vektor, kolem kterého se bude ruka otáčet.

$$(x, y, z) \times (0, 0, -1) = (-y, x, 0)$$



Obrázek 14: Ilustrační obrázek pro výpočet 3D souřadnic

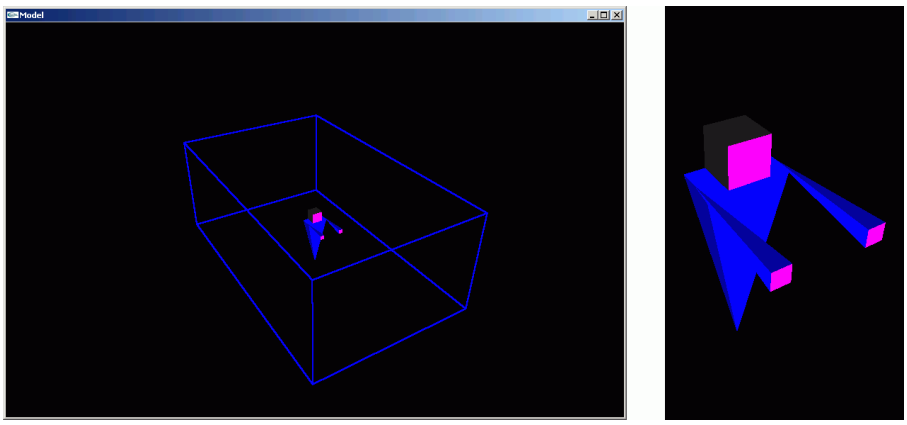
Úhel, o který se ruka otočí, je vypočítán:

$$\alpha = arcos\ \frac{z}{r}, \text{ kde } r = \sqrt[2]{x^2 + y^2 + z^2}$$

5.3 Vytvoření 3D modelu

Pro znázornění, kterým směrem osoba ve videu ukazuje, byl vytvořen 3D model. Tento model je vytvořen třídou `Detector`, která také řídí běh celého programu.

Model byl vytvořen pomocí knihovny OpenGL [35] a Glut [36]. Jde o jednoduché znázornění osoby v prostoru. Modelem může uživatel otáčet, přibližovat ho a oddalovat (viz Obrázek 15).



Obrázek 15: Ukázka 3D modelu – vlevo pohled na model, vpravo přiblížení na osobu

5.4 Běh detektoru

Jak již bylo zmíněno v předchozí kapitole, o běh detektoru se stará třída `Detector`. Tato třída načítá snímky z videa a tyto snímky zobrazuje, volá metody jednotlivých tříd pro detekci obličeje a trackování částí těla, překresluje 3D model atd.

Po inicializaci všech potřebných částí detektoru je zavolána funkce knihovny OpenGL `glutMainLoop`, která v daných intervalech volá metodu (callback funkci) `onTimer` třídy `Detector`, která zpracovává vstupní data.

Metoda `onTimer` naplánuje své opětovné volání podle následujícího vzorce:

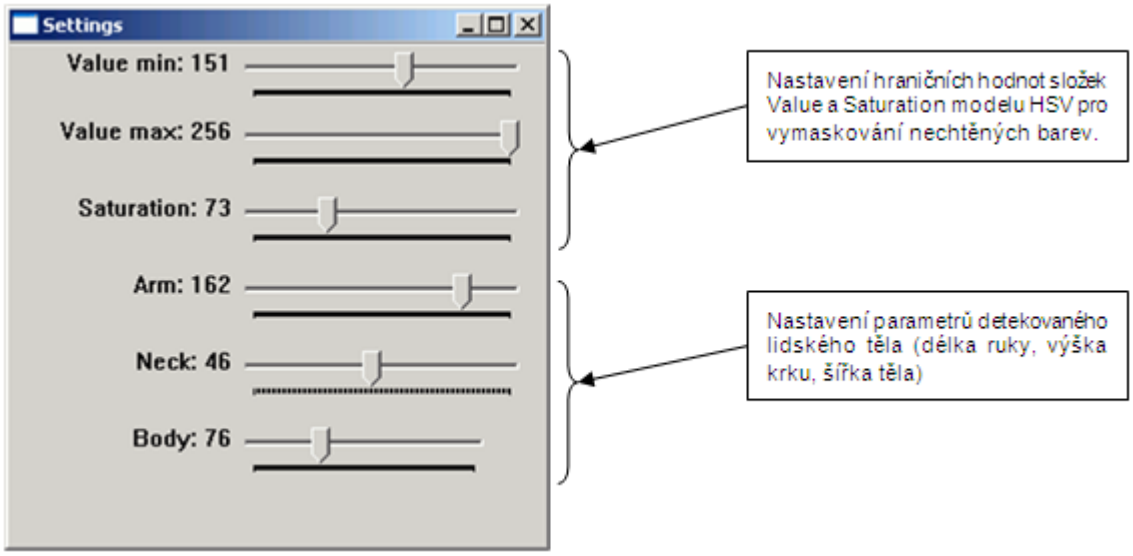
$$\text{interval} = \left(\frac{1}{\text{FPS}} - \frac{\text{počet Tiků}_{\text{začátekOnTimeru}} - \text{počet Tiků}_{\text{konecOnTimeru}}}{\text{frekvence Tiků}}\right) \times 1000$$

Interval je potom doba mezi voláními funkce `onTimer` v milisekundách.

Popis kroků prováděných při běhu detektoru popisuje Obrázek 8.

5.5 Ovládání programu

Program je konsolová aplikace, která při spuštění očekává jeden parametr. Pokud je tímto parametrem číslo, pokusí se program načíst jako zdroj dat kameru s identifikačním číslem zadaným parametrem. Pokud je zadán řetězec, program jej interpretuje jako název video souboru, který má být načten. V případě, že parametr není zadán, je za něj dosazeno číslo 0 a program se proto pokusí načíst zařízení s odpovídajícím identifikátorem.



Obrázek 16: Popis uživatelem nastavovaných parametrů

Model (viz Obrázek 15) je ovládán myší. Při stisku levého tlačítka a tažení myši je model otáčen, při stisku tlačítka pravého je model přibližován a oddalován.

Některé klávesy na klávesnici mají také svůj význam při ovládání:

A – zobrazuje a skrývá inicializační obdélník rukou

B – přepínání mezi zobrazením barevným a vymaskovaným backprojection (Obrázek 11)

P – pozastaví trackování

R – vynucené detekování obličeje

S – zobrazuje a skrývá okno Settings (Obrázek 16)

Ovládání klávesami funguje pouze při aktivním okně Model. Toto okno je totiž vytvořeno knihovnou OpenGL, která obstarává zachytávání vstupů z klávesnice.

Rozhraní detektoru využívá pouze mechanismů zabudovaných přímo do knihoven OpenCV a OpenGL. Díky tomuto je snížena paměťová náročnost programu a umožněna kompatibilita mezi operačními systémy. Dalším možným přístupem by bylo využití knihovny wxWidgets [37]. Při propojení s ní však často musí být kopírovány buffery obsahující data snímků, což zvyšuje paměťovou náročnost programu.

6 Testování

Testování probíhalo ve třech etapách rozdělených podle použitých testovacích souborů.

Testování bylo provedeno pod operačním systémem Microsoft Windows XP, avšak program byl navržen tak, aby byla zachována přenositelnost i na jiné platformy.

6.1 První etapa

V první etapě byly využity videozáznamy vytvořené ještě před vlastní implementací programu. Jedná se o video soubory:

- test-long.avi – tričko s dlouhým rukávem
- test-short.avi – tričko s krátkým rukávem

Obrázek 17: Testovací data (zleva) test-long.avi, test-short.avi

S těmito soubory se pracovalo již v průběhu implementace a je při nich dosaženo výborných výsledků. Obličej je rychle detekován a směr, kterým ukazuje osoba ve videu je určen také přesně.

6.2 Druhá etapa

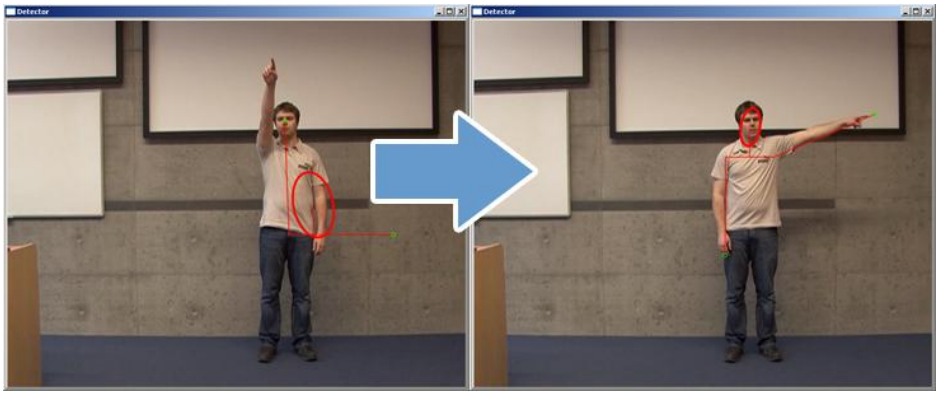
V této etapě byly také využity testovací videozáznamy pořízené před implementací programu. Tyto však byly určeny pro ověření adaptibility programu po dokončení implementace.

- test-skin.avi – tričko s barvou podobající se barvě kůže
- test-someone-else.avi – osoba procházející kolem trackované osoby

Záznam test-someone-else.avi funguje dobře. Osoba procházející za i před detekovanou osobou nepůsobí detektoru velké potíže. Detektor na okamžik ztratí pozici obličeje, ale ta je okamžitě opětovně nalezena a trackování probíhá dál (viz Obrázek 18).

Obrázek 18: Ztráta pozice obličeje a jeho opětovné určení

Větší problém se vyskytnul u testovacího videa test-skin.avi, při němž je zpočátku detekována místo obličeje levá část těla s rukou (Obrázek 19). Toto je dáno vlastnostmi detektoru obličeje, který pravděpodobně považuje oblast mezi levou ruku a levou polovinou těla za linku nosu a tmavý pruh za tělem za linku očí. Od okamžiku, kdy je levá ruka zdvižena, je při opakované detekci obličeje (stiskem klávesy R viz 5.5) nalezen obličej a detektor podává opět dobré výsledky.



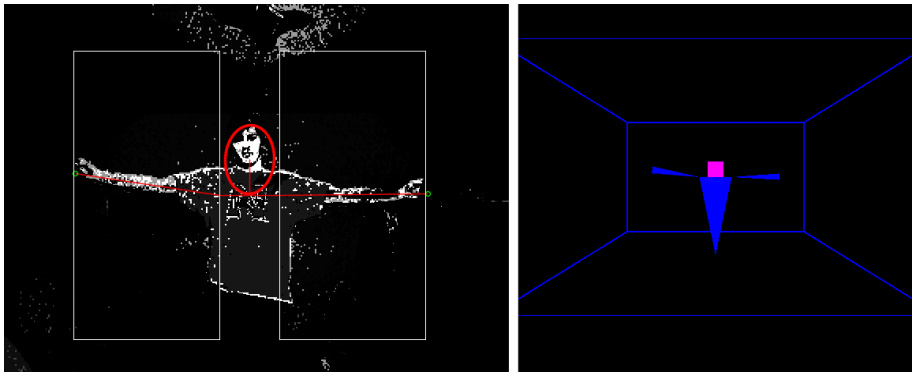
Obrázek 19: špatně detekovaný obličej

6.3 Třetí etapa

Poslední etapa testování proběhla při realtimovém detekování před webovou kamerou. Zpočátku byl s tímto problém, protože OpenCV nebylo schopno u několika webových kamer snímat video s vyšším rozlišením než 320 x 240px. Toto rozlišení videa je pro detektor příliš nízké, protože při detekování musí být celá postava detekované osoby v obraze. Přesněji se do obrazu musí vlézt inicializační obdélníky rukou.

Pro detekci je tedy potřeba mít na vstupu video s vyšším rozlišením. Ideální rozlišení videa je 640 x 480px. Toho se nakonec povedlo docílit s použitím kamery Chicony CNF7246.

Výsledek detekce se lišil podle toho, jaké byly světelné podmínky, při nichž testování probíhalo. Na Obrázek 20 je vidět příklad úspěšné detekce.

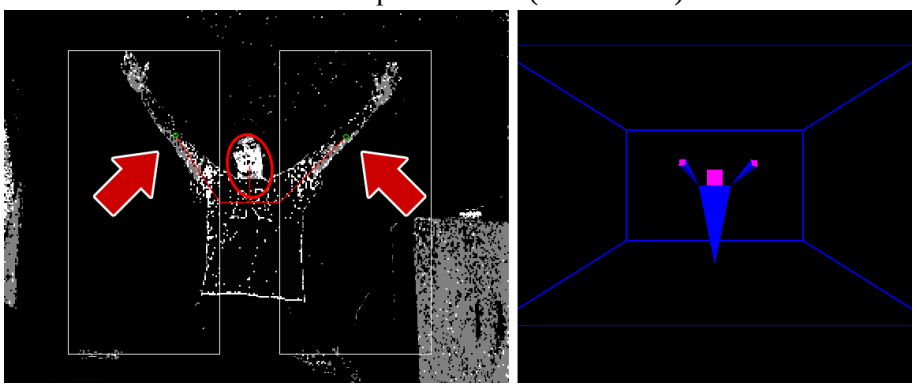


Obrázek 20: Příklad přesné detekce

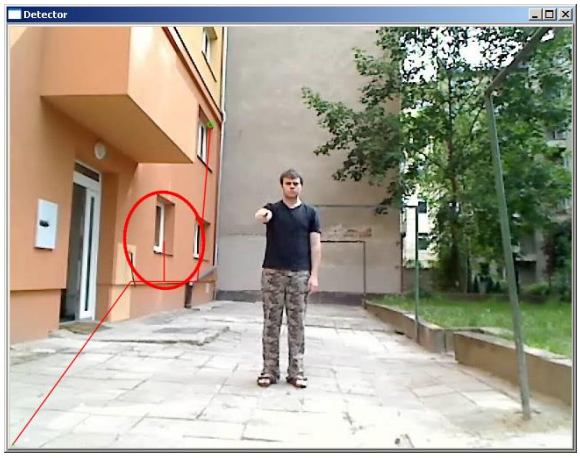
Při detekci může nastat okamžik, kdy není paže rovnoměrně osvícená, a proto její část neodpovídá pravděpodobnostnímu rozložení barvy kůže. Tento případ je zachycen na Obrázek 21. X-ovou a Y-ovou složku určuje detektor správně. Z-ová složka je však kvůli špatně detekované délce paže určena nepřesně (viz Obrázek 21)

Druhý problém se objevil při detekci s videozáznamem, který obsahoval komplikované pozadí, případně pozadí s barvou podobnou kůži (viz). V takovém případě detektor není schopen detekovat

správně obličej, protože komplikované pozadí často obsahuje část, která může být podobná obličeji a kontrola pomocí detekce kůže kvůli barvě pozadí selže (Obrázek 22).



Obrázek 21: Špatně detekované konce paží



Obrázek 22: Video se složitým počasím

6.4 Výsledky

Testováním se ukázaly některé nedokonalosti detektoru. Při detekci obličeje se může stát, že je detekován jiný objekt. Původně měla být detekce obličeje plně automatická, avšak zmíněný problém vyvolal nutnost implementace znovu-detekování obličeje v případě, že není určen správně.

Problém komplikovaného pozadí a pozadí barvy kůže může být odstraněn pomocí implementování odečítání pozadí.

Detektor po nalezení obličeje a nastavení parametrů ve většině případů detekuje správně. Chyby se objevují převážně z důvodu nerovnoměrného osvícení a nekvalitně vymaskovaného pozadí.

7 Závěr

Detektor určující pozici rukou ve videu je zajímavý projekt, který by mohl být v budoucnu využit jako interaktivní nástroj sloužící k ovládání přístrojů v místnosti, jako virtuální ukazovátko při prezentacích nebo jako nástroj pro ovládání počítačových her.

V současném stavu je detektor navržen tak, aby detekoval osobu stojící čelně ke kameře. Dalším možným rozšířením by tedy mohla být detekce rukou více lidí ve video sekvenci a to i takových, kteří nestojí ke kameře čelně.

V budoucnu tak třeba bude stačit ukázat na nějaký přístroj a říct: „Zapnout!“ a televize, nebo jakékoliv jiné zařízení uposlechne našeho příkazu.

Apendix

Instalace a spuštění

Pro spuštění programu detector je potřeba nainstalovat balíček Microsoft Visual C++ 2005 Redistributable Package (verze pro x86 je přiložena na CD, verze pro další platformy jsou ke stažení na http://www.microsoft.com/Downloads/). Po nainstalování tohoto balíčku je program připraven pro spuštění.

Program detektor očekává při spuštění jeden parametr. Tento parametr určuje odkud je bráno vstupní video sekvence. Pokud je parametrem řetězec, je tento brán jako jméno video souboru s cestou. Pokud je parametrem číslo, snaží se program načíst zdroj videa z kamery připojené k počítači, identifikované právě tímto číslem.

V případě, že chceme program přeložit, je potřeba nainstalovat Microsoft Visual Studio [38], knihovnu OpenCV [34] a knihovnu OpenGL [35].

Literatura

[1] *Barevný model* [online], Poslední modifikace: 8. 1. 2009. Dostupné na URL: http://cs.wikipedia.org/wiki/Barevn%C3%BD_model.

[2] *RGB color space* [online], Poslední modifikace 27. 12. 2008. Dostupné na URL: http://en.wikipedia.org/wiki/RGB_color_space.

[3] SKARBEK, W., AND KOSCHAN, A.: *Colour image segmentation – a survey –*. Tech. rep., Institute for Technical Informatics, Technical University of Berlin, 1994. Dostupné na URL: http://imaging.utk.edu/~koschan/paper/coseg.pdf

[4] *HSV* [online], Poslední modifikace: 23.4.2009. Dostupné na URL: http://cs.wikipedia.org/wiki/HSV

[5] VEZHNEVETS, V., SAZONOV, V., AND ANDREEVA, A.: *A Survey on Pixel-Based Skin Color Detection Techniques*. Moscow State University. Moscow, Russia.

[6] KOVAČ, J., PEER, P., AND SOLINA, F.: *Human Skin Colour Clustering for Face Detection*. University of Ljubljana, Ljubljana, Slovenia

[7] JONES, M. J., AND REHG, J. M.: *Statistical color models with application to skin detection*. In Proceeding of the CVPR ’99, vol. 1, 1999, s. 274–280.

[8] ZARIT, B. D., SUPER, B. J., AND QUEK, F. K. H.: *Comparison of five color models in skin pixel classification.* In Proceeding of the ICCV’99 Int’l Workshop on recognition, analysis and tracking of faces and gestures in Real-Time systems, 1999, s. 58–63.

[9] *Bayes' theorem* [online], Poslední modifikace 15. 5. 2009. Dostupné na URL: http://en.wikipedia.org/wiki/Bayes%27_theorem.

[10] *Normální rozdělení* [online], Poslední modifikace 24. 4. 2009. Dostupné na URL: http://cs.wikipedia.org/wiki/Norm%C3%A1ln%C3%AD_rozd%C4%9Blen%C3%AD.

[11] *Color in Computer Graphics* [online], Poslední modifikace 21. 8. 2002. Dostupné na URL: http://viz.aset.psu.edu/gho/sem_notes/color_2d/html/primary_systems.html.

[12] *Aplikace Picasa 3* [online], Poslední modifikace 11. 4. 2009. Dostupné na URL: http://picasa.google.com/.

[13] YANG, M.-H., KRIEGMAN, D. J., AND AHUJA, N.: *Detecting Faces in Images: A Survey*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 1, 2002.

[14] YANG, G., AND HUANG, T. S.: *Human Face Detection in Complex Background*. Pattern Recognition, vol. 27, no. 1, 1994, s. 53-63.

[15] LEUNG, T. K., BURL, M. C., AND PERONA, P.: *Finding Faces in Cluttered Scenes Using Random Labeled Graph Matching*. In Proceeding of the Fifth IEEE Int’l Conf. Computer Vision, 1995, str. 637-644.

[16] DAI, Y., AND NAKANO, Y.: *Face-Texture Model Based on SGLD and Its Application in Face Detection in a Color Scene*. Pattern Recognition, vol. 29, no. 6, 1996, s. 1007-1017.

[17] MCKENNA, S., GONG, S., AND RAJA, Y.: *Modelling Facial Colour and Identity with Gaussian Mixtures*. Pattern Recognition, vol. 31, no. 12, 1998, s. 1883-1892.

[18] KJELDSEN, R., AND KENDER, J.: *Finding Skin in Color Images*. In Proceeding of the Second Int’l Conf. Automatic Face and Gesture Recognition, 1996, s. 312-317.

[19] CRAW, I., TOCK, D., AND BENNETT, A.: *Finding Face Features*. In Proceeding of the Second European Conf. Computer Vision, 1992, s. 92-96.

[20] LANITIS, A., TAYLOR, C. J., AND COOTES, T. F.: *An Automatic Face Identification System Using Flexible Appearance Models*. Image and Vision Computing, vol. 13, no. 5, 1995, s. 393-401.

[21] TURK, M., AND PENTLAND, A.: *Eigenfaces for Recognition*. J. Cognitive Neuroscience, vol. 3, no. 1, 1991, s. 71-86.

[22] SUNG, K.-K., AND POGGIO, T.: *Example-Based Learning for View-Based Human Face Detection*. IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 20, no. 1, 1998, s. 39-51.

[23] ROWLEY, H., BALUJA, S., AND KANADE, T.: *Neural Network-Based Face Detection*. IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 20, no. 1, 1998, s. 23-38.

[24] OSUNA, E., FREUND, R., AND GIROSI, F.: *Training Support Vector Machines: An Application to Face Detection*. In Proceeding of the IEEE Conf. Computer Vision and Pattern Recognition, 1997, s. 130-136.

[25] SCHNEIDERMAN, H., AND KANADE, T.: *Probabilistic Modeling of Local Appearance and Spatial Relationships for Object Recognition*. In Proceeding of the IEEE Conf. Computer Vision and Pattern Recognition, 1998, s. 45-51.

[26] RAJAGOPALAN, A., KUMAR, K., KARLEKAR, J., MANIVASAKAN, R., PATIL, M., DESAI, U., POONACHA, P., AND CHAUDHURI, S.: *Finding Faces in Photographs*. In Proceeding of the Sixth IEEE Int'l Conf. Computer Vision, 1998, s. 640-645.

[27] LEW, M. S.: *Information Theoretic View-Based and Modular Face Detection*. In Proceeding of the Second Int'l Conf. Automatic Face and Gesture Recognition, 1996, s. 198-203.

[28] *Face Detection using OpenCV* [online], Poslední modifikace 28. 5. 2008. Dostupné na URL: http://opencv.willowgarage.com/wiki/FaceDetection.

[29] BERGGREN, K., AND GREGERSSON, P.: *Camera focus controlled by face detection on GPU* [online], Poslední modifikace 16. 10. 2008. Dostupné na URL: http://graphics.cs.lth.se/theses/projects/facerecognition/.

[30] VIOLA, P., AND JONES, M.: *Rapid Object Detection using a Boosted Cascade of Simple Features*. 2001.

[31] BERGGREN, K., AND GREGERSSON, P.: *Camera focus controlled by face detection on GPU*. Master Thesis, Lund University, Sweden, 2008.

[32] *Video tracking* [online], Poslední modifikace 27. 2. 2009. Dostupné na URL: http://en.wikipedia.org/wiki/Video_tracking.

[33] BRADSKI, G. R.: *Computer Vision Face Tracking For Use in a Perceptual User Interface*. Intel Technology Journal, No. Q2. Microcomputer Research Lab, Santa Clara, CA, Intel Corporation, 1998.

[34] *OpenCV Wiki* [online], Poslední modifikace 18. 3. 2009. Dostupné na URL: http://opencv.willowgarage.com/wiki/.

[35] *OpenGL API Documentation Overview* [online], Poslední modifikace 26. 4. 2009. Dostupné na URL: http://www.opengl.org/documentation/.

[36] *The OpenGL Utility Toolkit* [online], Poslední modifikace 26. 4. 2009. Dostupné na URL: http://www.opengl.org/resources/libraries/glut/.

[37] *wxWidgets* [online], Poslední modifikace 27. 4. 2009. Dostupné na URL: http://www.wxwidgets.org/.

[38] *Microsoft Visual Studio* [online], Poslední modifikace 27. 4. 2009. Dostupné na URL: http://www.microsoft.com/express/download/.

Seznam příloh

CD

Složky v kořenovém adresáři:

- *run*
  - *detector*: obsahuje binární soubor detector.exe, který spouští program a dll knihovny potřebné pro běh programu
  - *install*: obsahuje instalační balíček nutný pro běh programu
- *source_code*: obsahuje projekt Visual Studia 2005 se zdrojovými soubory programu
- *test_data*: obsahuje testovací video sobory (viz 6)
- *text*: obsahuje bakalářskou práci ve formátu pdf a docx