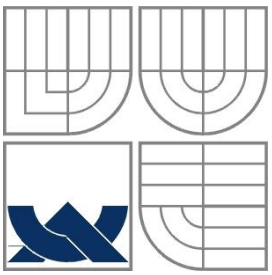
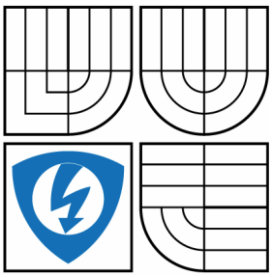


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

ROZPOZNÁVÁNÍ GEST RUKY V OBRAZE

HAND GESTICULATION RECOGNITION IN IMAGE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. STANISLAV MRÁZ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. KAREL HORÁK, Ph.D

BRNO 2011

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student:	Bc. Stanislav Mráz	***ID:***	98502
Ročník:	2	***Akademický rok:***	2010/2011

NÁZEV TÉMATU:

Rozpoznání gest ruky v obrazu

POKYNY PRO VYPRACOVÁNÍ:

Cílem studenta je seznámit se s možnostmi detekce částí lidského těla v obrazech se zaměřením především na detekci rukou a navrhnout algoritmus, který rozpozná několik základních statických gest ruky. Gesta slouží k jednoduchému ovládání PC. Podmínkou je zpracování obrazu v reálném čase s předpokladem na snímcích z webové kamery.

DOPORUČENÁ LITERATURA:

Hlaváč, V., Šonka, M.: Počítačové vidění. Praha: Grada, 1992.
Haußecker H., Geißler P.: Handbook of Computer Vision and Applications. San Diego: Academic press, 1999.

Termín zadání:	7.2.2011	***Termín odevzdání:***	23.5.2011

Vedoucí práce: Ing. Karel Horák, Ph.D.

prof. Ing. Pavel Jura, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Abstrakt

Tato diplomová práce se zabývá rozpoznáváním jednoduchých statických gest ruky za účelem ovládání počítače. Úvodní část práce je věnována teoretickému přehledu metod používaných pro nalezení ruky v obraze. Dále pak jsou popsány přístupy využívané pro klasifikaci gesta. Druhá část této práce je věnována výběru vhodného způsobu pro segmentaci ruky na základě barvy kůže a na základě pohybu. Poté jsou popsány metody pro rozpoznání gesta. Poslední část této práce se věnuje popisu navrhnutého řešení.

Klíčová slova

Gesta ruky, rozpoznání gesta, ovládání PC, neuronová síť, AdaBoost, kontury, detekce barvy pokožky, detekce pohybu, počítačové vidění, OpenCV

Abstract

This master's thesis is dealing with recognition of an easy static gestures in order to computer controlling. First part of this work is attended to the theoretical review of methods used to hand segmentation from the image. Next methods for hang gesture classification are described. The second part of this work is devoted to choice of suitable method for hand segmentation based on skin color and movement. Methods for hand gesture classification are described in next part. Last part of this work is devoted to description of proposed system.

Keywords

Hand gestures, gesture recognition, computer controlling, neural network, AdaBoost, contours, skin color detection, movement detection, komputer vision, OpenCV

Bibliografická citace:

MRÁZ, S. Rozpoznávání gest ruky v obraze. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. 81 s. Vedoucí diplomové práce Ing. Karel Horák, Ph.D.

Prohlášení

„Prohlašuji, že svou diplomovou práci na téma Rozpoznávání gest ruky v obraze jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: **23. května 2011** ……

podpis autora

Poděkování

Děkuji Ing. Iloně Kalové, Ph.D a Ing Karlu Horákovi, Ph.D za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne: **23. května 2011**

……
podpis autora

Obsah

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

1 ÚVOD

Počítače a obecně informační technologie zasahují do každodenního života téměř každého z nás. Bez mobilního telefonu, notebooku nebo osobního počítače si mnoho lidí již svůj běžný den nedokáže představit. S klesající cenou těchto zařízení docházelo k jejich rozšiřování mezi širší veřejnost. Postupem času se tato zařízení vyvíjela, rostl výkon, objevovaly se nové možnosti využití, zmenšovaly se také jejich rozměry, měnily se způsoby ovládání. Počítače (dále jen PC) byly pro ovládání vybaveny nejprve pouze klávesnicí. Příchod grafických rozhraní znamenal nutnost modifikace ovládání. Klasické ovládání pomocí klávesnice bylo doplněno ovládáním pomocí myši. Mezi další vstupní periferie, jejichž funkce je podobná myši lze například zařadit trackball a touchpad. Výše zmiňované periferie mají společný jmenovatel a to nutnost přímé interakce uživatele s danou periferií což může působit problémy například pohybově postiženým uživatelům. Tato skutečnost přispěla k vývoji alternativních způsobů ovládání PC. Mezi tyto lze zařadit například ovládání hlasem nebo ovládání PC pohybem oka. Ovládáním PC pohybem oka se zabývá mnoho prací a v dnešní době je již k dispozici několik komerčních produktů. Mezi něž patří například systém i4control. Další alternativou ovládání PC je jeho ovládání pomocí pohybu těla nebo pomocí gest ruky. Tento způsob ovládání PC není komerčně zatím příliš rozšířen. Ovládání pomocí gest ruky a pomocí pohybu celého těla využívá například společnost Sony u svého produktu EyeToy, dalším produktem využívajícím pro hraní her pohybu těla je „projekt Natal“ společnosti Microsoft dnes běžně dostupný i v české republice pod názvem Kinect.

Cílem této práce je navrhnout a implementovat rozhraní pro ovládání počítače pomocí gest ruky. Gesta ruky budou snímána pomocí běžně dostupné web kamery, nebo web kamery implementované přímo v těle notebooku. Podmínkou je práce v reálném čase.

První část této práce pojednává o způsobech detekce částí lidského těla respektive ruky v obraze. Následuje přehled metod, které lze použít pro klasifikaci jednotlivých gest. V těchto kapitolách jsou uváděny metody a postupy, které využili ve svých pracích různí autoři.

Další kapitoly a podkapitoly uvádějí vlastní postup prací, vyzkoušené metody, výběr metod a dosažené výsledky.

2 DETEKCE RUKY V OBRAZE

V této části práce budou zmíněny různé metody a přístupy pro nalezení částí lidského těla v obraze. Hlavně se zde budu věnovat nalezení ruky v obraze. Je zřejmé, že detekce ruky v obraze a její rozlišení od ostatních objektů v obraze je prvním krokem k rozpoznávání gest. Kvalita nalezení ruky v obraze do značné míry ovlivňuje úspěšnost rozpoznání gesta. Správnou detekcí ruky také dochází k redukci zpracovávaných dat. K detekci ruky v obraze existuje několik různých přístupů. V následujícím textu je uveden jejich stručný popis.

2.1 Detekce ruky na základě barvy kůže

Detekce ruky na základě barvy kůže patří k nejčastěji používaným přístupům. Jedná se o poměrně snadnou a rychlou metodu, kdy se u každého pixelu rozhoduje, zda jeho barva odpovídá barvě lidské kůže. Barva lidské kůže nabývá specifických odstínů. U různých osob a v závislosti na osvětlení se liší převážně pouze v jasové složce. Pro detekci oblastí lidské kůže je velice důležitý výběr barevného modelu pro reprezentaci barvy lidské kůže. Jak již bylo zmíněno, barva kůže nabývá specifických hodnot a k největším změnám dochází pouze v jasové složce. Proto je vhodné použít barevný model, u kterého je pro reprezentaci barvy využívána oddělená jasová složka.

Je zřejmé, že při snímání scény web kamerou se v obraze může vyskytovat více oblastí, které budou odpovídat barvě kůže. Například obličej nebo jiné objekty, velice často dřevěné, které svou barvou odpovídají barvě kůže. Proto je nutné metodu detekce ruky na základě barvy kůže používat v kombinaci s dalšími metodami, které budou zmíněny dále. Obecně při řešení úloh počítačového vidění je většinou nutné použít kombinaci segmentačních metod. Jen velmi málo metod je schopno poskytovat uspokojivé výsledky bez použití dalších doplňkových segmentačních metod.

2.1.1 Barevné prostory

V průběhu let a pro různé aplikace a případy bylo vyvinuto mnoho barevných prostorů s odlišnými vlastnostmi. Pro detekci barvy kůže různí autoři prací využili různé barevné prostory. V této kapitole vycházím především z práce [31], kde je uveden přehled barevných prostorů a je také diskutována jejich vhodnost pro detekci barvy lidské kůže.

2.1.1.1 RGB Barevný prostor

Jedná se o barevný prostor, který se používá pro vyjádření barev například v monitorech. Jde o takzvaný aditivní prostor kdy je barva vytvářena mícháním třech barevných složek R – red, G – green, B – blue. Tento prostor není příliš vhodný pro detekci objektů na základě barvy, protože je zde velká závislost mezi jednotlivými složkami a není u něj oddělena jasová složka. [31], [24]

2.1.1.2 Normalizovaný RGB prostor

Tento prostor je již pro detekci pokožky daleko vhodnější, protože zde dochází ke zmenšení závislosti na jasu pomocí normalizace. Lze jej získat z RGB prostoru pomocí normalizačních vztahů:

$$r = \frac{R}{R+G+B} \qquad g = \frac{G}{R+G+B} \qquad b = \frac{B}{R+G+B} \tag{2.1}$$

Dále pak platí že: $R+G+B = 1$. Z toho vyplývá, že třetí složka b nenese žádnou informaci a může být vynechána. Význačnou vlastností tohoto prostoru je to, že pro matné povrchy a při zanedbání okolního osvětlení je invariantní vůči natočení povrchu ke zdroji světla. Nelze zanedbat také jednoduchý převod RGB modelu na normalizovaný RGB barevný prostor. [31], [24]

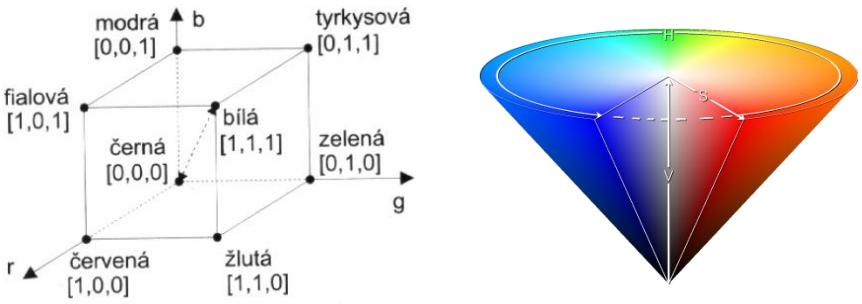
2.1.1.3 HSV Barevný prostor

Barevný prostor HSV (H – hue – barevný odstín, S – saturation – sytost, V –value – hodnota jasu) je svým uspořádáním velice blízký vnímání barvy člověkem. Pro detekci barvy pokožky je vhodný, protože je u něj oddělena jasová složka. Jeho nevýhodou je jeho nespojitost v H složce. Nevýhodou může také být složitost převodu RGB prostor na prostor HSV. Převod lze provádět pomocí následujících vztahů:

$$H = \arccos \frac{\frac{1}{2}\cdot\left((R-G)+(R-B)\right)}{\sqrt{\left((R-G)^2+(R-G)\cdot(G-B)\right)}} \tag{2.2}$$

$$S = 1 - 3\cdot\frac{\min(R,G,B)}{R+G+B} \tag{2.3}$$

$$V = \frac{1}{3}\cdot(R+G+B) \tag{2.4}$$



a) b)

Obr. 2.1: a) Barevný prostor RGB, b) Barevný prostor HSV

Převzato z [24], [12]

2.1.1.4 YCbCr barevný prostor

Jedná se o barevný prostor, který byl pro segmentaci částí lidského těla použit v pracích mnoha autorů. Jeho výhodou je oddělení barevné a jasové složky což je velice důležitým faktorem pro výběr barevného prostor při detekci částí lidského těla. Jednotlivé barevné složky tohoto prostoru lze získat převodem z prostoru RGB a to pomocí následujících vztahů:

$$Y = 0{,}299 \cdot R + 0{,}587 \cdot G + 0{,}114 \cdot B \tag{2.5}$$

$$C_r = R - Y \tag{2.6}$$

$$C_b = B - Y \tag{2.7}$$

Barva je reprezentována jasem Y, který je vypočítán pomocí vztahu (2.5) a dvěma rozdílovými složkami C_r *a* C_b vypočítanými pomocí vztahů (2.6) a (2.7). [31], [24]

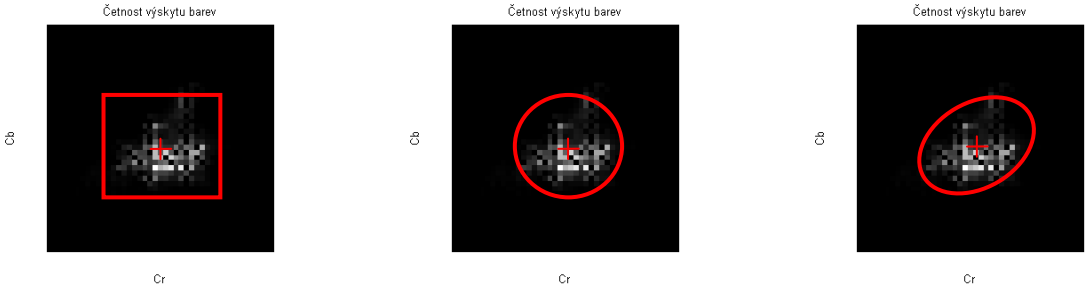
2.1.2 Model barvy kůže

Výběr modelu barvy kůže je dalším krokem k oddělení pozadí od oblasti ruky, která nás primárně zajímá. Možným řešením je vytvoření histogramu ze vzorků kůže a zjištění četnosti výskytu jednotlivých barevných odstínů. Obecně je třeba vytvořit rozhodovací pravidlo (klasifikátor) které rozhodne o tom, zda daný pixel obrazu náleží třídě kůže či nikoli. Problémem výběru barevného prostoru se je poměrně obsáhle popsán v práci [31], ze které také čerpám.

2.1.2.1 Explicitně definované barevní rozsahy kůže

Podstatou této metody je definovat pro určitý barevný prostor pravidla pro jeho jednotlivé složky, kterými bude určen rozsah barvy kůže. Obecně lze pro jednotlivé složky barevného prostoru použít několik způsobů, kterými lze vymezit barvu kůže. Prvním z nich je vymezení čtvercové či obdélníkové oblasti pro každou složku barevného prostoru. Pixely, jejichž hodnota spadá do vymezené obdélníkové oblasti, jsou označeny za pixely pokožky. Tento způsob vymezení není příliš vhodný, jelikož zde dochází k poměrně velké chybě klasifikace a to zejména u pixelů, nacházejících se v rozích této oblasti. Tuto chybu lze částečně eliminovat tak, že pro vymezení oblasti pokožky použijeme kružnici či elipsu. Za pixely pokožky jsou pak opět označeny ty, které spadají do oblasti vymezené zmiňovanými geometrickými útvary. Jednotlivé případy jsou znázorněny na obrázku Obr. 2.2 jsou znázorněny jednotlivé případy a) vymezení oblasti čtvercem/obdélníkem, b) vymezení oblasti kruhem c) vymezení oblasti elipsou.

Obecně tato metoda vyniká svou jednoduchostí a tudíž i svou nízkou výpočetní náročností. Nevýhodou je obtížnost stanovení hranic a volba vhodného barevného prostoru. [31]



a) **b)** **c)**

Obr. 2.2: Příklady geometrických útvarů pro explicitně definované rozsahy barvy pokožky

2.1.2.2 Modely založené na histogramu

Dalším možným způsobem pro vytvoření modelu barvy kůže je využít některou metodu využívající histogram. Prvním z nich může být takzvaná normalizovaná vyhledávací tabulka (Look Up Table LUT). Ze vzorků barvy kůže je vytvořen histogram, který je následně normalizován. Normalizované hodnoty vyhledávací tabulky udávají pravděpodobnost, zda daný pixel je oblastí pokožky.

$$P_{pok} = \frac{pok[c]}{norm} \tag{2.8}$$

Kde *pok[c]* udává počet výskytů určité barvy a *norm* je součet všech vyskytujících se barev v histogramu.

Pro vytvoření rozhodovacího pravidla byl v některých pracích použit bayesův klasifikátor. Při použití bayesova klasifikátoru, je vytvořena takzvaná podmíněná pravděpodobnost. To znamená, že pravděpodobnost pro daný pixel se stanovuje pouze pro určitou konkrétní barvu. Pravděpodobnost je tedy dána vztahem:

$$P(pok \mid c) = \frac{P(c \mid pok) \cdot P(pok)}{P(c \mid pok) \cdot P(pok) + P(c \mid -pok) \cdot P(-pok)} \tag{2.9}$$

Ze vztahu (2.9) je patrné, že pro výpočet výsledné pravděpodobnosti musíme znát pravděpodobnosti, zda daný pixel je pokožka - *P(c|pok)* a zda není pokožka *P(c|-pok).* Tyto pravděpodobnosti se získají z histogramu vzorků pokožky a vzorků, které pokožku neobsahují. Za pixel obsahující pokožku je pak prohlášen takový, pro který je pravděpodobnost větší než určitá stanovená mez.

Nevýhodou modelů získávajících pravděpodobnosti z histogramu je jejich závislost na vybraných vzorcích pokožky, ze kterých se sestavují pravděpodobnosti. Jejich výhodou je naopak rychlost. [31]

2.1.2.3 Modely s gaussovým rozložením

Velice často se pro definici barvy pokožky používají modely s gassovým rozložením. Jednak může být použit model s jedním gaussiánem nebo model směsí gaussovských funkcí. Pro model s jendím gaussíánem platí následujíí vztahy:

$$p(c \mid pok) = \frac{1}{2 \cdot \pi \cdot \left| \sum_s \right|^{1/2}} \cdot e^{-\frac{1}{2}(c-\mu_s)^T \cdot \sum_s^{-1} \cdot (c-\mu_s)} \tag{2.10}$$

$$\mu_s = \frac{1}{n} \sum_{j=1}^{n} c_j \tag{2.11}$$

$$\sum_s = \frac{1}{n-1} \cdot \sum_{j=1}^{n} (c_j - \mu_s) \cdot (c_j - \mu_s)^T \tag{2.12}$$

Kde c_j je vzorek z trénovací množiny o n prvcích, μ_s je střední hodnota respektive vektor středních hodnot a $\sum_s$ je kovarianční matice, která ve vícerozměrných případech gaussova rozložení nahrazuje rozptyl.

Model směsí gaussovských funkcí je v podstatě zobecnění modelu s jedním gaussiánem. Výsledná pravděpodobnost je pak dána vztahem:

$$p(c \mid pok) = \sum_{i=1}^{k} \pi_i \cdot p_i(c \mid pok) \tag{2.13}$$

Jedná se o sumu pravděpodobností p_i, kde k je počet komponent směsí, π_i je váha. Jednotlivé pravděpodobnosti p_i mají gaussovo rozložení. [15], [31]

2.1.2.4 Ostatní používané modely

V předchozích kapitolách byly zmíněny modely využívající histogramy – neparametrické a modely s gaussovým rozložením – parametrické. Tento výčet však není kompletní. Mezi další používané modely lze zařadit samoorganizující se tabulku Self-organizing map – SOM. Tato patří do neparametrických modelů. Mezi další modely parametrické lze zařadit eliptický model, který při stejné rychlosti a jednoduchosti dosahuje podobných výsledků jako model s jedním gaussiánem. Protože se mohou podmínky (světelné, barva kůže různých lidí atd.) s časem měnit, byly navrženy modely, které přepočítávají rozložení barvy kůže. Tyto modely se nazývají dynamické. Více o těchto modelech se lze dočíst v práci: [31].

2.2 Detekce ruky na základě pohybu

Jak již bylo řečeno, metodu detekce ruky v obraze podle barvy kůže je vhodné kombinovat s dalšími metodami. Při snímání scény web kamerou, pro kterou platí předpoklad statického umístění, se může v obraze vyskytovat více objektů, které svou

barvou odpovídají barvě kůže. Například obličej, dřevěná skříň nebo jiné dřevěné objekty atd. Zásadní rozdíl mezi těmito objekty a námi hledanou rukou je ten, že u ruky provádějící gesto se předpokládá větší potažmo rychlejší změna polohy v obraze nežli například u obličeje, který se sice může také pohybovat, ale změny polohy nebudou již tak výrazné.

Proto se jeví jako vhodné použít některou z metod pro nalezení pohybujících se objektů v obraze. Jejich výčet a popis je uveden v následujícím textu.

2.2.1 Rozdílové metody

Patří mezi nejjednodušší metody, u kterých se zjišťuje rozdíl dvou po sobě následujících snímků. Pokud se jasová úroveň pixelů liší o více než určitou stanovenou mez ε je tento pixel prohlášen za pixel pohybujícího se objektu. Zásadní nevýhodou je to, že nerozlišujeme směr pohybu. Matematicky lze danou metodu popsat takto:

$$d(x,y)=\begin{cases}0 & |f_1(x,y)-f_2(x,y)|<\varepsilon\\ 1 & |f_1(x,y)-f_2(x,y)|\geq\varepsilon\end{cases} \tag{2.14}$$

Tuto metodu lze ještě dále rozdělit na jednosměrnou a obousměrnou. U jednosměrné se uvažují pouze kladné změny jasu prvního snímku oproti snímku druhému. V tomto případě by ve vzorci (2.14) nebyla absolutní hodnota. U obousměrné metody se uvažují jak kladné tak záporné změny jasu.

Pro získání informace o směru pohybu lze použít metodu kumulativního rozdílového snímku, kdy se vypočítává rozdíl referenčního snímku a snímku aktuálního. Výsledný rozdílový snímek se pak přičte k předcházejícímu rozdílovému snímku. Rozdílové snímky mohou mít různé váhy w_i [11].

$$d_{kum}(x,y)=\sum_{i=1}^{N} w_i\cdot|f_1(x,y)-f_2(x,y)| \tag{2.15}$$

2.2.2 Estimace modelu prostředí

Metoda estimace modelu prostředí vyžaduje možnost nasnímat statickou scénu bez přítomnosti objektu. V nejjednodušším případě je za model prostředí považován první snímek. Toto řešení však není příliš vhodné například kvůli možnosti výskytu šumu. Mnohem častěji se využívá model prostředí sestavený z průměru n snímků. Toto řešení vede k odstranění chyb způsobených šumem nebo mírnými změnami osvětlení. U této metody jako i u metod předchozích vyvstává problém se stanovením prahu ε [11].

$$d(x,y)=\begin{cases}0 & |f(x,y)-b(x,y)|<\varepsilon\\ 1 & |f(x,y)-b(x,y)|\geq\varepsilon\end{cases} \tag{2.16}$$

Ve vzorci (2.16) *f(x,y)* je aktuální snímek a *b(x,y)* je model prostředí. Metoda estimace modelu prostředí se objevuje v mnoha modifikacích. Jako například v [9] kde je uveden modifikace vyhledávání objektu pomocí modelu prostředí. Model je zde postupně aktualizován podle vztahu:

$$b_n = \alpha \cdot b_{n-1} + (1-\alpha) \cdot f_n \qquad (2.17)$$

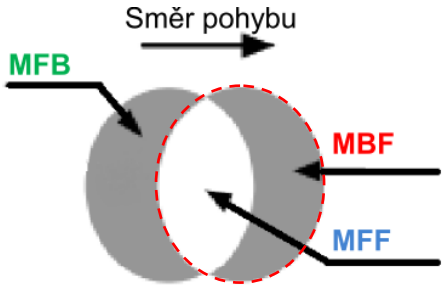
Koeficient α značí, jak moc bude aktuální snímek měnit model prostředí, čím menší je α, tím rychleji dochází k aktualizaci modelu. Koeficient alfa může nabývat hodnot 0 až 1.

Při aktualizaci modelu vyvstává problém, pokud dojde k rychlému pohybu bodu v pozadí nebo k náhlému zastavení bodu popředí. Jelikož je pozadí aktualizováno se stejným koeficientem (1 – α) bude tento chybný bod v modelu obsažen dlouho (než dojde ke kompletní aktualizaci modelu) a tudíž dojde k detekci falešného objektu.

Řešením je aktualizovat model podle toho jak velké změny nastaly ve dvou po sobě následujících snímcích. Podle vztahu (2.18) vypočítáme binární rozdíl dvou snímků $D_f(x,y)$.

$$D_f(x,y) = \begin{cases} 0 & |f_n(x,y) - f_{n-1}(x,y)| < \varepsilon \\ 1 & |f_n(x,y) - f_{n-1}(x,y)| \geq \varepsilon \end{cases} \qquad (2.18)$$

Pixely, které jsou obsaženy v absolutním binárním rozdílu $D_f(x,y)$, se dále dělí do tří skupin: popředí *MBF*, díry v popředí *MFF* a falešné pixely MFB. Význam jednotlivých kategorií je názorně zobrazen na obrázku Obr. 2.3. Kdy je pohybujícím se objektem elipsa vyznačena červenou přerušovanou čárou.



Obr. 2.3: Skupiny pixelů pohybujícího se objektu

Poté je možné měnit model prostředí pro jednotlivé kategorie odděleně. Pixely spadající do kategorie falešných pixelů se změní co nejrychleji, pixely kategorie pozadí se nebudou měnit vůbec. Pro každou skupinu je definována binární maska podle následujících vztahů:

- Popředí $MBF = D_f \wedge d$ (2.19)
- Díry v popředí $MFF = \neg D_f \wedge d$ (2.20)
- Falešné objekty $MFB = \neg D_f \wedge \neg d$ (2.21)

Potom pozadí bude aktualizováno následovně:

$$b_n(x,y)=\begin{cases}\alpha_1\cdot b_{n-1}(x,y)+(1-\alpha_1)\cdot f_n(x,y)\\ MBF(x,y)=1\\ \alpha_2\cdot b_{n-1}(x,y)+(1-\alpha_2)\cdot f_n(x,y)\\ MFF(x,y)=1\\ \alpha_3\cdot b_{n-1}(x,y)+(1-\alpha_3)\cdot f_n(x,y)\\ MFB(x,y)=1\\ \alpha_4\cdot b_{n-1}(x,y)+(1-\alpha_4)\cdot f_n(x,y)\\ jinak\end{cases}$$

K aktualizaci pozadí dochází za přispění čtyř koeficientů $\alpha_1 = 1$ a slouží k redukci vlivu popředí na pozadí, $\alpha_2 = 1$ neboť oblast díry v popředí nijak nepůsobí na pozadí, $\alpha_3 = 0$ neboť pixely, které byly v předchozím snímku v popředí, se v následujícím snímku přesunou do pozadí. Pro ostatní nezařazené pixely se volí koeficient α_4 v rozsahu nula až jedna. [34], [9]

2.2.2.1 Otsu metoda

Problematická volba prahu ε je v práci [9] řešena pomocí metody Otsu. Předpokládá se šedotónový obraz obsahující L úrovní šedi, který lze pomocí určitého prahu *t* rozdělit na objekty popředí a objekty pozadí.

$$C_1=\{0,1,....,t\},\qquad C_2=\{t+1,t+2,....,L-1\} \tag{2.22}$$

K výpočtu prahu se používá histogram šedotónového obrazu. Respektive je počítáno rozložení pixelů pro obě třídy

$$\omega_1(t)=\sum_0^t p_i\ ,\qquad \omega_2(t)=\sum_{t+1}^{L-1} p_i \tag{2.23}$$

Kde $p_i = n_i / n$ a n_i je počet pixelů o hodnotě *i* a *n* je celkový počet pixelů. Následně je vypočítána průměrná hodnota jasu v obou třídách:

$$u_1(t)=\sum_0^t i\cdot p_i/\omega_1(t),\qquad u_2(t)=\sum_0^t i\cdot p_i/\omega_2(t) \tag{2.24}$$

Metoda otsu následně hledá maximální rozdíl dvou tříd:

$$T=\arg\max_{0\le t\le L-1}\{\sigma^2(t)\} \tag{2.25}$$

Kde $\sigma^2(t)$ reprezentuje rozdíl dvou tříd a platí pro něj následující vztah:

$$\sigma^2(t)=\omega_1(t)\cdot(u_1(t)-u_0)^2+\omega_2(t)\cdot(u_2(t)-u_0)^2 \tag{2.26}$$

$$u_0=\sum_0^{L-1} i\cdot p_i \tag{2.27}$$

Metoda otsu pracuje dobře, pokud jsou v obraze respektive v histogramu zřetelně patrné objekty pozadí a popředí. Její výsledky mohou být do značné míry ovlivněny šumem. [9]

3 ROZPOZNÁNÍ GESTA

Pro rozpoznání gesta ruky je nejprve důležité si uvědomit, co to vlastně gesto je potažmo jaká gesta budeme klasifikovat. Variabilita gest je velice široká. Gesta mohou být prováděna pouze dlaní ruky a různým postavením prstů – dále tato gesta budeme nazývat gesty statickými. Nebo mohou být gesta prováděna pohybem ruky – gesta dynamická. Další kategorií jsou gesta prováděná oběma rukama naráz a jejich vzájemným postavením vzhledem k celému tělu člověka. Variabilita gest nevyvstává pouze z velkého množství gest, ale také z toho, že různí lidé provádějí gesta odlišně.

Z principu není možné, aby jedno shodné gesto bylo vykonáno naprosto stejně v podání různých lidí. Dokonce ani stejné gesto vykonávané opakovaně stejnou osobou není provedeno vždy shodně.

Z textu výše vyplývá, že rozpoznání gesta a jeho klasifikace je velice náročným úkolem. V dalším textu budou zmíněny metody, které se pro klasifikaci gesta dají použít, zejména se budeme zabývat statickými gesty, která jsou předmětem této diplomové práce. Pro zajímavost budou uvedeny také některé metody, které se používají pro rozpoznání gest dynamických.

3.1 Metody založené na kontuře

Tyto metody se využívají pro klasifikaci jednoduchých statických gest. Lze pomocí nich například rozlišit ruku zaťatou v pěst nebo ruku s roztaženými prsty. Prvním krokem u této metody je určení kontury objektů. K tomuto účelu se používá segmentační metoda, která se nazývá aktivní kontury nebo také snakes.

3.1.1 Aktivní kontury

Jedná se o metodu vyšší úrovně segmentace, kdy dochází k iterativnímu tvarování uzavřené řízené kontury postupně směrem k hranám segmentovaného objektu. Tvarování je prováděno na základě vnitřních, obrazových a vnějších sil. Vnitřní síly E_N zajišťují hladkost průběhu kontury, obrazové síly E_I směřují konturu směrem k hranám objektu a vnější síly E_T jsou výsledkem počátečního umístění kontury.

Uvažujeme – li konturu jako sadu diskrétních bodů:

$$p_n = [x_n, y_n], \; pro \; N = 0,1,...N \tag{3.1}$$

Poté pro výslednou pozici kontury, která odpovídá lokálnímu minimu energie lze psát:

$$E_S = \sum_{n=1}^{N} E_N\{p_n\} + \sum_{n=1}^{N} E_I\{p_n\} + \sum_{n=1}^{N} E_T\{p_n\} \tag{3.2}$$

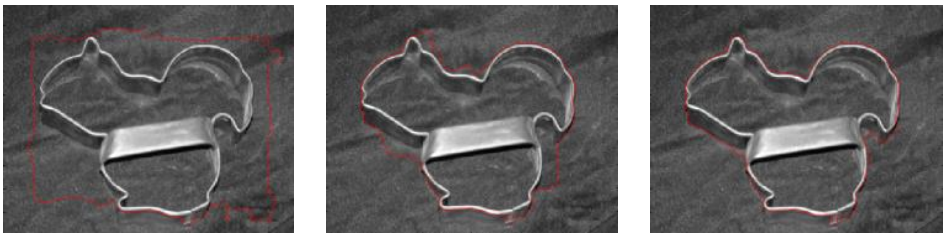
Pro měření velikosti energie bylo navrženo mnoho postupů. Lze například zmínit takzvaný nenasycený algoritmus, kde je vnitřní energie definována jako energie spojitosti E_C a energie zakřivení E_K.

$$E_N = \alpha(n) \cdot E_C\{p_n\} + \beta(n) \cdot E_K\{p_n\} \tag{3.3}$$

$$E_C = \frac{d - |p_n - p_{n-1}|}{\max\{d - |p_n(j) - p_{n-1}|\}} \tag{3.4}$$

$$E_K = \frac{|p_{n-1} - 2 \cdot p_n + p_{n+1}|^2}{\max\{|p_{n-1} - 2 \cdot p_n(j) + p_{n+1}|^2\}} \tag{3.5}$$

Kde $\alpha(n)$ a $\beta(n)$ udávají pružnost kontury, d udává průměrný obvod kontury a $p_n(j)$ reprezentuje osmi-okolí bodu p_n. [16], [28]

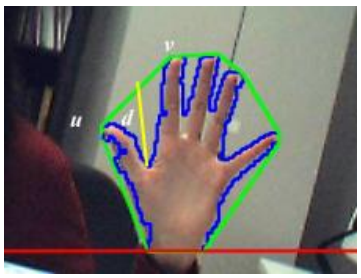


Obr. 3.1: Postupné tvarování kontury k hranicím objektu

Převzato z [16]

3.1.2 Určení gesta

Po nalezení kontury přichází na řadu nalezení konvexní obálky. V dalším kroku dochází ke hledání bodů ležících na konvexní obálce a počítání maximální vzdálenosti konvexní obálky od kontury mezi dvěma sousedními body ležícími na konvexní obálce. Zprůměrováním těchto vzdáleností dostáváme příznak d, podle kterého se rozhoduje o typu gesta. Počítá se tedy počet defektů d, jejich velikost nebo jejich vzájemná pozice. Situace je znázorněná na Obr. 3.2. [19]

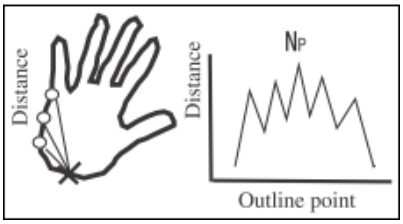


Obr. 3.2: Rozpoznání gesta pomocí kontury

Převzato z [19]

Pomocí kontur a konvexní obálky lze získávat i jiné příznaky, jako například obvod objektu (respektive délku kontury), lze také opsat danému objektu obdélní nebo elipsu a z těchto geometrických útvarů získávat další informace, jako je například poměr hlavní a vedlejší osy elipsy nebo její natočení. U opsaného obdélníka lze porovnávat například poměr stran a z tohoto stanovovat podlouhlost. Jedná se tedy o poměrně rychlou a spolehlivou metodu klasifikace jednoduchých statických gest.

Dalším možným využitím kontury objektu pro klasifikaci gesta je hledání výběžků na kontuře viz Obr. 3.3, kterou zmiňují autoři práce [29].



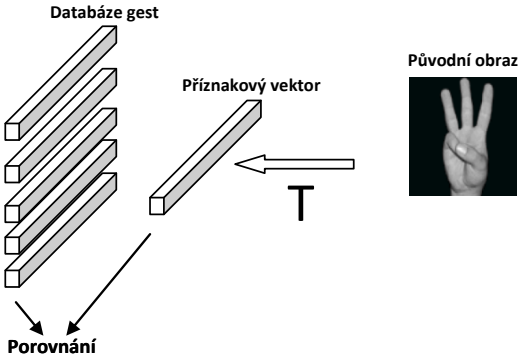
Obr. 3.3: Rozpoznání gesta pomocí výběžku kontury

Převzato z [29]

3.2 Metoda histogramů orientace

Jedná se o nízko-úrovňovou metodu, která vyniká velkou rychlostí a má dostatečně velkou diskriminační schopnost. Je minimálně závislá na změně osvětlení. Nejprve je ze vstupního obrazu pomocí transformace *T* získán vektor příznaků, který je dále porovnáván s vektory příznaků trénovací množiny gest. Situace je znázorněna na obrázku Obr. 3.4.

Abychom pro stejné gesto v různých místech obrazu získaly stejný vektor příznaků, používají se jako vektory příznaků lokální histogramy orientace.



Obr. 3.4: Obecný princip určení gesta

Postup pro výpočet histogramů orientace je následující. Nejprve jsou ve vstupním šedotónovém obraze vypočítány změny kontrastu zvlášť pro souřadnici *x* a pro souřadnici *y*. Získáme tak hodnoty *dx* a *dy* pro každý dílčí bod obrazu. Podle vzorce (3.6) vypočítáme matici kontrastů a podle vzorce (3.7) vypočítáme matici orientací.

$$C(x, y) = \sqrt{dx^2 + dy^2} \tag{3.6}$$

$$d(x, y) = \arctan(dx, dy) \tag{3.7}$$

Matici orientací však nelze použít rovnou jako příznakový vektor, protože její hodnoty jsou závislé na pozici ruky v obraze. Není invariantní vůči posunutí klasifikovaného objektu. Invariantnost vůči posunutí docílíme tím, že počítáme četnost výskytu jednotlivých orientací. Vytvoříme tak jakýsi histogram orientací.

V práci [8] je použit histogram orientací o 36 kontejnerech a to z důvodu pravidelného členění polárního prostoru po deseti stupních.

Při vytváření historamu orientací se ještě rozhoduje o tom, zda bude daná orientace do histogramu započítána. To zda je započítána se určuje porovnáním hodnoty v matici kontrastů *C(x,y)* s *určitým* prahem. Tato podmínka je vyjádřena následujícím předpisem:

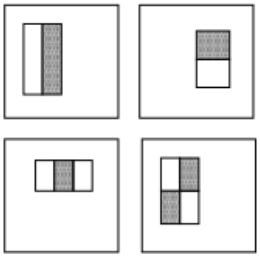
$$d(x, y) \in pv \quad \text{pokud} \quad C(x, y) > k_a \cdot k \tag{3.8}$$

Kde k_a udává průměrnou hodnotu kontrastu v matici *C(x,y)* a *k* je konstanta volená v rozmezí 1,2 až 2,7 a *pv* je příznakový vektor. Takto vytvořený histogram je dále filtrovan. Jedná se v podstatě o průměrování za použití pěti-prvkové masky *M(x)=[1 4 6 4 1]* .

Pro klasifikaci gesta je v práci [8] použita euklidovská metrika. U této metody je počítána vzdálenost mezi testovacím příznakovým vektorem a jednotlivými příznakovými vektory uloženými v databázi gest. [8], [34]

3.3 Ostatní metody pro statická gesta

Mezi dalšími metodami lze zmínit metodu navrženou pány Violou a Jonesem podle kterých je také pojmenována Viola&Jones tato metoda je blíže popsána v dokumentu [32]. Jako příznakový vektor je v této metodě použito velké množství takzvaných haarových příznaků, které lze vypočítat za použití masek na obrázku Obr. 3.5.



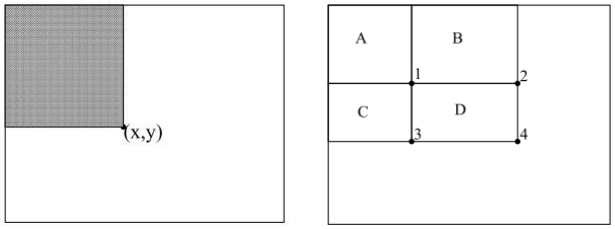
Obr. 3.5: Příklady masek používaných pro výpočet Haarových příznaků

Převzato z [32]

Hodnota příznaku se určuje tak, že se vybraná maska umístí nad obraz a počítá se suma hodnot pod maskou, s tím, že hodnoty pixelů pod bílou částí masky se počítají s kladnými znaménky a hodnoty pod šedou respektive černou částí masky se zápornými znaménky. Zrychlení výpočtu se dosahuje použitím takzvaného integrálního obrazu. Bod integrálního obrazu je dán předpisem:

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') \tag{3.9}$$

Kde *ii(x,y)* je bod integrálního obrazu a *i(x´, y´)* je bod obrazu původního. Poté lze pomocí integrálního obrazu snadno vypočítat sumu pixelu pod jakoukoli obdélníkovou částí obrazu. Toto je výhodné právě pro výpočet sumy hodnot pod maskou. Například plochu pod obdélníkem D na obrázku Obr. 3.6 b) lze vypočítat pouze pomocí čtyř bodů. Platí tedy $D_s = 4+1 - (2+3)$.

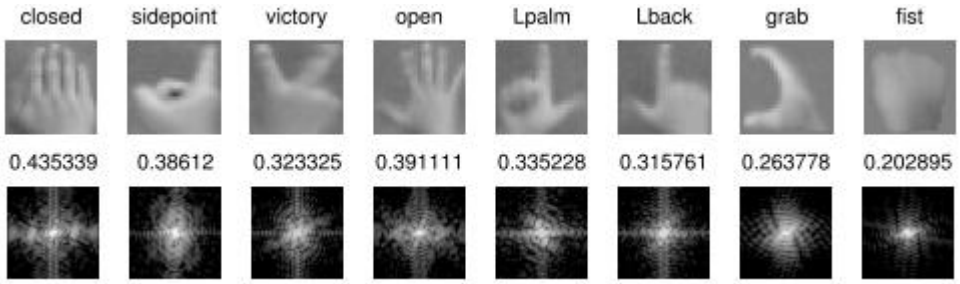


Obr. 3.6: a) Bod integrálního obrazu je dán součtem hodnot pixelu vlevo a nad tímto bodem b) Členěný integrální obraz

Převzato z [32]

Pro zrychlení výpočtu bývá často daná metoda rozdělena do několika úrovní, kdy se vždy pracuje jen s částí obrazu. Nevýhodou této metody je velká variabilita Haarových příznaků a dlouhá doba trénování. [32]

V práci [17] autoři klasifikují osm pozic ruky, tyto jsou uvedeny na Obr. 3.7.



Obr. 3.7: Příklady gest použitých v práci [17]

Převzato z [17]

Pro natrénování klasifikátoru byly použity výřezy gest o velikosti 25x25 pixelů respektive jejich reprezentace ve frekvenční oblasti. Pro převod byla použita Fourierova

transformace podle vzorce (3.10). Následně byla stanovena Fourierova transformace čistého snímku označovaná jako *P(u,v).* Poté je vypočítána rozdílová transformace *D(u,v)* dle vztahu (3.12). Posledním krokem této metody je výpočet součtu všech amplitud frekvencí, který je ještě normalizován. Tato suma je pak výsledkem pro daný klasifikátor. [17]

$$F(u,v) = \frac{1}{25 \cdot 25} \sum_{m=0}^{24} \sum_{n=0}^{24} I(m,n) \cdot e^{-i2\pi\left(\frac{mu}{25}+\frac{nv}{25}\right)} \tag{3.10}$$

$$P(u,v) = \frac{1}{25 \cdot 25} \sum_{m=0}^{24} \sum_{n=0}^{24} I(m,n) \cdot \frac{1}{2} \cdot e^{-i2\pi\left(\frac{mu}{25}+\frac{nv}{25}\right)} \tag{3.11}$$

$$D(u,v) = \log\left|F(u,v) - P(u,v)\right| \tag{3.12}$$

$$s = e^{\frac{1}{k} \cdot \sum_{u,v} D(u,v)} \tag{3.13}$$

3.4 Klasifikace statického gesta

Metody popisované v kapitolách 3.2 a 3.3 se obecně řečeno zabývají pouze získáním příznaků respektive příznakových vektorů pro klasifikaci gest. Mechanismy pro klasifikaci gest jsou tedy samostatnou kapitolou. U histogramů orientace navrhují autoři práce [8] použít pro klasifikaci gesta jednoduchou euklidovskou metriku kde se porovnává pouze vzdálenost jednotlivých bodů příznakového vektoru a bodů vektorů trénovacích. V práci [26] navrhuje autor práce použít jako příznakový vektor histogram orientace, avšak pro klasifikaci navrhuje použít neuronové sítě. V práci [32] pánové Viola a Jones používají pro klasifikaci metodu AdaBoost. Obecně lze říci, že neuronové sítě a metoda AdaBoost respektive Boosting se pro klasifikaci gest používají poměrně často. Proto k těmto dvěma klasifikátorům bude uvedena základní teorie.

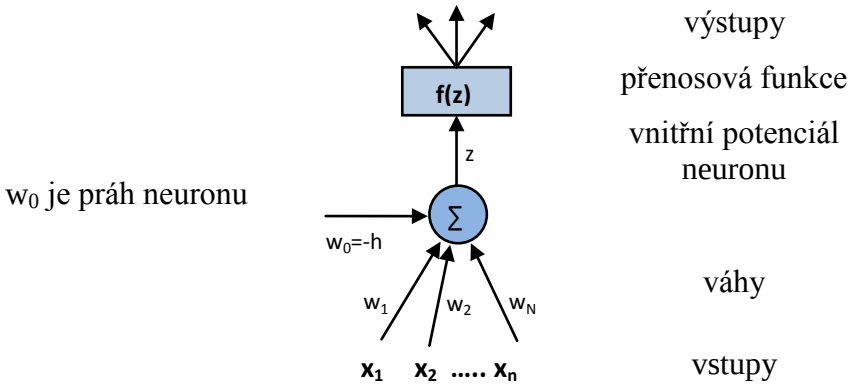
3.4.1 Neuronové sítě

Neuronová síť, jak již název napovídá, je uspořádaný systém neuronů, které jsou mezi sebou navzájem propojeny vazbami tak, aby byly schopny účelně zpracovávat informace. Neuronové sítě jsou inspirovány biologickými systémy. Je pro ně charakteristické paralelní zpracování informace. Znalost neuronové sítě je uložena především pomocí síly vazeb mezi jednotlivými neurony. Neuronová síť je charakterizována typem použitých neuronů (jejich přenosových funkcí), topologií (vzájemným uspořádáním a propojením neuronů), způsobem učení a způsobem vybavování. Jako nevýhodu neuronových sítí lze jmenovat problematickou volbu topologie sítě a nemožnost zpětné interpretace způsobu dosažení výsledku. V této kapitole a následujících podkapitolách čerpám především z prací [13], [14].

3.4.1.1 Neuron

Hodnoty přivedené na vstup neuronu jsou váhovány a následně sečteny. K tomuto součtu je ještě přičtena hodnota prahu w_0. Neuron do značné míry charakterizuje jeho přenosová funkce. Běžně se používají přenosové funkce typu skok, sigmoida, lineární a mnoho dalších. Pro výstup neuronu lze psát rovnici:

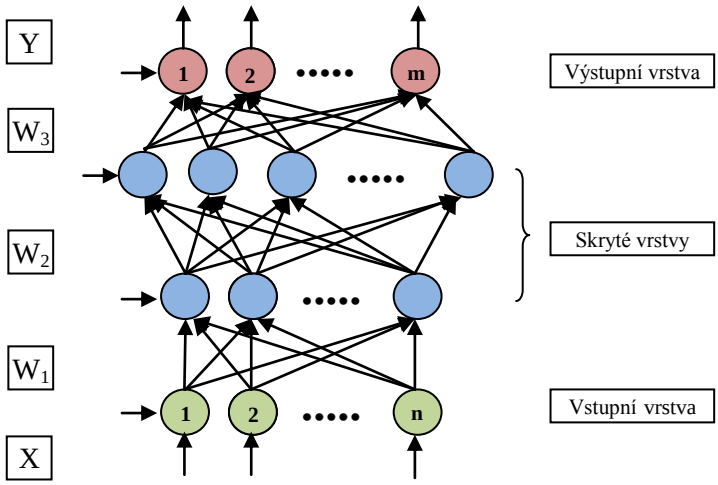
$$y = f\left(w_0 + \sum_{i=1}^{N} x_i \cdot w_i \right) \tag{3.13}$$



Obr. 3.8: Vysvětlení pojmů souvisejících s neuronem

3.4.1.2 Topologie

Topologie neuronové sítě nám říká, jak jsou jednotlivé neurony vzájemně uspořádány a propojeny. Propojení neuronů můžeme dále rozdělit na mezivrstvové a propojení mezi neurony téže vrstvy. Lze jistě jmenovat mnoho typů uspořádání. Mezi nejběžnější však patří uspořádání zobrazené na obrázku Obr. 3.9. Toto uspořádání se nazývá vícevrstvá neuronová síť. Jejím charakteristickým rysem je to, že má vstupní vrstvu, jednu nebo více vrstev skrytých a vrstvu výstupní. [13], [14]



Obr. 3.9: Vícevrstvá neuronová síť

3.4.1.3 Učení a vybavování

Neuronové sítě jako takové vyžadují trénování. Trénování je iterativní proces, kdy jsou neuronové síti předkládány jednotlivé trénovací vzory. Postupně jsou upravovány hodnoty vah, pomocí kterých je zakódovaná znalost. Velice často je pro trénování neuronové sítě používán algoritmus backpropagation. Tento algoritmus lze zjednodušeně shrnout do třech základních kroků:

- Dopředné šíření vstupního signálu tréninkového vzoru
- Zpětné šíření chyby (výpočet chyby sítě)
- Adaptace vah (minimalizace chyby sítě)

Vybavování je oproti učení procesem jednorázovým. Po předložení vzoru se očekává reakce sítě v co nejkratším čase. [13], [14]

3.4.2 Boosting

Boosting spadá do kategorie meta learningu. Využívá slabé klasifikátory, které mají úspěšnost klasifikace o něco větší než 50 %. Metoda tedy obecně spojuje slabé klasifikátory za účelem získání jednoho silného klasifikátoru. Jako slabé klasifikátory se často používají rozhodovací stromy. Boosting metoda je schopna klasifikovat pouze do dvou tříd. Proto je pro každé gesto nutné natrénovat vlastní klasifikátor. Algoritmus boostingu je znám v mnoha modifikacích. Nejznámější z nich je AdaBoost (Adaptive Boosting)

3.4.2.1 AdaBoost

Metoda se nazývá adaptivní, jelikož v $k+1$ kroku je zvětšena váha chybně klasifikovaným záznamům a oslabena váha u správně klasifikovaných záznamů.

Obecný princip metody AdaBoost je následující:

1) Máme trénovací množinu o N záznamech (x_i,y_i), $i = 0,1,\ldots,N$ a $y \in Y = \{-1,+1\}$

2) Proveď inicializaci vah $W_1(i) = \frac{1}{N}$, $i = 0,1,\ldots,N$

3) Hledej klasifikátor h_k, který vykazuje nejmenší chybu vzhledem k váhám $W_k(i)$ a počítej trénovací chybu E_k daného klasifikátoru.

4) Pokud je chyba $E_k < 0{,}5$ pokračuj, jinak ukonči učení

5) Urči váhu klasifikátoru α_k : $\alpha_k = \frac{1}{2} \cdot \ln\left(\frac{1-E_k}{E_k}\right)$

6) Přepočítej váhy dle následujícího předpisu:

$$W_{k+1}(i) = \frac{W_k(i) \cdot e^{-\alpha_k \cdot y_i \cdot h_k(x_i)}}{Z_k}$$

$$Z_k = \sum_{i=1}^{m} W_k(i) \cdot e^{-\alpha_k \cdot y_i \cdot h_k(x_i)}$$

7) Opakuj, doku platí podmínka v bodě 4.

Zde uvedený postup je pouze zkráceným vysvětlením obsáhlé problematiky. Při studiu této metody jsem čerpal zejména ze zdrojů [10], [23].

3.5 Metody pro klasifikaci dynamických gest

Přestože se práce primárně zabývá klasifikací gest statických, považuji za vhodné uvést, alespoň přehledově, několik metod sloužících pro klasifikaci gest dynamických. Tento stručný nástin může být použit jako základ pro další hlubší studium klasifikace dynamických gest.

3.5.1 Skryté markovovy modely

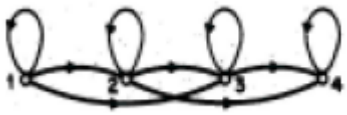
Skryté markovovy modely lze s výhodou použít pro klasifikaci dynamických gest, kde oproti statickým gestům vyvstávají další problémy. Jedním z nich je to, že stejné gesto zpravidla není prováděno vždy stejně dlouho.

Skryté markovovy modely (Hidden markov model HMM) jsou ve své podstatě konečné stavové automaty, které jsou charakterizovány jako:

$$\lambda = (N, M, A, B, \pi) \tag{3.14}$$

Kde N je vektor skrytých stavů, M je vektor pozorovatelných stavů, A je matice pravděpodobnosti přechodu mezi skrytými stavy, B je matice pravděpodobnosti přechodu k pozorovatelným stavům a π je vektor počátečních pravděpodobností stavů. U skrytých markovových modelů je třeba ošetřit případ, kdy se v sekvenci vektorů příznaků vyskytne takový vektor, pro který není ze stavu definován přechod. Takové případy nastávají například vlivem šumu nebo jiné chyby. Výše zmíněný problém je u skrytých markovových modelů, řešen pomocí klasifikátoru, který ohodnotí pravděpodobnost, že vektor příznaků byl vyslán stavem. Chybové vektory příznaků jsou pak ohodnoceny nízkou pravděpodobností. Markovovy modely lze rozdělit na diskrétní a spojité. Vstupem do spojitých markovových modelů jsou vektory příznaků. Spojité markovovy modely bývají pro klasifikaci gest využívány častěji. U většiny markovových modelů je definován právě jeden počáteční a koncový stav. Markovovy modely se používají v několika topologiích. Pro klasifikaci gest se často požívá takzvaný levopravý model jinak nazvaný dopředný model. U tohoto typu je povolen postup pouze od počátečního stavu ke koncovému stavu. Není povoleno navracení, to

znamená, že pravděpodobnost přechodu ze současného stavu do stavu minulého je rovna nule. Na obrázku Obr. 3.10 je ukázka levo-pravého markovova modelu.



Obr. 3.10: Ukázka levo-pravého markovova modelu

Převzato z [23]

Při práci s markovovymi modely se setkáváme s několika úkoly, které musíme řešit. Prvním z nich může být určení pravděpodobnosti pro vyslání sekvence *O* pozorování při známých parametrech modelu $\lambda=(\pi,A,B)$, druhým úkolem může být nalezení sekvence stavů, jimiž projde markovův model, je li dána sekvence *O*. Třetím častým úkolem může být nalezení parametrů modelu $\lambda=(\pi,A,B)$,takových aby byla pravděpodobnost přijmutí sekvence *O* maximální. První úkol můžeme řešit prostřednictvím výpočtu dopředných, nebo zpětných pravděpodobností, druhý úkol je řešen Vitterbiho algoritmem, který vybírá maxima pro všechny přechody. Nejsložitějším problémem je trénování modelu. Pro trénování modelu se používá takzvaného Baum Velchova algoritmu, který pro natrénování modelu využívá v podstatě směs gaussova rozložení a pro odhad jeho parametrů metodu Expectation Maximization. V bodech lze natrénování shrnout asi takto:

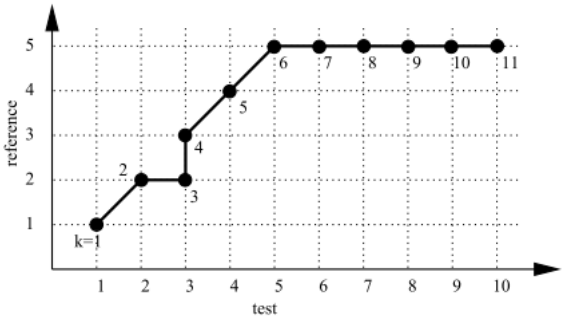
1) Hrubý odhad parametrů
2) Určení pravděpodobnosti $P_{j,}$ že se v daný čas nacházíme ve stavu j
4) Odhad nových parametrů
5) Opakování bodu dva a tři

Tato problematika je blíže popsána v dokumentu [6], z kterého také vycházím.

3.5.2 Dynamické borcení času

Jedná se o techniku, která je rovněž vhodná pro klasifikaci dynamických gest. Metoda vychází z porovnávání vzdálenosti dvou sekvencí. Její výhodou je, že při porovnávání dvou sekvencí (gest) není kladen důraz na jejich stejnou délku.

V této metodě je zavedena obecná časová proměnná *k* a dvě sekvence, které se mezi sebou porovnávají – *r(k)* je referenční sekvence a *t(k)* je testovací sekvence. Pomocí těchto sekvencí je vyjádřena cesta *C*, která má délku *K*. Grafické znázornění situace je na Obr. 3.11.



Obr. 3.11: Cesta pro srovnání dvou sekvencí

Převzato z [6]

Pro určitou cestu délky K_c s průběhem funkcí $r_c(k)$ a $t_c(k)$ lze pak vypočítat vzdálenost dvou sekvencí S_1 a S_2 pomocí vztahu:

$$D_C\left(\overrightarrow{S_1},\overrightarrow{S_2}\right)=\frac{\sum_{k=1}^{Kc} d\left[\left(\overrightarrow{S_1}t_C(k)\right),\left(\overrightarrow{S_2}r_c(k)\right)\right]\cdot W_C(k)}{N_C} \tag{3.15}$$

Kde $d[S_1,S_2]$ značí vzdálenost dvou vektorů, N_c je faktor pro normalizaci a $W_c(k)$ je váha k-tého vektoru. Vzdálenost dvou sekvencí je pak dána minimální vzdáleností přes všechny možné cesty:

$$D\left(\overrightarrow{S_1},\overrightarrow{S_2}\right)=\min_{\{C\}} D_C\left(\overrightarrow{S_1},\overrightarrow{S_2}\right) \tag{3.16}$$

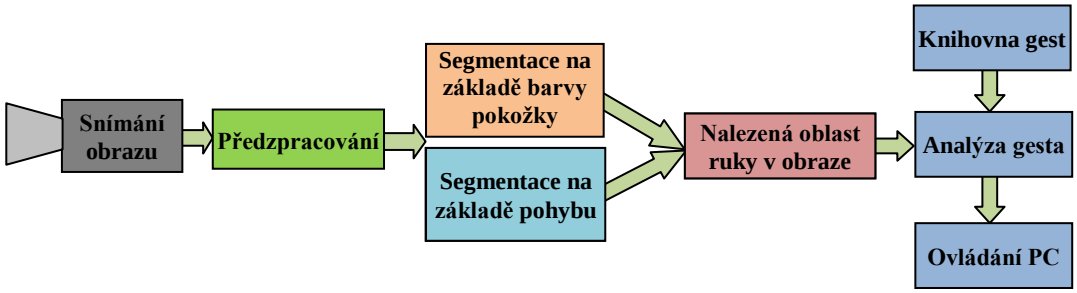
V práci [6] jsou dále uvedeny úkoly které je nutné při použití dynamického borcení času řašit. Je třeba provést omezení $r(k)$ a $t(k)$ tak, aby se cesta nevracela zpět. Dále je zapotřebí definovat váhovací funkci $W_c(k)$ a definovat normalizační faktor N_c. Jako poslední je třeba provést návrh algoritmu pro co nejrychlejší výpočet $D\left(\overrightarrow{S_1},\overrightarrow{S_2}\right)$. Jejich řešení rovněž uvádí dokument [6].

4 NÁVRH ŘEŠENÍ

Tato kapitola a příslušné podkapitoly jsou věnovány samotnému návrhu systému. Postupně jsou popsány vývojové kroky a postupy, které byly během práce na systému vyzkoušeny. Je zde popsáno ideové schéma celého systému a jeho jednotlivé části jsou zkoumány a vysvětlovány podrobněji. Jedná se zejména o nalezení ruky na základě barvy pokožky v kombinaci s nalezením ruky na základě pohybu. Dále pak budou popsány mechanismy pro extrakci příznaků a pro samotnou klasifikaci gesta.

Samotná implementace byla prováděna pomocí vývojového prostředí Visual Studio 2008 a knihovny OpenCV 2.1 od společnosti Intel, která již obsahuje řadu algoritmů pro zpracování obrazu potažmo videa. V některých případech bylo pro ověření výsledků a pro zrychlení programovaní použito prostředí Matlab R2009b, které je programátorsky přívětivější avšak rychlost zpracování obrazu je oproti OpenCV a Visual Studiu nesrovnatelně nižší.

4.1 Blokové schéma systému



Obr. 4.1: Principiální blokové schéma systému

Blokové schéma na obrázku Obr. 4.1 znázorňuje principiální uspořádání systému a hrubý nástin sekvence operací. Důležitým prvkem, který do značné míry ovlivňuje kvalitu celého systému, je blok s názvem *Snímání obrazu.* Tento blok reprezentuje běžnou web kameru, jejíž použití nám ukládá zadání. Dalším blokem je blok *Předzpracování* reprezentující operace jako je filtrace obrazu či jeho převzorkování. Pro nalezení ruky v obraze je používána kombinace metod nalezení barvy pokožky a detekce pohybu v obraze. Tyto dvě metody jsou reprezentovány bloky *Segmentace na základě barvy pokožky* a *Segmentace na základě pohybu.* Po nalezení oblasti výskytu ruky v obraze lze již přistoupit k samotné analýze gesta. Kdy je právě vykonávané gesto porovnáno z databází gest a podle rozpoznaného gesta je vykonána akce, která je danému gestu přiřazena.

4.2 Snímání obrazu

Zadání diplomové práce nám stanovuje podmínku použití běžně dostupné webkamery. Webkamery běžně zabudované do notebooků nebo webkamery používané uživateli u různých komunikačních programů se obecně vyznačují ne příliš vysokou kvalitou obrazu. Druhým neméně závažný problém je ten, že některé typy web kamer knihovna OpenCV 2.1 vůbec nepodporuje. Jedná se například o webkameru Genius iSlim 310. Proto již v této fázi lze konstatovat, že navrhované rozhraní nemůže být zcela univerzální. Jednotlivé kamery se samozřejmě liší i v mnoha dalších parametrech a to nejenom v těch výrobcem běžně uváděných, ale také v různých možnostech nastavení. Tuto skutečnost zde uvádím proto, že nastavení kamery má pro výslednou detekci pokožky v obraze zásadní vliv. Další výsledky a skutečnosti jsou uváděny pro webkameru výrobce MSI a to konkrétně pro typ StarCam Clip.

Vybrané parametry webkamery:

- Senzor: CMOS 1,3M
- Objektiv: 1/4" skleněná čočka
- Rozlišení: 640x480
- Rychlost snímků: 30fps(640x480)
- Rozlišení statického obrazu: 640x480 (interpolované 1280x1024)
- Zaostření: 3 cm ~ nekonečno
- Viditelný úhel objektivu: 56° diagonálně
- Formát obrazu: RGB24&I420
- Rozhraní: USB1.1
- LED osvětlení

4.3 Nalezení ruky na základě barvy pokožky

Kapitola pojednává o volbě barevného prostoru, výběru modelu barvy kůže a o jednotlivých postupech pro nalezení objektů barvy kůže použitých v navrhnutém systému.

4.3.1 Výběr barevného prostoru

Jak je již zmíněno v kapitole 2.1.1 pro detekci barvy kůže je velice důležité vybrat vhodný barevný prostor. Při výběru barevného prostoru jsem vycházel z předpokladu, že je nutné volit barevný prostor s oddělenou jasovou složkou a také z prací dalších

autorů, kde jsou barevné prostory pro detekci pokožky hodnoceny. Nejčastěji se v pracích zabývajících se detekcí částí lidského těla na základě barvy pokožky objevují prostory HSV a YCbCr. Tyto prostory samozřejmě vyhovují i prvnímu požadavku – mají oddělenou jasovou složku. Proto jsem se rozhodl zabývat se primárně těmito dvěma výše zmíněnými barevnými prostory.

Pro oba prostory byly tedy provedeny testy detekce pokožky na základě barvy. Detekce byla prováděna na fotografiích obsahujících barvu pokožky. Jako model bylo při těchto testech použito rozložení s jedním gaussiánem. Pro zjištění parametrů barvy pokožky byly použity výřezy z obrázků obsahující barvu pokožky. Nejprve byla vybraná fotografie převedena do daného barevného prostoru. Poté byla vypočítána pravděpodobnost, zda daný pixel je pixelem pokožky dle vztahu (2.10) za použití středních hodnot a kovariančních matic pro příslušný barevný prostor. Při těchto testech se již potvrzovaly teoretické předpoklady o tom, že barva některých objektů, například dřevěných, ale i jiných, je velice blízká barvě lidské pokožky. Markantnější byl ovšem tento problém při použití barevného prostoru HSV. Jak znázorňuje obrázek Obr. 4.2. Kde lze vidět originální obraz – a), obraz s vyhledanými oblastmi pokožky (stupně šedi zobrazují pravděpodobnost výskytu pokožky), a naprahovaný obraz – c), který je z obrazu b) vytvořen prostým prahováním s prahem hodnoty 0.2.

a) **b)** **c)**

Obr. 4.2: Vyhledání pokožky v barevném prostoru HSV

a) **b)** **c)**

Obr. 4.3: Vyhledání pokožky v barevném prostoru YCbCr

Oproti tomu při vyhledávání pokožky v barevném prostoru YCbCr nedocházelo k tak výraznému postižení oblastí „podobných“ barvě pokožky. Všimněme si například oblasti dveří, která je v prostoru HSV označena celá avšak v prostoru YCbCr nikoli.

Obr. 4.3 znázorňuje originální obraz - a), obraz s vyhledanými oblastmi pokožky – b), a naprahovaný obraz s prahem 0.2 – c).

Na základě těchto skutečností, které byly patrny ve větší či menší míře u převážné části vyhodnocovaných obrazů, jsem se rozhodl nadále pracovat s barevným prostorem YCbCr.

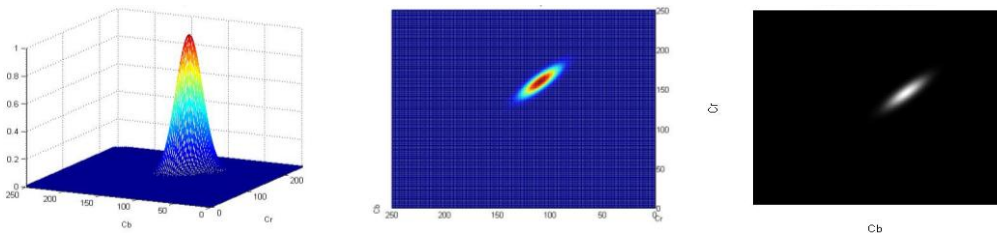
4.3.2 Model barvy kůže

Vzorek barvy pokožky je získáván přímo od uživatele aplikace. Uživatel je po spuštění aplikace vyzván, aby umístil svou ruku do prostoru vymezeného červeným obdélníkem. Po stisku příslušného tlačítka je oblast výřezu procházena prvek, po prvku a počítá se suma hodnot v jednotlivých barevných složkách. Výsledná suma je dělena celkovým počtem vzorků kůže respektive bodů uvnitř výřezu *n*.

Následným krokem je výpočet kovarianční matice dle vzorce (2.12). Pro lepší pochopení vzorce (2.12) je záhodno zapsat střední hodnoty vypočítané v předchozím kroku jako vektor: $\mu_s = \begin{bmatrix} \mu_{sCb} \\ \mu_{sCr} \end{bmatrix}$.

Výřez je opět procházen bod po bodu a je vypočítávána suma přes všechny body koláže, kdy je vždy od vektoru hodnot $c_j = \begin{bmatrix} C_{bj} \\ C_{rj} \end{bmatrix}$ na aktuální pozici *j* odečten vektor středních hodnot μ_s. Takto získaný vektor je vynásoben stejným vektorem, ovšem transponovaným. Vzniká tak matice 2 x 2, která je poté vydělena počtem prvků *n-1*. Tímto postupem dostáváme kovarianční matici $\sum_s$. Tuto pak dosadíme do vzorce (2.10) a získáme tak výslednou pravděpodobnost, zda daný pixel je pixelem pokožky.

Na obrázku Obr. 4.4 a) je znázorněn model s jedním gaussiánem vymezující barvu pokožky. Obr. 4.4 b) znázorňuje tento model při pohledu z vrchu a konečně obrázek Obr. 4.4 c) ukazuje stejný pohled pouze v odstínech šedi.



a) b) c)

Obr. 4.4: a) Model s jedním gaussiánem 3D b) Model s jedním gaussiánem pohled z vrchu c) Model s jedním gaussiánem pohled z vrchu – odstíny šedi

Výsledek detekce za použití modelu s jedním gaussiánem jsou na obrázku Obr. 4.6. Obrázek je získán pouze detekcí barvy pokožky bez jakýchkoli úprav v podobě filtrací či morfologických operací.

4.3.2.1 Další testované přístupy pro definici rozsahu barvy pokožky

Při výběru modelu barvy kůže byly odzkoušeny i další varianty detekce pokožky. A to hlavně metody s explicitně definovanými rozsahy. Tyto metody byly zkoušeny hlavně s výhledem zrychlení činnosti celé aplikace.

Konkrétně tedy byly zkoušeny přístupy, kdy je oblast pokožky definována čtvercovým respektive obdélníkovým výřezem. Dále pak bylo odzkoušeno kruhové vymezení oblasti barvy pokožky. Tyto dvě metody opět využívaly vzorek barvy pokožky získaný od uživatele umístěním ruky do vymezené oblasti. Z tohoto vzorku pokožky byly vždy nalezeny střední hodnoty v jednotlivých barevných složkách C_b a C_r, které udávali střed dané oblasti. Poté byla experimentálně nalezena velikost oblasti vymezující barvu pokožky. Tyto dva případy jsou znázorněny na obrázku Obr. 2.2.

Dalším zkoušeným přístupem bylo vytvoření modelu s jedním gaussiánem, ze sesbíraných vzorků pokožky (intenet, vlastní fotografie). Při výběru byl kladen důraz na to, aby byly vzorky pokožky pořízeny za různého osvětlení. Je nutné ovšem poznamenat, že vzorky pokožky byly sesbírány pouze z fotografií, na kterých se nacházeli pouze lidé bílé barvy pleti. Příklad vzorků pokožky je znázorněn na obrázku Obr. 4.5.

Obr. 4.5: Výřez koláže určené pro zjištění rozsahu barev pokožky

I přes snahu vybrat vzorky pokožky pořízené pod různým osvětlením, bylo u tohoto přístupu nutné před zahájením vlastní detekce měnit jednotlivé parametry kamery. Nastavení parametrů bylo nutné provádět pomocí knihovny VideoInput. Jedná se o volně dostupnou knihovnu pro sejmutí obrazu z kamery a pro nastavení jejich různých parametrů. Zde vyvstává otázka použití funkcí implementovaných přímo v OpenCV 2.1 `cvSetCaptureProperty()` pro nastavení parametrů kamery nebo `cvGetCaptureProperty()` pro zjištění aktuálně nastavených parametrů kamery. Tyto funkce nebyly použity, jelikož v OpenCV 2.1 jsou sice implementovány, ale pro nastavení parametrů kamery nejsou funkční.

Tab. 4.1: Nastavení kamery pro různá osvětlení

název	rozsah	default	přirozené	bílé	žluté
Brightness	-64 až 127	15	15	15	15
Contrast	-0 až 127	42	**40**	**10**	**10**
Gamma	-72 až 255	102	102	102	102
Hue	-40 až 40	12	**-40**	**-19**	**-19**
Saturation	-0 až 255	67	**255**	**110**	**27**
Sharpnes	-0 až 63	45	45	45	45
Gain	-0 až 100	5	**0**	5	5
Backlight comp.	-0 až 2	0	0	0	0

Tabulka Tab. 4.1 ukazuje přehled parametrů, které lze nastavit u kamery, jejich rozsahy a také defaultně nastavenou hodnotu. Parametry, které byly měněny v závislosti na osvětlení, jsou v tabulce zvýrazněny tučně. Jednotlivé hodnoty byly zjištěny experimentálně. Za přirozené osvětlení považujme denní osvětlení, za bílé považujme scénu osvětlenou například zářivkou a za žluté scénu osvětlenou běžnou žárovkou.

Kvůli již zmiňované nutnosti nastavení typu osvětlení byla tato metoda zavrhnuta. Vytvoření gaussova modelu ze vzorku barvy pokožky uživatele, který aplikaci používá, se ukázalo jako nejlepší řešení. Výsledná kvalita detekce byla upřednostněna před rychlostí, která je u explicitně definovaných modelů značně vyšší.

a) b)

Obr. 4.6: Detekce pokožky při použití modelu s gaussovým rozložením

a) Originální obraz b) Obraz s detekovanými oblastmi pokožky

a) b)

Obr. 4.7: Detekce pokožky při použití explicitně definovaných rozsahů - čtverec

b) Originální obraz b) Obraz s detekovanými oblastmi pokožky

a) b)

Obr. 4.8: Detekce pokožky při použití explicitně definovaných rozsahů - kruh

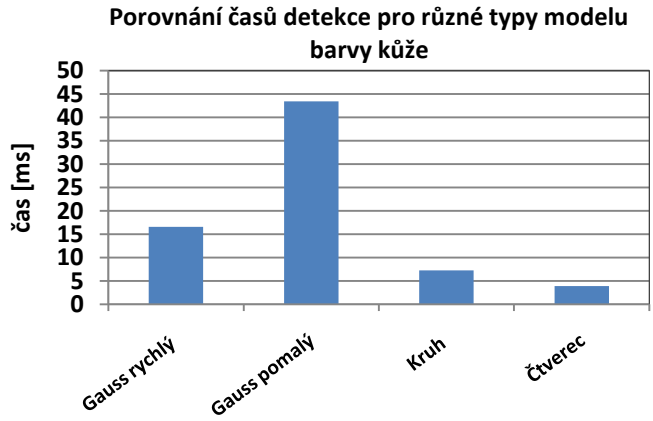
a) Originální obraz b) Obraz s detekovanými oblastmi pokožky

Na obrázcích Obr. 4.6 až Obr. 4.8 jsou znázorněny výsledky segmentace při zkoumání jednotlivých metod. Při použití explicitně definovaných modelů barvy pokožky se projevuje teoretický předpoklad, že geometrické útvary čtverce respektive kruhu ohraničují oblast pokožky ne zcela přesně. Vzniká tak chyba zejména v okrajových částech geometrického útvaru (například rohy čtverce), která značně negativně ovlivňuje výsledky segmentace. Tato chyba se například projevuje detekcí barvy pokožky v oblastech, kde se nenachází, jako je například stěna na ukázkových obrázcích.

Porovnání časů detekce jednotlivými metodami je uvedeno v tabulce Tab. 4.2.

Tab. 4.2: Časy detekce pro jednotlivé zkoušené modely

Typ modelu	Čas detekce [ms]
Gaussián rychlý	16,58
Gaussián pomalý	43,42
Kruh	7,25
Obdélník	3,92



Obr. 4.9: Graf porovnání časů detekce barvy pokožky

Časy udávané v tabulce byly dosaženy u obrazu velikosti 320 x 240 pixelů a na notebooku Acer Aspire 3690 s procesorem Intel Celeron M 1,6 GHz, operační paměť 1,25 GB DDR 2, integrovanou grafickou kartou. Na notebooku je nainstalován operační systém Windows 7 Proffesional. Všechny časy dále uváděné v práci byly dosaženy na výše zmiňované stanici.

V tabulce si lze všimnout, že byly testovány dva typy gaussova modelu. V tabulce jsou označeny jako *Gassián rychlý* a *Gaussián pomalý*. Lépe řečeno jedná se o jeden a ten samý gaussův model reprezentace barvy pokožky ovšem při detekci barvy pokožky jsou využívány odlišné přístupy k jednotlivým pixelům obrazu, což vede k poměrně markantním rozdílům času detekce. Gaussián rychlý používá takzvaný přímý přístup k pixelům obrazu, kdy se pro přístup ke konkrétnímu pixelu na pozici *i, j* složky *Cb* obrazu v barevném prostoru YCbCr používá následující syntaxe:

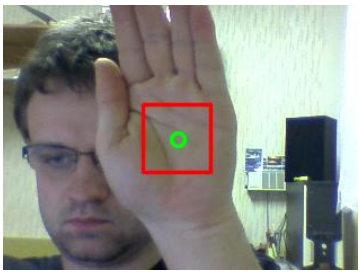
```
Cb = data[i * obraz->widthStep + j*obraz->nChannels + 1]
```

Oproti tomu Gaussián pomalý používá pro přístup k pixelům takzvaný nepřímý přístup. Při použití tohoto přístupu byl čas detekce pokožky téměř třikrát delší než při použití přístupu přímého. Pro nepřímý přístup k prvkům pixelů se používají funkce `cvSet2D(obraz,i,j,d)` a `cvGet2D(obraz,i,j)`. Tyto funkce slouží pro získání respektive nastavení vektoru hodnot představujících jednotlivé barevné složky obrazu na pozici *i, j*.

4.3.3 Algoritmus detekce barvy pokožky

Po inicializaci kamery a nastavení příslušných parametrů (rozlišení obrazu 320 x 240) již dochází k samotné detekci ruky v obraze.

Po spuštění aplikace uživatel umístí ruku do vyznačené oblasti. Tuto situaci ukazuje obrázek Obr. 4.10.



Obr. 4.10: Získání vzorku barvy pokožky

Po stisku příslušného tlačítka (podrobný popis ovládání aplikace je uveden v kapitole 4.6.5.2) je právě aktuální obraz pomocí funkce `cvCvtColor()` převeden do formátu YCbCr, u snímku je provedeno odstranění šumu pomocí gaussova filtru respektive funkce `cvSmooth()`. Následně je volána funkce `void gauss(IplImage* Ycbcr)` jejímž vstupem je převedený a vyfiltrovaný obraz. Tato funkce prochází vyznačenou oblast a počítá parametry gaussova rozložení barvy pokožky podle teoretického základu uvedeného v kapitole 2.1.2.3 respektive postupu výpočtu v kapitole 4.3.2.

Dále již dochází k cyklickému zpracování obrazu. Na začátku každého cyklu je sejmutý obraz pomocí funkce `cvFlip()` otočen kolem svislé osy. Otočení se provádí kvůli zlepšení orientace uživatele. Pokud by se otočení obrazu neprovedlo, ovládání aplikace by pro uživatele nebylo přirozené, protože pohyb ruky a vůbec její pozice v obraze by byla zrcadlově otočená, což se v průběhu práce na projektu ukázalo jako silně matoucí a nepřirozené. Následně je otočený obraz pomocí funkce `cvCvtColor()` převeden do vybraného barevného prostoru – YCbCr. Pro zlepšení výsledků segmentace se provádí odstranění šumu pomocí gaussova filtru respektive funkce `cvSmooth()`. Tyto kroky vedou k získání obrazu, ve kterém se budou vyhledávat oblasti barvy pokožky. K tomuto účelu byla napsána funkce `void VyhledaniGaussBarva(IplImage *obraz,CvRect rect)`. Jejím vstupem je obraz získaný výše popsanými úpravami – (převedený do YCbCr, převrácený a s odstraněným šumem). Tato funkce prochází aktuální snímek a pro každý jeho bod je dle vztahu (2.10) počítána pravděpodobnost, zda daný pixel lze považovat za pixel pokožky. Za pixel pokožky je prohlášen každý pixel, jehož pravděpodobnost je větší než 0.6. Provádí se tak v podstatě prahování šedotónového pravděpodobnostního obrazu s prahem 0.6.

Jak je patrné z obrázku Obr. 4.6 segmentace ruky na základě barvy pokožky je bez provedení nezbytných úprav nedostatečná. Na binární obraz jsou tedy aplikovány morfologické operace eroze, dilatace a uzavření. Jsou aplikovány přesně v tomto pořadí. Jako strukturální element tyto morfologické operace používají elipsu o velikosti 3 řádky x 3 sloupce. Eroze se provádí jedenkrát, dilatace a uzavření je provedeno dvakrát.

a) **b)** **c)**

Obr. 4.11: Vylepšení segmentace pomocí morfologických operací

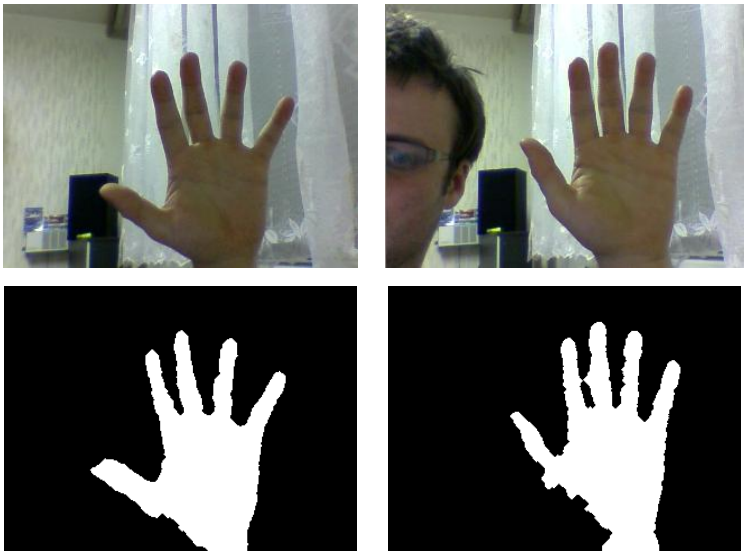
Obrázek Obr. 4.11 ukazuje výsledky segmentace na základě barvy kůže a) je původní obraz, b) je obraz bez jakýchkoli úprav a c) je binární obraz po provedení morfologických operací. Je zřejmé, že se kvalita segmentace znatelně zlepšila. Ovšem situace ještě není ideální. Proto byla implementována funkce `void odstran(void)`. Tato funkce primárně využívá knihovny *cvbobslib* a její funkce pro nalezení spojitých oblastí v obraze o definované velikosti `CBlobResult()`. Tato funkce nalezne všechny spojité oblasti v obraze. V dalším kroku jsou odstraněny spojité oblasti, které nesplňují

podmínku minimální velikosti. Odstraněny jsou všechny spojité oblasti, které jsou menší než 3000 pixelů.

a) **b)** **c)**

Obr. 4.12: Vylepšení segmentace odstraněním malých objektů

Obrázek Obr. 4.12 a) ukazuje původní obraz, b) ukazuje binární obraz masky kůže vylepšený morfologickými operacemi a c) binární masku po odstranění nevýznamných objektů. Na uvedených obrázcích si lze všimnout skutečnosti, že v obraze jsou ve většině případů detekovány dvě významné oblasti pokožky - ruka a hlava. Je tedy zřejmé, že oblast hlavy je nutné z obrazu nějakým způsobem odstranit. Nejjednodušším způsobem by bylo předpokládat, že v obraze se bude vyskytovat pouze ruka, nebo že hlava v obraze zabírá pouze malou část a je tedy odstraněna jako malá nevýznamná spojitá oblast. Tyto situace jsou znázorněny na obrázku Obr. 4.13 a) respektive b).



a) **b)**

Obr. 4.13: Omezující předpoklady pro nalezení ruky

Ovšem takto zásadní omezení nelze akceptovat. Nemluvě o případu kdy se ve snímaném obraze vyskytují například objekty dřevěné (polička na zdi detekovaná jako oblast pokožky). Dalším neméně vážným problémem je časová náročnost morfologických operací a postupu pro nalezení a odstranění malých spojitých oblastí. Je

tedy nutné přesněji vymezit oblast výskytu ruky. Tímto se zmenší oblast, ve které bude vyhledávána barva pokožky, zmenší se i oblast na které je nutno vykonávat morfologické operace a další úpravy.

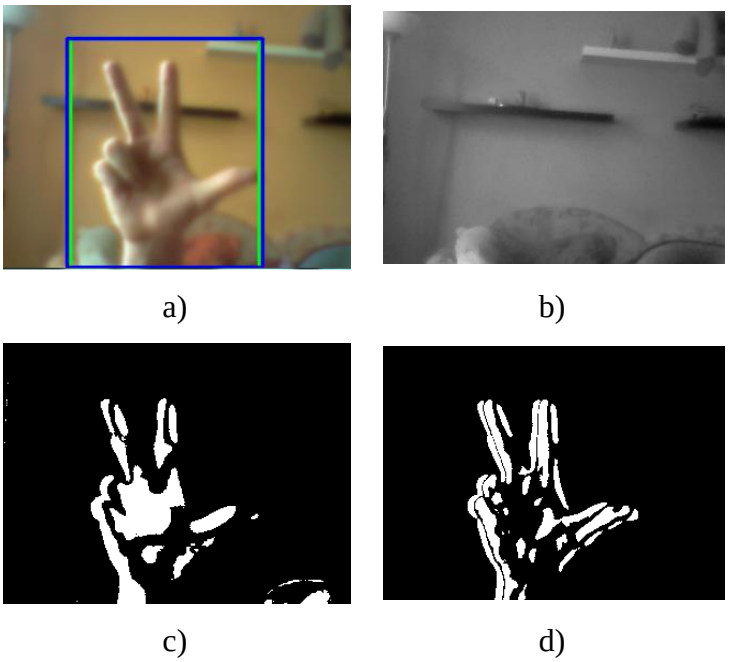
4.4 Nalezení ruky na základě pohybu

Metoda detekce barvy kůže byla v projektu doplněna detekcí pohybu v obraze.

Jelikož se v diplomové práci zabývám klasifikací gest statických, jevila se zprvu jako výhodnější metoda využívající estimaci modelu prostředí. Zde ovšem narážíme na problém tvorby respektive aktualizace modelu prostředí.

4.4.1 Algoritmus estimace modelu prostředí

Každý sejmutý snímek je nejprve převeden na šedotónový pomocí funkce `cvCvtColor()`. Následně je z prvních 30 snímků vytvořen model prostředí. Model je vytvořen sečtením všech snímků a následným vydělením počtem snímků, ze kterých se model vytváří. Dalším krokem je výpočet absolutního rozdílu vytvořeného modelu a aktuálního snímku a také výpočet absolutního rozdílu dvou po sobě následujících snímků pomocí funkce `cvAbsDiff()`. Z těchto rozdílů jsou prahováním vytvořeny binární snímky. Problém volby vhodného prahu je zde řešen funkcí `cvThreshold()`, kde lze nastavit automatickou volbu prahu pomocí metody Otsu. Výše uvedený postup lze vyjádřit pomocí vzorců (2.16) a (2.18).



Obr. 4.14: Detekce ruky v obraze na základě pohybu

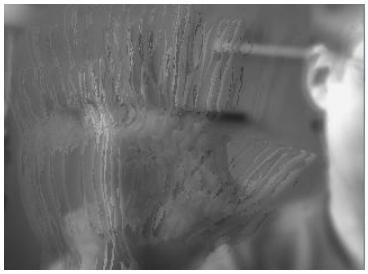
Na obrázku Obr. 4.14 a) lze vidět originální obraz, b) je příklad modelu prostředí, c) je výsledek získaný pomocí estimace modelu prostředí a konečně d) je výsledek získaný rozdílovou metodou. Na originálním obraze je modrým obdélníkem vyznačena oblast ruky získaná rozdílovou metodou – předpokládá se, že ruka je v pohybu, zeleným obdélníkem je vyznačena oblast ruky získaná estivací modelu prostředí.

Aktualizace modelu je prováděna za pomocí sady masek, které jsou získány dle rovnic (2.19), (2.20), (2.21).

MBF MFF MFB

Obr. 4.15: Příklad jednotlivých používaných masek

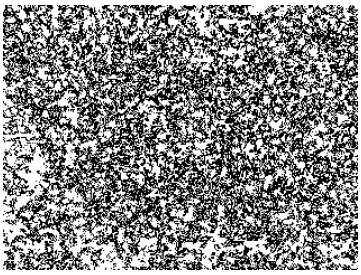
Části modelu jsou aktualizovány podle předpisu uvedeného v kapitole 2.2.2. Koeficienty $\alpha_1 - \alpha_4$ jsou nastaveny následovně α_1=0.9, α_2=0.7, α_3=0.2, α_4=0.9. Počáteční nastavení bylo provedeno dle dokumentu [34]. Je samozřejmé, že model prostředí nemůžeme aktualizovat stále, jelikož budeme vyhodnocovat statická gesta. Ruka by se tedy během poměrně krátké chvíle stala součástí modelu a nebyla by v obraze správně detekována. Proto k aktualizaci modelu dochází pouze tehdy, když se ruka nebo jiný objekt v obraze pohybuje. Tuto informaci snadno získáme z rozdílového snímku. Model je tedy aktualizován, pouze pohybuje-li se dostatečně významný objekt v obraze. Pro dosažení lepších výsledků je navíc model aktualizován pouze jednou za 5 snímků. Ukázku aktualizace modelu lze vidět na obrázku Obr. 4.16.



Obr. 4.16: Aktualizace modelu prostředí

U zmiňované metody Otsu pro automatické stanovení prahu se vyskytuje problém, pokud se v obraze nenalézá dostatečně velký objekt. Na obrázku Obr. 4.17 je znázorněn rozdílový naprahovaný snímek v situaci, kdy se ruka nehýbe, nebo v obraze vůbec není. Pro zvýšení kvality výsledné segmentace je prováděna morfologická operace dilatace, která nám pomáhá i v této situaci. Pokud na takovémto obrazu provedeme operaci dilatace, dostaneme celý bílý obraz. Proto je v obraze kontrolován počet bílých pixelů.

Pokud jejich množství překročí stanovenou mez, je obraz vynulován a neprovádí se detekce.



Obr. 4.17: Prahovaný rozdílový snímek bez významného objektu

Jak již bylo zmíněno, pro zlepšení výsledků segmentace se provádí morfologická operace dilatace. Jako strukturální element je použita elipsa o velikosti 5 řádků x 5 sloupců. Dále pak jsou z obrazu odstraněny malé objekty. Výsledky segmentace na základě pohybu jsou uvedeny na obrázku Obr. 4.18 a) původní obraz s vyznačenou oblastí ruky b) binární obraz po provedení dilatace, c) binární obraz po odstranění malých objektů

a) b) c)

Obr. 4.18: Výsledná segmentace na základě pohybu

Při pokusech s detekcí pohybu metodou upravené estimace modelu prostředí a výše popsaného algoritmu se ukázalo, že daná metoda je vhodná spíše pro malé a méně členité objekty, které se nepohybují příliš rychle. Její výpočetní náročnost nebyla úměrná kvalitě segmentace, proto tato metoda v konečné aplikaci není použita.

4.4.2 Upravený algoritmus rozdílových snímků

Základem metody je prostá detekce pohybu zjištěného z rozdílu dvou po sobě jdoucích snímků. Nejprve je tedy vstupní obraz převeden na šedotónový pomocí funkce `cvCvtColor()`. Následně je proveden výpočet absolutního rozdílu dvou po sobě jdoucích snímku v čase *t* a v čase *t+1*. Takto získaný šedotónový rozdílový obraz je prahován pomocí funkce `cvThreshold()`. Volba prahu je zajišťována pomocí metody otsu. Tímto postupem získáme binární snímek, který vyjadřuje oblasti, kde právě dochází k pohybu objektu v obraze. Následně je pro ucelení oblastí pohybu aplikována morfologická

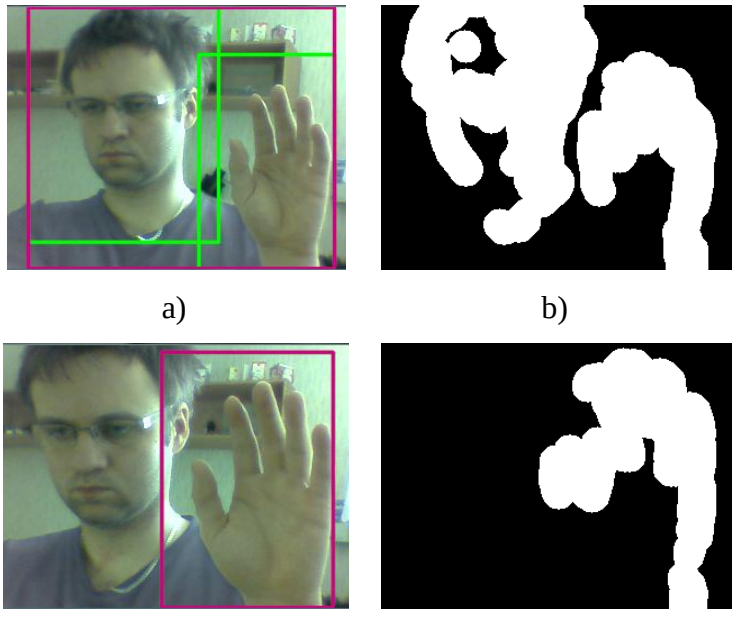
operace dilatace. Výsledky takto získané jsou znázorněny na obrázku Obr. 4.19 a) originální snímek, b) naprahovaný rozdílový snímek, c) upravený rozdílový snímek.

a) **b)** **c)**

Obr. 4.19: Metoda rozdílových snímků

Jelikož je při prahování použita metoda otsu, je opět kontrolován počet bílých pixelů. Pokud tento počet překročí určitou stanovenou mez značící, že v obraze není žádný pohyb, je obraz vynulován. Vysvětlení je podáno v předešlé kapitole.

Je zřejmé, že v obraze může docházet k pohybu více oblastí. Například pohyby hlavy, ty oproti pohybu ruky nejsou tolik výrazné avšak je s nimi nutné počítat. Algoritmus detekce pohybu byl tedy doplněn o funkci `CvRect oblast(void)`, která vyhledá v obraze všechny dílčí oblasti pohybu. Jejím výstupem je celková oblast pohybu v obraze.



a) b)

c) d)

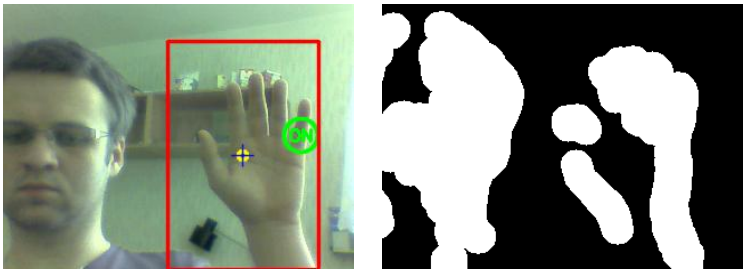
Obr. 4.20: Detekce dílčích oblastí pohybu v obraze

Na obrázku Obr. 4.20 a) je znázorněn případ, kdy se v obraze pohybuje více dílčích oblastí. Jednotlivé oblasti jsou označeny zeleně, růžová oblast je celková oblast pohybu

daná nalezením maximálních a minimálních souřadnic z jednotlivých dílčích oblastí pohybu. Obr. 4.20 b) znázorňuje upravený binární rozdílový snímek. Dále pak obrázek Obr. 4.20 c) ukazuje případ, kdy se v obraze pohybuje pouze jeden objekt. Výsledná celková oblast je tedy shodná s jednou dílčí oblastí pohybu. Obr. 4.20 d) je příslušná binární maska pohybu.

Dále pak bylo nutné omezit možnost výskytu dvou pohybujících se objektů v obraze a vyřešit případ, kdy se ruka v obraze nachází, ale nepohybuje se. K tomuto účelu byla implementována funkce `CvRect podm(void)`. Funkce obsahuje sekvenci podmínek, pomocí kterých se ověřuje, zda pohybující oblast je skutečně ruka, či nikoli. Vychází se z předpokladu, že uživatel pro odebrání vzorku barvy pokožky musí umístit ruku do vymezené oblasti obrazu. Poté se tedy předpokládá, že k pohybu nastane právě v okolí této oblasti. Dále je ve funkci testována velikost oblasti pohybu. Kdy maximální šířka pohybující se oblasti je omezena na rozmezí od 90 do 240 pixelů. Hodnoty byly určeny experimentálně. Dále se pak kontroluje rozdíl pozice středu, rozdíl výšky a rozdíl šířky oblastí ve dvou po sobě jdoucích snímcích. Maximální rozdíl pozice středů oblastí ve dvou po sobě jdoucích snímcích byla stanovena pro osu *x* i *y* na 30 pixelů, maximální rozdíl šířky byl stanoven na 60 pixelů a rozdíl výšky na 40 pixelů. Hodnoty byly opět stanoveny experimentálně. Pokud v obraze aktuálně nedochází k žádnému pohybu nebo je porušena některá z výše zmiňovaných podmínek, je jako oblast pohybu ponechána oblast ze snímku, ve kterém byl pohyb naposledy detekován. Spodní hranice oblasti pohybu je vždy nastavena na úroveň spodního okraje, protože se předpokládá, že ruka v obraze vystupuje přibližně vedle hlavy, tak jak je znázorněno například na obrázku Obr. 4.19.

Výsledná detekce ruky na základě pohybu je znázorněna na obrázku Obr. 4.**21** a), kdy je z rozdílového binárního obrazu b) jasně patrné, že navržený algoritmus je schopen vybrat pouze oblast pohybující se ruky, která je vyznačena červeným obdélníkem. Střed oblasti pohybu je vyznačen žlutým bodem.

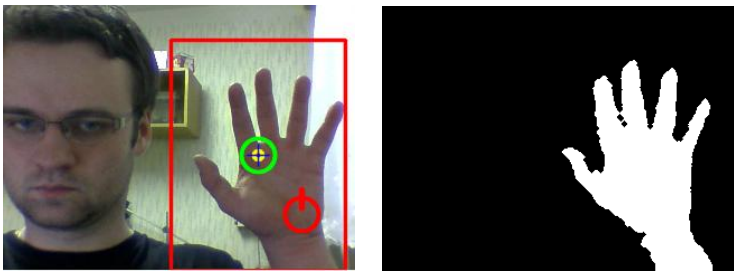


a) b)

Obr. 4.21: Finální detekce pohybu

4.5 Spojení obou algoritmů

Konečný algoritmus detekce ruky v obraze kombinuje nalezení barvy pokožky a detekci pohybu v obraze. Konkrétně je nejprve nalezena oblast pohybující se ruky, uvnitř které jsou hledány objekty barvy pokožky. Tento přístup nám zajišťuje dostatečně spolehlivé odlišení oblasti ruky od dalších objektů v obraze, které mají barvu pokožky. Podmínkou ovšem je to, aby klasifikované gesto nebylo vykonáváno například před obličejem nebo na části obrazu, kde se nachází objekt nesoucí barvu pokožky. V těchto případech by kvalitní segmentace ruky byla velice problematická. Zvolený přístup nám také umožňuje zvýšení rychlosti algoritmu, kdy je vyhledávání barvy pokožky, a další s tím spojené operace popsané v kapitole 4.3.3., prováděno pouze v omezeném prostoru. Čas detekce barvy pokožky a dalších nezbytných úprav prováděných na celém obraze je 34 ms, pokud se tyto operace provádějí pouze na vymezené části obrazu, byla detekční doba 21 ms, což je o třetinu kratší čas. Uváděné časy jsou vypočítány jako průměr času detekce ruky na sto snímcích.



a) b)

Obr. 4.22: Výsledek segmentace ruky při kombinaci segmentace na základě pohybu a detekce barvy pokožky

4.6 Klasifikace gesta

Klasifikace gesta je dalším krokem pro vytvoření aplikace ovládající počítač na základě rozpoznaného gesta. V průběhu práce byly vyzkoušeny v zásadě tři typy klasifikace statických gest. První z nich je klasifikace gesta na základě kontury, kdy je pak samotné gesto vyhodnocováno sérií podmínek, další pak klasifikace gesta metodou histogramů orientace, kdy jako klasifikátor sloužila neuronová síť. Třetím a posledním zkoušeným typem klasifikace byla opět metoda histogramů orientace, kdy byla pro klasifikaci použita metoda AdaBoost. Jednotlivé typy klasifikace gest jsou popsány v následujících kapitolách.

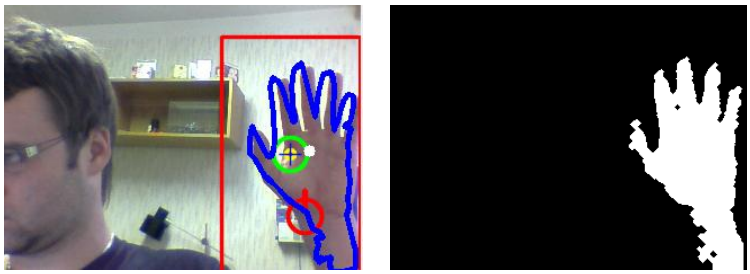
4.6.1 Klasifikace gesta založená na kontuře

Pro klasifikaci gesta na základě kontury byla implementována funkce `void kontury(void)`. Prvním krokem je nalezení kontur neboli obrysů binárního

obrazu obsahujícího detekovanou ruku. Následně je proveden algoritmus pro nalezení oblasti dlaně ruky. Dalším krokem je určení konvexní obálky a defektů v konvexní obálce. Poté je již uveden výběr příznaků pro klasifikaci gesta a jsou uvedeny jednotlivé podmínky pro určení gesta.

4.6.1.1 Nalezení kontury

Kontury byly nalezeny pomocí funkce `cvFindContours()`. Funkce vrací všechny kontury nalezené v binárním obraze. V některých případech se může stát, že v binárním obraze může být obsazeno více objektů. Proto je zkoumána délka každé dílčí kontury. Kontura s maximální délkou je prohlášena za konturu ruky. Tento postup také přispívá ke zlepšení segmentace, kdy jsou malé, povětšinou parazitní, objekty dále ignorovány. Pokud je tedy v obraze nalezena kontura, jejíž minimální délka je alespoň 300 bodů, a zároveň její délka není větší než 1500 bodů, je aplikován algoritmus takzvaných aktivních (hadích) kontur. Tento algoritmus je reprezentovaný funkcí `cvSnakeImage()`. Tato funkce vrací pole bodů, které reprezentují výslednou konturu. Výsledek nalezení ruky pomocí metody aktivních kontur je patrný na obrázku Obr. 4.23 a) vyznačená ruka nalezená metodou aktivních kontur, b) je vstupní binární obraz, jehož kontury jsou hledány.



a) b)

Obr. 4.23: Nalezení ruky metodou aktivních kontur

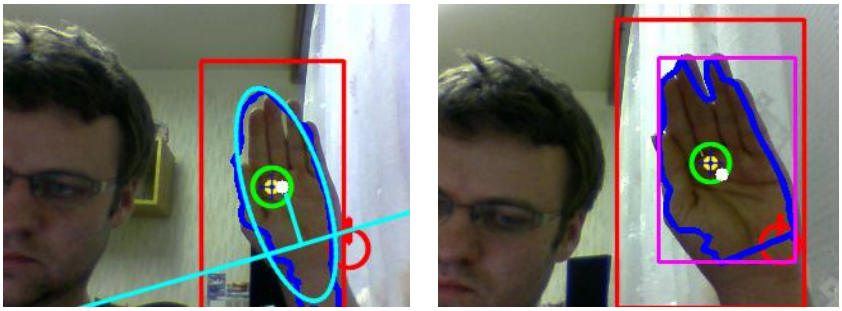
Výše popsaný postup trpí jedním zásadním nedostatkem – je označena celá část ruky, my však potřebujeme pro detekci gesta pouze dlaň a prsty. Nejjednodušší by bylo předpokládat, že uživatel bude mít při používání aplikace vždy dlouhý rukáv, který by v podstatě vymezoval hranici mezi dlaní a zbytkem ruky. Takto striktní omezení však není nejlepším řešením. Proto byl navrhnut postup, pro oddělení oblasti dlaně a zbytku ruky v případě, že uživatel nemá dlouhý rukáv.

4.6.1.2 Vymezení oblasti dlaně

Pomocí funkce `cvFitEllipse2()` je do prostoru vymezeného jednotlivými body kontury vepsána elipsa. Její parametry jsou poté využity pro vymezení oblasti dlaně. Konkrétně je využíván úhel natočení elipsy, střed elipsy a šířka vedlejší osy elipsy.

Obrázek Obr. 4.24 a) znázorňuje princip oddělení oblasti dlaně od zbytku ruky. Ze středu elipsy je pod úhlem odpovídajícím natočení elipsy vedena úsečka o délce

4/3 délky vedlejší osy elipsy. Tato hodnota byla zjištěna experimentálně. Kolmo k této úsečce je pak vedena další úsečka vymezující spodní okraj ruky. Tato úsečka protíná konturu právě ve dvou bodech.



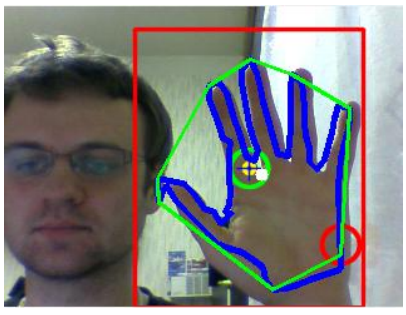
a) b)

Obr. 4.24: Nalezení oblasti dlaně

Nalezením souřadnic těchto dvou mezních bodů získáváme meze pro úpravu množiny bodů popisujících konturu. Body kontury, jejichž souřadnice *y* se nachází v oblasti pod omezující úsečkou, jsou dále děleny na dvě skupiny. Konkrétně je testována souřadnice *x* těchto bodů. Pokud souřadnice *x* těchto bodů spadá do úseku vymezeného průsečíky kontury a omezující přímky, je těmto bodů přiřazena nová souřadnice *y* a to taková, aby dané body ležely na omezující přímce. Pokud ovšem souřadnice bodu leží mimo úsek vymezený průsečíky kontury a omezující přímky, jsou tyto body odstraněny. Tímto postupem je získána nová množina bodů popisující konturu. Tato množina již popisuje pouze úsek dlaně ruky. Obrázek Obr. 4.24 b) již ukazuje výsledek s upravenou množinou bodů kontury. Dlaň je potom vyznačena růžovým obdélníkem.

4.6.1.3 Konvexní obálka a defekty v konvexní obálce

Pro nalezení konvexní obálky ohraničující sekvenci bodů definujících konturu ruky byla použita funkce `cvConvexHull2()`, Funkce vrací pole bodů definujících místa dotyku konvexní obálky a kontury ruky. Po nalezení konvexní obálky již můžeme přistoupit k nalezení samotných defektů v konvexní obálce.



Obr. 4.25: Konvexní obálka oblasti ruky

Pro nalezení defektů v konvexní obálce slouží funkce `cvConvexityDefects()`, která vrací strukturu obsahující vlastnosti jednotlivých defektů.

Každý defekt je charakterizován svým počátečním bodem, bodem udávajícím hloubku defektu a koncovým bodem. Na obrázku Obr. 4.26 a) jsou tyto označeny zelenou, červenou a žlutou barvou.

a) b) c)

Obr. 4.26: Defekty v konvexní obálce

Defekty jsou dále děleny na takzvané velké defekty a defekty malé. Dělení probíhá porovnáním hloubky defektu s velikostí hlavní osy elipsy vepsané do oblasti ruky vymezené konturou. Za velký defekt je prohlášen takový, jehož hloubka je větší než čtvrtina délky hlavní osy elipsy. Za defekt malý je prohlášen takový, jehož hloubka je menší než čtvrtina délky elipsy. Dále pak jsou ignorovány defekty, jejichž hloubka je menší než pět pixelů nebo větší než 80 pixelů. Obrázek Obr. 4.26 a) ukazuje detekci pouze velkých defektů, b) ukazuje detekci malých defektů a c) ukazuje detekci velkých i malých defektů.

4.6.1.4 Klasifikace gesta

Pomocí metody založené na kontuře je klasifikováno čtrnáct gest. Gesta a jejich názvy uvádí tabulka Tab. 4.3.

Tab. 4.3: Klasifikovaná gesta a jejich názvy

gesto	název	gesto	název
	Fist		Victory
	Stop		Three
	Finger		V-L
	Fist-L		Quoins-L
	Stop-L		Four
	Finger-L		Three-L
	Quoins		Palm

Gesto je klasifikováno na základě několika příznaků. Prvním z nich je počet velkých defektů. Podle počtu velkých defektů se gesta dělí do pěti skupin. První skupinou jsou gesta, kde je počet velkých defektů nulový. Jedná se o gesta Fist, Stop a Finger. Gesta Fist a Stop jsou mezi sebou odlišena na základě poměru hlavní a vedlejší osy elipsy vepsané gestu. Pokud je poměr *hlavní osa/vedlejší osa* větší než 1.9 gesto je klasifikováno jako Stop, pokud je poměr menší, gesto je klasifikováno jako Fist. Pokud jsou u předváděného gesta s nulovým počtem velkých defektů detekovány jeden nebo dva malé defekty, je gesto klasifikováno jako Finger.

Pokud je detekován jeden velký defekt vybíráme mezi gesty Fist-L, Stop-L, Finger-L, Quoins a Victory. V tomto případě již používáme i další příznaky a to například úhel α mezi dvěma úsečkami označujícími defekt Obr. 4.27 a), nebo poměr dvou úseček vyznačujících defekt (poměr zeleně označené úsečky a červeně označené úsečky) Obr. 4.27 b).

a) b)

Obr. 4.27: Ukázky používaných příznaků

Gesto je klasifikováno jako Fist-L pokud úhel svíraný dvěma úsečkami je v rozsahu 80° až 120° a zároveň pokud poměr dvou úseček vyznačujících defekt je menší než 1.3. Počet malých defektů v případě gesta Fist-L je menší nebo roven dvěma. Pokud je úhel svíraný dvěma úsečkami velkého defektu v rozsahu 80° až 120° a poměr úseček je větší než 1.5, je gesto klasifikováno jako Stop-L. Počet malých defektů u tohoto gesta je roven nule. Gesto Finger-L má parametry stejné, ovšem počet malých defektů je roven dvěma. Gesto Quoins a gesto Vitory se od sebe v zásadně liší velikostí úhlu mezi úsečkami vymezujícími defekt. U gesta Quins je úhel v rozsahu 45° až 80°, u gesta Victory je rozsah 10° až 45°

Kategorii gest se dvěma velkými defekty tvoří gesta Three, V-L a Quoins-L. Mezi těmito gesty se rozlišuje podle velikosti úhlů, které svírají úsečky vymezující jednotlivé defekty. Respektive je počítáno, kolikrát se vyskytuje úhel v rozmezí 10° až 45° (označen jako *úhel30*) a úhel v rozmezí 70° až 120° (označen jako *úhel 90*). Jsou-li zjištěny dva *úhly30* je gesto označeno jako Three, je – li zjištěn jeden *úhel30* a jeden *úhel90* je detekováno gesto V-L, jsou li zjištěny dva *úhly90* je gesto detekováno jako Quoins-L.

Do kategorie gest se třemi velkými defekty spadají gesta Four a Tree-L. Tato gesta se opět mezi sebou rozlišují podle počtu *úhlů30* a *úhlů90*. Gesto Four je charakteristické třemi *úhly30* a gesto Three-L je charakteristické dvěma *úhly30* a jedním *úhlem90*. Poslední kategorii tvoří jediné gesto a tím je gesto Palm. Toto gesto je charakterizováno čtyřmi velkými defekty.

Metoda klasifikace gesta pomocí kontury je poměrně dobře funkční. Zvolené příznaky jsou invariantní jak vůči poloze ruky v obraze tak vůči natočení ruky. Problematické ovšem je popsat jednotlivá gesta vhodnými příznaky. Dalším problémem také je, pokud se rozhodneme přidat další gesta do slovníku již klasifikovaných gest. Poté je nutné upravovat podmínky u celého slovníku gest což je značně časově náročné.

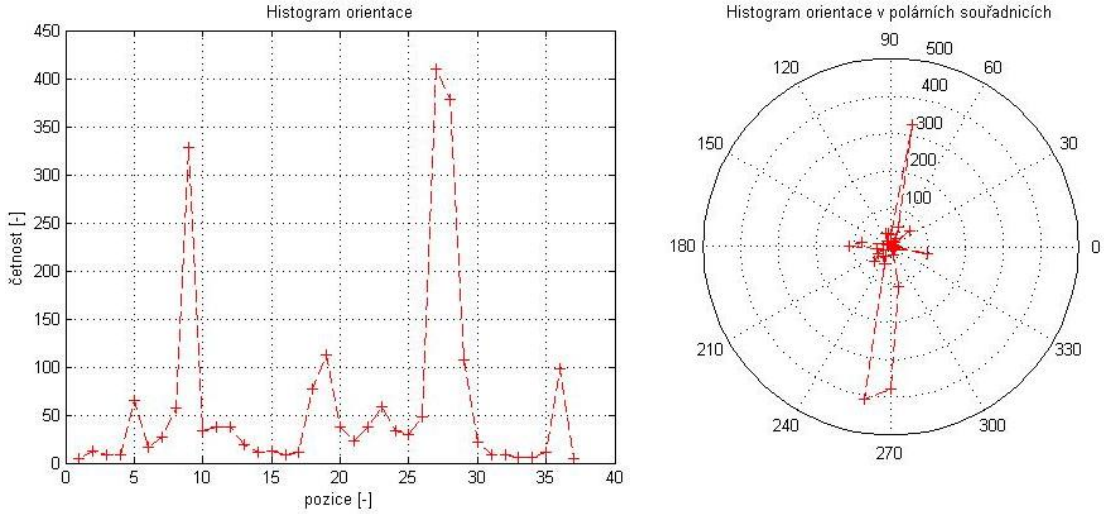
4.6.2 Metoda histogramů orientace

Další odzkoušenou metodou pro klasifikaci gesta, respektive pro získání příznaků popisujících gesto, je metoda histogramů orientace. Teoretický základ k této problematice je uveden v kapitole 3.2. Pro výpočet histogramů orientace byla implementována funkce `void orientHist (IplImage *vstup)`. Vstupem funkce je analyzovaný obraz.

Vstupní obraz je nejprve nutné převézt na šedotónový. Následně je tento obraz procházen prvek po prvku a jsou počítány diference v ose *x* a *y*. Pro každý bod obrazu je vypočítán kontrast a orientace dle vztahu (3.6) a (3.7). V cyklu je také počítán průměrný kontrast. Dalším krokem je vytvoření samotného histogramu orientací. Procházíme tedy matici kontrastů *C(i, j)* a porovnáváme aktuální hodnotu na pozici *i, j* s hodnotou průměrného kontrastu vynásobenou hodnotou 2.5. Konstanta 2.5 je volena dle doporučení v práci [8] . Je – li aktuální hodnota kontrastu větší než daný práh, vyzvedneme hodnotu z matice orientací na odpovídající pozici. Orientace jsou v rozmezí 0 – 360°. Pro klasifikaci gesta slouží 36prvkový histogram orientací. Dělíme tedy hodnoty orientací po deseti stupních. Orientace na příslušné pozici je tedy dělena číslem deset a takto získaná hodnota je zaokrouhlena nahoru. Získáme tak číslo v rozsahu 0 – 36, které odpovídá pozici v histogramu orientace, na které je inkrementována hodnota.

Takto získaný vektor příznaků je dále filtrován. K tomuto účelu byla napsána funkce `void filtruj(void)`, která provádí průměrování maskou *M(x)=[1 4 6 4 1]*. Hodnota filtrovaného vektoru je získána vynásobením prvků vektoru původního, které leží pod maskou *M(x)*, prky masky. Vynásobené hodnoty jsou sečteny a vyděleny počtem prvků masky. Takto získaná hodnota je zapsána na odpovídající pozici filtrovaného vektoru.

Histogramy orientace se nejčastěji zobrazují buď v kartézské souřadné soustavě Obr. 4.28 a) nebo v polární souřadné soustavě Obr. 4.28 b).

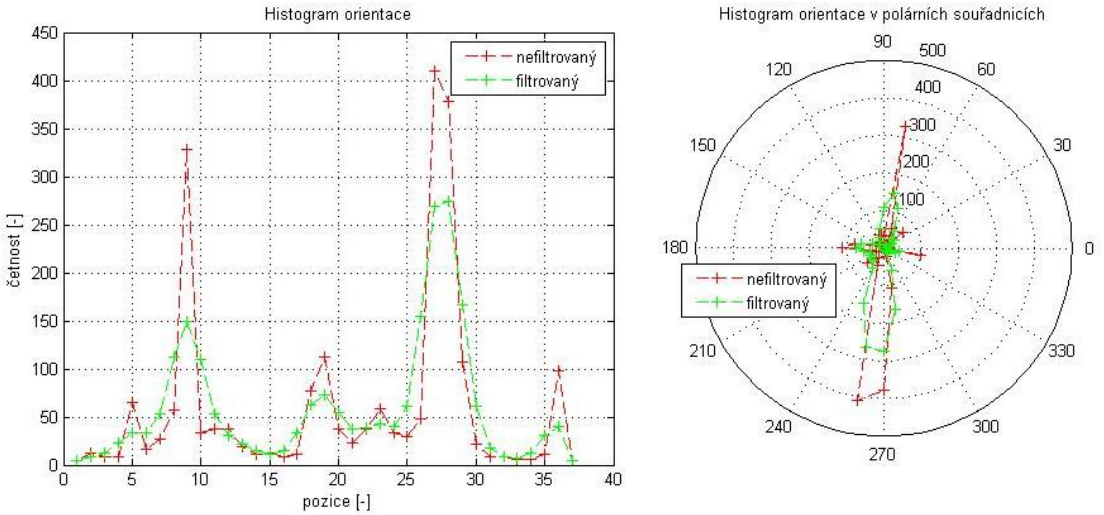


a) b)

Obr. 4.28: Úkazky nefiltrovaných histogramů orientace

a) v kartézské souřadné soustavě b) v polárním vyjádření

Filtrací histogramu orientací dosáhneme jejich celkového vyhlazení. Eliminací ostrých přechodů eliminujeme například vliv malých objektů při nedokonalé segmentaci ruky. Rozdíl mezi filtrovaným a nefiltrovaným vektorem příznaků je znázorněn na obrázku Obr. 4.29. Kde je zřetelně vidět celkové vyhlazení histogramu.



a) b)

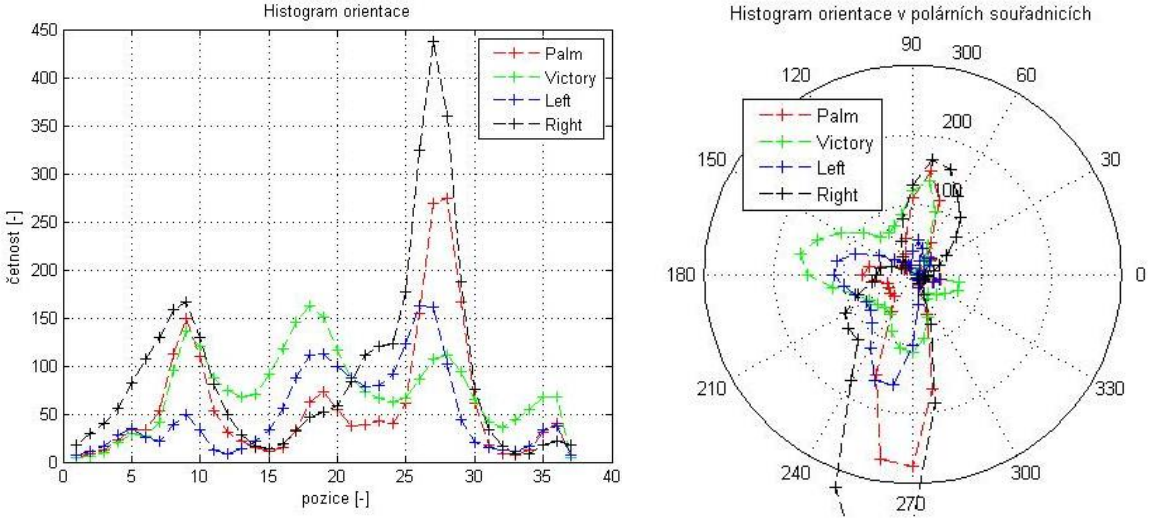
Obr. 4.29: Rozdíl mezi filtrovaným a nefiltrovaným vektorem příznaků

a) v kartézské souřadné soustavě b) v polárním vyjádření

Problémem této metody je fakt, že vektory příznaků některých gest jsou si velice podobné. Proto je třeba vybírat gesta taková, aby jejich histogramy orientace byly dostatečně odlišné. Porovnání histogramů orientace pro čtyři různá gesta je znázorněno na obrázku Obr. 4.30. Vzhledy těchto čtyř gest a jejich názvy jsou uvedeny v tabulce Tab. 4.4.

Tab. 4.4: Příklady gest a jejich názvy

Gesto	Název
	Palm
	Left
	Right
	Victory



a) b)

Obr. 4.30: Porovnání histogramů orientace čtyř gest

a) v kartézské souřadné soustavě b) v polárním vyjádření

Na obrázku Obr. 4.30 si lze všimnout, že histogramy orientace pro tyto čtyři gesta jsou dostatečně odlišné. Pokud již pracujeme s rozsáhlejším slovníkem gest, je výběr vhodných gest o hodně náročnější.

V dokumentu [8] jsou histogramy orientací počítány z šedotónového obrazu bez jakékoli předchozí segmentace ruky. Předpokládá se, že ruka je v obraze dominantním prvkem. V mojí práci jsem volil přístup, který volil například autor práce [34], kdy je histogram orientace počítán až z šedotónového obrazu vzniklého aplikací binární masky

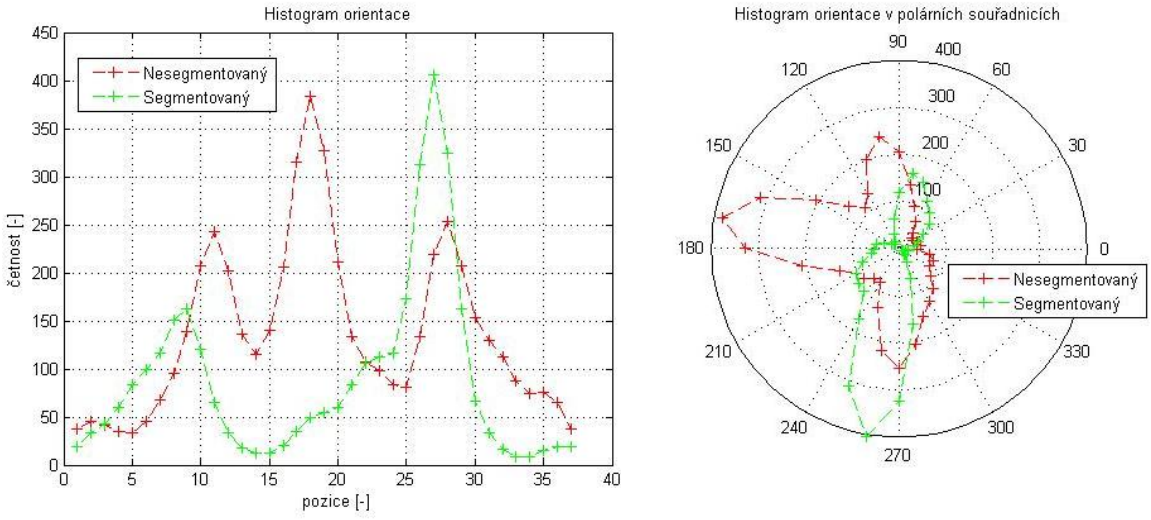
oblasti ruky. Tímto přístupem odstraníme závislost příznakového vektoru na ostatních objektech obsažených v obraze.

a) b)

Obr. 4.31: Vstupní obrazy pro výpočet histogramů orientace

a) nesegmentovaný, b) po aplikaci masky oblasti ruky

Na obrázku Obr. 4.32 je vidět rozdíl mezi příznakovým vektorem získaným z obrazu kde se kromě ruky vyskytují i jiné objekty a příznakovým vektorem získaným z plně segmentovaného obrazu. Příznakový vektor z nesegmentovaného obrazu obsahuje více vrcholů. Tyto vrcholy lze přisoudit ostatním významným objektům v obraze. Lze tedy s jistotou říci že, metoda získávání příznakového vektoru z plně segmentovaného obrazu povede k lepším výsledkům při klasifikaci.



a) b)

Obr. 4.32: Příznakové vektory stejného gesta získané z nesegmentovaného a plně segmentovaného obrazu

a) v kartézské souřadné soustavě b) v polárním vyjádření

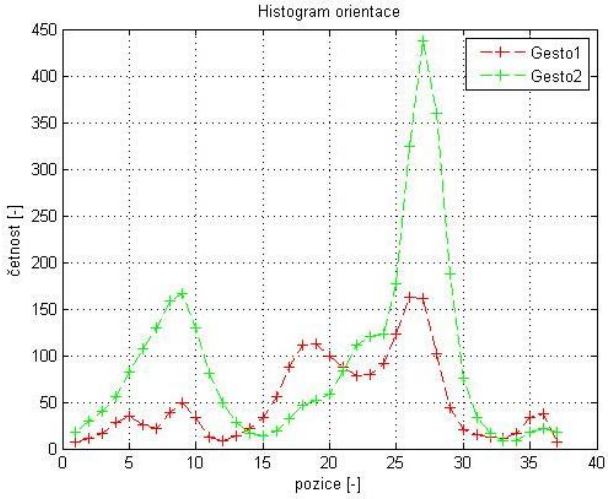
4.6.3 Klasifikace gesta podle histogramů orientace

Pro klasifikaci gest popsaných histogramem orientace byly vyzkoušeny tři metody. První z nich byla metoda měření euklidovské vzdálenosti mezi příznakovým vektorem daného gesta a sadou vektorů trénovacích. Dále byla pro klasifikaci gesta použita neuronová síť a v neposlední řadě byla gesto klasifikována pomocí sady klasifikátorů AdaBoost.

4.6.3.1 Měření eukllidovské vzdálenosti

Klasifikace gesta pomocí euklidovské vzdálenosti byla použita například v práci [8]. Tato metoda porovnává aktuální histogram orientace s histogramy orientací uloženými v souboru na pevném disku. Je zřejmé, že tato metoda vyžaduje sběr příznakových vektorů jednotlivých gest, která chceme klasifikovat. Pro sběr příznakových vektorů byla implementována funkce `void sbirejVzorky(char souborFiltr[])`. Vstupním parametrem funkce je adresa a název souboru kam mají být vzorky ukládány. Sběr vzorků probíhá tak, že uživatel vykonává gesto jednoho typu. Z každého obrazu je vypočítán histogram orientace příslušného gesta. Jednotlivé histogramy jsou na odpovídajících si pozicích sčítány. Po dosažení předem nadefinovaného počtu trénovacích cyklů daného gesta je vektor součtů jednotlivých histogramů orientace vydělen počtem trénovacích cyklů. Vzniká tak jakýsi průměrný histogram orientací pro dané gesto. Tento průměrný vektor je pak uložen do souboru na pevný disk. Proces je pak opakován pro všechna gesta, která chceme rozpoznávat.

Pro klasifikaci gesta pak byla napsána funkce `void EuklVzdal(void)`. Funkce postupně porovnává aktuální příznakový vektor s vektory uloženými v souboru a hledá nejmenší vzdálenost mezi dvěma vektory. Po prvotních pokusech byla však tato metoda zavrhnuta. Klasifikace byla správná pouze tehdy, rozhodovalo-li se mezi dvěma velmi odlišnými gesty. Pokud bylo rozhodováno mezi více gesty, metoda naprosto selhávala a její chybovost byla neúnosná. Lze tedy konstatovat, že tato metoda je v praxi nepoužitelná. O její použitelnosti by se dalo uvažovat, pokud by se jednalo o statické snímky gest, kde by byla ruka segmentována manuálně a kvalita obrazu by byla vysoká. Další podmínkou použití v praxi je to, že se snímky trénovací a testovací příliš neliší. To však zcela určitě není náš případ. Na obrázku Obr. 4.33 jsou vidět histogramy orientací dvou velmi odlišných gest, kdy metoda fungovala.



Obr. 4.33: Příklad histogramů orientace dvou značně odlišných gest

4.6.3.2 Klasifikace pomocí neuronové sítě

Dalším zkoušeným klasifikátorem gest byla neuronová síť, která byla pro klasifikaci gest ruky využita například v práci [26]. Je známo, že neuronová síť vyžaduje trénování, proto bylo nutné sesbírat trénovací data v podobě histogramů orientací jednotlivých gest. K tomuto účelu byla napsána funkce `void sbirejVzorky(char souborFiltr[])`. Tato funkce zajišťuje sběr a ukládání vzorků do souboru na pevný disk. Ke sběru vzorků se používá konzolová aplikace, kdy uživatel zadá číslo gesta, které chce trénovat. Poté je nutné, aby po celou dobu sběru vzorů jednoho gesta vykonával stejné gesto. V každém snímku je nalezena ruka a z šedotónového obrazu nalezené ruky je extrahován histogram orientace. Tento histogram je uložen do souboru v podobě řádkového vektoru. Dále je do stejného souboru za tento vektor uložen vektor výstupů. Jedná se o vektor dvaceti prvků. Na pozici trénovaného gesta je zapsána hodnota 10 a na ostatní pozice hodnota nula. Při sběru vzorků jednoho gesta je vždy uloženo 300 příznakových vektorů daného gesta. Postup popsaný výše je nutné opakovat pro všechna gesta, která chceme klasifikovat.

Při sběru vzorků pro námi vytvořenou aplikaci bylo dbáno na to, aby byla gesta vykonávána v různých světelných podmínkách, v různých vzdálenostech od kamery a také aby byla gesta vykonávána pod různým úhlem natočení avšak maximálně v rozmezí přibližně od -10°do +10° od svislé osy gesta.

Postup získávání vektorů orientací přímo z videa popsaný výše umožňuje velice rychlý sběr příznakových vektorů pro mnoho gest. Během krátké chvíle je tedy možné vytvořit poměrně rozsáhlou databázi příznakových vektorů různých gest.

Dalším krokem je tedy samotné trénování neuronové sítě. Pro trénování neuronové sítě byla napsána funkce `void trenujNN(char adresa[])`. Vstupem funkce je adresa a název souboru s příznakovými vektory. Procesu trénování neuronové sítě předchází definice parametrů samotné neuronové sítě. Pro klasifikaci gest je používána vícevrstvá neuronová síť, která má 36 vstupů. Každý jeden vstup odpovídá jednomu prvku ve

vektoru histogramu orientací. Dále pak má dvě skryté vrstvy. První z nich má 100 neuronů a druhá má 72 neuronů. Počet vrstev a počet neuronů v každé vrstvě byl volen experimentálně. Výstupní vrstva obsahuje dvacet výstupů. Je tedy možné klasifikovat až do dvaceti tříd. Jako přenosová funkce neuronu je zvolena sigmoida. Pro práci s neuronovými sítěmi je používána knihovna *ml.lib* která je součástí OpenCV. Tato knihovna umožňuje vytvoření neuronové sítě pomocí funkce `create()`.

Vstupní trénovací data jsou načtena do matice vstupů, kdy jeden řádek matice vstupů odpovídá příznakovému vektoru jednoho gesta. Dále jsou do matice výstupů načteny vektory výstupů ze souboru obsahujícího trénovací data. Vektor výstupů je složen ze samých nul, pouze na pozici odpovídající právě trénovanému gestu je hodnota 10. Proces trénování je realizován pomocí funkce `train()` dostupné v knihovně *ml.h.* Funkce `train()` má jako své parametry matici vstupů, matici výstupů, způsob trénování, maximální počet iterací trénování a požadovanou chybu výstupu sítě. Neuronová síť je trénována za pomocí algoritmu backpropagation. Požadovaná chyba sítě je nastavena na 10^{-5}, maximální počet iterací je stanoven na 10 000. Po skončení procesu trénování je neuronová síť uložena na pevný disk pomocí funkce `save()`.

Proces predikce je realizován funkcí `int predpoved(void)`. Funkce vrací číslo klasifikovaného gesta. Histogram orientace aktuálně vykonávaného gesta je předložen neuronové síti. Výstupem pak je vektor dvaceti prvků. Tento mechanismus zajišťuje funkce `predict()` z knihovny *ml.h.* Dále je ve funkci `predpoved` prohledáván vektor výstupů a je hledána maximální hodnota ve vektoru výstupů. Pozice maximální hodnoty nalezené ve vektoru výstupu odpovídá klasifikovanému gestu.

4.6.3.3 Klasifikace pomocí metody AdaBoost

Pro trénování AdaBoost klasifikátoru byla napsána funkce `void trenujBoost(char adresa[],int gesto,char adresaOut[])`. Vstupem funkce je adresa souboru s příznakovými vektory, číslo trénovaného gesta a adresa kam má být uložen natrénovaný klasifikátor. Pro trénování je použit stejný soubor příznakových vektorů jako pro trénování neuronové sítě. Je nutné pouze změnit vektor výstupů. Jak známo klasifikátor AdaBoost je schopen klasifikovat pouze do dvou tříd. Je tedy nutné natrénovat pro každé gesto samostatný klasifikátor. Trénování klasifikátoru pro dílčí gesto začíná načtením všech vektorů příznaků ze souboru. Jako pozitivní vzory jsou označeny histogramy orientace právě trénovaného gesta. Jako negativní vzory jsou označeny histogramy orientací všech ostatních gest. Pro vytvoření, trénování a klasifikaci jsou využívány funkce knihovny *ml.h.* Samotné trénování je prováděno opět pomocí funkce `train()`. Parametry funkce pak jsou vektory příznaků a jim přiřazená hodnota +10 pro pozitivní vzor a -10 pro negativní vzor. Počet slabých klasifikátorů je stanoven na 900. Jako slabý klasifikátor se využívá rozhodovací strom. Natrénovaný klasifikátor je pak uložen na pevný disk pomocí funkce `save()`. Proces je opakován pro všechna gesta.

Předpověď je realizována ve funkci `int predpovedBst(void)`, která vrací číslo klasifikovaného gesta. Aktuální příznakový vektor je tedy předložen všem

natrénovaným klasifikátorům. Předložení je realizováno pomocí funkce `predict()`, která vrací hodnotu +10 nebo -10 podle toho zda se jedná o gesto klasifikované daným klasifikátorem. Po předložení příznakového vektoru je tedy hledán klasifikátor, který vrací hodnotu +10. Číslo tohoto klasifikátoru nám udává klasifikované gesto.

4.6.3.4 Trénovaná gesta

Celkem bylo pro oba klasifikátory natrénováno patnáct gest. To znamená, že pro klasifikaci metodou AdaBoost byl natrénován odpovídající počet klasifikátorů. Natrénovaná gesta, jejich název a pořadové číslo jsou uvedeny v tabulce Tab. 4.5. Pro každé gesto bylo nasbíráno 2100 příznakových vektorů, které byly získány z obrazů pořízených za různých podmínek. Celkem tedy trénovací soubor obsahuje 31 500 příznakových vektorů.

Tab. 4.5: Natrénovaná gesta, jejich čísla a názvy

Číslo	Gesto	Název	Číslo	Gesto	Název	Číslo	Gesto	Název
1		Palm	**6**		Finger	**11**		Phone
2		Left	**7**		Four	**12**		Finger-L
3		Right	**8**		Fist	**13**		V-L
4		Victory	**9**		Stop	**14**		Little Finger-L
5		Ou	**10**		Quoins	**15**		Palm-1

4.6.4 Testy a hodnocení klasifikátorů

4.6.4.1 Test 1 – úspěšnost klasifikátorů

Testovány byly tři klasifikátory, které jsou popsány v předešlých kapitolách. Testování probíhalo za normálního denního osvětlení a za umělého osvětlení zářivkou. Každé gesto bylo vykonáno desetkrát při obou typech osvětlení. Úspěšnost klasifikace byla pro oba typy osvětlení zaznamenána do tabulek Tab. 4.6 až Tab. 4.8. První číslo v každé buňce udává počet gest při normálním osvětlení, druhé udává počet gest za umělého osvětlení.

Tab. 4.6: Klasifikace pomocí kontury ruky

Klasifikované gesto \ Předváděné gesto															
	9/10	0/1	1/2	2/2	2/1	1/0	1/0				2/3				
	1/0	**10/9**													
			9/8	3/2											
				4/6		1/2	2/2								
					8/9	2/2									
						6/6									
				1/0			**7/8**				3/1				
								10/9	1/1	4/2		2/0			
								0/1	**9/9**	0/1		3/2			
										6/7			1/1	0/1	
											5/6				
												5/8			
													9/8	2/0	
													0/1	**8/9**	

Tab. 4.7: Klasifikace pomocí histogramů orientace a neuronové sítě

Klasifikované gesto \ Předváděné gesto															
	7/8						4/2							0/2	
		7/6	1/0					1/0				1/2			
			4/8												
	0/2			**5/5**									2/2		
		2/2	3/2		**9/9**										
						8/7						3/4			
	1/0			0/4			**6/7**	1/1							
		1/2	2/0		1/1			**8/9**	3/2						
									6/8						
						0/1			1/0	**6/5**	2/3				
										4/4	**4/5**				
				2/1		2/2						**3/2**			1/0
				3/0			0/1					3/2	**4/4**	2/3	1/4
	1/0										2/2		2/3	**4/5**	5/4
	1/0									0/1	2/0		2/1	4/0	**3/2**

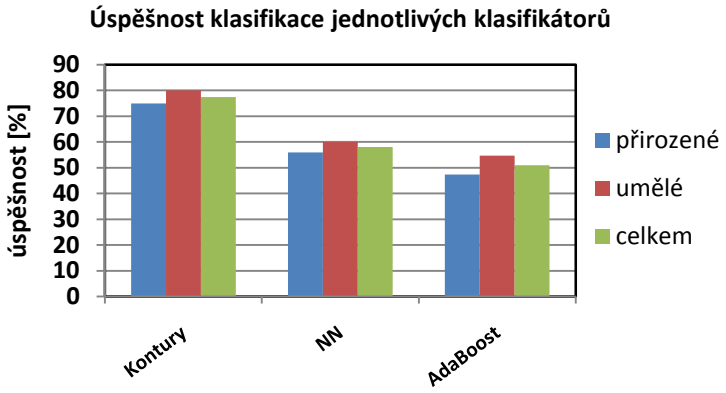
Tab. 4.8: Klasifikace pomocí histogramů orientace a metody AdaBoost

Klasifikované gesto \ Předváděné gesto															
	5/5						1/4								
		6/5	3/3								3/4	2/0			
			2/4												
				2/5		0/1									
		2/2	3/2		**9/8**			0/2							
						8/9						3/1			
	1/0						**5/6**								
		2/0	2/1		1/2			**10/8**	5/3						
		0/3							**5/7**						
										6/7	4/2				
										4/1	**3/4**				
						2/0						**2/4**	3/3	1/0	
				2/1								3/5	**4/3**	4/2	1/2
	4/5			6/4			4/0						1/1	**2/4**	7/5
										0/2			2/3	3/4	**2/3**

Dále pak je v tabulce Tab. 4.9 shrnuta úspěšnost pro jednotlivé klasifikátory a typy osvětlení. Obrázek Obr. 4.34 pak úspěšnost klasifikace vyjadřuje graficky.

Tab. 4.9: Úspěšnost klasifikace jednotlivých klasifikátorů

Klasifikátor	Přirozené	Umělé	Celkem
Kontury	75.00 %	80.00 %	77.50 %
HO+Neuronová síť	56.00 %	60.00 %	58.00 %
HO+AdaBoost	47.33 %	54.66 %	50.99 %

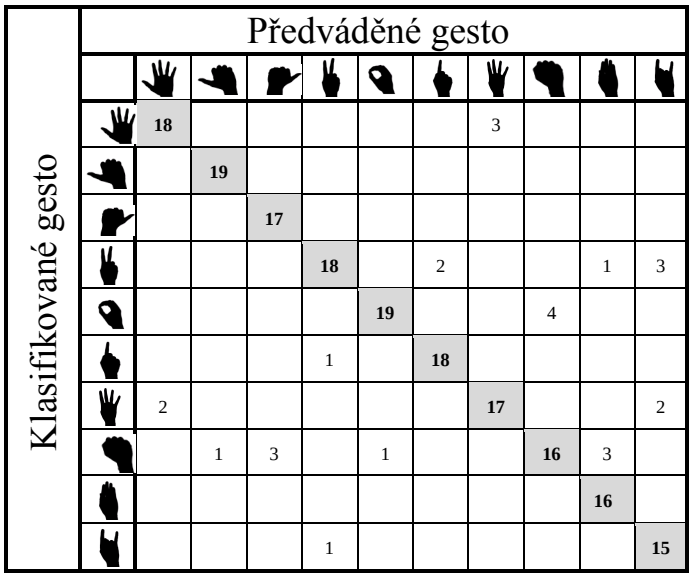


Obr. 4.34: Úspěšnost jednotlivých klasifikátorů

Klasifikátor založený na nalezení kontury ruky vykazuje dle testů nejlepší úspěšnost. Porovnání s klasifikací gesta s ostatními dvěma klasifikátory je silně zavádějící. Ke klasifikaci gesta se používá zcela odlišný vektor příznaků. Navíc byla gesta vybírána při programování tak, aby je mezi sebou bylo velmi snadné odlišit. Takto navržený klasifikátor trpí jednou zásadní nevýhodou. A tou je problematičnost přidávání dalších gest. Proto se již tímto klasifikátorem dále nezabývám a v koncové aplikaci není použit.

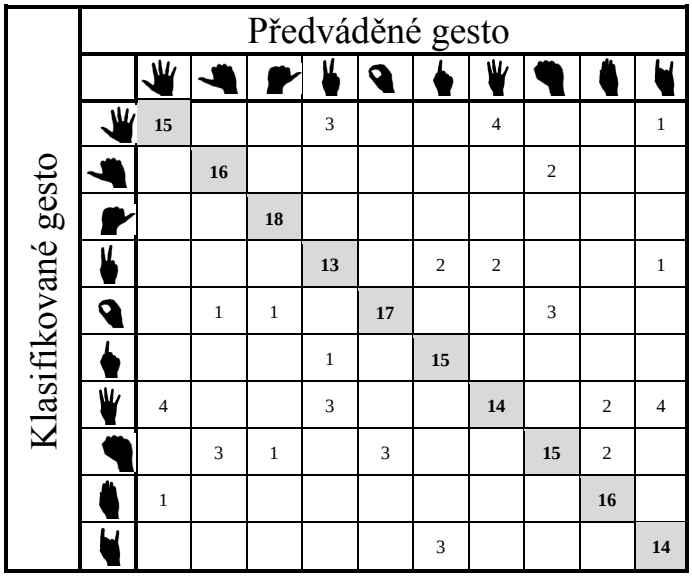
Oba zbývající klasifikátory vykazují jen velmi malou závislost na typu osvětlení. Toho je dosaženo tím, že je vzorek barvy kůže odebírán přímo za daných světelných podmínek. Úspěšnost klasifikace však není příliš vysoká. Lze nalézt dvě hlavní příčiny. Prvním z nich je patrná z tabulek Tab. 4.7 a Tab. 4.8. Nevhodným výběrem zejména posledních pěti gest vzniká velká chyba při klasifikaci. Histogramy orientací takto zvolených gest jsou si velice podobné. Proto dále budu pracovat s redukovaným slovníkem gest. Další příčinou nepříliš vysoké úspěšnosti klasifikace je volba poměrně členitého pozadí při testech. Pro další tety byla tedy natrénována neuronová síť pouze na deset gest a u AdaBoost klasifikátoru bylo použito pouze deset klasifikátorů. Pro oba takto sestavené klasifikátory byl proveden test, kdy bylo vykonáno každé gesto dvacetkrát.

Tab. 4.10: Klasifikace pomocí neuronové sítě – omezený slovník



	18						3			
		19								
			17							
				18		2			1	3
					19			4		
				1		**18**				
	2						**17**			2
		1	3		1			**16**	3	
									16	
				1						**15**

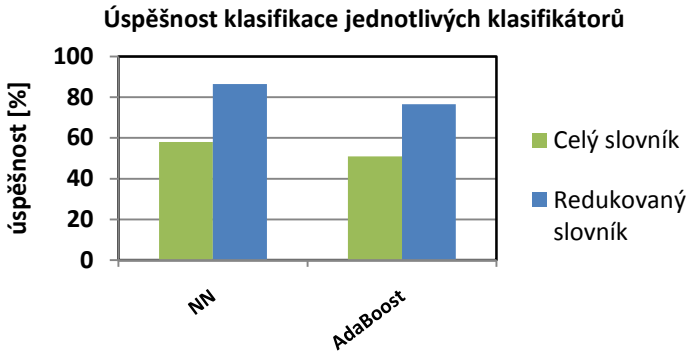
Tab. 4.11: Klasifikace pomocí metody AdaBoost – omezený slovník



	15			3			4			1
		16						2		
			18							
				13		2	2			1
		1	1		**17**			3		
				1		**15**				
	4			3			**14**		2	4
		3	1		3			**15**	2	
	1								**16**	
						3				**14**

Tab. 4.12: Úspěšnost klasifikace – redukovaný slovník gest

Klasifikátor	Celkem
H.O.+Neuronová síť	86.5 %
H.O.+AdaBoost	76.5 %



Obr. 4.35: Úspěšnost klasifikace při redukovaném počtu gest

Z tabulky Tab. 4.12 a grafu na obrázku Obr. 4.35 je patrné že redukce počtu gest vedla k výraznému zlepšení klasifikace. Ke zlepšení výsledků klasifikace také vedlo to, že gesta byla prováděna v málo členitém prostředí oproti prvotním testům. Na obrázku Obr. 4.36 a) můžeme vidět prostředí, při kterém byla vyhodnocována všechna gesta. Obrázek Obr. 4.36 b) ukazuje prostředí málo členité, ve kterém byly prováděny vyhodnocování redukovaného slovníku.

a) b)

Obr. 4.36: Prostředí při klasifikaci gest

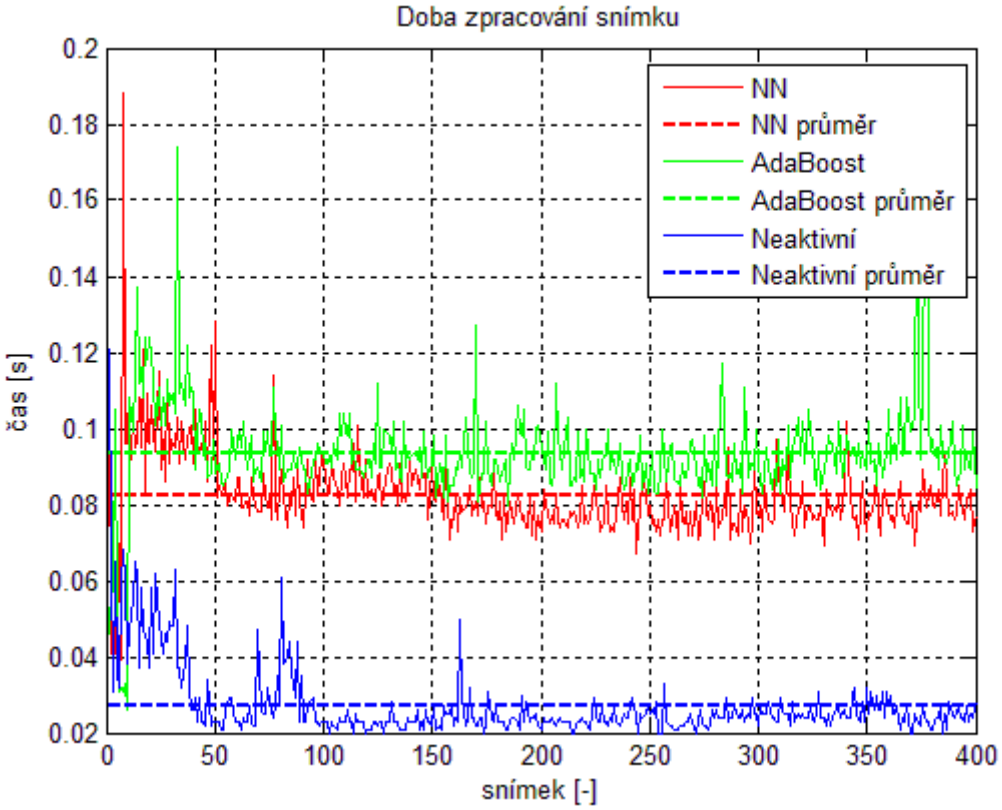
a) členité, b) ideální

4.6.4.2 Test 2 – test rychlosti klasifikátorů

Dále byla testována rychlost práce aplikace při použití neuronové sítě a při použití klasifikátoru AdaBoost. Na obrázku Obr. 4.37 jsou znázorněny celkové časy pro zpracování jednoho snímku a vyhodnocení gesta. Čas zpracování byl sledován na úseku 400 snímků. Počáteční nízké hodnoty jsou naměřeny ve stavu, kdy ještě nebylo uživatelem zahájeno vyhodnocování gesta. Dále pak jsou již hodnoty měřeny v aktivním stavu, to znamená při vyhodnocování gesta. Z naměřených dat vyplývá, že za použití neuronové sítě je celkový čas klasifikace gesta nižší než při použití klasifikátorů AdaBoost. To lze vysvětlit tím, že při použití klasifikátorů AdaBoost musíme předložit vektor příznaků aktuálního gesta všem klasifikátorům natrénovaným pro dílčí gesta. Počet predikcí pak odpovídá počtu klasifikovaných gest. Oproti tomu

u neuronové sítě se klasifikuje jen jednou. O výsledném gestu se pak usuzuje podle hodnot obsažených ve vektoru výstupů. Pro porovnání jsou v grafu uvedeny i hodnoty času zpracování snímků v neaktivním stavu vyhodnocování gesta.

Při vyhodnocování gesta pomocí neuronové sítě byla průměrná rychlost zpracování snímku 0.082 s (12 snímků/s). Při použití klasifikátorů AdaBoost je průměrný čas zpracování 0.092 s (10 snímků/s). Pokud je klasifikace neaktivní, to znamená, že se neprovádí ani vyhledání barvy pokožky a s tím spojené operace, je průměrná doba zpracování snímku 0.027 s (37 snímků/s).



Obr. 4.37: Časy zpracování snímků

Z výsledků prvního testu je zřejmé, že klasifikátor v podobě neuronové sítě dosahoval v ideálních podmínkách lepších výsledků nežli klasifikátor AdaBoost (Obr. 4.35. a Tab. 4.12). A rovněž v časovém hodnocení dosáhla lepších výsledků neuronová síť. Proto je v konečné navržené aplikaci jako klasifikátor neuronová síť. Následující testy už jsou prováděny pouze pro neuronovou síť.

4.6.4.3 Test 3 – doplňkové testy vybraného klasifikátoru

V tomto testu byl zjištěn vliv předmětů umístěných na ruku. To znamená, jaký vliv na úspěšnost klasifikace by měli například hodinky uživatele nebo prsten který by uživatel měl na ruce. Situace je znázorněna na obrázku Obr. 4.38. Byla vykonána 4 gesta, která byla v předešlých testech klasifikována nejspolehlivěji. Každé z těchto

gest bylo vykonáno desetkrát. Tabulka Tab. 4.13 ukazuje počty správně klasifikovaných gest. Výsledná úspěšnost je tedy 82.5 %. Je patrné, že vliv například hodinek na ruce na vliv klasifikace je minimální. Toto lze do značné míry přisoudit úpravě masky barvy kůže pomocí morfologických operací, kdy je případná mezera v masce barvy kůže úspěšně zacelena. Je ovšem zřejmé, že při výskytu větších objektů nežli je objekt zobrazený na obrázku Obr. 4.38, by byla kvalita segmentace do značné míry ovlivněna. Tím by i klasifikace gesta vykazovala horší přesnost.

Obr. 4.38: Cizí předmět na ruce

Tab. 4.13: Úspěšnost klasifikace s cizím předmětem na ruce

Klasifikované gesto	Předváděné gesto				
	7				
		9			
			9		
				8	

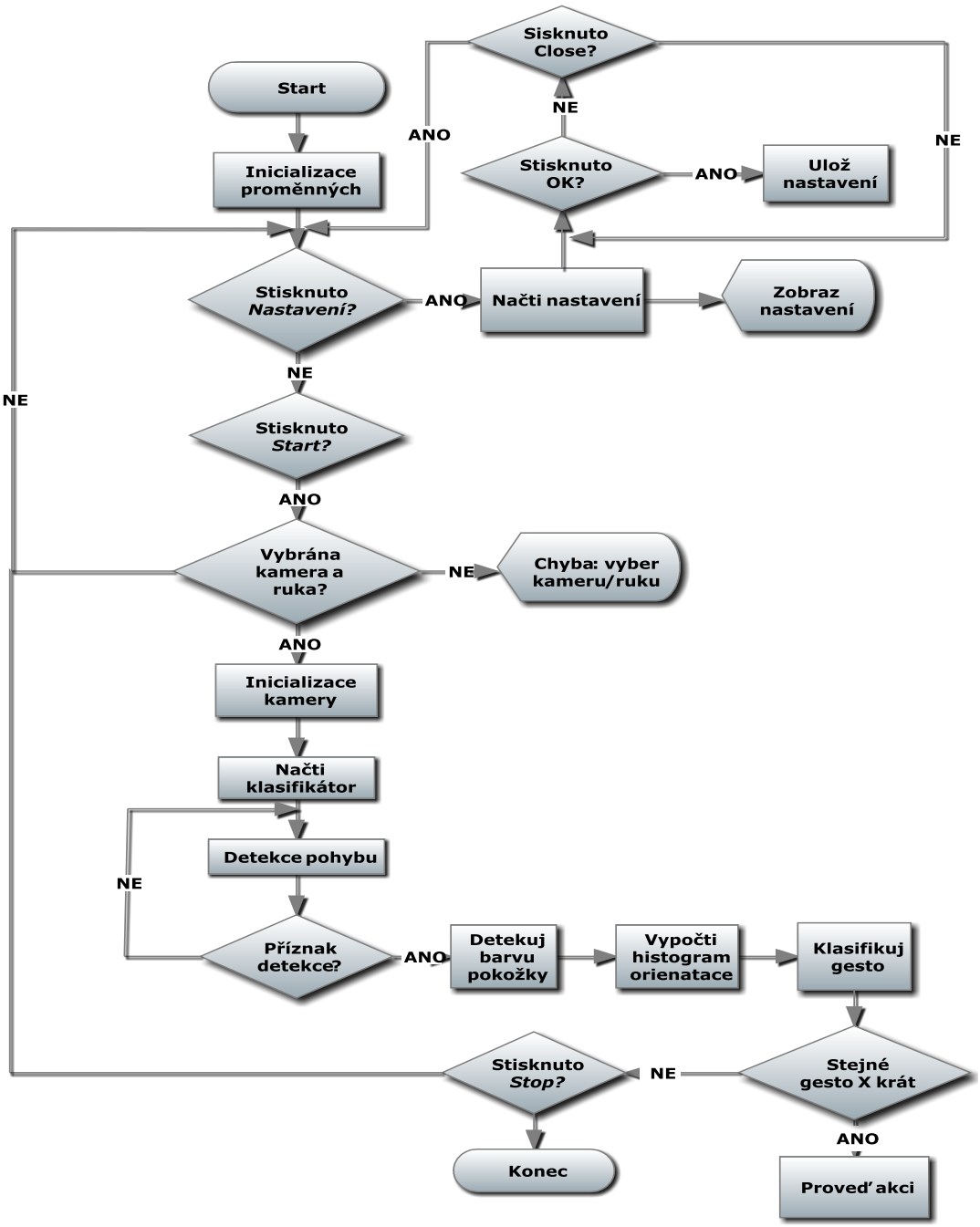
Dále byla zkoušena úspěšnost klasifikace gesta předváděného osobami, které byly pouze krátce seznámeny s funkcí aplikace. Ukázalo se, že osoby, které se testu účastnily, by potřebovaly mnohem delší čas na seznámení s aplikací respektive k osvojení ovládání aplikace. Testu se účastnily tři osoby a úspěšnost klasifikace se pohybovala od 60 % zhruba do 70 %. Při předvádění gest z tabulky Tab. 4.13.

Obecně tedy lze říct, že pro kvalitu klasifikace jsou důležité následující podmínky: kvalitní osvětlení, ne příliš členité prostředí, určitá praxe v používání systému, vypnutí veškeré automatiky kamery (vyvážení bílé …), kvalita obrazu snímaného kamerou.

4.6.5 Popis aplikace a implementační detaily

V této kapitole bude uveden podrobný popis aplikace a její ovládání. Princip práce bude popsán pomocí vývojového diagramu a popisu jednotlivých funkčních bloků. Dále pak bude formou obrázků a popisného textu vysvětlen způsob práce s navrženou aplikací. Navržené uživatelské rozhraní je realizováno v šabloně Windows forms applications.

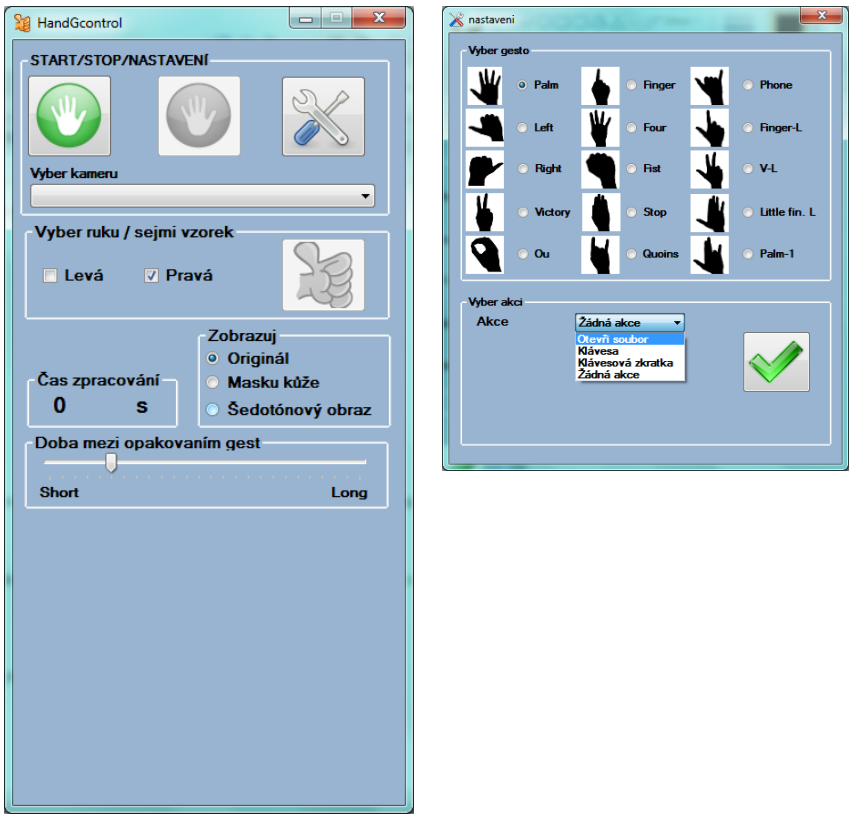
4.6.5.1 Vývojový diagram



Obr. 4.39: Vývojový diagram aplikace

4.6.5.2 Popis ovládání a příslušných funkcí

Po spuštění aplikace se uživateli zobrazí hlavní okno. Současně s otevřením aplikace je zjištěn počet připojených web kamer k počítači. Jejich názvy jsou vepsány do příslušného ComboBoxu. Uživatel před samotným spuštěním zpracování videa musí vybrat kameru, kterou chce používat. A také ruku, kterou bude gesto vykonávat. Dále pak stiskem tlačítka  může nastavit akci, která má být vykonána po rozpoznání gesta. Jedná - li se o první spuštění aplikace, je vygenerován soubor pro ukládání nastavení. Tento je uložen na pevný disk. Pokud jde již o opakované spuštění, z příslušného souboru jsou načteny nastavení uložená pro dané gesto. Uložení nastavení se provádí tlačítkem  . Pro operace související s nastavením byla napsána funkce `void prametryNacti(int rbIndex)`, která slouží k načtení parametrů nastavení pro dané gesto. Jejím vstupním parametrem je index gesta. Dále byla napsána funkce `parametry()`, která po stisku tlačítka  zapíše příslušná nastavení do konfiguračního souboru.



a) b)

Obr. 4.40: a) hlavní okno aplikace, b) okno nastavení akce

Uživatel má na výběr ze tří základních možností nastavení akce pro dané gesto. Vykonáním gesta lze generovat stisk klávesy, stisk klávesové zkratky nebo otevření souboru. Navržený způsob ovládání umožňuje použít vytvořenou aplikaci pro ovládání

velkého množství různých programů. Většina dnešních programů totiž umožňuje ovládání pomocí klávesových zkratek.

Tab. 4.14: Seznam použitelných kláves a klávesových zkratek

Prefix
Alt
Ctrl
Shift

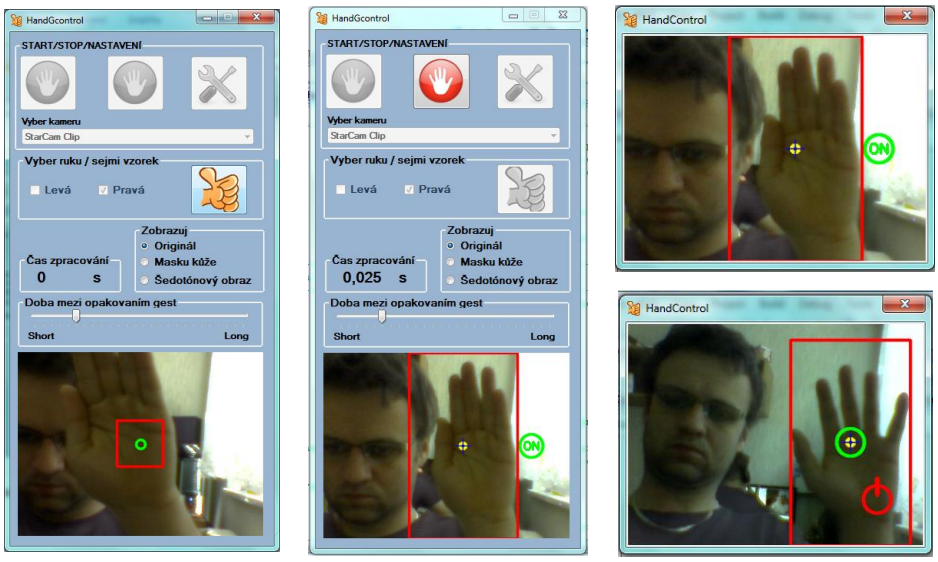
+

Klávesa	
a-z	LShift
0-9	Num Lock
Backspace	Space
Caps Lock	Tab
Enter	Up
Esc	Left
LAlt	Right
LCtrl	Home

Pokud před spuštěním samotné detekce ruky a klasifikace gest neprovedeme nastavení akcí pro jednotlivá gesta, je při detekci gesta pouze zobrazen jeho název a není vykonána žádna akce.

Je-li vybrána kamera a ruka, kterou bude gesto vykonáváno, můžeme již stiskem tlačítka  přistoupit k samotnému procesu detekce a klasifikace gesta. Po stisku tlačítka je provedena inicializace vybrané kamery, načtení natrénované neuronové sítě z disku (tuto je nutno mít ve stejné složce odkud je aplikace spouštěna pod názvem *ns1.m*) a další inicializace proměnných. Uživateli se začne zobrazovat obraz snímaný vybranou kamerou. V obraze je zobrazována oblast, kam má uživatel umístit svou ruku pro odebrání vzorku barvy kůže situaci znázorňuje obrázek Obr. 4.41 a). K odběru vzorku barvy pokožky slouží tlačítko  . Při stisku tohoto tlačítka je volána funkce `void gauss(IplImage* Ycbcr)`, která vypočítá gaussovo rozložení barvy pokožky. Zároveň se objevuje další menší samostatné okno, ve kterém je zobrazována detekovaná oblast ruky. Toto okno zůstává vždy v popředí. Uživatel si jej umístí do části obrazovky, kde mu nepřekáží při další práci.

Samotná klasifikace gest začne, když uživatel navede střed detekované oblasti ruky označeny žlutě do oblasti zeleného kruhu s nápisem ON. Detekce je signalizována tak, že se kolem žluté středové oblasti objeví zelený kruh a v dolním rohu okna (pravém nebo levém podle vybrané ruky, která provádí gesto) zobrazujícím obraz z kamery se zobrazí symbol pro vypnutí sledování. Vypnutí sledování se provede navedením středu oblasti ruky do symbolu vypnutí. Takto zvolený způsob ovládání umožňuje uživateli jednoduché zapnutí a vypnutí aplikace bez použití klávesnice nebo myši. Po vypnutí aplikace se po stanovené době zobrazí oblast pro umístění ruky uprostřed obrazu a uživatel při dalším požadovaném zapnutí musí vložit svou ruku do vymezené inicializační oblasti.



a) b) c)

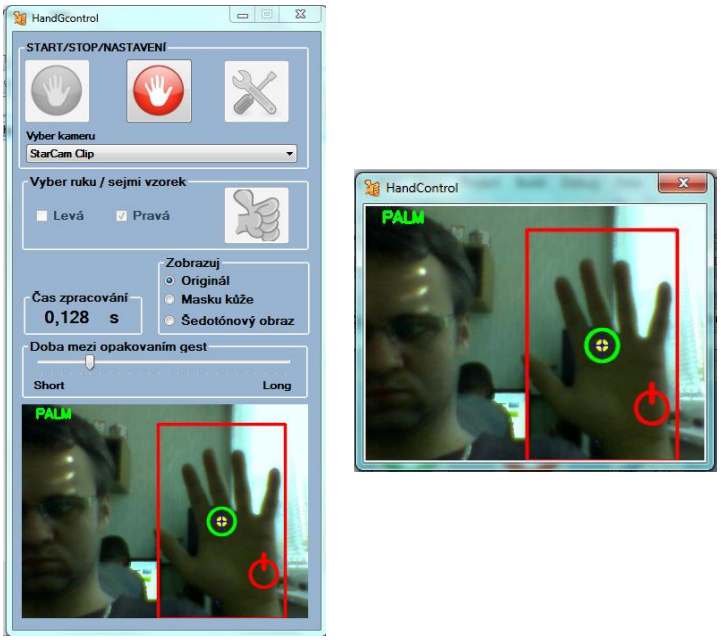
Obr. 4.41: a) hlavní okno odběr vzorku, b) Hlavní okno detekce pohybu, c) pomocné okno – nahoře detekce pohybu, dole klasifikace gesta

Pokud je sledování aktivní, je v oblasti pohybu vyhledávána barva pokožky pomocí funkce `void VyhledaniGaussBarva(IplImage *obraz,CvRect rect)`. Dále jsou prováděny další úpravy popsané v kapitole 4.3.3. Obraz nalezené ruky je převeden na šedotónový pomocí funkce `IplImage sedoton(IplImage *vstup)`. Z tohoto obrazu je pomocí funkce `void orientHist (IplImage *vstup)` získán vektor příznaků gesta – histogram orientací. Gesto v každém snímku je vyhodnoceno pomocí funkce `int predpoved(void)`, která vrací číslo rozpoznaného gesta. Aby byla provedena akce přiřazená k vykonávanému gestu, musí být dané gesto detekováno v několika snímcích po sobě. Zde ještě zmíním, že uživatel může nastavit rychlost opakování akce při vykonání určitého gesta. Citlivost je měněna na hlavním okně uživatelského rozhraní pomocí posuvníku. Pokud tedy má být akce opakována jednou za deset snímků, musí být dané gesto rozpoznáno v šesti případech z deseti. Ošetření této podmínky zajišťuje funkce `void action(int gesto)`. Vstupním parametrem této funkce je číslo gesta získané z funkce `int predpoved(void)`. Funkce tedy počítá četnost výskytu jednotlivých gest a při splnění podmínky zavolá funkci `void akce(int gesto)`, která již podle konečného rozpoznaného gesta vykoná akci ke gestu přiřazenou.

Uživatel pak ještě může měnit způsob zobrazení videa v hlavním ovládacím okně. Může být zobrazován původní obraz, stejný jako v malém pomocném okně, nebo může být zobrazována pouze binární maska oblasti ruky. U volby zobrazení binárního obrazu lze ještě měnit práh, který určuje, zda bude zkoumaný pixel prohlášen pixelem pokožky. Třetí možností je zobrazení obrazu šedotónového, z kterého jsou počítány histogramy orientace.

Cyklický běh programu je zajištěn voláním funkcí pomocí časovače. Kdy je perioda nastavena na 50 ms. Před zpracováním funkcí v dalším cyklu je ověřeno, zda již bylo dokončeno celé zpracování v cyklu předešlém.

Název klasifikovaného gesta je uživateli zobrazen v levém horním rohu obrazu jak v okně hlavním, tak v okně vedlejším – obrázek Obr. 4.42.



a) b)

Obr. 4.42: Zobrazení klasifikovaného gesta

Snímání obrazu je možné zastavit tlačítkem  . Při stisku tohoto tlačítka je zastaveno cyklické zpracování obrazu a jsou vynulovány potřebné proměnné.

5 ZÁVĚR

Cílem této práce bylo seznámit se s metodami detekce částí lidského těla především pak ruky v obraze a navrhnout algoritmus, který bude schopen rozpoznat několik základních statických gest ruky. Rozpoznaná gesta pak měla sloužit ovládání počítače.

V první části práce jsou popsány postupy, které se běžně k detekci ruky v obraze používají. Konkrétně se jedná o detekci ruky na základě barvy pokožky a o detekci ruky na základě pohybu. Dále jsou popsány metody používané pro klasifikaci statických gest ruky.

Druhá část práce se již věnuje výběru vhodných metod pro detekci ruky v obraze a výběru metod pro samotnou klasifikaci gesta. Pro detekci ruky v obraze byla zvolena kombinace detekce barvy pokožky a detekce pohybu. Pro detekci barvy pokožky byl na základě testů vybrán YCbCr barevný prostor. Jako model barvy pokožky byl vybrán model s jedním gaussiánem. Vzorek barvy pokožky je vybírán přímo z oblasti ruky uživatele, který aplikaci používá. Tento přístup při testech vykazoval nejlepší vlastnosti. Je jím minimalizována závislost kvality detekce barvy pokožky na typu osvětlení.

Dále byly pro detekci ruky v obraze testovány metody využívající detekci pohybu. Testovaná metoda detekce pohybu pomocí upravené estimace prostředí se ukázala jako nevhodná z důvodu členitosti ruky. Dalším problémem byl způsob aktualizace modelu prostředí, který je při vyhodnocování statických gest poměrně obtížný. Proto byla pro detekci ruky na základě pohybu využita upravená metoda rozdílových snímků, která ve spojení se sadou podmínek dokáže detekovat ruku i při jejím minimálním nebo nulovém pohybu. Tato metoda je také časově méně náročná než metoda estimace modelu prostředí. Obě metody byly sloučeny tak, že barva pokožky je detekována pouze v oblasti pohybu ruky. Takto navržený algoritmus detekce ruky v obraze je za jistých omezujících podmínek spolehlivě funkční. Gesto nesmí být prováděno v popředí, které má barvu podobnou barvě kůže (barva dřeva) a nesmí docházet k prudkým změnám osvětlení.

Pro samotnou klasifikaci gesta byly testovány tři metody. Úspěšnost klasifikace první metody, kdy se ke klasifikaci gesta používaly příznaky odvozené z kontury ruky, konvexní obálky ruky a defektů v konvexní dosahovala 77.5 %. Ovšem velkou nevýhodou takto navrženého klasifikátoru byla složitost dalšího přidávání gest do slovníku klasifikovaných gest. Dále byly pro klasifikaci gesta testovány neuronová síť a sada klasifikátorů AdaBoost. Jako příznakový vektor pro tyto klasifikátory sloužil histogram orientací. Při použití stejných příznakových vektorů vykazovala větší úspěšnost klasifikace neuronová síť. Konkrétně pro redukovaný slovník gest a pro málo členité prostředí dosahovala úspěšnosti 86.5 % oproti 76.5 % u klasifikátorů AdaBoost. Proto byla pro klasifikaci gesta vybrána neuronová síť.

Navržená aplikace tedy využívá pro nalezení ruky v obraze detekci barvy pokožky a detekci pohybu. Jako příznakový vektor pro popis gesta je využíván třiceti šesti prvkový histogram orientací. Klasifikaci gesta pak zajišťuje vícevrstvá neuronová síť, která má 100 neuronů v první skryté vrstvě a 72 neuronů ve druhé skryté vrstvě.

Při vývoji a testování aplikace se ukázalo, že na správnou klasifikaci má zásadní vliv výběr vhodných gest. Pokud jsou vybrána gesta s podobnými histogramy orientací, úspěšnost klasifikace rapidně klesá. Na úspěšnost klasifikace pak mají vliv i parazitní objekty vyskytující se v oblasti segmentované ruky. Jedná se například o dřevěné objekty v pozadí obrazu. Z toho tedy vyplývá, že výsledná klasifikace je závislá na kvalitě nalezení ruky v obraze.

Navržená aplikace je schopna rozpoznat deset statických gest ruky - Tab. 4.10. Pro každé gesto uživatel může nastavit akci v podobě stisku vybrané klávesy, stisku klávesové zkratky nebo otevření souboru.

Aplikace také splňuje požadavek na práci v reálném čase, kdy průměrná rychlost zpracování jednoho snímku byla na notebooku o konfiguraci uvedené v kapitole 4.3.2.1 0.082 s (12 snímků/s).

Literatura

[1] AGAM, Gady . *Introduction to programming with OpenCV* . [online]. 2006, [cit. 2011-01-16]. Dostupný z WWW: <http://www.cs.iit.edu/~agam/cs512/lect-notes/opencv-intro/opencv-intro.html#SECTION00071000000000000000>.

[2] BARTONČÍK, Michal. *Rozpoznávání výrazů tváře u neznámých osob*. Brno, 2010. 29 s. Semestrální práce. VUT Brno.

[3] BAŠTEK, Jozef. *Hra s ovládáním pomocí gest ruky*. Brno, 2008. 34 s. Bakalářská práce. VUT Brno.

[4] *Cognotics* [online]. 2008 [cit. 2011-05-18]. Machine Learning Reference. Dostupné z WWW: <http://www.cognotics.com/opencv/docs/1.0/ref/opencvref_ml.htm#ch_ann>.

[5] CORRADINI, Andrea. *Dynamic Time Warping for Off-line Recognition of a Small Gesture Vocabulary* .[online]., [cit. 2010-05-03]. Dostupný z WWW: <http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=00938914>

[6] ČERNOCKÝ, Jan. *Zpracování řečových signálů - studijní opora*. [online]. 2006, [cit. 2010-05-03]. Dostupný z WWW: <http://www.fit.vutbr.cz/study/courses/ZRE/public/opora/zre_opora.pdf>.

[7] *DsynFlo* [online]. 2010 [cit. 2011-05-18]. Cvblobslib with openCV Installation. Dostupné z WWW: <http://dsynflo.blogspot.com/2010/02/cvblobskib-with-opencv-installation.html>.

[8] FREEMAN, William T.; ROTH, Michal. *Orientation Histograms for Hand Gesture Recognition*. [online]. 1994, [cit. 2010-05-02]. Dostupný z WWW: <http://citeseerx.ist.psu.edu>.

[9] HONGBIN, Zhou, et al. *Real-time fish detection based on improved adaptive background*. [online]. 2008, [cit. 2010-05-01]. Dostupný z WWW: <http://www.wseas.us/e-library/conferences/2008/hangzhou/acacos/54-586-358.pdf>.

[10] HONZÍK, Petr. *Meta learning*. Brno: VUT Brno, [online]. 2011 [cit. 2011-05-12]. Dostupné z WWW: < https://www.vutbr.cz/www_base/priloha.php?dpid=45524>.

[11] HORÁK, Karel *Dynamické obrazy*. In *Počítačové vidění*. Brno : VUT Brno, [online]. 2008 [cit. 2010-04-25]. Dostupné z WWW:<https://www.vutbr.cz/studis/student.phtml?gm=gm_detail_predmetu&apid=86980>.

[12] HSV In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 28. 3. 2010 [cit. 2010-04-25]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/HSV>.

[13] JIRSÍK, Václav. *Umělé neuronové sítě.* Brno: VUT Brno, 2010 [cit. 2011-05-12].

[14] JIRSÍK, Václav. *Vícevrstvá neuronová síť s algoritmem učení backpropagation.* Brno: VUT Brno, 2010 [cit. 2011-05-12].

[15] JIŘÍK, Leoš. *Rozpoznávání pozic a gest*. Brno, 2008. 47 s. Diplomová práce. VUT Brno.

[16] KALOVÁ, Ilona. *Segmentace obrazových primitiv*. [online]. [cit. 2011-05-11]. Dostupný z WWW: <http://www.uamt.feec.vutbr.cz/vision/TEACHING/MPOV/05%20-%20Segmentace%20a%20detekce%20geometrickych%20primitiv.pdf>.

[17] KOLSCH, Mathias; TURK, Matthew. *Robust Hand Detection*. [online]. 2004, [cit. 2010-05-02]. Dostupný z WWW: <http://ilab.cs.ucsb.edu/projects/mathias/KolschTurk2004RobustHandDetection.pdf>.

[18] LAGANIÈRE, Robert. *Programming computer vision applications* [online]. 2009 [cit. 2011-05-18]. A step-by-step guide to the use of Microsoft Visual C++ and the Intel OpenCV library. Dostupné z WWW: <http://www.site.uottawa.ca/~laganier/tutorial/opencv+directshow/cvision.htm>.

[19] MANRESA, Cristina, et al. *Real – Time Hand Tracking and Gesture Recognition for Human-Computer Interaction . Electronic Letters on Computer Vision and Image Analysis* [online]. 2000, [cit. 2010-05-02]. Dostupný z WWW: <http://citeseerx.ist.psu.edu>. ISSN 1577-5097.

[20] MATĚJÍČKA, Miloš. *Ovládání počítače gestem ruky*. Brno, 2008. 24 s. Bakalářská práce. VUT Brno.

[21] NEELAMEGAN, Naren. *The code project* [online]. 2004 [cit. 2011-05-18]. Keyboard Events Simulation using keybd_event() function. Dostupné z WWW: <http://www.codeproject.com/KB/system/keyboard.aspx>.

[22] NOBUYUKI, Otsu. *A threshold selection method from gray-level histograms.* [online]. [cit. 2010-05-03]. Dostupný z WWW: <http://web.ics.purdue.edu/~kim497/ece661/OTSU_paper.pdf>.

[23] PETYOVSKÝ, Petr. *Kalsifikátory, strojové učení, automatické třídění 1*. *Počítačové vidění* [online]. 2008, [cit. 2010-05-03]. Dostupný z WWW: <https://www.vutbr.cz/studis/student.phtml?gm=gm_detail_predmetu&apid=86980>.

[24] PETYOVSKÝ, Petr. *Reprezentace a vlastnosti obrazových dat.* In *Počítačové vidění*. Brno: VUT Brno, [online]. 2008 [cit. 2010-04-25]. Dostupné z WWW:<https://www.vutbr.cz/studis/student.phtml?gm=gm_detail_predmetu&apid=86980>.

[25] SVOBODA, Tomáš. *Rozpoznávání gest*. Brno, 2009. 36 s. Bakalářská práce. VUT Brno.

[26] SYMEONIDIS, Klimis. Hand gesture recognition using neural network [online]. [s.l.], 2000. 68 s. Seminární práce. School of Electronic and Electrical Engineering. Dostupné z WWW: <http://personal.ee.surrey.ac.uk/Personal/T.Windeatt/msc_projects/symeonidis/Final%20Project.pdf>.

[27] ŠPANĚL, Michal. *Rozpoznávání gest ve video sekvencích*. Brno, 2003. 83 s. Diplomová práce. VUT Brno.

[28] ŠPANĚL, Michal; BERAN, Vítězslav. *Obrazové segmentační techniky.* - [online]. 2005, [cit. 2011-05-11]. Dostupný z WWW: < http://www.fit.vutbr.cz/~spanel/segmentace/>

[29] TANIBATA, Nobuhiko; SHIMADA, Nobutaka; SHIRAI, Yoshiaki. *Extraction of Hand Features for Recognition of Sign Language Words.* - [online].[cit. 2010-05-02]. Dostupný z WWW: <http://citeseerx.ist.psu.edu>

[30] VERNER, Jan. *Rozpoznání jednoduchých gest ruky*. Brno, 2007. 36 s. Bakalářská práce. VUT Brno.

[31] VEZHNEVETS, Vladimir*;* SAZONOV, Vassili *;* ANDREEVA, Alla. *A Survey on Pixel-Based Skin Color Detection Techniques. In* A Survey on Pixel-Based Skin Color Detection Techniques. *Moskva : Graphics and Media Laboratory,*. [online]. *2003 [cit. 2010-04-25]. Dostupné z WWW: <http://graphicon.ru/oldgr/en/publications/text/gc2003vsa.pdf*>.

[32] VIOLA, Paul; JONES, Michael. *Robust Real-time Object Detection.* - [online]. 2001, [cit. 2010-05-02]. Dostupný z WWW: http://research.microsoft.com/en-us/um/people/viola/pubs/detect/violajones_ijcv.pdf>.

[33] *Willowgarage* [online]. 2010 [cit. 2011-01-16]. OpenCV 2.1 C Reference. Dostupné z WWW: <http://opencv.willowgarage.com/documentation/c/index.html>.

[34] ZLOTÝ, Petr. *Rozpoznávání gest ruky v obraze*. Brno, 2009. 56 s. Bakalářská práce. VUT Brno.

Seznam zkratek

zkratka	význam	popis
HSV	H – hue – barevný odstín, S – saturation – sytost, V – value – hodnota jasu	Barevný prostor
PC	Personal Computer	Osobní počítač
RGB	R – Red – červená, G – green – zelená, B – blue – modrá	Barevný prostor
YCbCr	Y - jas, rozdílové složky C_r a C_b	Barevný prostor
LUT	Look up table	Vyhledávací tabulka
SOM	Self organizing map	Samo organizační mapa
MBF	-	Popředí
MFF	-	Díry v popředí
MFB	-	Falešné objekty