VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

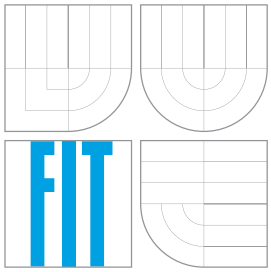
ROZPOZNÁVÁNÍ GEST

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE TOMÁŠ SVOBODA
AUTHOR

BRNO 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ROZPOZNÁVÁNÍ GEST

GESTURE RECOGNITION

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — TOMÁŠ SVOBODA

AUTHOR

VEDOUCÍ PRÁCE — Ing. MICHAL HRADIŠ

SUPERVISOR

BRNO 2009

Abstrakt

Tato bakalářská práce se zabývá rozpoznáváním gest rukou. Jsou diskutovány výhody a nevýhody různých barevných modelů pro detekci barvy kůže. Kůže je detekována pomocí vyhledávací tabulky. Vyhledávací tabulka je vytvořena z histogramu barvy kůže a volitelně Gaussova rozdělení, jehož parametry jsou z histogramu odhadnuty. Pro klasifikaci gest jsou využity Skryté Markovovy modely. Pro práci s těmito modely byl použit toolkit HTK. Rozpoznávání gest v reálném čase zajištuje vlastní dekodér Skrytých Markovových modelů využívající Viterbiho algoritmus. Bylo provedeno několik experimentů se systémem a datovými sadami pro 4 gesta. Výsledky experimentů jsou velmi dobré.

Abstract

This Bachelor's thesis is engaged in recognition hand gestures. The advantages and disavantages of various color models for skin color detection are discussed here. Skin is detected by look-up table. Look-up table is created from histogram of skin color and optional from Gaussian distribution, whose parameters are estimated from histogram. Hidden Markov models are used for gesture classification. The HTK toolkit have been used for working with the models. Own decoder of Hidden Markov models based on Viterbi algorithm was created for real-time gesture recognition. Several experiments were accomplished with data sets for 4 gestures. The results of the experiments are very good.

Klíčová slova

klasifikace, Skryté Markovovy modely, Gaussovo rozdělení, gesto ruky, histogram, vyhledávací tabulka, vektor příznaků

Keywords

classification, Hidden Markov models, Gaussian distribution, hand gesture, histogram, look-up table, feature vector

Citace

Tomáš Svoboda: Rozpoznávání gest, bakalářská práce, Brno, FIT VUT v Brně, 2009

Rozpoznávání gest

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Michala Hradiše.

.....

Tomáš Svoboda

15. května 2009

Poděkování

Rád bych poděkoval vedoucímu práce Ing. Michalu Hradišovi za vstřícný přístup a pomoc při řešení této práce. Dále bych rád poděkoval Ing. Josefu Mlíchovi za poskytnutí jeho skriptů pro práci s toolkitem HTK.

Obsah

Kapitola 1

Úvod

V dnešní době výkonných počítačů vzniká celá řada oblastí, kde lze tento výkon smysluplně využít. Jednou z nejzajímavějších je zpracování různých druhů multimediálních dat. V oblasti zpracování obrazu nebo zvuku probíhá intenzivní výzkum a aplikace založené na tomto výzkumu jsou stále častěji využívány v praxi. Cílem těchto aplikací může být například analýza obrazu využitelná v různých druzích průmyslu, systémech bezpečnostních kamer apod. Další využití zpracování multimédiálních dat najdeme v nových možnostech interakce člověka a počítače. Ovládání počítače hlasem nebo gesty ruky je velice zajímavá možnost.

Ve své bakalářské práci se zabývám rozpoznáváním gest rukou. Pohyby a gestikulace rukou mohou být velkým přínosem na poli ovládání počítače. V dnešní době již existuje několik případů, kdy byl pohyb ruky nebo předmětu, který v ruce držíme, úspěšně použit jako nový přístup k uživatelskému rozhraní počítače. Může se jednat například o systém vyvinutý na uviverzitě Ben Gurion v Izraeli. Tento systém umožňuje lékařům manipulaci s obrazovými daty mozku bez nutnosti využití klávesnice nebo myši[1]. Další příklad podobného systému můžeme nalézt v zábavním průmyslu. Tímto příkladem je ovládání konzole Nintendo Wii.

Rozpoznávání gest rukou tedy není nová oblast a věnuje se jí několik skupin vědců z různých univerzit světa.

Ming-Hsuan Yang a Narendra Ahuja z univerzity Illinois se zabývali rozpoznáváním 40 symbolů americké znakové řeči [7]. Využívali afinní transformace souřadnic trajektorie dlaně pro získání informací o pohybu dlaně a následné natrénování časově zpožděné neuronové sítě (TDNN).

Další, kdo zkoumal tuto problematiku, byl tým složený z vědců z univerzit Missouri a Tsinghua, kteří svou práci založili na algoritmu DTW[2][9] a rozpoznávali 12 gest rukou.

Posledním příkladem, který bych rád uvedl, je velmi zajímavá práce týmu z Bostonské

[1] http://www.vision-systems.com/display_article/340083/19/none/none/Techt/Hand-gesture-recognition-system-targets-medical-applications

[2] Dynamic time warping – Borcení časové osy

univerzity[1], jenž využívá rozšířený algoritmus DTW pro rozpoznávání čísel od 0 do 9. Výsledky všech zmíněných skupin vědců jsou velmi dobré.

Každý z týmů, které se rozpoznáváním gest již zabývaly, použil jiné metody a přístupy, na něž bude dále odkazováno v příslušných částech práce. Je tedy vidět, že k rozpoznávání gest rukou lze přistupovat mnoha různými způsoby. Způsob, který jsem zvolil já, a jeho výsledky budou popsány v této práci.

Text práce má tuto strukturu. Po úvodní kapitole následuje kapitola, ve které je nastíněn problém statistického rozpoznávání vzorů a jeho možná řešení. V další části této kapitoly jsou rozebrány metody odhadu funkce hustoty pravděpodobnosti. Poté jsou diskutovány výhody a nevýhody barevných modelů HSV a normalizovaného RGB. Dále jsou vysvětleny použité metody pro segmentaci obrazu za účelem nalezení oblastí barvy kůže. Následující část se zabývá parametrizací jednotlivých gest. Nakonec je v této kapitole uveden teoretický základ Skrytých Markovových modelů (HMM), které jsou využity pro vlastní rozpoznávání gest. Důraz je kladen na dekódování HMM pomocí Viterbiho algoritmu.

Ve třetí kapitole je práce rozebrána z algoritmického hlediska a věnuji se zde vlastnímu návrhu aplikace. Je tu popsána celá struktura systému. Vysvětlen je postup segmentace videa za účelem nalezení pohybující se ruky, postup získání parametrů pro klasifikaci gest a vlastní postup při klasifikaci jednotlivých gest.

Čtvrtá kapitola se zabývá prací na implementační úrovni. Tato kapitola je rozdělena do dvou částí. První část se zaměřuje především na knihovnu OpenCV, která byla využita pro zpracování obrazu. V druhé části je popsána práce s toolkitem HTK, jenž je využit pro klasifikaci gest. Také je zde zmíněna knihovna TinyXML, která je použita pro zpracování xml dokumentů, v nichž jsou uchovány informace o natrénovaných modelech pro rozpoznávání gest v reálném čase.

V páté kapitole jsou uvedeny typy gest, které jsem se rozhodl rozpoznávat. Dále je zde uvedeno, jakým způsobem jsem získával datové sady nutné pro trénování HMM a následné testování systému. Výčet a popis těchto datových sad se také nalézá v této kapitole.

Šestá kapitola je zaměřena na experimenty. V první části jsou popsány všechny druhy experimentů, které jsem prováděl. V druhé části následuje vyhodnocení všech provedených experimentů.

Poslední kapitolou je závěr, jenž obsahuje zhodnocení výsledků mé práce a nastiňuje možnosti dalšího rozvoje.

Kapitola 2

Teoretická východiska

V této kapitole jsou popsány teoretické základy práce. Nejdříve se zaměřím na statistické rozpoznávání vzorů. Tato teorie je základem celé práce a je využita jak u segmentace barvy kůže, tak při vlastním rozpoznávání gest. V následující části jsou popsány dva způsoby aproximace funkce hustoty pravděpodobnosti, která je využívána v teorii statistického rozpoznávání vzorů. Dále diskutuji výhody a nevýhody barevných modelů RGB, normalizované RGB a HSV z pohledu detekce barvy kůže. Další část této kapitoly popisuje možnosti parametrizace ruky, které jsou vhodné pro rozpoznávání gest. V poslední části je nastíněn teoretický základ Skrytých Markovových modelů.

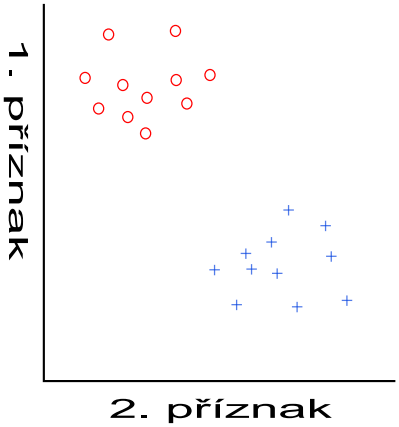
2.1 Statistické rozpoznávání vzorů

Základním úkolem rozpoznávání vzorů je klasifikovat příchozí data do několika tříd, z nichž každá obsahuje data stejného druhu.

Po nasnímání dat nějakým zařízením (kamera, mikrofon apod.) dochází k segmentaci vstupních dat, jejich zpracování a získávání informací relevantních pro rozdělení dat do tříd. Těmto informacím se říká příznaky. Z jednoho vstupního vzorku dat (snímek videa, časový úsek zvukového signálu apod.) můžeme získat několik příznaků, které jsou pak sdruženy do vektoru příznaků. Na obrázku 2.1 jsou vidět data dvou tříd v dvourozměrném příznakovém prostoru.

Úlohám, jejichž cílem je rozdělit data do dvou tříd, se často říká detekční. Cíl těchto úloh je obvykle formulován jako detekce nějakého „objektu“ (obličeje, mluvčího, vadného výrobku), což v praxi znamená, že data rozdělujeme do dvou tříd. První třída představuje detekovaný objekt a druhá zahrnuje všechny data, která daným objektem nejsou.

Klasifikace dat do několika tříd (např. dvou) může být prováděna různými způsoby, na základě různých přístupů. Společným znakem všech přístupů je to, že data jsou rozpoznávána vždy za pomoci určitých druhů pravidel nebo modelů. Tyto modely je třeba pro daný úkol nejdříve natrénovat na sadě dat, které se říká trénovací. Nástroj, jenž dané modely využívá a jehož cílem je rozdělení dat do příslušných tříd, se nazývá klasifikátor.



Obrázek 2.1: Zobrazení dat dvou tříd v dvourozměrném příznakovém prostoru. Data jsou lineárně oddělitelná.

Hlavním rozlišením různých přístupů ke klasifikaci dat je množství informací, které máme o trénovací sadě dat [6]. V případě, že známe zařazení trénovacích dat do jednotlivých tříd, říkáme, že se jedná o tzv. učení s učitelem. Pokud informaci o příslušnosti dat k dané třídě nemáme, jedná se o učení bez učitele, kterému se také říká shlukování. Principy nástrojů založených na učení bez učitele nejsou v této práci využity.

Modely, na jejichž základě provádíme klasifikaci, lze dále rozdělit do dvou skupin. Rozlišujeme modely diskriminativní a generativní. Diskriminativní modely se zaměřují na oddělení dat v příznakovém prostoru tak, aby co nejlépe rozhodly o příslušnosti data k určité třídě. Diskriminativní modely se nezajímají o rozložení dat v prostoru. Generativní modely se oproti diskriminativním snaží popsat rozdělení dat každé třídy v prostoru a třídy jsou od sebe odděleny na základě tohoto popisu.

Data na obrázku 2.1 lze lineárně oddělit přímkou vedenou mezi jednotlivými třídami. Tato přímka se nazývá rozhodovací linie a její nalezení je hlavním úkolem diskriminativních modelů. Na základě dimenze příznakového prostoru se nemusí jednat o přímku, ale obecně o hyperrovinu. Data poté můžeme klasifikovat podle toho, na jaké straně rozhodovací linie leží. Příkladem diskriminativního modelu jsou Support vector machines (SVM).

V některých případech není klasifikace pomocí diskriminativního modelu vhodná, proto lze využít generativní model. Jak již bylo řečeno, generativní modely se snaží popsat rozdělení dat v prostoru. Popisem rozdělení dat v prostoru se rozumí znalost nebo aproximace funkce rozdělení hustoty pravděpodobnosti. Pokud tuto funkci známe pro každou třídu, lze pro každý klasifikovaný vektor příznaků x určit věrohodnost (likelihood) $p(\mathbf{x}|\omega)$ pro danou třídu ω. Věrohodnost je možné spočítat jako funkční hodnotu funkce rozdělení hustoty pravděpodobnosti.

Bayesovo rozhodovací pravidlo pro minimalizaci chyby

V případě, že dokážeme získat věrohodnost pro každý vstupní vektor příznaků $\mathbf{x}$, můžeme pro klasifikaci dat využít Bayesova rozhodovacího pravidla [6].

Předpokládejme, že klasifikujeme data do J tříd $\omega_1 \dots \omega_j$. Pro každou z těchto tříd a každý vstupní vektor příznaků získáme za pomocí funkce hustoty pravděpodobnosti věrohodnost $p(\omega_1|\mathbf{x}) \dots p(\omega_j|\mathbf{x})$ dané třídy a příznakového vektoru. Dále jsou nám známy apriorní pravděpodobnosti[1] $P(\omega_1) \dots P(\omega_j)$ pro každou třídu ω_j. Z těchto údajů můžeme pomocí Bayesova teorému (rovnice 2.1) vypočítat posteriorní pravděpodobnosti[2] $P(\omega_1|\mathbf{x}) \dots P(\omega_k|\mathbf{x})$ daných tříd.

$$P(\omega_j|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_j)P(\omega_j)}{\sum_{i=1}^{J} p(\mathbf{x}|\omega_i)P(\omega_i)} \tag{2.1}$$

Porovnáním těchto pravděpodobností

$$P(\omega_m|\mathbf{x}) > P(\omega_k|\mathbf{x}) \quad k = 1, \dots, J; \quad k \neq m, \tag{2.2}$$

dostáváme pro vektor $\mathbf{x}$ příslušnou třídu ω_m.

Jmenovatel ve zlomku Bayeova teorému (rovnice 2.1) je shodný pro všechny třídy, proto lze vztah 2.2 přepsat na vztah

$$p(\mathbf{x}|\omega_m)P(\omega_m) > p(\mathbf{x}|\omega_k)P(\omega_k) \quad k = 1, \dots, j; \quad k \neq m, \tag{2.3}$$

který představuje Bayesovo rozhodovací pravidlo pro minimalizaci chyby. Klasifikace dat do tříd pomocí tohoto pravidla zaručuje nejmenší možnou chybu (více viz. [6]).

2.2 Odhad funkce hustoty pravděpodobnosti

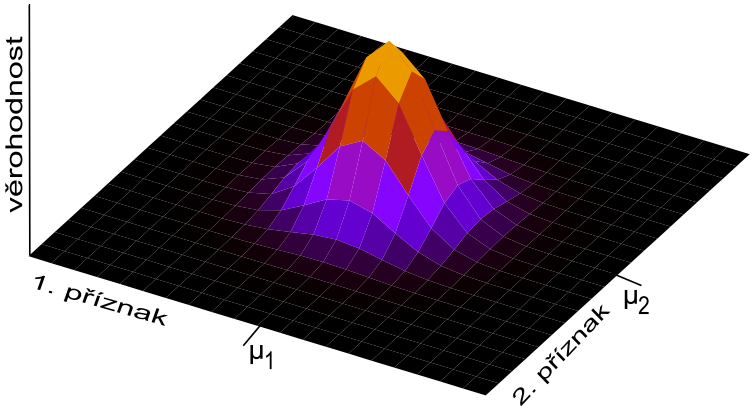
V předcházející části této kapitoly jsme velmi často pracovali s hodnotou věrohodnosti, což je funkční hodnota funkce hustoty pravděpodobnosti pro data dané třídy. Tvar této funkce obvykle neznáme, a tak jej odhadujeme z trénovací sady dat. Pro přesný tvar funkce bychom potřebovali nekonečné množství vzorků trénovacích dat, což v praxi není možné. Proto můžeme vytvářet pouze odhad této funkce. Odhad může být založen na parametrickém nebo neparametrickém modelu. Parametrickým modelem rozumíme to, že funkce hustoty pravděpodobnosti je aproximována nějakým existujícím rozdělením pravděpodobnosti. V takovém případě jsou pomocí trénovacích dat odhadovány parametry tohoto rozdělení. Nejčastěji využívaným rozdělením je Gaussovo (normální) rozdělení.

Gaussovo rozdělení

Gaussovo (normální) rozdělení je charakterizováno střední hodnotou μ a rozptylem σ pro jednorozměrné rozdělení, případně vektorem středních hodnot μ a kovarianční maticí Σ pro

[1] Pravděpodobnost výskytu dané třídy

[2] Pravděpodobnost dané třídy pro daný vektor x



Obrázek 2.2: Gaussovo rozdělení v dvoudimenzionálním příznakovém prostoru. Data nejsou korelována, proto je Gaussovo rozdělení popsáno diagonální kovarianční maticí (je vidět, že není v prostoru nijak natočeno).

vícedimenzionální rozdělení (obrázek 2.2). Obecná rovnice vícerozměrného Gaussova rozdělení lze zapsat jako

$$N(\mathbf{x};\mu,\mathbf{\Sigma}) = \frac{1}{(2\pi)^{\frac{d}{2}}|\mathbf{\Sigma}|^{\frac{1}{2}}}\exp[-\frac{1}{2}(\mathbf{x}-\mu)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\mu)], \tag{2.4}$$

kde d představuje počet dimenzí.

Odhad parametrů (vícerozměrného) Gaussova rozdělení se provádí pomocí Maximum likelihood odhadu (rovnice 2.5) [6] z trénovacích dat.

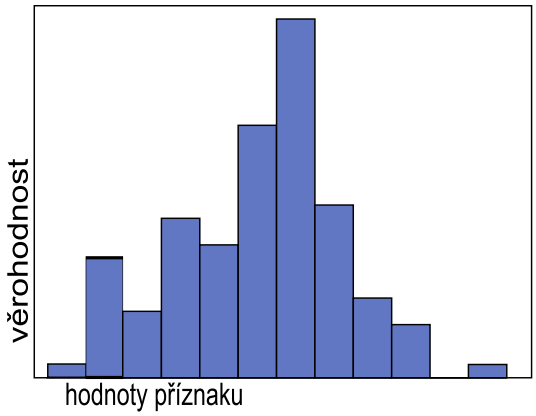
$$\mu = \frac{1}{N}\sum_{k=1}^{n}\mathbf{x_k}, \quad \mathbf{\Sigma} = \frac{1}{N}\sum_{k=1}^{n}(\mathbf{x_k}-\mu)(\mathbf{x_k}-\mu)^T. \tag{2.5}$$

Pokud nejsou data, která popisujeme Gaussovým rozdělením, navzájem korelováná (statisticky závislá), lze Gaussovo rozdělení vyjádřit pomocí diagonální kovarianční matice. Tímto se zjednoduší výpočet funkční hodnoty vícerozměrné Gaussovy křivky. Můžeme využít toho, že výsledná funkční hodnota je v takovém případě rovna součinu funkčních hodnot všech jednorozměrných rozdělení, ze kterých je vícerozměrné rozdělení složeno. Další možností je počítat výslednou funkční hodnotu vícedimenzionálního Gaussova rozdělení podle rovnice 2.4, která v případě diagonální kovarianční matice degraduje na rovnici

$$N(\mathbf{x};\mu,\mathbf{\Sigma}) = \frac{1}{\sqrt{(2\pi)^d\prod_{i=1}^{d}\sigma_i^2}}\exp[-\frac{1}{2}\sum_{i=1}^{d}\frac{(\mathbf{x}_i-\mu_i)^2}{\sigma_i^2}]. \tag{2.6}$$

Tato rovnice lze dále upravit na tvar násobení jednorozměrných Gaussovských rozdělení.

Často data nepopisujeme jedním Gaussovým rozdělením, ale jejich směsí. V takovém případě mají jednotlivá Gaussovská rozdělení své váhy a celková věrohodnost je rovna



Obrázek 2.3: Jednorozměrný normalizovaný histogram, kde hodnoty jednotlivých polí histogramu představují věrohodnost

součtu váhovaných věrohodností jednotlivých rozdělení. Pro trénování se používá algoritmus Expectation Maximization.

Histogram

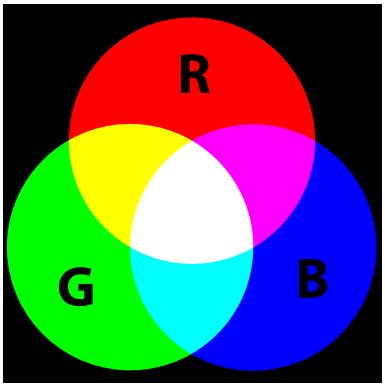
Pokud nechceme použít parametrický model pro vyjádření funkce hustoty pravděpodobnosti, můžeme využít často používaný neparametrický způsob aproximace této funkce, histogram (obrázek 2.3). Histogram vytvoříme na základě trénovací sady dat a normalizujeme ho, aby hodnoty jednotlivých polí histogramu byly v rozsahu $0 \ldots 1$. Poté již pro každý příznakový vektor příchozích dat zjistíme, do jakého pole histogramu patří, a hodnotu v tomto poli budeme považovat za věrohodnost příslušející danému vektoru příznaků. Nevýhodou tohoto přístupu je, že příznaky mohou ve všech dimenzích nabývat pouze diskrétních hodnot nebo je musíme diskretizovat.

2.3 Barevné modely

Pro segmentaci obrazu za účelem nalezení oblastí barvy kůže je třeba učinit rozhodnutí, pomocí jakého barevného modelu budeme jednotlivé snímky videa reprezentovat. Pro tento typ úkolu není vhodný každý barevný model.

Ming-Hsuan Yang a Narendra Ahuja [7] používají barevný model CIE LUV, který jsem blíže nezkoumal. Yuanxin Zhu a jeho tým [9] se při segmentaci barvy kůže zaměřil na barevný model HSV (HSI), kterým se v této kapitole budu zabývat později. Jonathan Alon se svým týmem [1] použil jako barevný model normalizovaný RGB, jenž bude v této kapitole také popsán. V této části práce vycházím především ze zdroje [5].

V klasickém RGB modelu je barva vyjádřena aditivně podílem červené, zelené a modré složky (viz. obr. 2.4). Tento model je používán pro zobrazení dat na monitor a další podobná zařízení. Pro segmentaci barvy kůže má model RGB několik nevýhod. Mezi hlavní nevýhody



Obrázek 2.4: RGB model – aditivní reprezentace barvy. Model nevhodný pro reprezentaci příznaků kvůli velké korelaci jednotlivých složek.

patří vysoký stupeň korelace (statistické závislosti) mezi jednotlivými složkami barevného modelu, čímž je snižována užitečná informace, kterou z této reprezentace získáme. Další nevýhodou je sloučení informace o barvě a jejím jasu. Tímto se tento model stáva velmi náchylným na změnu světelných podmínek.

Normalizovaný RGB model

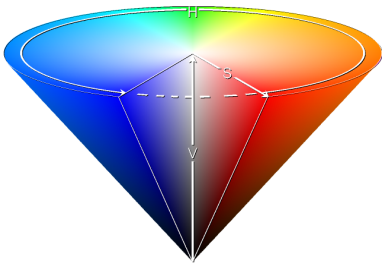
Podstatně vhodnější model pro segmentaci barvy lidské kůže je model normalizovaného RGB. Barva je stejně jako v klasickém RGB reprezentována třemi složkami, ale každá složka je normalizována tímto způsobem:

$$R_n = \frac{R}{R+G+B} \quad G_n = \frac{G}{R+G+B} \quad B_n = \frac{B}{R+G+B}. \tag{2.7}$$

Složky R,G,B reprezentují původní RGB model a složky označené dolním indexem n označují normalizované hodnoty. Součet všech hodnot popisujících barvu v tomto barevném prostoru je vždy roven jedné, z toho vyplývá, že třetí složka (modrá) nenese žádnou užitečnou informaci. To je výhodou především proto, že se tímto způsobem redukuje dimenze prostoru, ve kterém lze hodnoty v normalizovaném RGB zobrazit. Složky R a G se nazývají „čisté barvy (pure colors)“ [5]. Další výhodou tohoto modelu je, že závislost složek R a G na jasu barvy je snížena normalizací. Důležitá je také jednoduchost převodu do tohoto modelu z klasického RGB. Za další výhodu lze považovat to, že „model normalizovaného RGB je pro matné povrchy, pokud zanedbáme ambientní osvětlení, za určitých podmínek invariantní ke změně orientace povrchu vzhledem ke zdroji světla“ [5].

HSV model

Další možností, jak reprezentovat informaci o barvě, je využít modelu HSV (viz. obr. 2.5). Tento model je svou podstatou velmi blízký člověku, protože popisuje barvu způsobem, který je založen na lidském vnímání barvy. Barva je opět reprezentována třemi složkami.



Obrázek 2.5: HSV model – kuželová reprezentace. Reprezentace podobná vnímání člověka. Explicitní oddělení jasu barvy.

H představuje převládající barvu (tón barvy). S popisuje sytost barvy a V reprezentuje jas barvy. Výhodou této reprezentace je explicitní oddělení informace o vlastní barvě a jejím jasu, což je pro segmentaci barvy lidské kůže velmi vhodné. Pro HSV platí, stejně jako pro normalizované RGB, že je pro matné povrchy za určitých podmínek invariantní ke změně orientace povrchu vzhledem ke zdroji světla. Složky HSV modelu lze spočítat jako

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

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

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

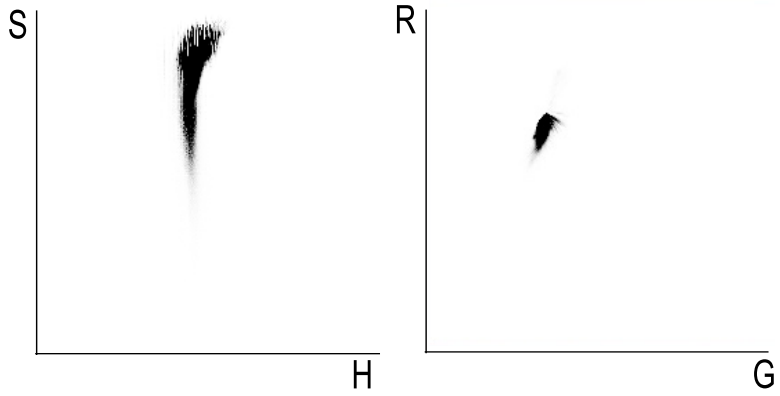
Z výpočtu vyplývá, že mezi nevýhody modelu HSV patří nespojitost složky H [5]. Další nevýhodou může být složitost převodu z klasického RGB.

2.4 Segmentace obrazu

Pro klasifikaci gest rukou je nejdříve nutné ruce v obraze detekovat. Až poté je možné získat o rukou informace, které později využijeme jako příznaky pro klasifikátor gest.

Nejčastějším přístupem k segmentaci obrazu za účelem nalezení rukou je využití specifičnosti barvy lidské kůže. Dobrou vlastností lidské kůže z pohledu její detekce je, že se pro různé osoby liší především v jasu barvy a hlavní barevné složky jsou velmi podobné. Z tohoto důvodu je vhodné pro reprezentaci barvy lidské kůže volit takové barevné modely, kde je jas barvy explicitně oddělen od informace o čisté barvě.

Možnosti reprezentace barvy a jejich vhodnost pro tento druh problému byly diskutována v kapitole 2.3. Po výběru vhodného barevného modelu pro reprezentaci barvy kůže je formulována detekční úloha, jejímž cílem je detekce barvy kůže. Častým řešením této úlohy (používá např. [1] a další) je vytvoření histogramu na základě pozitivních vzorků lidské kůže (obrázek 2.6). Pro lepší generalizaci tohoto modelu lze ze získaného histogramu



Obrázek 2.6: Histogram barvy lidské kůže. V prostoru HSV vyjádřen pouze složkami HS(vlevo). Histogram v normalizovaném RGB prostoru, barva vyjádřena složkami GR (vpravo)

vytvořit Gaussovo rozdělení popisující funkci hustoty pravděpodobnosti lidské kůže (obrázek 2.7). Pro tento přístup je však třeba většího množství trénovacích dat.

Kromě modelu popisujícího barvu kůže je třeba vytvořit model popisující negativní vzorky této detekční úlohy, což jsou pixely barvy neodpovídající barvě kůže. Existuje několik možností, jak k tomuto problému přistupovat. V případě, že detekci barvy kůže provádíme pouze v jednom prostředí, můžeme ze snímků neobsahujících lidskou kůži vytvořit podobný model, který jsme vytvořili pro barvu lidské kůže.

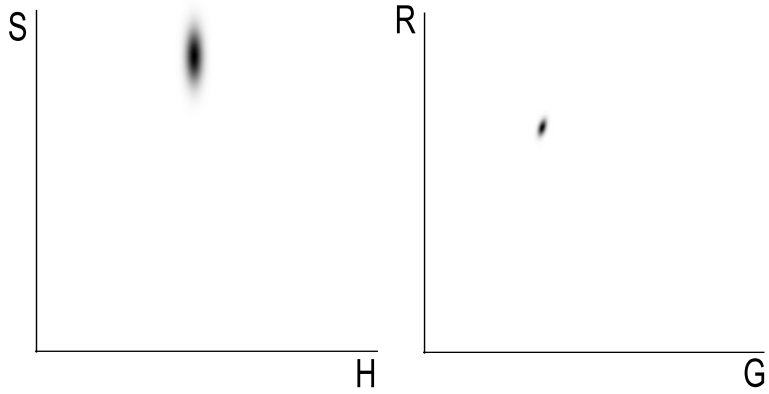
Pokud detekci barvy lidské kůže neprovádíme v jednom prostředí, je možné vytvořit obecný model okolí. Tímto modelem může být „histogram“ udávající pro všechny body barevného prostoru stejnou hodnotu věrohodnosti. Tímto zobecněním nedochází k velkým ztrátám na kvalitě detekce.

Na základě modelu kůže a okolí lze detekovat barvu kůže pomocí Bayesova teorému (rovnice 2.1). Pro rychlejší detekci lidské kůže je možné vytvořit pro celý barevný prostor vyhledávací tabulku (look-up table), jejíž hodnoty určují, zda pixel dané barvy je nebo není kůže.

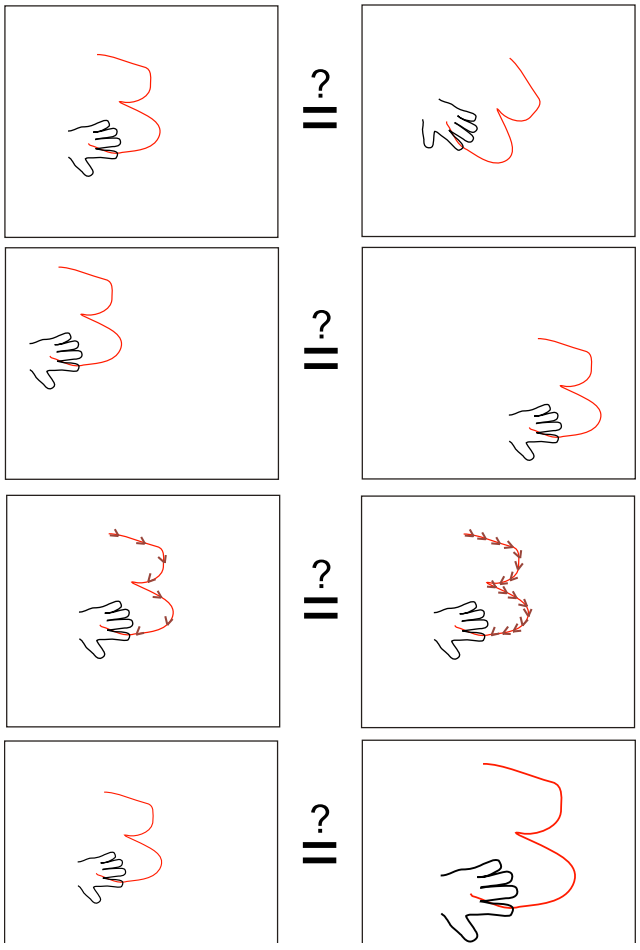
Pro další postup v segmentaci obrazu je vhodné vědět, jaké oblasti barvy lidské kůže se mohou ve snímku vyskytnout. V případě, že se nejedná o samotnou ruku, je třeba rozlišit mezi rukou a hlavou, případně mezi dvěma rukama a hlavou. Za předpokladu, že scénu považujeme za statickou, lze ruku od hlavy resp. druhé nepohybující se ruky odlišit tak, že hledáme pouze pohybující se oblasti lidské kůže. Pohybující se oblast lidské kůže lze poté považovat za ruku provádějící gesto.

2.5 Parametrizace

Jak již bylo řečeno dříve, klasifikace je obecně prováděna pomocí abstrahované informace o daném jevu. Pokud bychom například pro klasifikaci libovolného jevu nasnímaného



Obrázek 2.7: Gaussovo rozdělení reprezentující barvu lidské kůže. Dekorelovaná data v prostoru HSV (vlevo). Korelovaná data v normalizovaném RGB prostoru (vpravo)



Obrázek 2.8: Možné požadavky na invariantnost příznaků. Shora: invariantnost vůči natočení, invariantnost vůči posunu, invariantnost vůči rychlosti gesta, invariantnost vůči vzdálenosti od kamery (velikosti ruky) nebo také invariantnost vůči velikosti gesta

kamerou použili jako příznaky informaci o barvě každého pixelu daného snímku, vedlo by to při vyšších rozměrech snímku k velkému množství příznaků a úspěšnost takového klasifikátoru by byla spekulativní. Při vysokém rozlišení by také mohlo být obtížné dosáhnout zpracování v reálném čase. Z těchto důvodů se snažíme o sledovaném jevu získat co nejinformativnější příznaky. Zároveň se snažíme, aby těchto příznaků bylo při zachování kvality klasifikace co nejméně.

V kapitole 2.4 byly popsány možnosti detekce barvy kůže v obraze. Jako příznaky byla použita informace o barvě daného pixelu. V části 2.3, kde byly popsány barevné modely, je řečeno, že se snažíme snížit dimenzi prostoru příznaků vypouštěním barevné složky, která neobsahuje užitečnou informaci nebo vyjadřuje jas barvy.

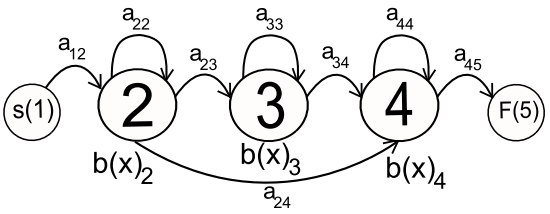
Další vlastností dobře zvolené parametrizace je, že volíme příznaky, které nejsou vzájemně korelovány. Tuto podmínku není vždy jednoduché splnit. Případnou korelaci příznaků je možné řešit transformacemi vektoru příznaků v příznakovém prostoru (např. natočení dat v prostoru tak, aby již příznaky nebyly lineárně závislé). Touto problematikou se však nebudu dále zabývat (více v [2]).

Ruku provádějící gesto lze popsat několika způsoby. Důležité je, jaké typy gest budeme klasifikovat. Nejdůležitější informací pro většinu typů gest je poloha dlaně ruky, která gesto provádí. Za bod určující polohu dlaně je obvykle považován střed geometrického obrazce, jímž dlaň opíšeme. Běžně je pro tyto účely využívána elipsa [1] nebo polygon. Pokud budeme oblast dlaně chápat jako elipsu, můžeme využít další parametry elipsy jako příznaky. Mezi příznaky pak může patřit úhel natočení elipsy, její velikost, poměr hlavní a vedlejší osy a další.

Po vybrání příznaků je třeba určit, jakým způsobem je budeme reprezentovat. Volba vhodné reprezentace příznaků je důležitá především ve vztahu k invariantnosti (nezávislosti) příznaků vůči různým jevům (viz. obrázek 2.8). Je třeba rozhodnout, jaká gesta budeme považovat za shodná. Při klasifikaci gest je žádoucí, aby gesta prováděná v různých místech obrazu klasifikátor označil za shodná. Příznaky tedy musí být invariantní vůči posunutí. Polohu ruky, stejně jako úhel natočení, není tedy vhodné reprezentovat absolutními hodnotami pozice resp. úhlu, ale jejich diferencemi (rozdíl dvou po sobě jdoucích příznaků stejného druhu). Stejným způsobem lze zajistit invarianci dalších příznaků (velikost elipsy, poměr os) vzhledem k velikosti ruky.

Pokud se rozhodneme reprezentovat příznaky ve tvaru diference, musíme zvolit referenční bod v příznakovém prostoru, který spolu s aktuálním bodem umožní její využití. Tímto bodem může být bod v příznakovém prostoru odpovídající předcházejícímu vektoru příznaků.

Při snaze zajistit invariantnost příznaků vůči různým vlivům je třeba zvážit, jaký druh informace o gestu chceme zachovat. Pokud bychom se snažili zajistit invariantnost vůči některému vlivu neuváženě, může dojít ke zkreslení informace o daném gestu. Jako stejná gesta budou klasifikovány i takové datové vzorky, které spolu nesouvisí. Příkladem může být invariantnost vůči rotaci gesta v prostoru (viz. obrázek 2.8).



Obrázek 2.9: Levopravý Skrytý Markovův model s 5 stavy z nichž 3 jsou vysílací. Stav S je první (počáteční) nevysílací stav. Stav F je koncový nevysílací stav. Model umožňuje přeskočení jednoho následujícího vysílacího stavu.

Před vlastním trénováním klasifikátoru je třeba zvážit, jaké příznaky použijeme a jakou reprezentaci zvolíme, abychom dosáhli co nejpřesnějšího a nejjednoduššího popsání gesta.

2.6 Skryté Markovovy modely

Problém klasifikace gest již není problémem klasifikace jednoho vektoru příznaků, jak tomu bylo například u detekce kůže. Gesto je popsáno několika vektory příznaků, které jdou v určitém pořádí za sebou. Jedná se tedy o klasifikaci sekvencí vektorů příznaků. V této části práce vycházím především ze zdrojů [3] a [4].

Sekvence lze klasifikovat několika nástroji. Příkladem může být neuronová síť ([7]) nebo nástroj využívající algoritmus DTW ([9] a [1]). Dalším použitelným nástrojem je Skrytý Markovův model (HMM[3]). HMM je založen na stochastickém konečném automatu (Markovův model), který se od normálního konečného automatu liší tím, že hrany definující přechody mezi stavy jsou pravděpodobnostně ohodnoceny. Markovovy modely však selhávají v případě, že se v sekvenci vyskytne vektor příznaků, pro který z daného stavu není definován přechod. Tento problém řeší HMM tím, že každý stav obsahuje klasifikátor (nazývá se vysílací funkce nebo vysílací model), jenž produkuje věrohodnost toho, že daný vektor „je vyslán“ tímto stavem. Tento mechanismus zabezpečuje, že model neselže ani v případě, kdy se v sekvenci vektorů vyskytne šum, chyba apod. V takovém případě je pouze vyprodukována nízká věrohodnost daného vektoru příznaků. Existují dva druhy HMM, kontinuální a diskrétní. Diskrétní HMM pracují s diskrétní abecedou vstupních symbolů. Těmi se dále nebudu zabývat. Vstupem kontinuálních HMM jsou vektory příznaků, jejichž hodnoty mohou nabývat libovolných hodnot z oboru reálných čísel. Obvykle má každý HMM právě jeden počáteční a koncový stav. Do počátečního a koncového stavu bývá umožněn přechod pouze z jednoho stavu (obrázek 2.9). Skrytý Markovův model je definován:

- Množinou stavů $\mathbf{S_1} \ldots \mathbf{S_n}$, kde první a poslední stav jsou tzv. „nevysílací“
- Množinou přechodových pravděpodobností mezi stavy $\mathbf{a_{1,i}} \ldots \mathbf{a_{i,n}}$ pro $i = 2 \ldots n-1$

[3] Hidden Markov model

- Množinou vysílacích funkcí $\mathbf{b(x)_2} \ldots \mathbf{b(x)_{n-1}}$

Nevysílací stavy neobsahují žádné vysílací funkce a slouží pouze jako počáteční a koncový stav modelu.

Jako vysílací model může sloužit libovolný klasifikátor, jehož výstupem je věrohodnost vstupního vektoru příznaků. Mezi používané klasifikátory patří neuronové sítě nebo Gaussovo rozdělení (případně směs Gaussovských rozdělení). Pro rozpoznávání běžných druhů sekvencí se používají tzv. „levopravé modely“, což jsou modely, kde je povolen postup pouze od počátečního stavu ke koncovému stavu a není povoleno navracení. Například u gesta předpokládáme průběh od počátku do konce, navracení nedává příliš smysl.

Dekódování modelů

Odezvou Skrytého Markovova modelu na vstupní sekvenci vektorů příznaků $\mathbf{O}$ je věrohodnost této sekvence pro daný model. Délku sekvence vektorů příznaků označíme T. Skrytou informací v HMM je rozložení vektorů příznaků mezi stavy. Existuje několik způsobů dekódování Skrytých Markovových modelů. Prvním způsobem je algoritmus, který produkuje Baul-Welchovu věrohodnost, což je součet věrohodností počítaných přes všechny cesty v modelu. Druhou možností je výpočet Viterbiho věrohodnosti, což je věrohodnost nejlepší cesty v modelu. Věrohodnost generování sekvence $\mathbf{O}$ po cestě X modelem M (předpokládá se jeden přechod ze vstupního stavu 1) lze spočítat jako:

$$p(\mathbf{O}, X|M) = a_{x(1)x(2)} \prod_{t=1}^{T} b_{x(t)}(\mathbf{o}_t) a_{x(t)x(t+1)}, \tag{2.8}$$

kde $\mathbf{o}_t$ označuje jeden vektor, jehož pořadí v celé sekvenci $\mathbf{O}$ je rovno t. Tato informace je často interpretována tak, že daný vektor $\mathbf{o}_t$ byl vyslán v čase t.

Vyhodnocení všech cest modelem a následné porovnání všech věrohodností je však časově i paměťově náročná operace, proto se pro vyhodnocení Viterbiho věrohodnosti používá algoritmu „token passing“ [3]. Token je označení kontejneru, ve kterém se průběžně počítá věrohodnost dané cesty. Aby se předešlo problémům s přetečením hodnoty tokenu a došlo k zefektivnění výpočtů, přechází se z rovnice 2.8 na její ekvivalent v logaritmické podobě:

$$\log p(\mathbf{O}, X|M) = \log a_{x(1)x(2)} + \sum_{t=1}^{T} \log b_{x(t)}(\mathbf{o}_t) + \log a_{x(t)x(t+1)}. \tag{2.9}$$

Algoritmus „token passing“ pro výpočet Viterbiho věrohodnosti lze popsat v několika bodech:

1. Do každého vstupního stavu vlož token s hodnotou 0
2. Pro každý čas (vektor) $t = 1 \ldots T$ prováděj kroky 3 a 4
3. V každém stavu i, který obsahuje token, projdi všechny provázané stavy j a v nich nastav hodnotu tokenu na token v předcházejícím stavu plus $\log a_{ij} + \log b_j[o(t)]$

4. Pokud se již v některém provázaném stavu vyskytuje token, ponech jen ten s vyšší hodnotou

5. Do posledního stavu N vlož všechny tokeny ze stavů i, které jsou s ním propojeny a k těmto tokenům přičti $\log a_{iN}$

6. Vyber token s nejvyšší hodnotou v posledním stavu. Tato hodnota je rovna Viterbiho věrohodnosti

Trénování modelů

Trénování HMM, které používají jako vysílací funkci Gaussovo rozdělení, se provádí pomocí Baum-Welchova algoritmu [4]. Tento algoritmus využívá mechanismu pro trénování modelů založených na směsi Gaussovských rozdělení, který se nazývá Expectation Maximization. Principem Baum-Welchova algoritmu je rozdělení vektorů dané sekvence mezi stavy a následné upravení parametrů směsi nebo jediného Gaussova rozdělení v každém stavu na základě přiřazených vektorů. Pro rozdělení vektorů mezi stavy modelu se využívá částečných dopředných a zpětných věrohodností.

Kapitola 3

Návrh a struktura systému

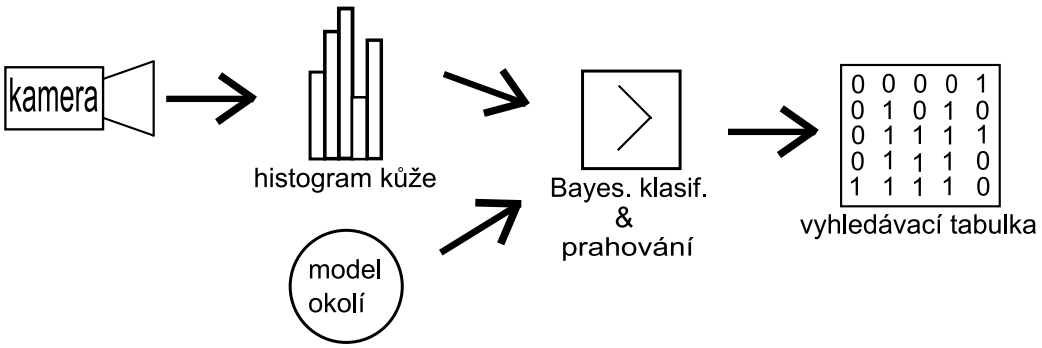
V této kapitole se budu zabývat popsáním struktury výsledného systému na algoritmické úrovni. Zaměřím se na všechny oblasti, které jsem již teoreticky popsal. Nejdříve vysvětlím, jakým způsobem získávám z jednoho snímku videa informace o dlani ruky. Poté se zaměřím na vlastní získávání příznaků a nakonec vysvětlím princip fungování rozpoznávání gest v reálném čase a rozpoznávání gest obecně.

3.1 Nalezení oblasti ruky

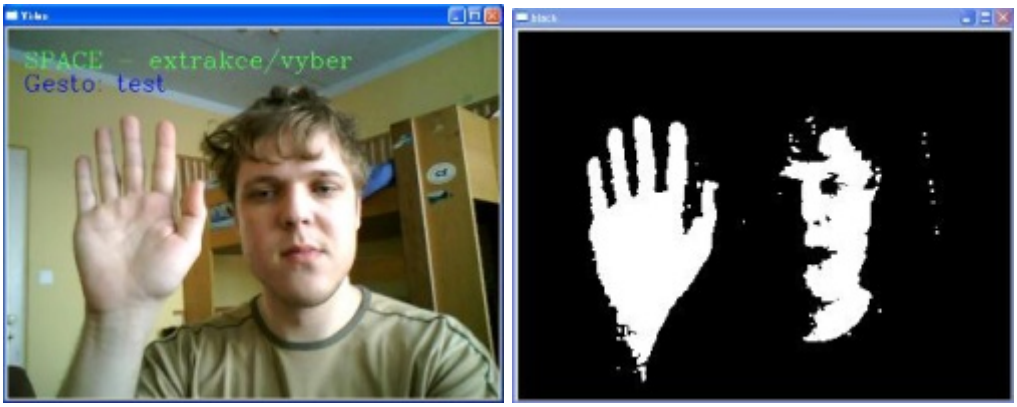
V této části se budu zabývat nalezením oblastí s barvou kůže a následnou identifikací ruky.

Pro nalezení oblasti, kterou označíme za ruku, je nejdříve třeba nalézt všechny oblasti barvy kůže. K tomuto účelu využívám vyhledávací tabulku vytvořenou za pomoci histogramu barvy kůže. Proces tvorby vyhledávací tabulky je popsán v kapitole 2.4.

Ve snímku je vyznačena oblast, do které je třeba vložit ruku, a poté je možné spustit proces tvorby histogramu. Výřez snímku s oblastí ruky je převeden do zvoleného barevného prostoru (normalizované RGB nebo HSV) a následně je na základě hodnot pixelů počítán histogram. Jsou využity jen ty barevné složky modelu, které mají pro detekci lidské kůže



Obrázek 3.1: Schéma zobrazující proces tvorby vyhledávací tabulky, která je použita pro detekci barvy kůže.



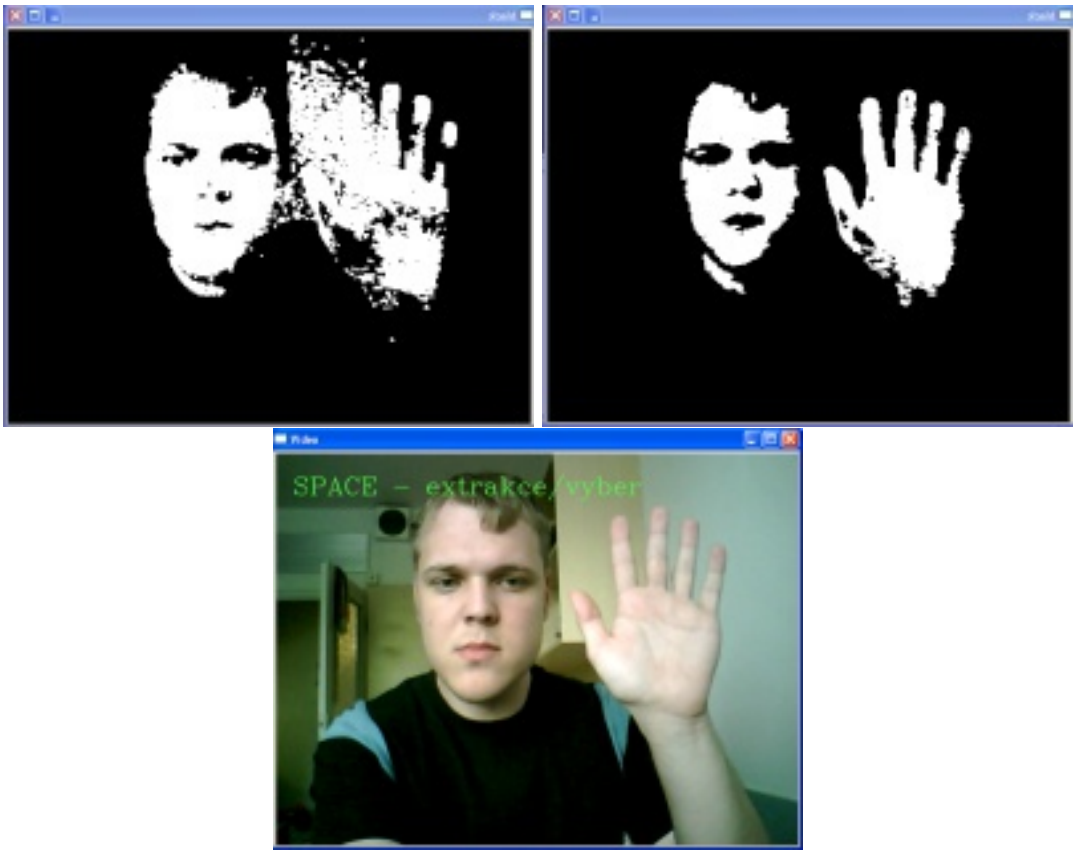
Obrázek 3.2: Převod snímku videa (vlevo) na binární obraz (vpravo). Bílé oblasti v binárním obraze jsou oblasti detekované jako kůže. Je použit barevný prostor normalizovaného RGB.

význam (viz. 2.3). Pokud je tvořen histogram na základě hodnot z prostoru normalizovaného RGB, jsou tyto hodnoty nejprve převedeny do diskrétních hodnot v rozsahu $0 \ldots 255$. Parametry modelu okolí jsou nastaveny tak, aby se věrohodnost produkovaná tímto modelem pro všechny body barevného prostoru sčítala do jedničky. Apriorní pravděpodobnosti jednotlivých tříd jsou odhadnuty na $0,3$ pro kůži a $0,7$ pro okolí.

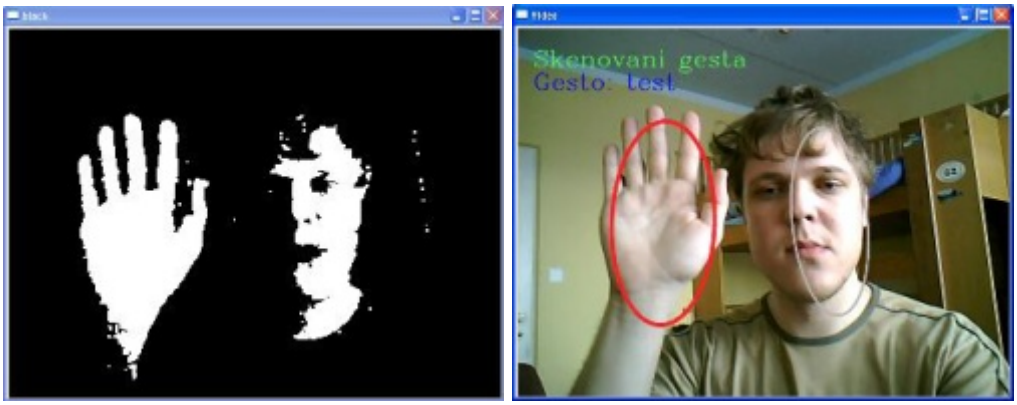
Vyhledávací tabulku lze z histogramu získat dvěma způsoby. Prvním je normalizace histogramu a následné vyhodnocení Bayesova teorému pro model kůže (histogram) a model okolí. Získaná pravděpodobnost je porovnána s prahem $0,7$ a hodnoty větší nebo rovny tomuto prahu jsou dále považovány za kůži. Ve vyhledávací tabulce jsou body barevného prostoru považované za kůži označeny jedničkou, ostatní body jsou označeny nulou.

Druhý způsob tvorby vyhledávací tabulky se od prvního liší tím, že pro získání věrohodnosti lidské kůže není využit samotný histogram, ale data z histogramu jsou využita pro tvorbu Gaussova rozdělení. Pro barevný model HSV, kde nedochází k výrazné korelaci mezi složkami H a S, je využito Gaussovo rozdělení s diagonální kovarianční maticí. Pokud pracujeme v barevném prostoru normalizovaného RGB, využívá se Gaussovo rozdělení s plnou kovarianční maticí, protože použité složky R a G jsou vzájemně korelovány. Tímto rozlišením se snažím dosáhnout jednoduššího a rychlejšího modelu tam, kde je to možné. Pokud je histogram tvořen z menšího množství dat (např. 45 snímků 100×100 pouze z jedné oblasti dlaně), nepřináší využití Gaussova rozdělení lepší výsledky než použití čistého histogramu. V případě, že pro tvorbu histogramu použijeme více dat (histogram se tvoří z většího množství snímků např. z celé dlaně a paže), přináší využití Gaussova rozdělení výhodu větší odolnosti proti šumu a vyšší míru generalizace (viz. obrázek 3.3). Schéma tvorby vyhledávací tabulky je vidět na obrázku 3.1.

Za pomoci vyhledávací tabulky je již možné provést detekci barvy kůže v obraze. Každý pixel snímku získaného z kamery je na základě své pozice v barevném prostoru a odpovídající hodnoty ve vyhledávací tabulce zařazen buď do třídy kůže, nebo okolí. Tímto způsobem je získán binární obraz, ve kterém jsou vyznačeny oblasti kůže (obrázek 3.2). Na binární



Obrázek 3.3: Dva binární obrazy a původní snímek videa. Je využit prostor HSV. Histogram byl vytvořen z většího množství dat. Pro tvorbu binárního obrazu vlevo byl použit pouze histogram barvy kůže a v obraze je zřetelně vidět šum. Binární obraz vpravo je tvořen za pomocí Gaussova rozdělení s plnou kovariantční maticí a je vidět, že je proti šumu více odolný.



Obrázek 3.4: Binární obraz (vlevo) a snímek se zakreslenými elipsami (vpravo). Červená elipsa označuje pohybující se ruku. Je použit barevný prostor normalizovaného RGB.

obraz je poté aplikována morfologická operace otevření (eroze a dilatace), jejímž cílem je odstranění šumu v obraze.

Dalším krokem při segmentaci obrazu je nalezení hranic spojitých oblastí a následné opsání těchto oblastí elipsou. K tomuto účelu jsou použity prostředky knihovny OpenCV. Je vybráno vždy tolik největších oblastí, kolik uživatel definuje (např. dvě, pokud předpokládá, že se bude snímku ve vyskytovat hlava a jedna ruka). Elipsa je podle získaných parametrů kreslena do snímku (obrázek 3.4).

Problémem je hledání kontextu mezi jednotlivými snímky videa a sledováním objektu dlaně ruky. Tato problematika bude popsána níže. Jakmile jsou určeny středy oblastí (elips), je jako ruka označena oblast, jejíž střed se napříč snímky pohybuje. Scéna, která je kamerou snímána, je považována za statickou, a proto se předpokládá existence pouze jednoho pohybujícího se objektu barvy kůže, kterým je ruka provádějící gesto.

Při tvorbě histogramu, vyhledávací tabulky a následné tvorbě binárního obrazu je důležitá kvalita videa. Kamera, kterou jsem při práci používal, prováděla automatické vyvažování bílé barvy, což vedlo k šumu v binárním obraze. Proto bylo nutné před vlastním používáním takové kamery automatické vyvažování bílé barvy vypnout.

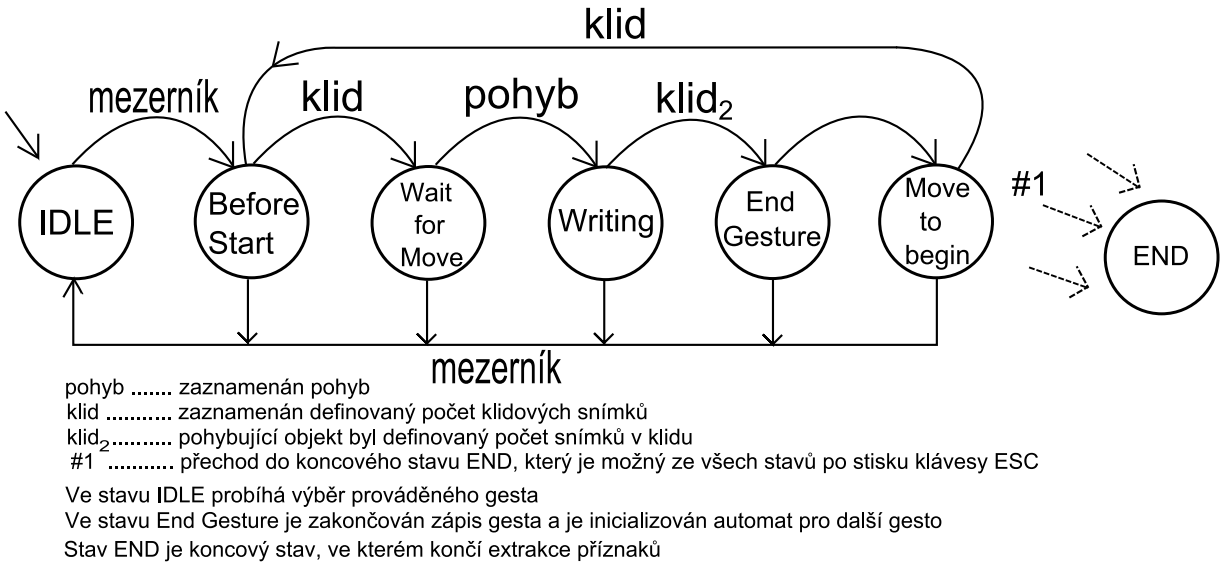
3.2 Získávání příznaků

V této sekci rozeberu způsob, jakým jsou z informací o jednotlivých oblastech lidské kůže získávány příznaky pro klasifikátor gest.

Oblast lidské kůže je opsána elipsou a parametry elipsy jsou použity jako příznaky. Těmito parametry jsou střed, úhel natočení a velikost elipsy. Dále je jako příznak použit poměr vedlejší a hlavní poloosy elipsy. Za parametr určující velikost elipsy považuji obdélník tvořený hlavní a vedlejší poloosou. Úhel natočení elipsy je reprezentován v rozsahu $-\frac{\pi}{2} \ldots \frac{\pi}{2}$, což je úhel, ve kterém je člověk přirozeně schopen natočit dlaň. Příznaky jsou zvoleny tak, aby bylo možné popsat několik způsobů natočení dlaně a tím zvětšit množinu gest, která bude možné rozpoznat.

Pokud je třeba ve snímku detekovat více než jednu oblast barvy kůže (např. ruku a obličej), vyvstává problém správného přiřazení odpovídajících oblastí v rámci sekvence snímků. Tento problém, kterému se říká sledování objektu (object tracking), je možné řešit několika způsoby. V [7], kde je třeba sledovat napříč snímky více objektů, je tento problém formulován jako problém porovnávání a transformace grafů. Každý snímek je reprezentován jako graf, kde objekt představuje uzel grafu a sousedství objektů je reprezentováno hranami grafu. Rozhodl jsem se přistupovat k tomuto problému jednoduše. Za odpovídající oblasti napříč snímky označuji ty oblasti, jejichž středy jsou si v rámci dvou sousedních snímků nejblíže.

Systém může selhat v několika případech. Prvním z nich je situace, kdy dojde k překrytí oblastí barvy kůže a ty jsou následně detekovány jako jedna oblast. Při opětovném rozdělení může dojít k chybě při přiřazování odpovídajících oblastí. Původní pohybující se oblast,



Obrázek 3.5: Konečný automat reprezentující proces získávání příznaků.

která překryla jinou, může být s touto oblastí při oddělení zaměněna. Další problém vzniká, pokud se ve snímku nevyskytuje požadovaný počet oblastí barvy kůže (ruka, obličej). V takovém případě je jako chybějící oblast označena další největší oblast, která je rozpoznána jako kůže. Z těchto dvou problémů vyplývají dvě omezující podmínky pro použití systému. V celém průběhu extrakce příznaků je nutné nepřejíždět rukou přes obličej nebo druhou ruku a zároveň je třeba, aby se v záběru kamery vždy vyskytovaly všechny požadované oblasti barvy kůže. Posledním doporučením pro správné fungování systému je skutečnost, že člověk, kterého kamera snímá, musí mít na sobě oblečení s dlouhým rukávem. Tento oděv zakryje paže a je tak detekována pouze dlaň.

Proces extrakce příznaků lze popsat konečným automatem (obrázek 3.5). Většina přechodů mezi stavy je podmíněna zaznamenáním určitého počtu klidových snímků (tento počet lze nastavit). Klidovým snímkem je míněn snímek, ve kterém není zaznamenán pohyb žádné oblasti kůže. Za pohyb se považuje přesun středu oblasti mezi snímky o hodnotu větší, než je definovaný práh. Pomocí klidových snímků je detekován počátek a konec gesta. Příznaky jsou zapisovány do souboru pouze, pokud se automat nachází ve stavu „writing“. Tento stav je ukončen, jakmile je sledovaný objekt (ruka) definovaný počet snímků v klidu. Celý proces extrakce příznaků se skládá z několika kroků. Pokud je aplikace spuštěna s požadavkem na rozpoznávání gest v reálném čase, proces rozpoznávání je spuštěn ve stavu „End Gesture“ a jeho výsledek je zobrazen.

Během vlastní extrakce příznaků jsou uchovávány informace o oblastech barvy kůže vždy pro aktuální snímek a snímek předcházející. Poté, co jsou určeny odpovídající oblasti, jsou od sebe jednotlivé dvojice příznaků odečteny a výsledné diference jsou poté použity pro vlastní klasifikaci gest. Použitím diference se v případě středu elipsy a úhlu natočení dosáhne invariantnosti vůči posunu. U velikosti a poměru os tímto odečtením dochází

ke zvýšení odolnosti vůči proměnlivé velikosti ruky nebo vzdálenosti od kamery. Pro vyšší míru odolnosti vůči těmto vlivům by bylo vhodné reprezentovat rozdíl těchto příznaků pro sousední snímky procentuálně. Tento fakt jsem si však uvědomil až po nasbírání datové sady. I poté by však mohlo docházet k problémům, protože velikost rozpoznané oblasti dlaně se i pro jednoho člověka často velmi liší.

Takto získané příznaky však nejsou invariantní vůči rychlosti prováděného gesta. Tohoto jevu by šlo dosáhnout tak, že by pozice dlaně nebyla reprezentována posunutím od předchozí pozice, ale úhlem, v jakém byl posun proveden.

3.3 Klasifikace gest

V této části popíši princip klasifikace gest a zpracování sekvencí vektorů příznaků před jejich vlastním použitím pro trénování HMM nebo testování. Pro trénování Skrytých Markovových modelů používám toolkit HTK, kterému se budu více věnovat v následující kapitole. Tento toolkit je také používán pro dekódování modelů při vyhodnocení experimentů. Pro rozpoznávání gest v reálném čase jsem vytvořil vlastní dekodér, který využívá Viterbiho algoritmu. Využívám Skryté Markovovy modely s jedním Gaussovým rozdělením v každém stavu, toto Gaussovo rozdělení má diagonální kovarianční matici.

Před vlastním trénováním modelů jsou sekvence vektorů zkontrolovány a jsou vyloučeny sekvence, které nejsou správně ukončeny. Dále jsou vyloučeny sekvence, ve kterých dochází k velmi velkým posunům pozice dlaně mezi snímky. Vysoká hodnota příznaku označující posun dlaně obvykle znamená, že v průběhu zaznamenávání příznaků došlo k šumu nebo chybě a jako dlaň byla rozpoznána jiná oblast barvy kůže. Poslední typy sekvencí, které jsou vyloučeny, jsou příliš krátké sekvence. Pokud uživatel ve stavu programu, který čeká na pohyb ruky, provede výrazný pohyb hlavou, dochází k tomu, že hlava je označena jako pohybující se ruka. Tyto sekvence jsou však obvykle velmi rychle ukončeny, a proto jsou z použitých dat krátké sekvence vyloučeny. Z každé sekvence je odebráno několik posledních vektorů, které označovaly klid a ukončovaly dané gesto. Následně je již možné provést trénování modelů nebo testování systému se zkontrolovanými sekvencemi.

Pro rozpoznávání gest v reálném čase jsou natrénované modely převedeny do xml reprezentace, kterou je má aplikace schopná zpracovat. Z těchto dokumentů jsou následně načteny parametry natrénovaných modelů. Každý model je pro rozpoznávanou sekvenci dekódován a jako rozpoznané gesto je vybráno to, jehož model vyslal vstupní sekvenci vektorů příznaků s největší věrohodností. Dochází tak k minimalizaci klasifikační chyby podle Bayesova rozhodovacího pravidla pro minimalizaci chyby (2.1). Nepředpokládá se vyšší výskyt některého z gest, proto jsou apriorní pravděpodobnosti všech gest shodné. Výsledkem dekódování je Viterbiho věrohodnost, která je získána algoritmem token passing (2.6). Vyhodnocení vícerozměrných Gaussovských rozdělení je prováděno pomocí rovnice 2.6. Části této rovnice, které nezávisí na daném vektoru příznaků, jsou předpočítány, aby se výpočet urychlil.

Kapitola 4

Implementační detaily

V této kapitole stručně popíši nástroje, které jsem použil při tvorbě aplikace. Celá část práce, která se zabývá zpracováním obrazu a extrakcí příznaků, je napsána v jazyce C++ a využívá prostředků knihovny OpenCV. Pro práci s toolkitem HTK, který je použit pro trénování a dekódování HMM, jsem vytvořil sadu skriptů v jazyce Python. Při tvorbě těchto skriptů jsem vyšel ze skriptů pro unixový shell bash, které mi poskytl Ing. Josef Mlích. Rovněž jsem využil program pro převod textové reprezentace příznaků do formátu RAW, který mi také poskytl Ing. Mlích. HTK neumí zpracovávat vstup v textové formě, proto je třeba převést textovou reprezentaci příznaků do formátu RAW. Práce byla vytvořena pro platformu Windows (funkčnost programu byla vyzkoušena na Windows XP a Windows Vista) s využitím vývojového prostředí MS Visual Studio 2008.

4.1 Tvorba vyhledávací tabulky a extrakce příznaků

Pro zpracování videa, tvorbu vyhledávací tabulky a extrakci příznaků jsem využil knihovnu OpenCV. Součástí této knihovny jsou pokročilé algoritmy zaměřené na detekci pohybu, sledování objektu apod.

Z knihovny OpenCV jsem využil především funkce pro snímání videa z kamery a další funkce pro manipulaci s obrazem. Pro převod do barevného prostoru HSV používám funkci *cvCvtColor*, která umožňuje převod mezi mnoha typy barevných prostorů. Převod z klasického RGB do jeho normalizované podoby jsem implementoval vlastními silami, protože to zmíněná funkce neumí. Za největší přínos knihovny OpenCV pro mou práci považuji funkci pro nalezení hranic spojitých oblastí v binárním obraze. Jedná se o funkci *cvFindContours*, kterou jsem využil pro hledání hranic oblastí barvy lidské kůže. Následně jsem použil funkci *cvFitEllipse2*, která dokáže spočítat parametry elipsy, kterou je možné nalezenou oblast kůže opsat. Tyto dvě funkce jsou stěžejní pro část získávání příznaků. Dále jsem využil možnost ukládání různých datových struktur do xml dokumentu pomocí funkce *cvSave*. Poslední užitečná funkce, kterou bych rád zmínil, umožňuje otočení obrazu tak, aby nebyl zrcadlově převrácen. Toto ocenili především lidé, kteří mi pomáhali tvořit datové sady

a které zrcadlově převrácený obraz mátl. Jedná se o funkci *cvFlip*.

Všechny sekvence příznaků získané při jednom spuštění aplikace jsou uloženy do jednoho souboru. Každý vektor příznaků je označen číslem snímku, které však není dále využito s slouží jen pro označení místa, odkud byl daný vektor příznaků získán. Číslo snímku je číslem druhého ze dvojice snímků, ze kterých byly získány diference jednotlivých příznaků. Každá platná sekvence vektorů je ukončena informací zahrnující název a délku gesta společně s délkou vektoru příznaků a počtem klidových snímků ukončujících gesto.

4.2 Klasifikace gest

Při tvorbě klasifikátoru gest jsem vyzkoušel několik nástrojů pro práci se Skrytými Markovovými modely. Ze začátku jsem zkoumal možnosti knihoven UMDHMM[1] a GHMM[2]. Vybral jsem si knihovnu GHMM, protože UMDHMM nepodporuje práci s kontinuálními HMM. Nakonec jsem však od knihovny GHMM přešel na toolkit HTK, neboť jsem narazil na problémy s trénováním HMM.

HTK je profesionální toolkit pro HMM, který se používá především pro rozpoznávání řeči. Při práci s tímto toolkitem jsem čerpal z jeho obsáhlé dokumentace [8]. Vytvořil jsem sadu skriptů pro jazyk Python, které automatizují trénování modelů a následné testování. Skript *NetHMM.py* nejdříve vytvoří seznam hlásek (fonémů). Tento seznam je při zpracování řeči výčtem jednotlivých HMM reprezentujících hlásky, ze kterých se dále skládají slova. V mém případě se také jedná pouze o výčet gest. Dále je vytvořen slovník, jenž v pojetí zpracování řeči definuje skladbu slov. V mém případě je však každé gesto chápáno jako jedno slovo. Nakonec je vytvořena síť modelů, která reprezentuje možnou posloupnost jednotlivých gest a tato gesta mohou vyskytovat nezávisle na sobě. Tato síť je převedena do vnitřní reprezentace HTK, tzv. „latice“, programem *Hparse*.

Před vlastním trénováním modelů je třeba data validovat (kontroly sekvencí jsou popsány v 3.3) skriptem *PrepareData.py*, který také dokáže vybrat jen určitou skupinu příznaků, s nimiž chceme dále pracovat. Průběh trénování modelů automatizuje skript *TrainHMM.py*. Tento skript trénovací data převede do formátu RAW a následně vytvoří jejich seznam v HTK reprezentaci. Parametry prototypů modelů jsou před trénováním inicializovány hodnotami vypočítanými z celé trénovací sady pomocí programu *HCompV*. Nakonec je spuštěn program *HERest*, který natrénuje modely pomocí Baum-Welchova algoritmu.

Pro tvorbu modelů se směsí Gaussovských rozdělení v jednotlivých stavech, jež jsou použity v jednom z experimentů, jsem využil program *HHed*. Jednou z možností tohoto programu je rozdělení stávajících Gaussovských rozdělení na více rozdělení podle směru, ve kterém je v datech nejvyšší variance. V modelech jsou vždy využita Gaussovská rozdělení s diagonální kovarianční maticí.

[1] http://www.kanungo.com/software/software.html
[2] http://www.ghmm.org/

Pro účely rozpoznávání gest v reálném čase jsem za pomoci modulu ElementTree pro Python, který ovládá manipulaci a tvorbu xml dokumentů pomocí objektového modelu DOM, vytvořil skript *HmmToXML.py*. Tento skript převede natrénované modely z HTK reprezentace do formy xml dokumentů. Převod je možný jen pro modely, které jako vysílací funkce využívají jedno Gaussovo rozdělení. Tyto xml dokumenty zpracovává aplikace prostřednictvím knihovny TinyXML. Knihovna TinyXML přistupuje k dokumentům opět pomocí objektového modelu DOM.

Pro testování systému je třeba testovací sadu dat také validovat skriptem *PrepareData.py* a následně převést do formátu RAW. Dekódování modelů je prováděno programem *HVite*. Celý průběh a vyhodnocení testu provádí skript *TestHMM.py*. Pro vyhodnocení experimentů jsem vytvořil skipt *runTest.py*, kterému je předán jako parametr adresář s prototypy modelů a datovými sadami. Skript následně natrénuje modely a vyhodnotí úspěšnost klasifikátoru pro testovací sadu dat.

Kapitola 5

Datové sady

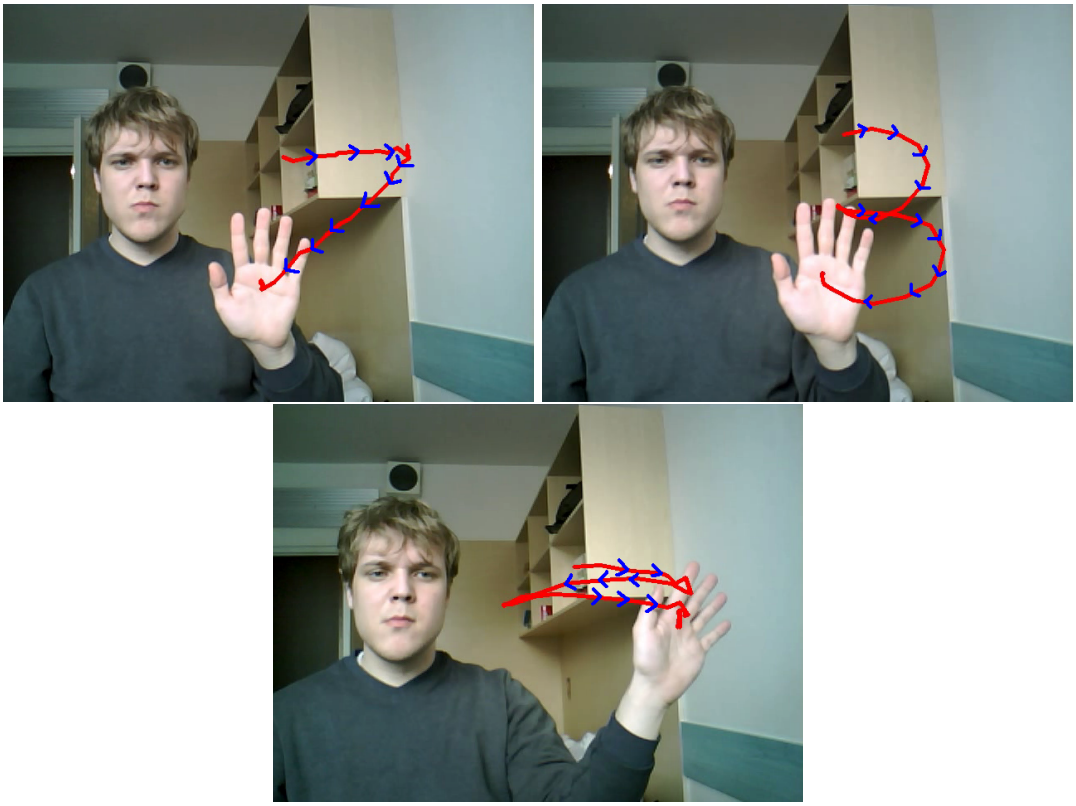
Tato kapitola je zaměřena na výběr klasifikovaných gest a získávání datových sad pro trénování HMM a následné vyhodnocení experimentů.

Rozhodl jsem se klasifikovat gesta, která jsem nazval *sedmička*, *trojka*, *tady* a *ke mě* . První tři gesta jsou společně s vyznačeným směrem pohybu vidět na obrázku 5.1. Gesto *ke mě* je složeno ze tří „mávnutí“ celou dlaní směrem k sobě, která běžně člověk považuje za signál toho, že má přijít k osobě, jež gesto provádí (gesto je vidět v natočených videosekvencích). Gesta *ke mě* a *tady* (zamávání třikrát ze strany na stranu) jsou gesta, která člověk běžně provádí, což byl hlavní důvod, proč jsem je zvolil. Gesta *sedmička* a *trojka* jsou gesta představující čísla. Zařadil jsem je proto, že podobné typy gest jsou často rozpoznávány (např. [1]) a představují jednoduchou trajektorii (*sedmička*) stejně jako složitou trajektorii (*trojka*).

Datové sady mi pomohlo získat 8 dalších osob, které souhlasily s využitím získaných dat pro mou práci. Jednotlivé videosekvence byly nahrávány ve 4 různých prostředích. Vždy bylo dbáno na co nejlepší osvětlení. Data byla sbírána na 3 různých počítačích, ale vždy byla využita stejná kamera. V každém prostředí byl vytvořen vlastní histogram pro barvu kůže. Použil jsem barevný prostor normalizované RGB bez využití Gaussova rozdělení. Každá skupina gest (pro jedno spuštění aplikace) je reprezentována textovým souborem s příznakovými vektory a zaznamenanou videosekvencí, která slouží spíše pro ilustraci a případné budoucí použití při podobném projektu. Před vlastním záznamem videa a extrakcí příznaků jsem vypnul automatické vyvažování bílé barvy pomocí programu *VLC media player*.

Gesto	**Počet trénovacích dat**	**Počet testovacích dat**
Ke mě	90	40
Tady	90	40
Trojka	91	40
Sedmička	91	40

Tabulka 5.1: Počet dat v trénovací a testovací sadě



Obrázek 5.1: První tři gesta, která jsem se rozhodl rozpoznávat. Poslední gesto (*ke mě*) není zobrazeno, protože jeho trajektorie není podstatnou informací a její zobrazení je matoucí. Zobrazena jsou gesta (zleva) *sedmička*, *trojka* a *tady*.

Kůže většiny osob byla dobře rozpoznána i s histogramem vytvořeným pomocí mé kůže, ale pro lepší výsledky jsem obvykle pro každého člověka vytvářel vlastní histogram.

Bylo natočeno celkem 60 videosekvencí, z nichž bylo po provedení kontroly skriptem *prepareData.py* získáno 522 validních sekvencí vektorů příznaků. Na jedno gesto připadá přibližně 130 sekvencí. Získaná data jsem rozdělil do dvou sad. Trénovací sada obsahuje přibližně $\frac{2}{3}$ všech dat a testovací sada zahrnuje zbylou $\frac{1}{3}$ dat (viz. tabulka 5.1). Testovací data byla vybírána tak, aby se v ní nevyskytovaly vzorky, které byly použity pro trénování.

Kvůli malému množství osob, které mi pomáhaly s tvorbou datových sad, jsem testovací sadu nevytvořil pouze z gest osob, jejichž záznamy nebyly vůbec použity pro trénování, ale testovací sada byla vytvořena z datových vzorků z celé množiny nasbíraných dat. Je možné, že díky tomuto výběru dochází k určitému zkreslení výsledků experimentů, pro které byly datové sady využity. Vyzkoušel jsem však i vytvořit testovací sadu pro jedno gesto složenou pouze z dat jednoho člověka, kterého jsem vůbec nevyužil pro trénování a výsledky byly prakticky shodné. Experimenty jsou popsány v následující kapitole.

Kapitola 6

Experimenty

V první části této kapitoly jsou zaznamenány druhy a cíle experimentů, které jsem provedl. V další části kapitoly je popsáno vyhodnocení experimentů. Snahou experimentů je nalézt takovou množinu parametrů klasifikátoru gest (počet stavů a struktura HMM, typy použitých příznaků), pro kterou bude klasifikátor vykazovat nejlepší výsledky.

6.1 Popis experimentů

V této části jsou popsány prováděné experimenty. U každého experimentu je vysvětlen jeho cíl a způsob, jakým se experiment bude vyhodnocovat. Všechny modely mají společné to, že Gaussovská rozdělení v jednotlivých stavech mají diagonální kovarianční matici.

Experiment č.1 – Vliv počtu stavů

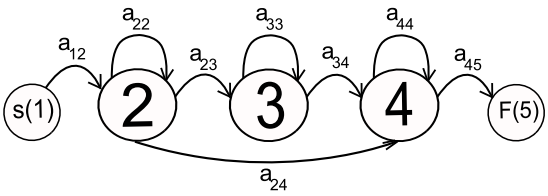
Tento experiment má za cíl zjistit, jakým způsobem se mění průměrná chyba klasifikace (error rate) při změně počtu stavů. Bude vyzkoušena úspěšnost klasifikátoru na modelech s 3 až 16 stavy (1 až 14 vysílacích stavů). Více stavů by mělo za následek nemožnost použití kratších sekvencí, a proto takové testy již neprovádím.

Výsledkem tohoto experimentu bude graf určující trend chyby klasifikace při zvyšování počtu stavů Skrytých Markovových modelů. Dalším výsledkem tohoto experimentu bude optimální počet stavů pro modely, které se použijí v dalších experimentech.

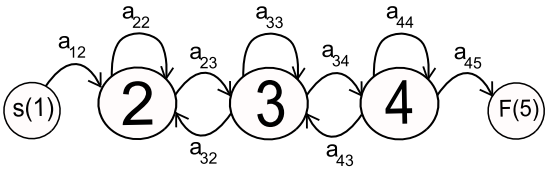
Pro zjednodušení vyhodnocení tohoto testu se pro každé gesto bude používat model o stejném počtu stavů, i když je pravděpodobné, že pro jednodušší gesta lze použít méně-stavový model bez ztráty kvality rozpoznávání.

Experiment č.2 – Vliv použitých příznaků

Cílem tohoto experimentu je zjištění, jak kvalitu rozpoznání jednotlivých gest ovlivní klasifikace na základě omezené množiny původních příznaků. Bude provedeno několik testů



Obrázek 6.1: Příklad levopravého modelu umožňujícího přeskočení jednoho následujícího vysílacího stavu. Model má 3 vysílací stavy.



Obrázek 6.2: Příklad Skrytého Markovova modelu, ve kterém je umožněn návrat na předchozí stav. Tento model není typický pro rozpoznávání sekvencí.

pro různé skupiny použitých příznaků. Každý test bude vyhodnocen pomocí matice záměn[1] (confusion matrix). Skryté Markovovy modely budou mít počet stavů, který bude vybrán na základě experimentu č. 1.

Experiment č.3 – Vliv struktury modelu

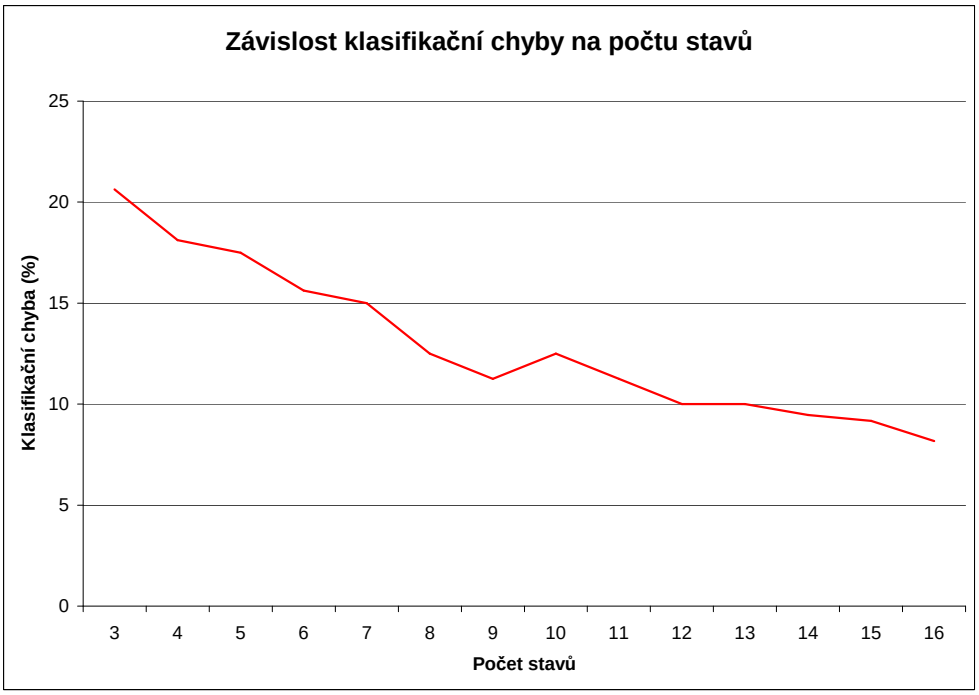
Tento experiment zkoumá vliv struktury modelu na úspěšnost rozpoznávání. Testováno bude na modelech s 5 až 16 stavy. Tyto modely však již nebudou pouze levopravé bez přeskakování stavů nebo navracení. Modely budou navrženy tak, aby do koncového stavu šlo přejít pouze z jednoho stavu a stejně tak, aby se z počátečního stavu dalo přejít pouze do jednoho stavu.

Provedeny budou dvě skupiny testů. Jedna pro modely s přeskakováním jednoho následujícího vysílacího stavu (obrázek 6.1) a jedna pro modely s umožněním navracení na předchozí stav (obrázek 6.2). Klasifikační chyba pro různé počty stavů modelu bude zobrazena pomocí grafu. Obě skupiny testů společně s výstupem experimentu č. 1 budou pro lepší srovnání zobrazeny v jednom grafu.

Experiment č.4 – Vliv počtu Gaussovských rozdělení v jednom stavu HMM

Pro potřeby tohoto experimentu zavádím poprvé směs Gaussovských rozdělení v každém stavu HMM. Cílem experimentu je snaha zjistit závislost klasifikační chyby na počtu Gaussovských rozdělení v každém stavu HMM. Bude provedeno několik testů pro různý počet Gaussovských rozdělení ve stavech. Budou použity modely s počtem stavů, který bude vybrán na základě experimentu č. 1. Vyhodnocením experimentu bude graf určující trend chyby klasifikace.

[1] Matice záměn – každý řádek obsahuje informaci o jednom druhu gesta. Ve sloupcích je číselně vyjádřeno, jak bylo dané gesto rozpoznáno.



Obrázek 6.3: Graf znázorňující závislost klasifikační chyby na počtu stavů jednotlivých HMM. Je vidět, že při zvyšování počtu stavů chyba klesá.

6.2 Vyhodnocení experimentů

V této části se věnuji vyhodnocení navržených experimentů. Pro vyhodnocení jsou použity matice záměn a grafy. U každého experimentu je uveden závěr, který z jeho vyhodnocení vyplývá.

Experiment č.1 – Vliv počtu stavů

V tomto experimentu jsem zkoumal, jaký vliv má počet stavů modelu na chybu klasifikace gest. Z grafu na obrázku 6.3 je jasně patrné, že se zvyšujícím se počtem stavů klesá průměrná klasifikační chyba (součet všech klasifikačních chyb jednotlivých modelů dělený počtem modelů).

Při zvyšování počtu stavů je však třeba brát ohled na délku nejkratších validních sekvencí. V mém případě má nejkratší sekvence, která byla při sběru dat pořízena, délku 11 vektorů. Aby bylo možné použít sekvenci o délce 11 vektorů, je nutné zvolit modely nejvýše o 13 stavech. Nakonec jsem zvolil použít modely o 12 stavech. Rozhodl jsem se tak také proto, že výsledky na mé testovací sadě byly pro 12 a 13 stavů shodné.

Experiment č.2 – Vliv použitých příznaků

V tomto experimentu jsem zkoumal úspěšnost rozpoznávání při použití různých skupin příznaků. Cílem bylo zjistit, zda při některé konfiguraci příznaků nedochází ke zvýšení

průměrné úspěšnosti klasifikace nebo k výrazným změnám úspěšnosti klasifikace u jednotlivých gest. Všechny modely využité v tomto experimentu mají 12 stavů.

Pro vyhodnocení všech testů, které jsou součástí experimentu, je využito maticí záměn. V tabulce vždy uvádím jeden sloupec navíc, v němž je uvedena celková úspěšnost rozpoznání pro dané gesto.

Tabulkou 6.1 je znázorněna matice záměn při využití všech příznaků. S touto maticí budu srovnávat výsledky ostatních testů, které jsou součástí tohoto experimentu.

Matice záměn znázorněná tabulkou 6.2 je vyhodnocením testu, při kterém byly jako příznaky použity pouze diference x-ové a y-ové souřadnice středu dlaně. U jednoduchých gest typu „sedmička“ a „tady“, kde je hlavní informací trajektorie pohybu, dochází oproti původní skupině příznaků (tabulka 6.1) ke zlepšení. U složitějších gest však použitá skupina příznaků není vhodná.

V testu, jehož vyhodnocením je tabulka 6.3, byly jako příznaky použity všechny příznaky kromě příznaků popisujících posun pozice dlaně. Je vidět, že dochází k nárustu chyby u gest se složitou trajektorií jako je například „trojka“. Stejně tak u gesta „tady“ dochází k nárustu chyby, protože se jedná o gesto, kde je trajektorie pohybu nejdůležitější informací. Gesto „sedmička“ vykazuje v tomto testu zajímavé výsledky oproti očekávání. Důvodem je pravděpodobně to, že většina lidí při provádění tohoto gesta mění náklon ruky při šikmém pohybu dolů (viz. videa). Tento fakt pravděpodobně přispěl k tomu, že úspěšnost rozpoznání tohoto gesta neklesla tolik, kolik jsem očekával. Gesto „ ke mě“ vykazuje stejné výsledky jako při použití všech příznaků, protože trajektorie pohybu není v tomto případě podstatnou informací.

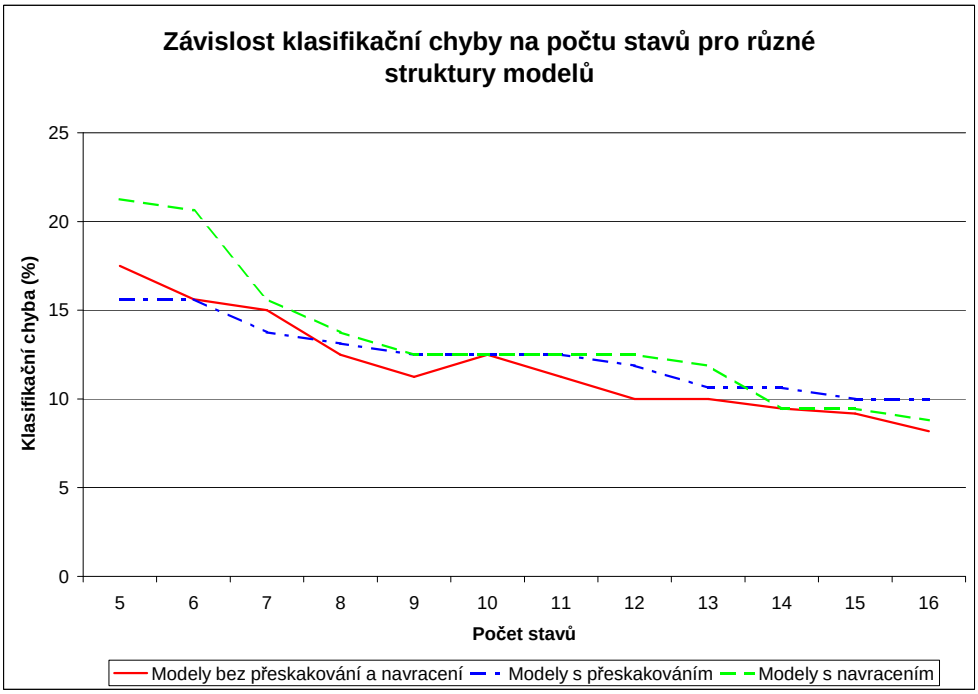
Poslední test, který je součástí tohoto experimentu, je vyhodnocen v tabulce 6.4. Test byl proveden s příznaky polohy a natočení dlaně, protože příznaky popisující velikost dlaně a poměr os elipsy nejsou plně invariantní vůči velikosti dlaně a vzdálenosti od kamery. Výsledky ukazují, že tato skupina příznaků by byla vhodná pro všechna gesta kromě gesta „ ke mě“. Gesto „ke mě“ je z velké části popsáno právě velikostí a poměrem os elipsy. Pro gesta založená na pohybu a natočení dlaně (např. číslice, písmena apod.) by byla skupina příznaků použitá v tomto testu lepší volbou než původní příznaky.

Z výsledků testů tohoto experimentu vyplývá, že vzhledem ke zvoleným typům gest je nejvhodnější použít jako příznaky všechny dostupné informace (poloha a natočení dlaně, velikost a poměr os elipsy).

Experiment č.3 – Vliv struktury modelu

Tento experiment zkoumal průběh chyby klasifikace v závislosti na počtech stavů při určitě struktuře modelů. Byly zkoumány modely s možností přeskočení jednoho stavu (obrázek 6.1) a modely s možností navracení do předchozího stavu (obrázek 6.2). Výsledek experimentu je vidět v grafu na obrázku 6.4.

Po zhodnocení výsledků experimentu lze konstatovat, že jako nejlepší struktura modelů se ukázala základní struktura bez přeskakování stavů nebo navracení.



Obrázek 6.4: Graf znázorňující závislost klasifikační chyby na počtu stavů jednotlivých HMM pro různou strukturu modelů. Výsledky jsou velmi podobné, ale jako nejlepší se ukázaly modely bez přeskakování nebo navracení.

	Sedmička	**Trojka**	**Ke mě**	**Tady**	**Úspěšnost (%)**
Sedmička	39	1	0	0	97,5
Trojka	0	39	1	0	97,5
Ke mě	0	8	32	0	80
Tady	1	2	3	34	85

Tabulka 6.1: Matice záměn pro test, kdy byly použity všechny příznaky. V tabulce je také vidět úspěšnost rozpoznávání pro jednotlivá gesta.

	Sedmička	**Trojka**	**Ke mě**	**Tady**	**Úspěšnost (%)**
Sedmička	40	0	0	0	100
Trojka	1	35	4	0	87,5
Ke mě	1	8	31	0	77,5
Tady	0	2	1	37	92,5

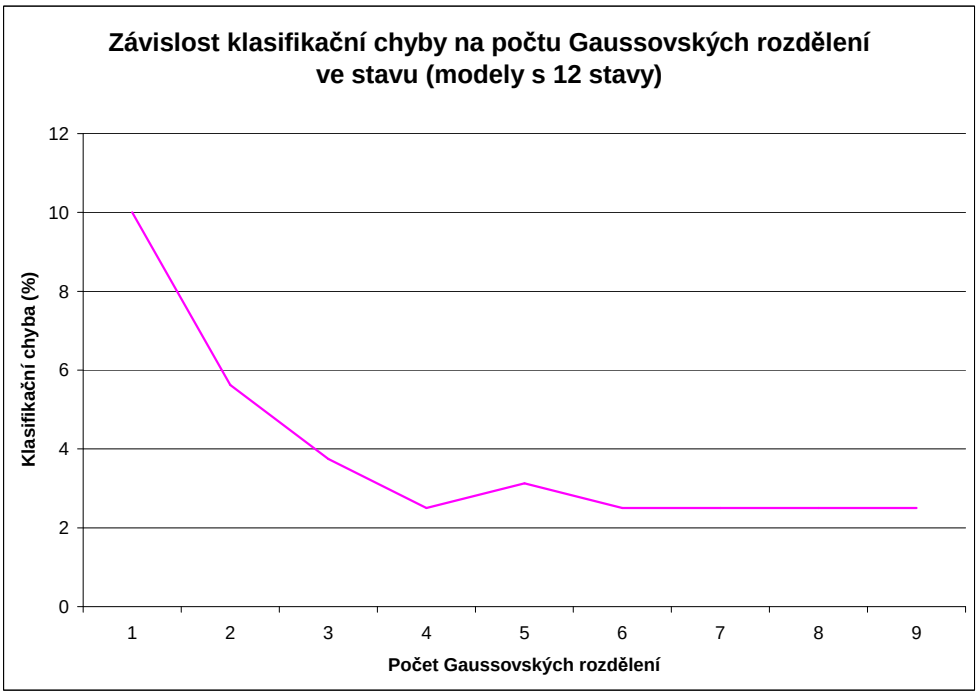
Tabulka 6.2: Matice záměn pro test, kdy byly použity pouze příznaky popisující polohu dlaně (diference x-ové a y-ové souřadnice). V tabulce je také vidět úspěšnost rozpoznávání pro jednotlivá gesta.

	Sedmička	Trojka	Ke mě	Tady	Úspěšnost (%)
Sedmička	32	3	0	5	80
Trojka	10	28	1	1	70
Ke mě	1	6	32	1	80
Tady	16	3	4	17	42,5

Tabulka 6.3: Matice záměn pro test, kdy byly použity příznaky popisující poměr os elipsy, úhel natočení a velikost dlaně. V tabulce je také vidět úspěšnost rozpoznávání pro jednotlivá gesta.

	Sedmička	Trojka	Ke mě	Tady	Úspěšnost (%)
Sedmička	39	0	1	0	97,5
Trojka	1	38	1	0	95
Ke mě	2	14	24	0	60
Tady	0	2	2	36	90

Tabulka 6.4: Matice záměn pro test, kdy byly použity pouze příznaky popisující polohu a úhel natočení dlaně. V tabulce je také vidět úspěšnost rozpoznávání pro jednotlivá gesta.



Obrázek 6.5: Graf znázorňující závislost chyby klasifikace na počtu Gaussovských rozdělení v jednom stavu HMM. Použité modely měly 12 stavů. Je vidět, že chyba ze začátku prudce klesá, později však již nedochází ke zlepšení.

Experiment č.4 – Vliv počtu Gaussovských rozdělení v jednom stavu HMM

Tento experiment si kladl za cíl zjistit, jak ovlivní průměrnou chybu klasifikace zavedení směsi Gaussovských rozdělení v jednotlivých stavech HMM. Bylo provedeno 9 testů pro 1 až 9 Gaussovských rozdělení v jednom stavu HMM. Použity byly modely s 12 stavy. Po každém zvýšení počtu Gaussovských rozdělení ve stavech HMM byly modely znovu přetrénovány.

Graf určující trend chyby klasifikace v závislosti na počtu Gaussovských rozdělení ve stavech je vidět na obrázku 6.5. Použití směsi Gaussovských rozdělení vede k výraznému snížení klasifikační chyby, ale pro vyšší počet rozdělení ve stavech již ke změně klasifikační chyby nedocházelo. Z testu vyplývá, že směs Gaussovských rozdělení s diagonální maticí pokrývá data lépe než jedno Gaussovo rozdělení s diagonální maticí.

Kapitola 7

Závěr

Cílem této bakalářské práce bylo navrhnout a implementovat systém pro rozpoznávání gest a provést s tímto systémem několik experimentů nad získanou datovou sadou. Tento cíl se mi podařilo splnit. Mezi funkce výsledné aplikace patří tvorba vyhledávací tabulky pro detekci kůže, získávání příznaků pro klasifikaci gest a rozpoznávání gest v reálném čase. Dále bylo vytvořeno několik skriptů pro práci s toolkitem HTK. Tyto skripty zajišťují přípravu dat, trénování HMM a vyhodnocení úspěšnosti klasifikace pro testovací sadu dat. Rozpoznávání gest v reálném čase jsem implementoval jako rozšíření aplikace, protože nebylo předmětem zadání.

Výsledky experimentů jsou velmi dobré, stejně tak obstojným způsobem funguje rozpoznávání gest v reálném čase. Výhodou navržené aplikace je možnost přidávání dalších gest pro rozpoznávání, pokud získáme dostatek vzorků takových gest.

Další rozvoj práce je možný především na poli detekce kůže a nalezení dlaně ruky. Bylo by vhodné použít algoritmy, které si poradí se zmizením dlaně ze scény (nebo překrytím dlaně a hlavy) případně s tím, že člověk, který aplikaci používá, nemá oblečen oděv s dlouhým rukáv. Dále by bylo vhodné učinit detekci kůže a nalezení dlaně více odolné vůči změně světelných podmínek.

V současné podobě by bylo možné využít funkci rozpoznávání gest v reálném čase pro jednoduché ovládání počítače. Větší využití v praxi by však bylo problematické vzhledem k omezením týkajících se nalezení dlaně ruky.

Práce byla z mého pohledu velmi zajímavá. Během její tvorby jsem se seznámil s možnostmi segmentace a zpracováním obrazu. Dále jsem prostudoval několik typů klasifikátorů a soustředil se na Skryté Markovovy modely. Také jsem se naučil pracovat s knihovnou OpenCV a toolkitem HTK.

Literatura

[1] Alon, J.; Athitsos, V.; Yuan, Q.; aj.: Simultaneous Localization and Recognition of Dynamic Hand Gestures [online].
http://www2.computer.org/portal/web/csdl/abs/proceedings/wacv-motion/2005/2271/02/227120254abs.htm.

[2] Burget, L.: Klasifikace a rozpoznávání - extrakce příznaků [online].
http://www.fit.vutbr.cz/study/courses/IKR/public/prednasky/03_extrakce%20priznaku/IKR3.pdf.

[3] Černocký, J.: Systémy zpracování řeči - Dekódování HMM [online].
http://www.fit.vutbr.cz/study/courses/SRE/public/prednasky/05_dekodovani_zaklad/04_decode.pdf.

[4] Černocký, J.: Systémy zpracování řeči - Skryté Markovovy modely [online].
http://www.fit.vutbr.cz/study/courses/SRE/public/prednasky/04a_hmm_honza/sre_03_hmm.pdf.

[5] Vezhnevets, V.; Sazonov, V.; Andreeva, A.: A Survey on Pixel-Based Skin Color Detection Techniques [online].
http://graphics.cs.msu.ru/en/publications/text/gc2003vsa.pdf.

[6] Webb, A. R.: *Statistical Pattern Recognition second edition*. Butterworth Heinemann, 2002, 1–24, 33–37, 83 s., iSBN 0-470-84513-9.

[7] Yang, M.-H.; Ahuja, N.: Recognizing Hand Gesture Using Motion Trajectories [online].
http://vision.ai.uiuc.edu/mhyang/papers/cvpr99.pdf.

[8] Young, S.; Evermann, G.; Gales, M.; aj.: *The HTK Book*. Cambridge University Engineering Department, 1996.

[9] Zhu, Y.; Ren, H.; Xu, G.; aj.: Toward Real-time Human-computer Interaction with Continuous Dynamic Hand Gestures [online].
http://www2.computer.org/portal/web/csdl/abs/proceedings/fg/2000/0580/00/05800544abs.htm.

Seznam příloh

Příloha A

Obsah CD

Adresářová struktura přiloženého CD:

- **Aplikace**
  - **bin** (Projekt přeložený pomocí MS Visual Studia 2008 pod Windows XP)
  - **src** (Zdrojové soubory ve formě projektu pro MS Visual Studio 2008)
  - **VS_patchs** (Balíčky obsahující části C++ knihoven pro VS, které jsou nutné pro běh programů zkompilovaných pod Visual Studiem 2008 a 2005. Na některých počítačích se bez těchto balíčku nedaří aplikaci spustit. Na jiných funguje bez problémů.)
- **Data** (Nahrané videosekvence společně s textovými soubory s příznaky)
- **HTK_scripts** (Skripty pro práci s HTK, adresářová struktura je popsána v Manuálu)
- **Knihovny** (Knihovny OpenCV a TinyXML)
- **Plakat** (Obsahuje plakát prezentující tuto bakalářskou práci)
- **Zprava** (Zdrojové soubory tohoto dokumentu v systému LaTeX. Také obsahuje tento dokument ve formátu pdf)

Příloha B

Manuál k programu

V této příloze je popsána práce s aplikací *gestures* a se skripty pracujícími s toolkitem HTK. Pro úspěšnou kompilaci aplikace a funkčnost všech částí práce je třeba nainstalovat:

- **knihovna OpenCV** (používaná verze 1.1pre1a[1])
- **toolkit HTK** (používaná verze 3.4[2] zkompilovaná pod Windows XP)
- **skriptovací jazyk Python** (používaná verze 2.6.1[3] pro Windows)
- **Knihovna TinyXML**[4]
- **kodek DIVX**[5] (používá se pro ukládání videa)
- **MS Visual Studio 2008**

Použití aplikace *gestures*

Aplikace předpokládá adresářovou strukturu:

- **cfg** – konfigurační soubory, budou popsány níže
- **data** – adresář, kam se ukládájí soubory s příznaky a videosekvence
- **models** – xml dokumenty popisující natrénované HMM, které se používají pro rozpoznávání gest v reálném čase
- **skin** – adresář pro uložení histogramu barvy kůže a vyhledávací tabulku

[1] http://sourceforge.net/project/showfiles.php?group_id=22870&package_id=16937
[2] http://htk.eng.cam.ac.uk/download.shtml
[3] http://www.python.org/download/releases/2.6.1/
[4] http://sourceforge.net/projects/tinyxml
[5] http://www.divx.com/en/downloads

Po načtení projektu pomocí MS Visual Studia 2008 a jeho přeložení vzniká aplikace s názvem *gestures.exe*. Pokud program nepřekládáme a spustíme již přeložený projekt v adresáři **aplikace/bin**, může se stát, že se program nepodaří spustit. V takovém případě je nutné nainstalovat chybějící komponenty knihoven C++, které jsou nutné pro spuštění programů zkompilovaných pod VS 2008 nebo 2005. Tyto balíčky se nachází v adresáři **aplikace/VS_patchs**. Potřebný by měl být především soubor it vcredist_x86_2008sp1.exe, což je balíček pro Visual Studio 2008 se Service Packem 1, které bylo pro kompilaci aplikace použito.

V hlavičkovém souboru *const.h* je možné přenastavit několik konstant ovlivňujících běh programu. Jedná se především o konstantu *NUMBLOBS*, která udává kolik oblastí barvy kůže je v obraze očekáváno. Tato konstatnta je nastavena na hodnotu 2 (ruka a obličej). Dále je v tomto souboru několik konstant, jež slouží především pro ladění aplikace a umožňují sledovat binární obraz kůže, výsledný histogram, trajektorii gesta a podobně.

Program je ovládán parametry příkazové řádky. Spuštění této aplikace bez parametrů vypíše nápovědu k programu. Aplikace akceptuje jeden vstupní parametr, který definuje požadovanou akci. Zvolená akce je vždy předcházena znakem pomlčky (např. -h). Je možné vybrat následující akce:

- **h** – Tvorba histogramu. Pokud již byl dříve vytvářen histogram, jsou nové hodnoty přičítány k uloženému histogramu.
- **c** – Tvorba histogramu. Je vytvořen nový histogram.
- **s** – Probíhá analýza obrazu. Je detekována pohybující se ruka, ale nejsou ukládány příznaky.
- **f** – Probíhá analýza obrazu. Je detekována pohybující se ruka a příznaky jsou ukládány do adresáře data.
- **v** – Rozpoznávání gest v reálném čase. Příznaky nejsou ukládány.
- **k** – Rozpoznávání gest v reálném čase. Příznaky jsou ukládány do adresáře data.

Před spuštěním aplikace je vhodné vypnout automatické vyvažování bílé barvy (např. pomocí programu VLC media player), pokud kamera tuto funkci má. Při zapnutém automatickém vyvažování bílé barvy dochází k šumu v binárním obraze a tedy k nekvalitní detekci lidské kůže. Aplikaci je také třeba používat v dobrém osvětlení a je vhodné být oblečen v oděvu s dlouhým rukávem.

V režimu tvorby histogramu lze vytvořit vyhledávací tabulku stiskem klávesy „**g**“ pro vytvoření tabulky pomocí Gaussova rozdělení nebo stiskem klávesy „**t**“, což vede k vytvoření tabulky bez využití Gaussova rozdělení. Histogram je tvořen vždy z oblasti označené modrým čtvercem, do kterého je třeba vložit ruku. Proces tvorby histogramu je spuštěn

zmáčknutím „**mezerníku**“ a ukončen tvorbou tabulky nebo klávesou „**esc**“, v tom případě je histogram pouze uložen a vyhledávací tabulka není vytvořena.

Proces extrakce příznaků začíná vždy volbou gesta. Klávesy pro volbu gesta jsou definovány v souboru *gests.txt*. Tento soubor bude popsán níže. Výběr gesta je potvrzen zmáčknutím „**mezerníku**“ a stejnou klávesou se lze vždy k výběru vrátit. Po výběru gesta se již v levém horním rohu zobrazují stavové informace aplikace. Druhy stavových informací:

- **Čeká se na klid** – Aplikace čeká, až budou všechny detekované oblasti kůže definovaný počet snímků v klidu.
- **Čeká se na pohyb** – Aplikace čeká na pohyb ruky, kterou poté označí červenou elipsou.
- **Zapisují se příznaky** – Probíhá zapisování příznaků prováděného gesta. Jakmile je pohybující se ruka definovaný počet snímků v klidu, gesto je ukončeno.
- **Skenování gesta** – Stejné jako předchozí případ, nejsou však zapisovány příznaky.
- **Přesun do počátku** – Aplikace umožňuje po ukončení gesta přesunout ruku opět do výchozího bodu gesta. Při tomto stavu nejsou zapisovány příznaky. Stav je ukončen, pokud jsou všechny detekované oblasti kůže definovaný počet snímků v klidu.

Počet snímků „klidu“ potřebný pro určení různých přechodů mezi stavy programu je definován v konfiguračním souboru.

Práce s toolkitem HTK

Pro práci s toolkitem HTK jsem vytvořil sadu skriptů (vycházejí z podobných skriptů Ing. Mlícha). Pro jejich funkčnost je třeba mít v systémové proměnné PATH adresář s toolkitem HTK. Dále je nutné mít nainstalovaný skriptovací jazyk Python, ve kterém jsou skripty napsány. Adresář *HTK_scripts* na CD má specifickou strukturu. Každý podadresář má svůj účel. Tato struktura vypadá následovně:

- **config** – Obsahuje konfigurační soubor, který především definuje právě popisovanou adresářovou strukturu, proto nebude tento konfigurační soubor dále popsán
- **convert_util** – Obsahuje program pro převedení příznaků z textové podoby do RAW. Program je přeložený. Adresář však obsahuje i zdrojový kód a Makefile (celý program je dílem Ing. Mlícha)
- **data** – Adresář pro práci s daty
  - **ALL** – Slouží pro přípravu a kontrolu dat (skriptem *prepareData.py*. Obsahuje podadresáře pro připravená (zkontrolovaná) a nepřipravená data a také pro data setříděná podle názvu modelu (skriptem *sortData.py*). Do adresáře pro

nepřipravená data je třeba nahrát sekvence získané aplikací *gestures*. Skript *prepareData.py* je zkontroluje a upravené vloží do adresáře pro připravená data.

  - **test** – Testovací data. Obsahuje podadresáře pro připravená data (výstup *prepareData.py*) a data zkonvergovaná do RAW.
  - **train** – Trénovací data. Adresář má stejnou strukturu jako adresář pro testovací data.

- **hmm** – Adresář s definicemi modelů HMM
  - **init** – Inicializované modely
  - **prototypes** – Prototypy modelů. Obsahuje adresář **modely** s různými druhy prototypů modelů, které jsem v průběhu práce používal.
  - **trained** – Natrénované modely
  - **xml** – Definice modelů ve formě xml dokumentů
- **lists** – Seznamy trénovacích a testovacích dat
- **mlf** – Mapování trénovacích sekvencí na modely a také vyhodnocení testovací sady dat
- **network** – Definovaná síť modelů a její HTK reprezentace latice
- **TESTY** – Obsahuje skript *runTest.py* a adresáře s daty pro provedení všech v práci popsaných experimentů.

Každý skript v sobě obsahuje nápovědu, která se zobrazí, pokud je spuštěn s parametrem **-h**, proto se funkčností skriptů nebudu příliš zabývat. Postup práce se skripty je následující:

1. Příprava dat pomocí skriptu *prepareData.py* (je možný výběr jen určité skupiny příznaků)
2. Vytvoření trénovací a testovací sady, které je třeba nakopírovat do adresářů *prepared* v adresářích pro trénovací a testovací data
3. Vytvoření (výběr) prototypů modelů a nakopírování prototypů do příslušného adresáře
4. Vytvoření seznamu hlásek, slovníku atd. skriptem *NetHMM.py*
5. Natrénování modelů skriptem *TrainHMM.py*
6. Vyhodnocení úspěšnosti rozpoznávání na testovací sadě pomocí skriptu *TestHMM.py*

Skript *HmmToXML.py* slouží pro převod natrénovaných modelů na xml dokumenty. Tyto dokumenty jsou využity pro rozpoznávání gest v reálném čase a je třeba je nahrát do příslušného adresáře aplikace *gestures*. Tento skript neumí převádět modely se směsí Gaussovských rozdělení.

Skript *runTest.py* v adresáři TESTY slouží k opětovnému vyhodnocení experimentů, případně k zjednodušení celého procesu trénování HMM a vyhodnocení úspěšnosti na testovací sadě dat. Jako jediný parametr je skriptu předán název adresáře s daty pro experiment, který obsahuje prototypy modelů a datové sady, a skript provede trénování příslušných modelů a vyhodnocení testovací sady dat. V adresáří TESTY jsou podklady ke všem experimentům, které jsem prováděl. Ostatní skripty nejsou pro práci stěžejní a slouží spíš jako podpora prováděných úkonů.

Příloha C

Popis konfiguračních souborů

V této části se nachází popis konfiguračních souborů aplikace *gestures*. Tato aplikace má dva druhy konfiguračních souborů, které jsou unístěny v adresáři **cfg**.

Výběr gesta

Prvním konfiguračním souborem je soubor *gests.txt*, který definuje klávesy pro výběr gesta daného jména. Stiskem zvolené klávesy je vybrán model daného jména. Jeden záznam v tomto souboru má tvar:

klávesa:gesto

Klávesa slouží pro výběr gesta na začátku extrakce příznaků pomocí aplikace *gestures* a gesto je jménem tohoto gesta, které bude uvedeno u každé zaznamenané sekvence. Jméno gesta se musí shodovat se jménem souboru příslušného prototypu HMM. Pokud sekvence nezaznamenáváme, můžeme vybrat jakékoli gesto nebo si pro tento případ definovat vlastní klávesu. Nyní jsou pro výběr gest nadefinovány klavásy *q*, *w*, *e*, *r* a *t* pro testování.

Konfigurace aplikace *gestures*

Některé vlastnosti a možnosti aplikace *gestures* je možné nastavit v jejím druhém konfiguračním souboru *gesture.conf* v adresáři **cfg**. Jedná se o prostý textový soubor. Jeden záznam v tomto souboru má tvar:

Název vlastnosti=hodnota

Nastavit lze tyto vlastnosti:

- **StartSequence** – Číslo sekvence. Je částí názvu souboru, do kterého jsou ukládány příznaky nebo videosekvence. Aplikace toto číslo sama při každém spuštění s požadavkem na ukládání souborů s příznaky inkrementuje, aby nedošlo k přepsání starších souborů. Hodnota je považována za celé číslo.

- **HistogramRange_X** – Rozsah x-ové osy histogramu. Slouží pro možnost využití menšího histogramu. Standardně je použit histogram, ve kterém je každému bodu v barevném prostoru přiřazeno jedno pole histogramu. Pokud je použit barevný prostor HSV, musí tato hodnota být menší nebo rovna 256. Hodnota je považována za celé číslo.

- **HistogramRange_Y** – Rozsah y-ové osy histogramu. Pokud je použit barevný prostor HSV, musí tato hodnota být menší nebo rovna 256. Hodnota je považována za celé číslo.

- **HistogramSquare** – Délka strany modrého čtverce, do kterého se vkládá dlaň při výpočtu histogramu. Hodnota je považována za celé číslo.

- **FPS** - Snímková frekvence kamery. Není třeba vyplňovat. Se svou kamerou jsem narazil na problém s knihovnou OpenCV, která nedokázala FPS mé kamery určit. Pokud program hlásí, že se mu nepodařilo FPS určit (v takovém případě je nastaveno na 30), je možné nastavit FPS explicitně touto hodnotou. Hodnota může být zadána jako desetinné číslo.

- **FPS_Divider** – Dělitel FPS, který slouží pro určení počtu klidových snímků. Touto hodnotou se dělí FPS a výsledkem je počet klidových snímků pro přechod mezi různými stavy aplikace. Pokud je hodnota např. 2, tak je třeba pro přechod mezi stavy zůstat v klidu půl sekundy apod. Hodnota může být zadána jako desetinné číslo.

- **ColorSpace** – Určuje používaný barevný prostor. Může nabývat hodnoty *NormRGB* pro prostor normalizovaného RGB nebo *HSV* pro barevný prostor HSV.