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

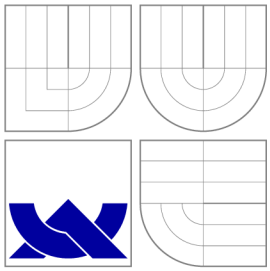
ROZPOZNÁVÁNÍ OBJEKTŮ A GEST V OBRAZE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

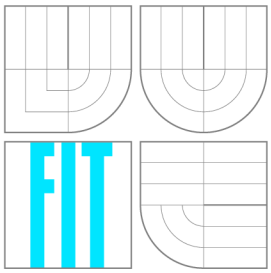
AUTOR PRÁCE
AUTHOR

Bc. DANIELA JOHANOVÁ

BRNO 2015



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 OBJEKTŮ A GEST V OBRAZE

RECOGNITION OF OBJECTS AND GESTURES IN IMAGE

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. DANIELA JOHANOVÁ

VEDOUCÍ PRÁCE
SUPERVISOR

prof. Dr. Ing. PAVEL ZEMČÍK

BRNO 2015

Abstrakt

Tato práce je zaměřena na rozpoznávání gest ve videu. Cílem práce bylo vytvořit aplikaci, která by umožnila pomocí obyčejné webové kamery rozpoznávání malého množství gest za účelem ovládání počítače. Pro tento účel bylo použito vybraných metod pro získávání deskriptorů z obrazu, pro sledování určitých oblastí ve videu a strojového učení.

Abstract

This thesis is focused on gesture recognition in video. The main purpose of this thesis was to create an algorithm and an application that can recognize selected gestures using a video obtained through a standard webcamera. The intention was to control an application program, such as video player. The approach used to achieve this goal was to exploit methods of feature extraction, tracking, and machine learning.

Klíčová slova

Gesta, strojové učení, HMM, Adaboost, sledování trajektorie ruky, detekce kůže, ovládání VLC.

Keywords

Gestures, machine learning, HMM, Adaboost, hand tracking, skin detection, VLC control.

Citace

Daniela Johanová: Rozpoznávání objektů a gest v obraze, diplomová práce, Brno, FIT VUT v Brně, 2015

Rozpoznávání objektů a gest v obraze

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracovala samostatně pod vedením pana prof. Dr. Ing. Pavla Zemčíka. Další informace mi poskytl Ing. Jozef Mlích.

.....
Daniela Johanová
27. května 2015

Poděkování

Děkuji panu profesoru Zemčíkovi za vedení při vytváření této diplomové práce a také za jeho cenné rady a připomínky. Dále děkuji panu Ing. Mlíchovi za odbornou pomoc. Děkuji také své rodině, která mne podporovala a pomáhala mi po celou dobu mého studia. Dík patří také mému příteli za oporu, trpělivost a porozumění, které mi poskytoval.

Obsah

Kapitola 1

Úvod

Oči jsou pro člověka jeden z nejdůležitějších smyslových orgánů. Vidění nám dává možnost snadno a rychle se zorientovat v prostoru, dokážeme rozpoznat předměty v blízkém i vzdálenějším okolí. Umožňuje nám také zhodnotit situaci, pokud se kolem nás něco pohybuje. Například stojíme-li v davu lidí, vidíme kam dav míří a můžeme na to adekvátně reagovat. Vnímání pohybu je velmi přínosná věc pro člověka, stejně tak to může být užitečné i pro počítač. Pro úspěšné rozpoznání hledaného objektu jsou zapotřebí další informace z pozorovaného prostředí. Člověk se už od dětství učí rozpoznávat barvy a textury, základní a později i složitější tvary. To jsou hlavní parametry objektu, které pro jeho identifikaci musíme zvládnut správně určit. Pro člověka je to snadná úloha, kterou dokáže provést takřka bez mrknutí oka. Mozek je v tomto ohledu výjimečně rychlá, výkonná a přesná jednotka. Pro počítač je to však velice obtížný úkol spojený s mnoha cykly rozpoznávání a klasifikace, které je třeba opakovaně vyhodnotit v co nejkratším čase.

Počítačové vidění, které se zabývá rozpoznáváním obsahu získaného obrazu (ať už z fotografií nebo z videí), je dnes jedním z nejrychleji se rozvíjejících odvětví v oboru informačních technologií. Získané poznatky se dále uplatňují například v robotice, při tvorbě zabezpečovacích programů pro monitorovací kamerové systémy nebo v medicínských aplikacích. Lze je také využít ke zpříjemnění a usnadnění běžného lidského života. A právě touto možností se budu ve své diplomové práci zabývat.

Rozlišování ukazované trajektorie mne zajímá z toho důvodu, že v tomto oboru vidím velký potenciál pro budoucí možný přínos pro lidstvo. Člověk denně bezděčně provádí celou řadu gest, které vypovídají o našem charakteru, náladě i duševním a zdravotním stavu. Detekce gest může být využita v bezpečnostních systémech například při zjištění potenciálně nebezpečného chování člověka na veřejných místech. Dle mého názoru je však detekce gest také dobře využitelná v běžném každodenním životě.

Za zajímavou oblast využití považuji ovládání domácího audio-video systému gesty. Mnoho lidí si pouští televizi při vaření, avšak poté je obtěžující si umývat ruce pokaždé, když chcete pomocí dálkového ovládání změnit hlasitost nebo přepnout program. Nevidomí lidé nebo lidé se špatným zrakem si rádi pouští rádio, kvůli hudbě i přísunu informací. Ale jen hledání samotného ovládacího zařízení pro ně může být problém. Oproti tomu otočení se ke zdroji hudby, kde se předpokládá i umístění kamery, a provedení naučeného gesta je jednoduché. Tento směr je podle mne důležitý a perspektivní – usnadnění a zlepšení lidského života, proto jsem v tomto duchu vypracovala i tuto práci.

Následující kapitola se zabývá vybranými metodami pro popis obrazu, které se využívají při rozpoznávání gest. Konec kapitoly je věnován metodám na sledování oblastí zájmu v obraze. Začátkem třetí kapitoly jsou vymezeny důležité pojmy týkající se gest. Následuje

popis metod strojového učení, které jsou vhodné pro klasifikaci gest. Ve čtvrté kapitole jsou shrnuty používané postupy pro řešení zadaného problému. Nachází se zde také specifikace postupů a softwaru, které byly využity při realizaci programové části. Podrobnější popis vytvořené aplikace na rozpoznávání gest se nachází v páté kapitole. Jsou zmíněny využité metody a je zde také popsána vytvořená datová sada. Šestá kapitola hodnotí vytvořenou aplikaci. Jsou zde uvedeny výsledky experimentů s programem a konec kapitoly se věnuje úvaze o možnostech pokračování práce.

Kapitola 2

Detekce a sledování objektu

Úloha zpracování videa a detekce objektů v obraze je obtížný úkol z kategorie počítačového vidění. Obraz se může zpracovávat jako celek nebo se lze zaměřit jen na jeho určité vytipované části. Metoda zpracování se volí s ohledem na to, jaký typ informace je třeba z obrazu získat.

Při snaze detekovat gesta ruky se nezískává z kamery pouze obraz pohybující se ruky. V obraze je spousta dalších informací vyplývajících z okolí sledovaného objektu. Jedná se především o množství předmětů rozličných barev a tvarů, které se mohou pohybovat nebo být jinak význačné. Proto je pro danou problematiku žádoucí vybrat vhodný popisovač obrazu, který bude reflektovat zvolený typ vyhledávaného objektu nebo akce. Kvalita vybraných obrazových příznaků (features) značně ovlivní výsledek celé detekce.

Popisovač obrazu (deskriptor) je reprezentován příznakovým vektorem (tzv. feature vector). Jako příznak může být brána jakákoliv společná charakteristika dané skupiny objektů (například barva nebo textura).

Příznaky mohou být děleny podle různých vlastností. První způsob kategorizace je podle toho, zda jsou příznaky extrahovány lokálně, tedy pouze z části obrazu, nebo globálně, tedy z celého obrazu. Další způsob dělení je dle opakování detekce – buď je příznak zjišťován pro každou pozici v obraze či jen pro vybrané body (dense/sparse). Příznaky se dělí také podle podstaty své funkce či způsobu zjišťování. Následující body uvádí některé z těchto skupin [27].

- Histogramy – histogramy gradientů (viz kapitola 2.2).
- Konvoluční – wavelety (Haarovy přáznaky – viz kapitola 2.3).
- Význačné body – MoSIFT (viz kapitola 2.4), Space-time feature (viz kapitola 2.5).

Výběr příznaků záleží především na povaze řešeného úkolu. Tato kapitola se věnuje metodám předzpracování obrazu, detekci objektů a sledování vybraných oblastí. Není zde uveden encyklopedický výčet všech možných způsobů práce s obrazem.

Kapitoly této sekce popisují vybrané metody a postupy, které se používají při rozpoznávání gest. První část této kapitoly se věnuje zpracování obrazu a detekci požadovaných objektů. Jako vyhledávaný objekt je zde zvolena oblast barvy kůže. Pokud je známo jen málo informací o hledaném objektu (například pouze tvar), popisuje se obraz pomocí deskriptorů. Je třeba volit vhodný popisovač scény tak, aby se z něj snadno mohla získat požadovaná informace. Další části této kapitoly rozebírají různé druhy obrazových příznaků. Jsou zde přiblíženy popisné vektory zaměřené na obsah obrazu (kapitoly o Histogramech

orientovaných gradientů a Haarových příznacích). Následují kapitoly o deskriptorech, které se hodí pro zjišťování pohybu v obraze (kapitoly o MoSIFT a Space-time features). Poslední tři podkapitoly této sekce se věnují vybraným metodám vhodným ke sledování trajektorie předmětu ve videu – v tomto případě ke sledování ruky.

Tato sekce neobsahuje encyklopedický výčet všech možných variant pro extrakci příznaků a sledování objektu. Jsou zde uvedeny postupy a metody, které se vztahují k tématu rozpoznávání gest.

2.1 Detekce podle barvy kůže

Vyhledávání částí lidského těla (nejčastěji rukou a hlavy) lze provádět několika způsoby. Tato kapitola popisuje metodu založenou na barvě pixelu.

Cílem souboru metod, založených na detekci barvy pleti, je vytvořit pravidlo, podle kterého se budou rozlišovat pixely, zda mají barvu pleti či nikoli. Tato metrika je definována metodou modelující barvu kůže [34]. Metody lze rozdělit do čtyř kategorií:

- explicitně definovaná oblast kůže,
- neparametrické distributivní modelování pleti,
- parametrické distributivní modelování pleti,
- dynamický distributivní model kůže.

Explicitně definovaná oblast kůže

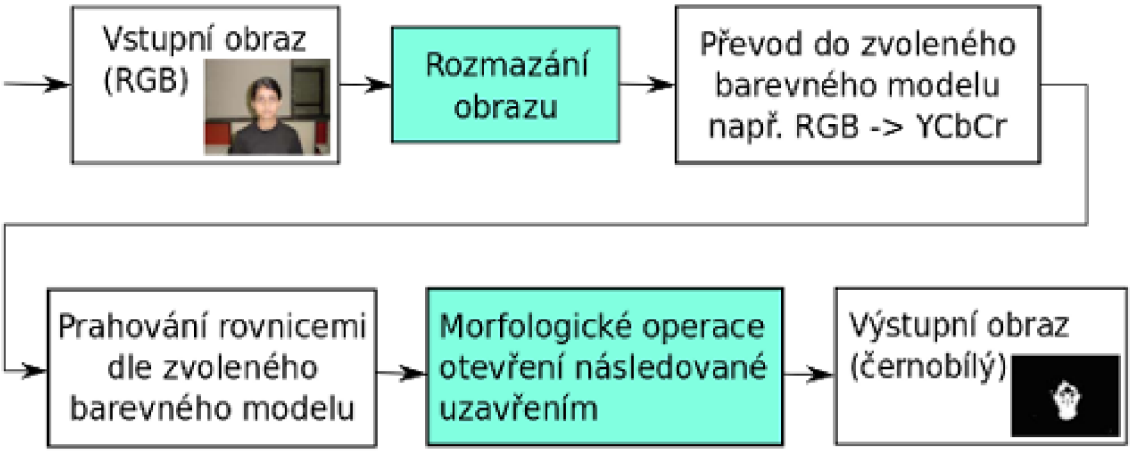
První způsob, jak rozpoznávat barevné pixely, je explicitně definovat rovnice vymezující charakter pleti v určitém barevném prostoru. Vstupní obraz se pak prahuje pomocí těchto rovnic. Nejčastěji se využívají barevné prostory YCbCr, HSV a RGB [32].

Příklad provedení detekce kůže je znázorněn na obrázku 2.1. Ve schématu je zachycen proces vytvoření binární mapy kůže. Obraz získaný z kamery se rozmaže a převede do barevného prostoru YCbCr. Provede se prahování podle vztahu 2.2 [4] a získá se tak mapa výskytu kůže v obraze. Vstupní obraz lze převést i do jiného barevného prostoru, například do RGB a následně prahovat rovnicemi 2.1 [34]. Modré části schématu jsou volitelné, ale zlepšují úspěšnost detekce.

$$\begin{aligned} &R > 95 \quad \wedge \quad G > 40 \quad \wedge \quad B > 20 \quad \wedge \\ &max(R,G,B) - min(R,G,B) > 15 \quad \wedge \\ &|R - G| > 15 \quad \wedge \quad R > G \quad \wedge \quad R > B \end{aligned} \tag{2.1}$$

$$80 \leq Cb \leq 120 \quad \wedge \quad 133 \leq Cr \leq 173 \quad kde\ Y, Cb, Cr = [0, 255] \tag{2.2}$$

Výhoda této metody tkví v tom, že rovnice pro rozpoznání kůže jsou jednoduché a tím je celý algoritmus velmi rychlý. Zmíněné rovnice jsou však také nedostatkem této metody, neboť je obtížné přesně definovat rozmezí barevnosti oblastí s pletí. Vhodný barevný model a rovnice se musí často pro řešený problém upřesnit empiricky.



Obrázek 2.1: Schéma algoritmu detekce barvy kůže, částečně převzato z [32].

Neparametrické distributivní modelování pleti

Hlavním principem rozpoznávačů této kategorie je snaha odhadnout rozložení barvy kůže z trénovacích dat bez explicitního vytvoření modelu barvy kůže. Výsledek těchto metod je někdy označován jako tzv. pravděpodobnostní mapa barvy pleti (Skin Probability Map – SPM). Hlavní výhoda těchto metod spočívá v rychlém trénování a použití modelu. Nevýhodou metod je vysoká paměťová náročnost a neschopnost interpolovat či generalizovat trénovací data [34].

Nejznámější zástupce této kategorie je Bayesovský klasifikátor, který pracuje s podmíněnou pravděpodobností pozorování kůže. Rovnice určující pravděpodobnost pozorování kůže vzhledem ke konkrétní hodnotě barvy c lze vyjádřit rovnicí 2.3. Proměnná $P(c|skin)$ v rovnici značí pravděpodobnost pozorování barvy c za předpokladu, že se jedná o barvu kůže $skin$.

$$P(skin|c) = \frac{P(c|skin)P(skin)}{P(c|skin)P(skin) + P(c|\neg skin)P(\neg skin)} \tag{2.3}$$

$P(c|skin)$ a $P(c|\neg skin)$ se přímo spočítají z histogramu barev kůže a barev okolí podle rovnice 2.4, kde $skin[c]$ je hodnota prvku histogramu odpovídající barevnému vektoru c a $Norm$ je koeficient normalizace.

$$P_{skin}(c) = \frac{skin[c]}{Norm} \tag{2.4}$$

$$P(skin|c) \geq \Theta \tag{2.5}$$

Následuje pravidlo pro detekci kůže 2.5, kde Θ je hodnota prahu. Článek [34] uvádí, že z vykreslené ROC křivky pro $P(skin|c) \geq \Theta$ lze vyčíst invarianci vůči volbě předchozí pravděpodobnosti (díky povaze Bayesova modelu). Tudíž je $P(skin)$ ovlivněna pouze volbou prahu Θ.

Parametrické distributivní modelování barvy pleti

Do této kategorie metod spadají následující metody:

- Single Gaussian – modeluje barvu lidské kůže pomocí normálního rozložení pravděpodobnosti,
- Gaussian Mixture model – je lineární kombinace několika normálních rozložení,
- Eliptic boundary model – za použití elipsy vytváří model barvy kůže, dává přitom stejný výsledek jako normální rozložení a prahování dohromady.

Metody této kategorie využívají pouze barevné složky modelu barevného prostoru, jasová hodnota se zanedbává. Více podrobností se lze dočíst v [34].

$$p(c|skin) = \frac{1}{2\pi|\Sigma_S|^{1/2}} \cdot e^{-\frac{1}{2}(c-\mu_S)^T\Sigma_S^{-1}(c-\mu_S)} \tag{2.6}$$

Rozložení barev u Single Gaussian lze popsat rovnicí eliptické Gaussovy sdružené funkce hustoty rozložení pravděpodobnosti 2.6, kde c je vektor barvy, μ_S je vektor průměru a Σ_S je kovarianční matice. Tyto parametry modelu se získávají z trénovacích dat rovnicemi 2.7 a 2.8, kde n je celkový počet vzorků barev kůže c_j. Hodnota pravděpodobnosti $p(c|skin)$ pak může být použita k rozhodování o barvě pixelu, zda je to barva kůže či nikoli.

$$\mu_S = \frac{1}{n}\sum_{j=1}^{n}(c_j) \tag{2.7}$$

$$\Sigma_S = \frac{1}{n-1}\sum_{j=1}^{n}(c_j - \mu_S)(c_j - \mu_S)^T \tag{2.8}$$

Single Gaussian je využit pro detekci kůže v článku [25]. Tento způsob detekce kůže se využívá také ve spojení s neuronovou sítí (viz kap. 3.3) pro detekci statických gest [22].

Dynamický distributivní model kůže

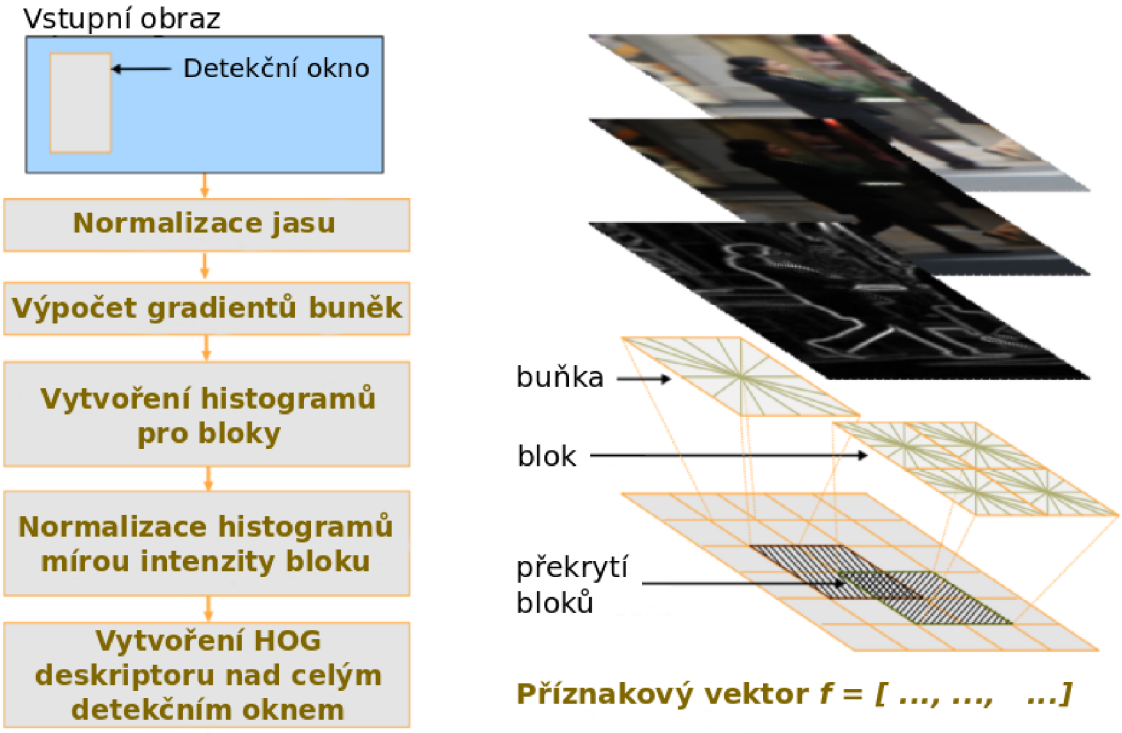
Statický model barvy funguje dobře s jednou kamerou a za stálých podmínek. Pokud se použije jiná kamera nebo se změní osvětlení, tak je model potřeba přetrénovat. K přetrénovaní barevného modelu kůže lze použít například detektor obličeje založený na Haarových příznacích (detekce tvaru). Tento postup vytváření modelu kůže se od ostatních metod liší v několika ohledech.

- Model kůže může být více konkrétní, zaměřen na daný pozorovaný případ či přímo na osobu a osvětlení scény.
- Inicializovat model je možné až když je obličej odlišen od pozadí (jiným klasifikátorem či ručně).

Vzhledem k těmto vlastnostem lze dosáhnout při rozpoznávání větší úspěšnosti detekce pixelů barvy kůže a menšího počtu falešných poplachů. Z vlastností modelu vyplývá, že běží v reálném čase a je nenáročný na výpočetní výkon. Jelikož se model sám aktualizuje, může se barevné rozložení sledované kůže měnit v čase (změna osvětlení). Mezi zástupce této kategorie patří neparametrická Normalized lookup table (LUT), online Expectation Maximization, dynamic histograms a Gaussian distribution adaptation [34].

2.2 Histogramy orientovaných gradientů

Histogram orientovaných gradientů (HOG) je globální metoda pro popis obrazu. Pracuje se změnami jasu, které jsou největší na hranách objektů. Metoda vypočte gradienty ve zpracovávaném rámci a dále určí jejich intenzitu a směr [13]. Schéma průběhu metody je vidět na obrázku 2.2.



Obrázek 2.2: Schéma získávání HOG příznaků, převzato a upraveno z [14].

Prvním krokem metody je předzpracování obrazu. Provádí se normalizace jasu a barvy v obraze (nad obrazovými bloky), aby se docílilo lepší schopnosti vyrovnat se se změnami osvětlení a stínování [13]. Pro získání deskriptoru se obraz nejprve pomyslně rozdělí do bloků typicky o velikosti 16x16 pixelů. Tyto obrazové bloky se však z poloviny překrývají, takže se dělí ještě na 4 podbloky zvané buňky (cells) o rozměrech 8x8 pixelů. Nad každou buňkou se spočítá 1-D histogram orientace gradientů. Typicky se orientace dělí do 9 kategorií (bins) po 20°. Pokud nějaký gradient nelze přesně zařadit do skupiny, jeho hodnota se mezi danými dvěma kategoriemi interpoluje.

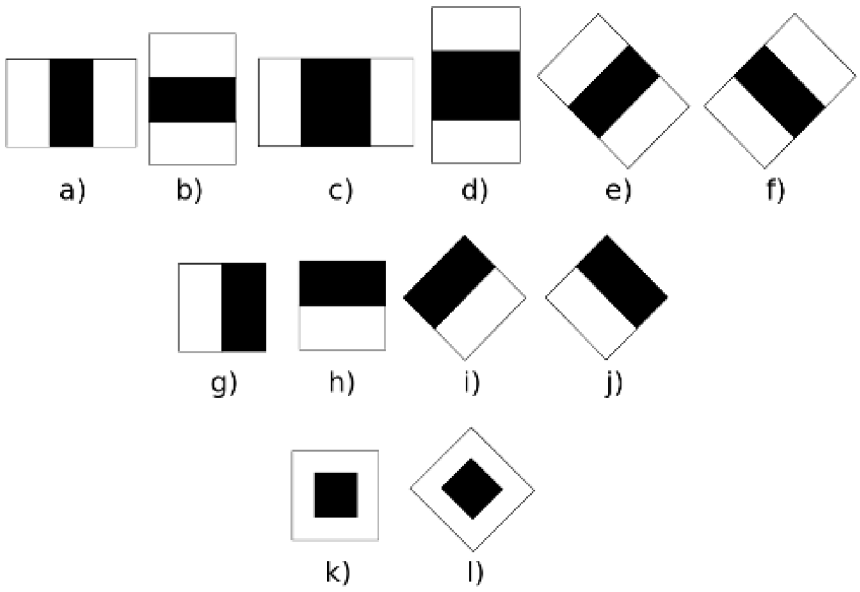
Po získání histogramů z každé buňky obrazu se vytvoří histogram pro celý blok, který vznikne složením dílčích histogramů. Následuje další normalizace buněk, kde se každá normalizuje hodnotou míry intenzity celého bloku [33]. Výsledný deskriptor obrazu vznikne konkatenací všech jednotlivých histogramů bloků obrazu.

Tato metoda se využívá ve spojení s SVM klasifikátorem (viz kapitola 3.5) pro detekci chodců v obraze. HOG příznaky se v kombinaci se svými modifikacemi, jako jsou například HOD nebo HOF příznaky nebo s jinými příznaky, používají také pro detekci gest. Pro více informací vizte [46, 51].

2.3 Haarovy příznaky

Haarovy obrazové příznaky představili pánové Viola a Jones jako součást klasifikační procedury pro detekování objektů v obraze [35]. Příznaky se v navrženém postupu používají ve spojení s kaskádou klasifikátorů natrénovanou pomocí metody AdaBoost (viz kapitola 3.4). Výpočet Haarových příznaků je založen na sčítání a odčítání intenzit částí obrazu, které se nacházejí pod Haarovým příznakem namapovaným na oblast obrazu. Ukázka příznaků je k vidění na obrázku 2.3.

Haarovy příznaky lze rozdělit do tříd podle toho, jaký objekt se od detekce požaduje. Záleží přesněji na povaze částí, ze kterých se objekt skládá. Na obrázku 2.3 je prezentována třída čárových příznaků (obrázky a až f), třída hranových příznaků (obr. g až j) a třída středových příznaků (obrázky k a l). Tyto příznaky se využívají i rotované, nejčastěji pod úhlem 45° [21].

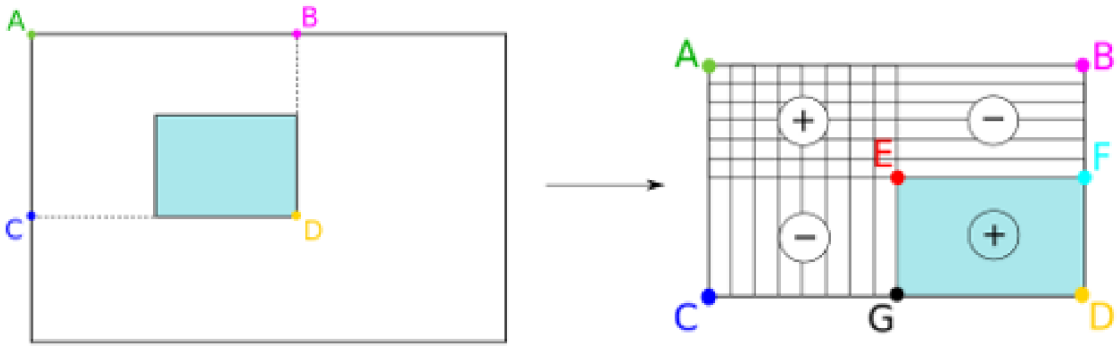


Obrázek 2.3: Příklad reprezentace Haarových příznaků, a - f třída čárových příznaků, g - j hranové příznaky, k a l středové příznaky.

Detekce Haarovými příznaky začíná volbou příznaku. Ten se vybere dle typu informace, která se má z obrazu získat. Následně se zvolí minimální velikost příznaku a minimální hodnota jeho posunu v prohledávacím okně [35]. Princip detekce je poté následující.

1. Nastaví se velikost příznaku.
2. Nastaví se velikost posunu v okně.
3. Opakuje se pro každou polohu prohledávacího okna. Příznak se iterativně posune v okně na každou možnou pozici a pro každou se vypočte hodnota Haarova příznaku. Výsledek se uloží.
4. Pokud se příznak posunul přes celé okno, jde se na bod 5, jinak na bod číslo 2.
5. Pokud je příznak větší než prohledávací okno, jde se na bod 6, jinak na bod 1.

6. Ukončení prohledávání a vyhodnocení uložených výsledků.



Obrázek 2.4: Náčrt výpočtu pomocí integrálního obrazu. Požadovanou sumu intenzit pro modrý obdélník lze spočítat rovnicí $\Sigma_{EFGD} = E - F - G + D$.

Samotný výpočet hodnoty Haarových příznaků se provádí podle vah jednotlivých částí modelů. Bílé obdélníky mají v příznacích pozitivní váhu, černé obdélníky negativní. Provede se suma intenzit pixelů pod bílým obdélníkem (nebo více obdélníky) a od této hodnoty se odečte suma intenzit pixelů, které se nacházejí pod černými částmi příznaku [35].

Neustálé přepočítávání sum intenzit v obraze je časově náročné. Proto se pro urychlení zpracování převádí obraz do takzvaného integrálního obrazu. Tento obraz obsahuje na každé své pozici (dané souřadnicemi x a y) součet všech intenzit, které se od dané pozice (x, y) nacházejí vlevo a nahoře. Tento výpočet znázorňuje rovnice 2.9 [21].

$$I_{\Sigma} = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(i, j) \tag{2.9}$$

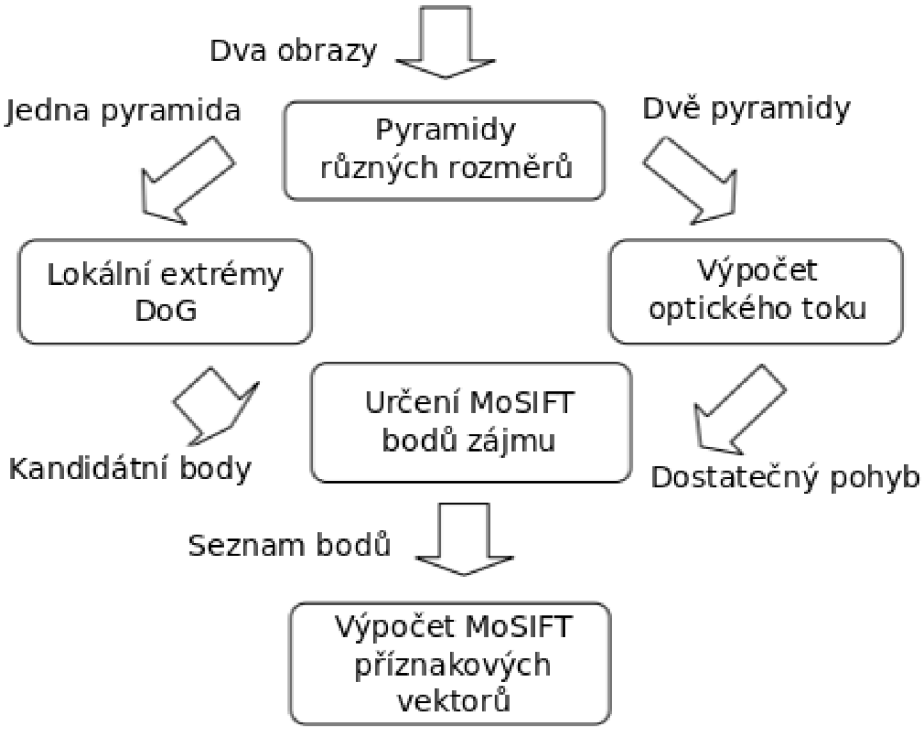
Obrázek 2.4 ukazuje příklad výpočtu sumy intenzit pro určitý (modrý) obdélník, který se nachází uprostřed obrazu.

Haarovy příznaky se využívají na detekci různých objektů v obraze. Nejčastěji jsou využívány jako Viola-Jones klasifikátor k detekci obličeje. Tento klasifikátor lze také využít pro detekci statických gest rukou [20] (popis statických gest je k nahlédnutí v kapitole 3.1).

2.4 Motion SIFT deskriptor

Název metody MoSIFT (anglicky Motion Scale Invariant Feature Transform) vznikl z hlavních principů tohoto algoritmu. Jedná se o zpracování optických toků pomocí SIFT algoritmu.

Metoda MoSIFT je zaměřena na rozpoznání pohybu mezi následujícími obrazy videa. Algoritmus získávání deskriptorů se zakládá na zpracování časoprostorových bodů zájmu. Pokud se však tyto příznaky extrahují ze všech snímků celého videa, je metoda velice časově náročná a nelze ji bez paralelizace provádět real time [10]. Body zájmu se proto běžně extrahují pouze z určité části obrazu, čímž se sníží objem zpracovávaných pixelů bez újmy na informaci o pohybu v obraze. Výsledné MoSIFT deskriptory jsou invariantní v prostorové doméně, ale závislé v čase.



Obrázek 2.5: Schéma algoritmu MoSIFT. Z několika obrázků videa se získají časoprostorové body zájmu. Hlavními postupy v algoritmu jsou SIFT detekce a výpočet optického toku, převzato a upraveno z [9].

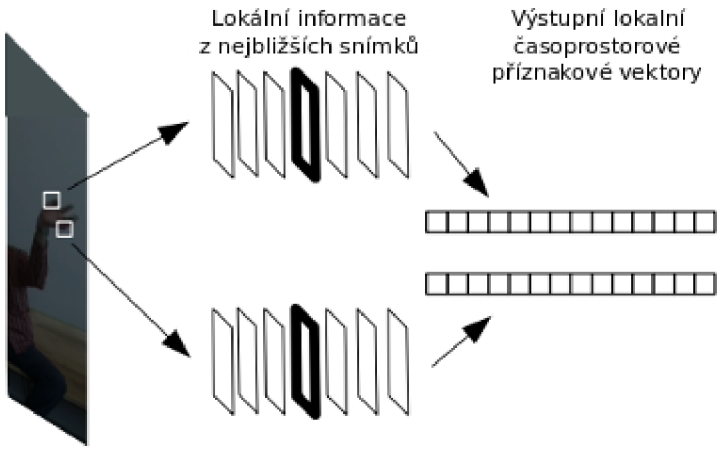
Vizualizace algoritmu je znázorněna na obrázku 2.5. Nejprve se z několika vstupních obrazů vypočítávají deskriptory HOF (Histogram of Oriented Flow) a DoG (Difference of Gaussian image). Výsledky výpočtů ve více rozměrech se ukládají do tzv. pyramidy. Histogramy orientovaného optického toku v obraze se získává podobně jako HOG (viz kap. 2.2). Výsledkem výpočtu je dvojrozměrný vektor rychlosti, který udává směr a velikost rychlosti pohybu [17]. DOG se vypočítává jako rozdíl dvou sousedních rozostřených šedotónových obrazů. Tyto šedotónové obrazy zarovnané vedle sebe v tzv. oktávě vznikají konvolucí nejprve původního obrazu s Gaussovým filtrem. Každý další obraz se vypočítává z obrazu získaného v předešlém kroku.

Po vypočtení DoG se přes něj ve více rozměrech získají lokální extrémy, které se používají jako body zájmu. Algoritmus projde všechna osmi-okolí bodů z vypočtené DoG pyramidy a detekuje možné body zájmu. Pyramidy optického toku jsou vytvářeny nad dvěma Gaussovskými pyramidami. Lokální extrémy z DoG se mohou stát body zájmu jen tehdy, mají-li dostatečný pohyb zaznamenaný v pyramidě optického toku. MoSIFT metoda je pak založena na SIFT algoritmu, kde optický tok v obraze detekuje směr a rozsah pohybu. Takže to lze přirovnat k obrazovým gradientům a můžeme tedy na optický tok aplikovat stejné agregační postupy ke zvýšení odolnosti vůči okluzi a deformaci [9].

Tato metoda se ve spojení s SVM s kvadratickou jádrovou funkcí (viz kapitola 3.5) používá k detekci a rozpoznávání gest rukou, jak bylo popsáno v [10].

2.5 Časoprostorové příznaky

Tento druh příznaků (space-time features) se vyznačuje tím, že se jeho extrakce provádí nad celým vstupním videem nebo jeho určitou částí. Algoritmy založené na extrakci tzv. space-time features se hodí například pro určování lidských činností v obraze [49].



Obrázek 2.6: Vizualizace vytváření časoprostorových bodů zájmu, převzato a upraveno z [50].

Extraktor časoprostorových příznaků produkuje spoustu vektorů. Příklad extrakce je naznačen na obrázku 2.6. Počet těchto vektorů se liší dle videa, ale mají stejnou dimenzi. Ta nezávisí na délce videa. Pro správné rozpoznání je důležité mít optimální délku vstupního videa, neboť jeho délka ovlivňuje kvalitu příznaků [52]. Video je zpracováváno po snímcích. Prostor je reprezentovaný výškou a šířkou jednoho obrazu. Čas je prezentován časovou doménou, která je digitalizována s využitím hodnoty rychlosti pořizování snímků [50].

Extraktor space-time features má 2 části:

1. vyhledávání – prohledávání dimenzí a hledání pozic význačných bodů v čase i prostoru,
2. detekce – zkoumání okolí detekovaného bodu.

Pro rozpoznávání akcí jsou vhodné následující extraktory časoprostorových příznaků.

- STIP extraktor – založen na vyhledávání pomocí Harrisova rohového detektoru. Pro každý ze zachycených bodů je vyhodnocen HOGHOF deskriptor – tzn. příznak složený z HOG (viz kap. 2.2) a HOF příznaků, které jsou jednoduše zřetězené. HOG zachycuje statický vzhled obrazu, HOF zachytí lokální informace o pohybu.
- Cuboid extraktor – založen na 2D Gaussovském jádru a kvadratickém páru 1D Gaborových filtrů. Provádí se potlačení nemaximálních hodnot v obraze a následné prahování, čímž se získá umístění klíčových bodů.
- HesSTIP extraktor – detektor zjišťuje asymetrii v prostoru a čase pomocí determinantu z 3D Hessianovy matice.

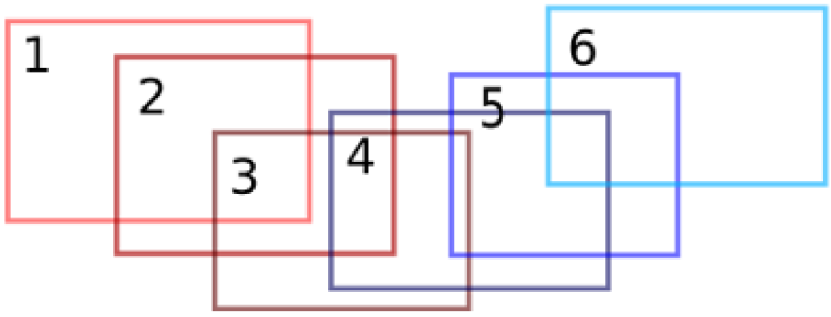
- Dense trajectories extraktor – pro detekci trajektorie používá HOG, HOF, MBx a MBy (MBH deskriptor dělí optický tok na vertikální (MBy) a horizontální (MBx)).

Extraktor obecně pracuje následovně. Nejprve se vyhledávají lokální extrémy přes prostor a čas. Když je extrém detekován, sousední pixely přes prostorovou a časovou doménu jsou použity pro získání příznakového vektoru popisujícího daný extrém [51].

2.6 Sledování objektů

Po určení a vyznačení určité části obrazu, ve které se nachází objekt zájmu, je žádoucí tuto část obrazu sledovat (angl. tracking). Pro sledování pohyblivých objektů (jako jsou například ruce) je třeba použít vhodný tracker. Tzv. tracker je označení pro metody, které jsou schopny sledovat či předpovídat polohu objektu zájmu. Mohou poskytovat také dodatečné informace o objektu, jako je například orientace oblasti či její tvar [45]. Tracker tedy dokáže předpovědět, kam se potenciálně pohne sledovaný objekt. Tím se usnadní identifikace objektu a jeho ztotožnění v novém snímku s objektem z předešlých obrazů.

Jednoduchý a často používaný způsob, jak sledovat objekty v čase, je pomocí sledování překrývajících se oblastí tzv. „Overlapping boxes“. Tato metoda vychází ze zjevného poznatku o pohybu objektů ve scéně (pokud se nepředpokládá extrémně rychlý pohyb sledovaného objektu). Tedy že pohybující se objekt mění svoji polohu pouze tak rychle, že ve dvou po sobě následujících snímcích se tento objekt z velké části překrývá. Princip metody ukazuje obrázek 2.7. Tato metoda je snadná a účinná a lze ji využít při rozpoznávání gest.



Obrázek 2.7: Náčrt metody Overlapping boxes. Obrázek ilustruje očekávané překrytí obalových boxů detekovaných oblastí kůže při pohybu (pohyb naznačují čísla obdélníků).

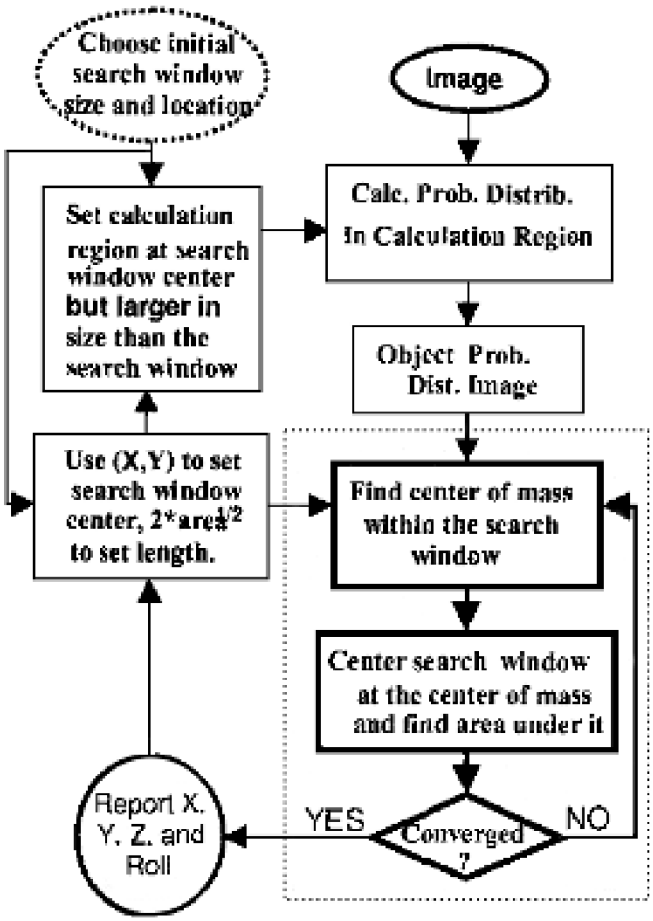
Následující kapitoly popisují vybrané metody sledování objektů. První následující kapitola zmiňuje Kalmanův filtr, což je jedna z nejčastěji využívaných metod pro sledování objektů. Další kapitola přibližuje méně známou metodu CAMShift, která je zaměřená na sledování objektů, které během pozorování mění svou vzdálenost od kamery.

2.7 Continuously Adaptive Mean Shift

Tato metoda zvaná též jako CAMShift byla vytvořena na základě Mean Shift algoritmu. Mean Shift dokáže sledovat objekt pomocí výpočtu pravděpodobnosti pohybu ze všech pixelů cílového objektu. Pokud se ale objekt od kamery vzdaluje a přibližuje, může tak

měnit své barevné rozložení a také se mění jeho velikost. S tímto problémem si už Mean Shift neporadí (více o jeho činnosti a vlastnostech se lze dočíst v [44]). Právě na překlenutí tohoto problému byl vytvořen CAMShift algoritmus. Je to postup, který také využívá barevného histogramu objektu. Je jednoduchý a účinný, proto se často používá na sledování objektů, u kterých se předpokládá změna vzdálenosti od kamery (např. pro sledování obličejů [44]).

Princip algoritmu je skoro stejný jako při vyhodnocení Mean Shift metody. Následující body popisují základní kroky metody CAMShift [6]. Metoda je také znázorněna schématem na obrázku 2.8.



Obrázek 2.8: Princip CAMShift metody, převzato z [6].

1. Určí se velikost a počáteční pozice prohledávacího okna.
2. Vypočítá se pravděpodobnostní rozdělení barev ve vyhledávacím okně.
3. Provede se Mean Shift algoritmus, získají se nové průměrné polohy a velikosti prohledávacího okna.
4. Posune se střed vyhledávacího okna do střední polohy získané v předchozím kroku a opakuje se krok 3.

5. Vypočítá se orientace a měřítko cíle pomocí rovnic 2.10, 2.11 a 2.12.

$$M_{20} = \sum_x \sum_y x^2 I(x,y) \tag{2.10}$$

$$M_{02} = \sum_x \sum_y y^2 I(x,y) \tag{2.11}$$

$$M_{11} = \sum_x \sum_y xy I(x,y) \tag{2.12}$$

Potom jsou délka l a šířka w definovány rovnicemi 2.13 a 2.14,

$$l = \sqrt{\frac{(a+c) + \sqrt{b^2 + (a-c)^2}}{2}} \tag{2.13}$$

$$w = \sqrt{\frac{(a+c) - \sqrt{b^2 + (a-c)^2}}{2}} \tag{2.14}$$

kde $a = M_{20}/M_{00} - x_c^2$, $b = M_{11}/M_{00} - x_c y_c$, $c = M_{02}/M_{00} - y_c^2$. Matice M značí moment prohledávacího okna. Přímý úhel θ je definován rovnicí 2.15.

$$\theta = \frac{1}{2} \cdot arctan\left[\frac{2(\frac{M_{11}}{M_{00}} - x_c y_c)}{(\frac{M_{20}}{M_{00}} - x_c^2) - (\frac{M_{02}}{M_{00}} - y_c^2)}\right] \tag{2.15}$$

S HOG příznaky (viz kapitola 2.2) a HMM klasifikátorem (viz kapitola 3.7) se algoritmus CAMShift využívá na real time detekci gest, jak je popsáno v článku [46].

2.8 Kalmanův filtr

Kalmanův filtr je technika vhodná pro predikci polohy pohybujících se objektů. Jedná se o dynamickou filtraci, která je schopná na základě současného stavu a měření s určitou pravděpodobností předpovědět následující polohu objektu.

Pro správnou funkci filtrace je nutné dodat dobrý model dynamického systému, který chceme sledovat. Z počátku není odhad polohy příliš přesný a může zabírat větší prostor, ve kterém se má objekt nacházet. S postupem sledování systém zpřesňuje své odhady.

$$x_{k+1} = Ax_k + Bu_k + w_k \qquad y_k = Cx_k + z_k \tag{2.16}$$

Filtr pracuje ve dvou krocích, které se neustále opakují. Nejprve je proveden odhad polohy a poté korekce chyby [45]. Princip filtru je následující. Předpokládá se diskrétní systém popsaný rovnicemi 2.16. Proměnná x zastupuje vektor stavu, y je výstup, u značí vstup. A značí matici dynamiky systému, B zastupuje vstupní matici a C je výstupní matice. Proměnná w značí systémový šum a z je šum měření. Výpočet budoucího stavu $\hat{x}_{k+1}$ v čase k vyjadřuje rovnice 2.17 [5].

$$\hat{x}_{k+1} = A\hat{x}_k + Bu_k \tag{2.17}$$

$$\hat{x}_{k+1} = x_{k+1} - \hat{x}_k \tag{2.18}$$

V následujícím čase $(k+1)$ lze zjistit chybu odhadu tak, že se srovná předchozí odhad stavu se skutečnou polohou, viz rovnice 2.18. Z odhadu chyby lze poté získat kovarianční matici dle rovnice 2.19.

$$P_k = E\left[\tilde{y}_{k+1}\tilde{y}_{k+1}^T\right] = CP_kC^T + S_z \tag{2.19}$$

$$\hat{x}_{k+1} = A\hat{x}_k + Bu_k + K_k(y_{k+1} - C\hat{x}_k) \qquad K_k = AP_kC^T(CP_kC^T + S_z)^{-1} \tag{2.20}$$

Když je známa chyba odhadu, je možné usměrnit odhad následujícího stavu pomocí matice K_k, což zmenší kovarianci chyby, viz rovnice 2.20.

Kalmanův filtr se používá pro sledování rozličných předmětů, například článek [26] popisuje využití této metody při sledování rukou při snaze o rozpoznání gest.

Kapitola 3

Rozpoznávání gest

Rozpoznávání objektů zájmu (nebo lidských činností) je složitý úkol. Před započetím identifikace nějakého objektu je nutné si přesně určit a definovat důležité pojmy, které popisují daný objekt zájmu, jenž se bude rozpoznávat. První kapitola této sekce se tedy zabývá vymezením pojmů, které jsou spojené s rozpoznáváním gest. Je zde také zmíněno dělení gest do skupin. Jsou popsány hlavní rysy těchto skupin a jejich typické příklady využití.

Následující kapitoly se věnují metodám strojového učení. První z nich shrnuje rozdělení těchto metod. Následující kapitoly popisují vybrané metody strojového učení, které jsou použitelné při detekci gest nebo při přidružených úkolech (např. při detekci obličeje). Podrobněji jsou líčeny metody vhodné pro detekci gest, jsou zmíněny jejich základní principy a způsoby jejich vytváření. V poslední kapitole této sekce jsou uvedeny příklady existujících aplikací na rozpoznávání gest.

3.1 Identifikace gest

Gesta jsou pro člověka přirozenou součástí komunikace a života vůbec. Člověk tím podporuje své vyjadřování, podtrhujeme svoji náladu. Pro sluchově handicapovaného člověka je znaková řeč, jež je tvořena souborem přesně definovaných gest, jediným možným způsobem dorozumívání s okolím.

Gesto jako takové si lze definovat jako určitou polohu ruky a prstů. Rukou je myšlena koncová část paže. Běžně se vyhledává dlaň (nebo hřbet ruky se zápěstím) a již zmíněné prsty. Obvykle se také očekává, že vyhledávaná část ruky má barvu pleti (tzn. ruka bez rukavice).

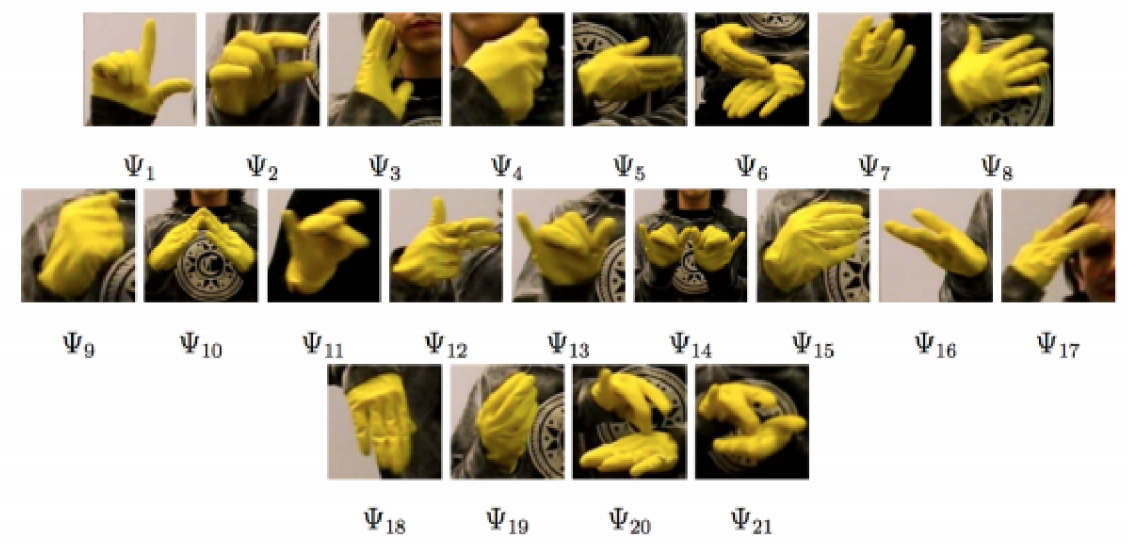
Gesto rukou může ale také nemusí být spojeno se specifickým pohybem. Rychlost či dynamika provedení určitého gesta se většinou nebere při rozpoznávání v potaz, pouze se určuje pohyb jako takový tzn. trajektorie ruky. Problém při rozpoznávání gest je určit, kdy dané gesto započalo a kdy skončilo. Při rozpoznávání pohybu v obraze je také obtížné určit, kdy určité gesto přešlo v jiné.

Gesta se dělí do dvou hlavních kategorií:

- statická gesta,
- dynamická gesta.

Statická gesta

Pojem gesto zde představuje určitou polohu ruky. Může se jednat o polohu pouze dlaně nebo celé paže. Bere se zde v potaz zvláštní natočení ruky, umístění v prostoru i poloha a stav jednotlivých prstů. Tyto gesta mají značné zastoupení ve znakové řeči, kde záleží také na poloze rukou vůči hlavě.



Obrázek 3.1: Příklad statických gest, kde záleží na poloze a natočení ruky a také na postavení a stavu prstů, převzato z [24].

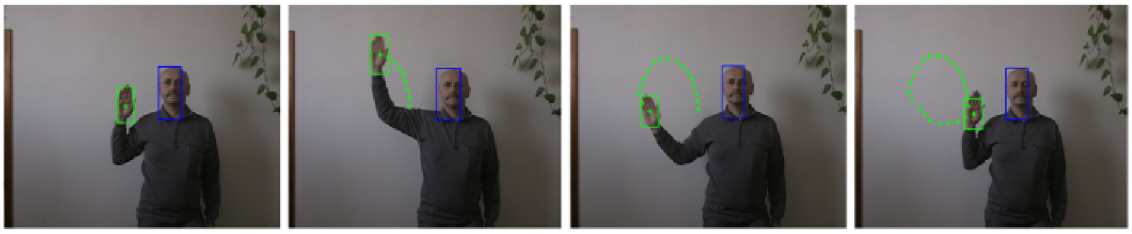
Obrázek 3.2: Příklad statických gest v kontextu okolí, převzato z [24].

Na obrázku 3.1 je příklad statických gest prováděných pouze rukou. Obrázek 3.2 ukazuje spojení gest rukou se zbytkem těla.

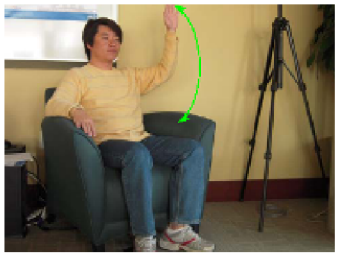
Při detekci těchto gest je snadné určit počátek a konec určité skupiny gest – když skončí pohyb v obraze, lze detekovat statické gesto. Problém je však v jemných rozdílech mezi gesty ruky, například rozdíly v poloze prstů. Pro rozpoznávání tohoto druhu gest se využívá např. neuronových sítí (viz kap. 3.3) [22].

Dynamická gesta

Dynamické gesto lze popsat jako polohu ruky měnící se v čase. Pod pojmem gesto se tedy bude nadále v této práci rozumět specifický pohyb rukou. V tomto typu gest nezáleží na přesném rozložení prstů ruky, stěžejní je zde trajektorie vzniklá pohybující se rukou.



Obrázek 3.3: Příklad dynamického gesta – kolečko.



Obrázek 3.4: Příklad dynamického gesta – zvednutí ruky, převzato z [10].

Příklad dynamického gesta je vidět na obrázku 3.3 a 3.4. Na druhém obrázku je vyobrazeno gesto zvednutí paže. Na prvním obrázku se subjekt snaží vykreslit v prostoru kruh. Z obrázku je patrné, že tyto gesta mají velkou variabilitu a v rozpoznávání se musí počítat s tím, že člověk nedokáže přesně vykreslit či projet rukou určenou trajektorii.

Problém těchto gest byl zmíněn již v předchozích odstavcích. Je zde obtížné určit, kdy dynamické gesto začalo a kdy skončilo, neboť během něho může být ruka v klidu na jednom místě. Tato gesta mohou být porovnávána časově (tzn. může být reflektována rychlost provádění gesta) či nemusí (pro tento typ se využívá HMM, více v kapitole 3.7).

3.2 Metody strojového učení pro rozpoznávání gest

Zpracováním obrazu metodami popsanými v předchozí kapitole 2 se získají vektory příznaků, které reprezentují gesta či jiné činnosti či objekty nacházející se v obraze. O získaných vektorech je třeba rozhodnout, zda-li představují požadovanou akci (případně objekt). Pro lidský mozek je snadné určit, zda předváděný pohyb patří do zamýšlené třídy pohybů, zda se jedná o očekávané gesto či nikoli. Pro počítač je to však komplikovaný problém.

Rozhodovat o vstupních datech může mechanismus vzniklý strojovým učením. Strojové učení je soubor postupných změn vnitřního stavu systému s cílem zlepšit jeho funkcionalitu. Strojové učení lze rozdělit do následujících čtyř kategorií [2].

- Učení sdružením – pracuje s podmíněnou pravděpodobností jevů. Na počátku jsou

dodána vstupní trénovací data s ohodnocením. Dle toho jsou předpokládány určité vlastnosti systému, které se uplatňují při zpracování množiny vstupních dat.

- Učení s učitelem – mechanismus se natrénuje pomocí vstupních dat s ohodnocením. Tato skupina se dělí na dvě část:
  - klasifikace – pro vstupní data vrací název třídy, do které patří,
  - regrese – pro vstupní data vrací jejich číselné ohodnocení.
- Učení bez učitele – na vstup jsou přivedena pouze data (bez ohodnocení). Účelem je najít ve vstupních datech určité zákonitosti, které mají v dodaných datech velké zastoupení.
- Učení posilováním – systém obsahuje zpětnou vazbu a dokáže se učit ze svých předešlých rozhodnutí za účelem zlepšení své činnosti.

Pro rozpoznání gest se využívá metoda klasifikace. Je tedy třeba množina vstupních dat s ohodnocením, o jakou třídu gest (případně obecně objektů) se jedná. Z toho plyne, že klasifikátory rozlišují vždy minimálně dvě třídy. Pokud je zapotřebí rozlišovat pouze jeden objekt od okolí, musí se specifikovat model pozadí (background model).

Pro trénování klasifikátorů je třeba dodat rozsáhlou vstupní množinu dat s ohodnoceními. Pokud by byl vstup málo obsáhlý, klasifikátor by nefungoval obecně. Pro testování je třeba druhá množina dat. Je možné použít ještě další množinu dat k validaci vytvořeného mechanismu. Všechny použité soubory dat by měly být navzájem disjunktní [2].

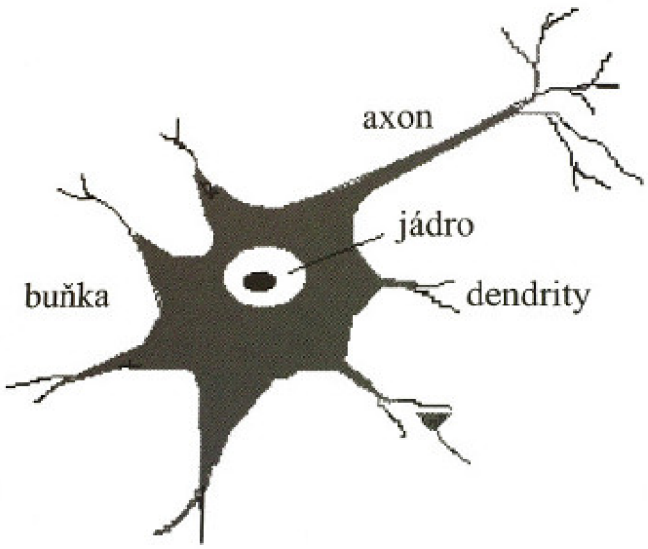
Klasifikátorů, které jsou vhodné pro rozpoznávání gest, je celá řada. Následující kapitoly přibližují vybrané druhy klasifikátorů, o kterých jsem se dočetla v článcích o rozpoznávání dynamických gest.

3.3 Neuronové sítě

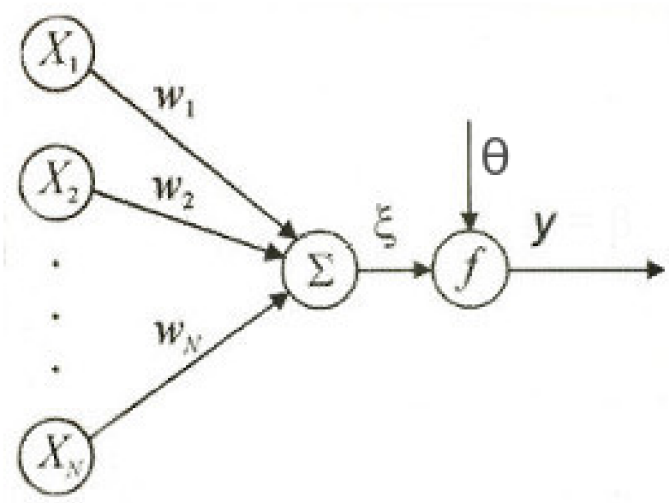
Model umělého neuronu je základním stavebním kamenem neuronových sítí. Tento model je inspirovaný reálnou funkcionalitou biologického neuronu, tedy nejnižšího prvku nervové soustavy. Příklad stavby biologického neuronu je na obrázku 3.5. Biologický neuron je specializovaná buňka schopná zpracovávat elektrické a chemické signály, které přijímá přes svá spojení s ostatními neurony – tzv. dendrity (spojení axonu a dendritu se nazývá synapse). Každý vstupní dendrit neuronu má určitou váhu. V neuronu se všechny vstupy s jejich vahami sumují. Po překročení určitého prahu buzení se neuron stane aktivním a sám vysílá ostatním signály přes svůj axon [28].

Stavba i funkcionalita umělého neuronu je velmi podobná tomu biologickému. Na obrázku 3.6 vidíme nákres stavby umělého neuronu. Ten může mít mnoho vstupů X_i (kde X je označení vstupu a $i = 1, 2, .., N$). Přes tyto spoje X_i vstupují do sumátoru neuronu Σ signály, které jsou ovlivňovány ohodnocením daného propojení. Toto ohodnocení (neboli síla spojení) je vyjádřeno váhou w_i. Výstup sumátoru vytváří vstupní veličinu ξ (zvanou postsynaptický potenciál) pro prahovou funkci f, jejímž parametrem je aktivační práh neuronu θ. Výstupem neuronu je hodnota y, jež můžeme vyjádřit rovnicí 3.1 [28].

$$y = f \sum_{i=1}^{N} (X_i \cdot w_i) + \theta \tag{3.1}$$



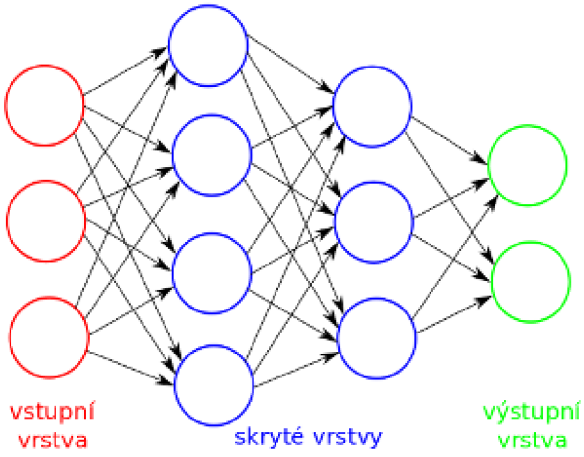
Obrázek 3.5: Příklad biologické stavby neuronu, převzato z [48].



Obrázek 3.6: Model umělého neuronu, převzato z [48].

Umělé neurony se spojují navzájem do sítí, v nichž tvoří vrstvy. Model neuronové sítě vidíme na obrázku 3.7. Typů neuronových sítí je celá řada. Vyobrazenému typu sítě se říká síť s dopředným šířením – výstup neuronů v jedné vrstvě tvoří vstup pro neurony ve vrstvě následující. Počátkem sítě je vstupní vrstva, kam jsou přivedena vstupní data. Následuje libovolný počet skrytých vrstev. Čím více skrytých vrstev síť obsahuje, tím je model složitější. Ve výstupní vrstvě získáme koncovou odezvu na vstupní signál.

Učení neuronových sítí lze provádět několika způsoby – korelačním, soutěživým či adaptačním učením. Nejčastěji se využívá poslední zmiňovaný způsob. Sítě mající zpětnou vazbou se učí následovně. Na vstup se přivedou vektory trénovací množiny. Výstupní odezva sítě se porovnává s požadovanými výstupními vektory. Odchylka vektorů se využívá



Obrázek 3.7: Model neuronové sítě, uzly grafy reprezentují neurony.

k úpravě všech vah neuronové sítě tak, aby se dosáhlo lepších výsledků. Druhá nejpoužívanější metoda – soutěživé učení, pracuje na velice podobném principu, ale výsledná odchylka je odezva jediného (nejsilnějšího a vítězného) neuronu. Odchylkou se pak upravují váhy pouze vítězného neuronu každé vrstvy (případně několika okolních neuronů kolem vítězného neuronu) [5].

Neuronové sítě vycházejí z předpokladu, že funkce lidského mozku a myšlení lze napodobit pomocí modelů, které provádějí základní výpočetní paradigma mozku [48]. Funkcionalita neuronových sítí však zatím nikdy nedosáhla výpočetní rychlosti a kvality práce živočišného mozku. Neuronové sítě se využívají především k rozpoznávání statických gest, například v kombinaci s metodou detekce lidské kůže (viz kapitola 2.1) [22].

3.4 AdaBoost

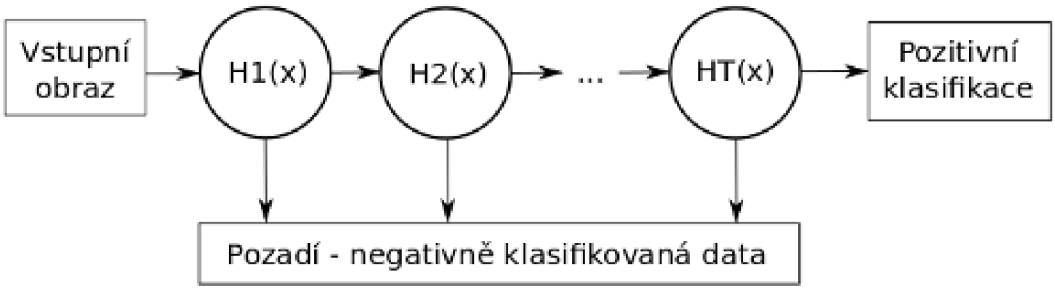
Název metody AdaBoost je zkratkou pro Adaptive Boosting, což je varianta metody strojového učení Boostingu. Hlavní cíl metody je, aby se jakékoliv jednoduché klasifikační metodě zvýšil výkon.

Pomocí metody AdaBoost se vytváří kaskáda klasifikátorů. Takto vzniklý klasifikační nástroj se pak využívá pro rozpoznávání naučených vzorů v obraze. Zpracovávaný obraz se převádí do stupňů šedi, proto je nezávislý na změnách osvětlení [35].

Při trénování klasifikátoru metodou AdaBoost se využívá lineární kombinace slabých klasifikátorů k tomu, aby se vytvořil nelineární silnější klasifikátor. Slabým se nazývá rozpoznávač, který má úspěšnost jen o něco málo větší, než je přesnost odhadu (tzn. klasifikátor musí mít úspěšnost více jak 50%). Obecně se při boostingu využívají za slabé klasifikátory velmi jednoduché funkce – například prahování nebo perceptron [29].

Pro natrénování silného klasifikátoru je třeba mít velkou datovou sadu, která obsahuje spoustu pozitivních i negativních příkladů rozpoznávaného vzoru. Datová sada musí být opatřena ohodnocením, které určuje, do které množiny vybraný obraz náleží – zda do pozitivní nebo negativní. Toto je praktikováno pomocí zápisu dvojice (x_m, y_m), kde $x_m \in X$ označuje jednotlivé obrazy z trénovací množiny a $y_m \in Y$ značí jejich ohodnocení. Množina $Y = \{-1, 1\}$ obsahuje pouze dva prvky, kde 1 značí kladný příklad a -1 označuje pozadí.

Hodnota proměnné p ($m = 1, 2, ..., p$) představuje počet všech obrazů z trénovací sady [5].



Obrázek 3.8: Schéma kaskády klasifikátorů. Jeden stupeň kaskády je znázorněn kolečkem, funkce HT popisuje vybraný klasifikátor.

Slabé klasifikátory se uspořádávají do takzvané kaskády. Jednoduchý model kaskády vidíme na obrázku 3.8. Kaskádu tvoří několik stupňů[1], jejichž počet označíme písmenem T. V každém stupni kaskády se vybírá z velkého množství slabých klasifikátorů ten, který má na trénovací sadě nejmenší chybu. Úspěšnost detekce každého vzoru se zaznamenává ve formě váhy Dt. Při následujícím výběru klasifikátoru jsou pak upřednostněny obrazy s největší vahou (tzn. ty, které se nejhůře rozpoznávají). Tím se docílí výběru optimálního klasifikátoru a snížení chyby detekce [29].

Algoritmus trénování klasifikátoru pomocí metody AdaBoost je následující. Nejprve se musí určit počáteční váhy pro všechny vzorky trénovací množiny podle rovnice 3.2.

$$D_1(i) = \frac{1}{M} \qquad i = 1, 2, ..., M \tag{3.2}$$

Poté se pro každý stupeň kaskády $t = 1, 2, ..., T$ opakují následující body [2, 35].

1. Vyhodnotí se každý slabý klasifikátor h_j z množiny H a vypočte se jeho chyba na váhovaných trénovacích datech, která je dána rovnicí 3.3.

$$\epsilon_j = \sum_{i=1}^{M} D_t(i)\Big[y_i \neq h_j(x_i)\Big] \tag{3.3}$$

2. Z vypočítané množiny se vybere klasifikátor h_t, který má nejmenší chybu ϵ_j. Výběr popisuje rovnice 3.4. Pokud je tato chyba $\epsilon_j > 0,5$, ukončí se cyklus.

$$h_t = arg \min_{h_j \in H} \epsilon_j \tag{3.4}$$

3. Pro vybraný slabý klasifikátor h_t se vypočítá váha v silném klasifikátoru podle rovnice 3.5.

$$\alpha_t = \frac{1}{2} \cdot log\left(\frac{1 - \epsilon_t}{\epsilon_t}\right) \tag{3.5}$$

[1] Původní algoritmus Boostingu, který v roce 1990 představil pan Schapire, pracoval pouze se třemi slabými klasifikátory [2].

4. Spočtou se nové váhy $D_{t+1}(i)$ pro každý prvek i z trénovací množiny M dle vzorce 3.6, kde Z_t je normalizační faktor.

$$D_{t+1}(i) = \frac{D_t(i)^{-\alpha_t y_i h_t(x_i)}}{Z_t} \tag{3.6}$$

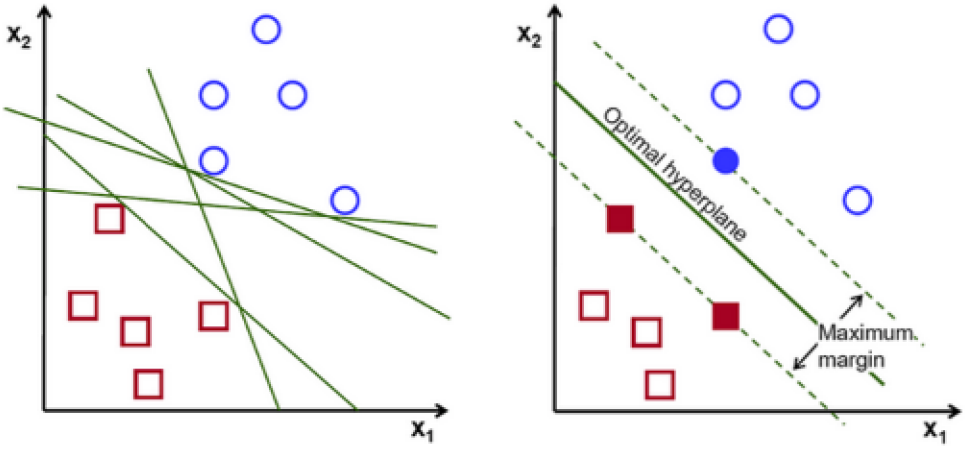
Výsledný klasifikátor, který lze popsat rovnicí 3.7, je rychlý a silnější než využité slabé klasifikátory. Vyznačuje se také tím, že s vzrůstajícím počtem stupňů kaskády exponenciálně snižuje chybu detekce. Tato chyba má tendenci konvergovat k nule. Pro jeho trénování je třeba široká trénovací datová sada. Je však třeba omezit počet průchodů algoritmu, aby nedošlo k přetrénování [7].

$$H(x) = sign\left[\sum i = 1T\alpha_i h_i(x)\right] \tag{3.7}$$

AdaBoost se jako součást Viola-Jones klasifikátoru používá pro rozpoznávání obličejů [35]. Metoda je využívána také pro rozpoznávání gest jak statických [15] tak dynamických [31].

3.5 Support vector machine

Support vector machine (SVM) je jeden z mnoha způsobů rozpoznávání naučených vzorů. Od ostatních metod (např. od neuronových sítí 3.3) se liší tím, že při trénování vždy najde globální minimum (netrpí uváznutím v lokálním minimu) a má snadnou geometrickou interpretaci [8].

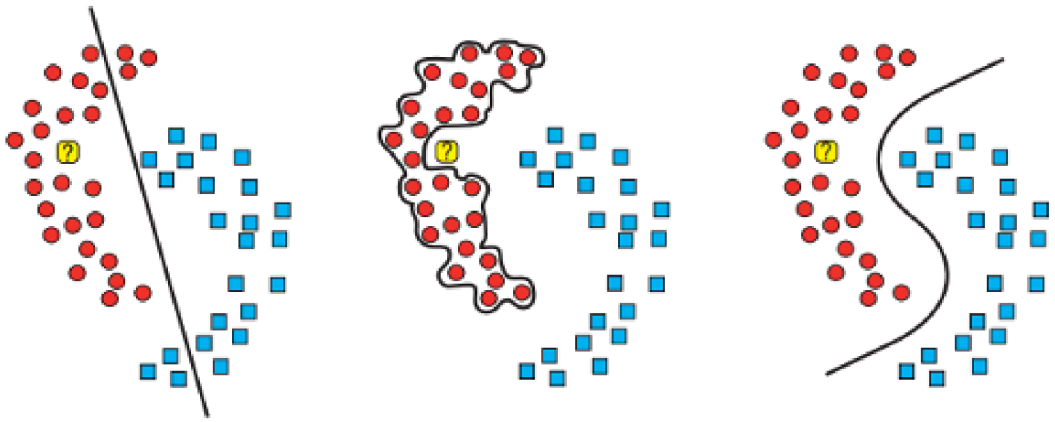


Obrázek 3.9: Příklad lineárního SVM. Obrázek vlevo demonstruje pokus o rozdělení prostoru lineální funkcí. Obrázek vpravo znázorňuje vložení pomocné nadroviny do skupiny vzorků, kde vzdálenost od prvků obou rovin je co největší, převzato z [38].

Prvním krokem metody je rozdělení vstupní množiny dat do dvou[2] tříd. Rozdělení probíhá dle zvolené jádrové funkce. Pokud vstupní množinu nejde triviálně rozdělit, lze provést transformaci jejího prostoru. Transformací se přidá nová dimenze, která umožní oddělit

[2] Existují různé modifikace této metody – např. multiclass SVM. V tomto druhu metody lze data dělit do více tříd.

požadované skupiny prvků [8]. Cílem metody je nalézt takovou polohu nadroviny (při přidání nové dimenze), kde bude vzdálenost prvků jedné i druhé třídy od dělící nadroviny co největší. Vzdálenost nejbližšího prvku jedné množiny od nadroviny by měla být ideálně stejná jako vzdálenost nejbližšího prvku druhé množiny od nadroviny. Tyto body jsou vykresleny plnými tvary na obrázku 3.9 napravo. Z těchto prvků obou množin se získávají podpůrné vektory, které se využívají pro klasifikaci v SVM. Ostatní body se dále při klasifikaci nevyužívají (tyto body jsou na obrázku 3.9 vpravo vyznačeny obrysy koleček a čtverců).



Obrázek 3.10: Příklad dělení vstupních dat jádrovou funkcí pro SVM klasifikátor. Obrázek vlevo prezentuje nevyhovující rozdělení množiny vstupních dat lineární funkcí. Obrázek uprostřed ukazuje přetrénování modelu. Ideálně rozdělená data pomocí sigmoidy ukazuje obrázek vpravo, převzato z [18].

Variabilita SVM se vyznačuje možností výběru jádrové funkce. Tato funkce je použita pro oddělování datových tříd. Na obrázku 3.10 je vidět příklad výběru jádra pro SVM pro množinu vstupních dat. Nejčastěji využívané jádrové funkce jsou [3]:

- lineární jádro – $k(x_i, x_j) = \langle x_i, x_j \rangle$,
- radiální bázová funkce (RBF) – $k(x_i, x_j) = exp\left(\frac{-\|x_i - x_j\|^2}{2{\sigma_0}^2}\right)$,
- polynomiální jádro – $k(x_i, x_j) = (s\langle x_i, x_j \rangle + c)^d$,
- sigmoidální jádro – $k(x_i, x_j) = tanh(s\langle x_i, x_j \rangle + c)$,
- konvexní kombinace jader – $k(x_i, x_j) = \lambda_1 k_1(x_i, x_j) + \lambda_2 k_2(x_i, x_j)$,
- normalizační jádro – $k(x_i, x_j) = \frac{k'(x_i, x_j)}{\sqrt{k'(x_i, x_i) k'(x_j, x_j)}}$,

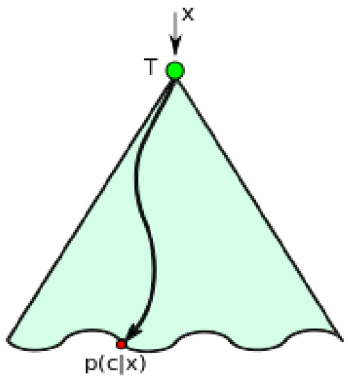
kde s, c, d a λ_i jsou specifické parametry jádra, ${\sigma_0}^2 = mean\|x_i - x_j\|^2$ a $k(x_i, x_j)$ značí jádrovou funkci SVM. Jak je vidět na zmíněném obrázku 3.10, výběr správné jádrové funkce je velice důležitý. Levá část obrázku ukazuje nevhodné dělení dat lineární funkcí. V takovém případě by klasifikátor nebyl příliš úspěšný, neboť funkce nerozděluje množinu bodů ideálně (jednalo by se o podtrénování klasifikátoru). Obrázek uprostřed ukazuje opačný problém klasifikátoru – a to přetrénování modelu. Tento problém je zapříčiněn přílišným upravením

jádrové funkce podle vstupních trénovacích dat (vyobrazeno těsné obalení dat jádrovou funkcí). To znamená, že u vstupních testovacích prvků (které mohou patřit do dané množiny – vyobrazeno žlutým bodem s otazníkem) nelze určit příslušnost do dané třídy. Ideální dělící funkce je ve vyobrazeném případě sigmoida.

Ve spojení s výše popsanými HOG příznaky (viz kapitola 2.2) se lineární SVM používá pro rozpoznávání chodců v obraze [13]. Pro rozpoznávání gest je možné použít SVM s kvadratickou jádrovou funkcí v kombinaci s MoSIFT příznaky (viz kapitola 2.4) [10].

3.6 Classification forest

Jednou z podtříd rozhodovacích stromů[3] je třída classification forest neboli klasifikační les. Jedná se o jednoduchou a snadno paralelizovatelnou metodu klasifikace, která je poměrně robustní vůči šumu a náhodným chybám [2]. Průchod obecným modelem stromu je ilustrován obrázkem 3.11.



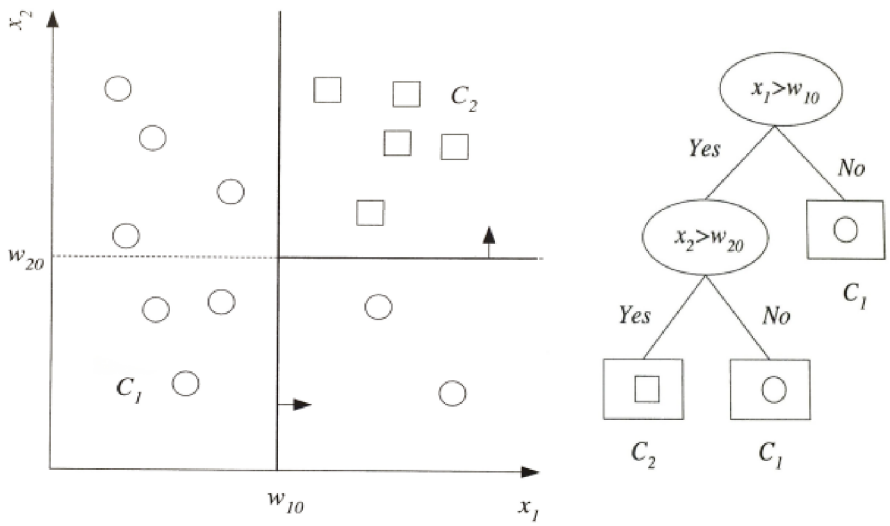
Obrázek 3.11: Model rozhodovacího stromu. Data určená ke klasifikaci x vstupují do kořene stromu T. Z listu stromu se získá pravděpodobnost $p(c|x)$ udávající příslušnost prvku x do dané třídy c.

Tato třída klasifikátorů se skládá z binárních stromů. Každý strom je trénován jednotlivě poměrnou částí trénovací sady [7]. Jeden strom obsahuje rozhodovací uzly a koncové uzly (viz obrázek 3.12). Každý z nekoncových uzlů m obsahuje testovací funkci $f_m(x)$, jejíž diskrétní výstupy označují příslušné výstupní větve uzlu. Při klasifikaci určitého vektoru se začíná porovnávat od kořene stromu. Postupně se prochází větvemi až k listům, kde se získá výsledek vyhodnocení pro příslušný vstup (viz obrázek 3.11). Každý uzel stromu dělí na menší části d-dimenzionální prostor, který do něj vstupuje. Díky tomu je postup stromem poměrně rychlý [2].

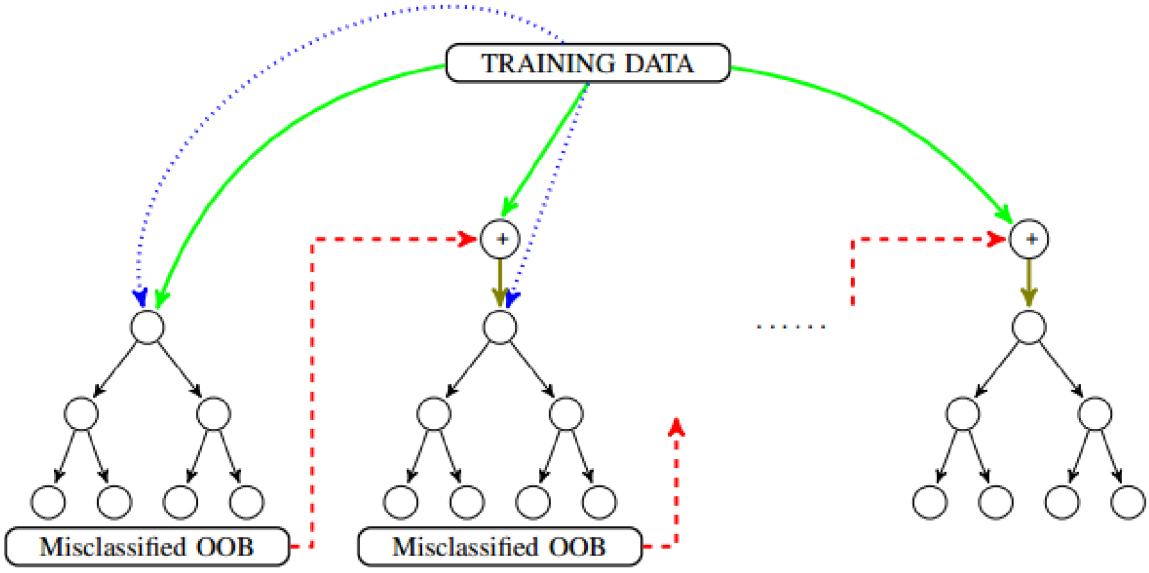
Jak již bylo zmíněno, trénování každého stromu probíhá samostatně. Pro trénování jednoho stromu se použije pouze určité množství dat. Aby se dosáhlo zlepšení funkčnosti klasifikátoru, lze data, která byla špatně klasifikována v jednom stromě, přivést na vstup dalšího. Tím lze docílit zlepšení klasifikace, viz obrázek 3.13. Postup vytváření stromu se obecně řídí následujícím postupem [2].

- Máme dán počet uzlů m. N_m je pak počet instancí trénování nad těmito uzly. Hodnota N_m^i z N_m náleží ke třídě C_i, $\sum_i N_m^i = N_m$. Pak odhad pravděpodobnosti pro

[3] Rozhodovací strom je hierarchická struktura vhodná jak ke klasifikaci, tak k regresi [2].



Obrázek 3.12: Ukázka dat a korespondujícího rozhodovacího stromu. Ovály jsou rozhodovací uzly, obdélníky jsou listy stromu, převzato z [2].



Obrázek 3.13: Ukázka trénování klasifikačního lesa. Zelená šipka značí pozitivní trénovací data. Modrá tečkovaná šipka reprezentuje negativní trénovací data. Červená čárkovaná šipka představuje data, která byla chybně klasifikována, převzato z [23].

třídu C_i je dán rovnicí 3.8.

$$\hat{P}(C_i|x,m) \equiv p_m^i = \frac{N_m^i}{N_m} \tag{3.8}$$

- Uzel m se nazývá čistým, pokud p_m^i pro všechna i dává hodnoty:

  - 0 – pokud žádná vstupní instance v uzlu m nepatří do třídy C_i,
  - 1 – pokud každá vstupní instance v uzlu m patří do třídy C_i.

  Pokud je uzel čistý, není potřeba jej dále větvit a lze přidat pouze koncové uzly (tzv. listy stromu) nazvané jako třída, pro kterou je $p_m^i = 1$.

- Pro měření čistoty uzlu lze používat různé funkce. Uvažujme příklad s dvěma třídami, kde $p^1 \equiv p$ a $p^2 = 1 - p$. $\phi(p, 1-p)$ je nezáporná funkce měření čistoty rozdělení. Funkce musí splňovat následující podmínky:
  - $\phi\left(\frac{1}{2}, \frac{1}{2}\right) \geq \phi(p, 1-p)$ pro každé $p \in [0, 1]$,
  - $\phi(0, 1) = \phi(1, 0) = 0$,
  - $\phi(p, 1-p)$ je rostoucí pro $p \in \left[0, \frac{1}{2}\right]$ a klesající pro $p \in \left[\frac{1}{2}, 1\right]$.

  Pak lze pro funkci na zjišťování čistoty uzly vybrat z následujících možností:

  - Entropie - $\phi(p, 1-p) = -p \log_2 p - (1-p) \log_2 (1-p)$,
  - Gini index - $\phi(p, 1-p) = 2p(1-p)$,
  - Chyba klasifikace - $\phi(p, 1-p) = 1 - max(p, 1-p)$.

Touto metodou lze rozpoznávat statická gesta, jak bylo popsáno v článku [23]. Klasifikační les lze také využít pro rozpoznávání pozice ruky v trojrozměrném prostoru [19].

3.7 Skryté Markovovy modely

HMM je zkratka pro Hidden Markov Models neboli Skryté Markovovy modely. Jedná se o klasifikátor založený na deterministickém konečném automatu. Jeho výstupem je pravděpodobnost (likelihood), jež udává, s jakou určitostí spadají vstupní data do dané třídy.

$$M = (N, M, A, B, \Pi) \tag{3.9}$$

Formálně lze HMM popsat podobně jako konečný automat respektive vztahem 3.9, kde [2]:

- N – značí počet prvků množiny Q, která obsahuje všechny stavy modelu $Q = \{S_1, S_2, ..., S_N\}$. Každý stav má v čase t hodnotu q_t (zápis $q_t = S_i$ znamená, že systém je v čase t ve stavu S_i).

- M – udává počet unikátních pozorování, tj. počet symbolů výstupní abecedy $V = \{v_1, v_2, ..., v_M\}$.

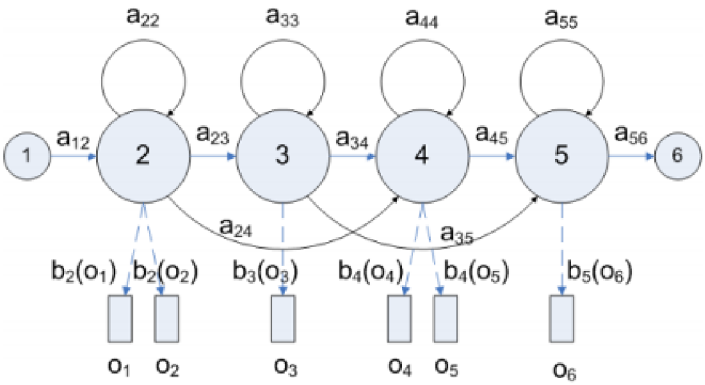
- A – přechodová matice čtvercového tvaru se stranou délky N. Hodnota jednoho prvku matice a_{ij} je pravděpodobnost, se kterou přejde automat ze stavu S_i do stavu S_j v čase q_t. Tento vztah lze zapsat jako: $a_{ij} \equiv P(q_{t+1} = S_j | q_t = S_i)$, přičemž $a_{ij} \geq 0 \quad \wedge \quad \sum_{j=1}^{N} (a_{ij}) = 1$

- B – matice pravděpodobnostních distribucí pro pozorované symboly o velikosti $N \times M$, prvek matice lze popsat jako $b_j(m) \equiv P(O_t = v_m | q_t = S_j)$, kde O značí danou pozorovanou sekvenci $O = \{O_1 O_2 ... O_T\}$.

- Π – značí počáteční pravděpodobnost, kde $\pi_i \equiv P(q_1 = S_i) \quad \wedge \quad \sum_{i=1}^{N}(\pi_i) = 1$

Pravděpodobnost přechodu do následujícího stavu automatu (S_j) závisí pouze na současném stavu HMM (S_i). Přechod mezi stavy prezentuje rovnice 3.10.

$$P(q_{t+1} = S_j | q_t = S_i, q_{t-1} = S_k, ...) = P(q_{t+1} = S_j | q_t = S_i) \tag{3.10}$$



Obrázek 3.14: Příklad Skrytého Markovova modelu. Velké kruhy (označené čísly 2–5) reprezentují skryté stavy modelu. Malé kruhy (označené čísly 1 a 6) značí počáteční a koncový stav. Obdélníky pod grafem značí pravděpodobnostní rozložení b_s generování symbolů o_r pozorované sekvence O, převzato z [26].

HMM si lze představit jako orientovaný graf, jak ilustruje obrázek 3.14. Graf má obvykle přechody jen mezi některými stavy. Přechod je název pro orientovanou spojnici dvou stavů grafu, jež značí diskrétní změnu stavu systému. Každý přechod je ohodnocen přechodovou pravděpodobností. Tato hodnota nám říká, s jakou určitostí systém provede změnu z daného stavu pomocí tohoto přechodu [2]. Všechny přechodové pravděpodobnosti modelu lze vyjádřit prostřednictvím takzvané přechodové matice. Ta nám udává, s jakou pravděpodobností lze přejít z kteréhokoliv vybraného stavu v grafu do jakéhokoliv jiného. Nuly v této matici znamenají, že daný přechod neexistuje. Celá přechodová matice pro obrázek 3.14 ilustrující Skrytý Markovův model je vidět na obrázku 3.15.

$$A = \begin{bmatrix} 0 & a_{12} & 0 & 0 & 0 & 0 \\ 0 & a_{22} & a_{23} & a_{24} & 0 & 0 \\ 0 & 0 & a_{33} & a_{34} & a_{35} & 0 \\ 0 & 0 & 0 & a_{44} & a_{45} & 0 \\ 0 & 0 & 0 & 0 & a_{55} & a_{56} \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Obrázek 3.15: Přechodová matice k HMM z obrázku 3.14 [26]. Číslo stavu přísluší k řádku v matici. Sloupce prezentují stavy, do kterých lze přejít. Na hlavní diagonále matice se nacházejí zpětnovazební smyčky stavů.

Vnitřní stavy HMM, tzn. všechny stavy systému kromě počátečního a koncového stavu, jsou skryty. To znamená, že během zpracování se přesně neví, ve kterém stavu se automat

nachází. Při průchodu systémem (po provedení jakéhokoliv přechodu) však aktuální skrytý stav generuje s určitou pravděpodobností výstupní symbol (na obrázku 3.14 je funkce hustoty rozložení pravděpodobnosti vysílaného symbolu označena jako $b_j(o_i)$ - což je funkce definující pravděpodobnost, že generovaná entita[4] o_i patří ke stavu j). S postupem času[5] je tak vytvářena řada výstupních symbolů, která se nazývá generovaná nebo také pozorovaná sekvence [5]. Vyslání výstupního znaku je tedy (většinou) vázáno na určitý časový úsek. Vnitřní stavy jsou nejčastěji reprezentovány Gaussovským rozložením. Pro stav j lze v určitém čase t rozdělit vektor dat do S nezávislých datových toků o_{st}. Pak lze GMM reprezentující stav vypočítat pomocí vzorce 3.11 [47].

$$b_j(o_t) = \prod_{s=1}^{S} \left[\sum_{m=1}^{M_s} c_{jsm} N(O_{st}; \mu_{jsm}, \Sigma_{jsm}) \right]^{\gamma_s} \tag{3.11}$$

$$N(o; \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \cdot e^{-\frac{1}{2}(o-\mu)' \cdot \Sigma^{-1} \cdot (o\ -\mu)} \tag{3.12}$$

M_s značí počet komponent v datovém toku s. Váha m-té komponenty je zastoupena proměnnou c_{jsm}, μ je vektor středních hodnot a Σ zastupuje kovarianční matici. Proměnná n je dimenze o, γ_s je váha datového toku. $N(\cdot; \mu, \Sigma)$ značí vícerozměrnou Gaussovu funkci, kterou lze spočítat vzorcem 3.12.

Před trénováním Skrytého Markovova modelu je třeba určit počáteční nastavení modelu, tzn. jeho výchozí parametry. Tyto parametry lze „přibližně“ odhadnout (jedná se o střední hodnotu a rozptyl, jež lze poprvé vypočítat pomocí rovnic 3.13 a 3.14). Hodnoty lze odhadnout z toho důvodu, že metody trénování HMM jsou iterativní. V každé iteraci se přepočítávají vlastnosti modelu tak, aby jejich nové nastavení lépe podporovalo pravděpodobnost pozorované sekvence. To znamená, že počáteční nastavení bude přepočítáno a změněno, proto příliš nezáleží na jeho přesnosti. Ideální nastavení parametrů modelu by tedy maximalizovalo pravděpodobnost pozorování posloupnosti znaků generované modelem [47]. Metoda na výběr optimálního nastavení však neexistuje. Je možné dosáhnout pouze lokálního umocnění pravděpodobnosti a to použitím iterační metody trénování. Mezi nejčastěji využívané metody trénování patří Baum-Welchova metoda a Expectation-Maximization.

$$\hat{\mu}_j = \frac{1}{T} \sum_{t=1}^{T} o_t \tag{3.13}$$

$$\hat{\sigma}_j = \frac{1}{T} \sum_{t=1}^{T} (o_t - \mu_j)(o_t - \mu_j)' \tag{3.14}$$

Suma všech sekvencí stavů, které jsou schopny vysílat určitou výstupní posloupnost znaků (tato posloupnost je označena jako „C“), udává celkovou pravděpodobnost generované sekvence C. Každá entita o_t generovaná modelem je popsána pravděpodobností vzhledem ke stavu, ve kterém byla alespoň jednou pozorována, a tedy se podílí na výpočtu parametrů pro tento stav (pro tyto stavy). Zmíněná okupační pravděpodobnost $L_j(t)$ (tzv. state occupation function) vyjadřuje, s jakou určitostí se model nachází v daném stavu j v čase t. Pak lze upravit rovnice 3.13 a 3.14 do váženého tvaru, vizte rovnice 3.15 a 3.16,

[4] Jako entita může být brán vektor nebo symbol.

[5] Čas je zde reprezentován určitým časovým segmentem, nejedná se tedy o jednotku času jako takovou.

což jsou vlastně přepočítávací rovnice Baum-Welchova algoritmu [47].

$$\hat{\mu}_j = \frac{\sum_{t=1}^{T} L_j(t) o_t}{\sum_{t=1}^{T} L_j(t)} \tag{3.15}$$

$$\hat{\sigma}_j = \frac{\sum_{t=1}^{T} L_j(t)(o_t - \mu_j)(o_t - \mu_j)'}{\sum_{t=1}^{T} L_j(t)} \tag{3.16}$$

Po přepočítání středních hodnot a rozptylu je třeba také upravit přechodovou matici. Výpočet nové přechodové pravděpodobnosti pro změnu ze stavu i do stavu j se provede pomocí vzorce 3.17. Aby bylo možné zmíněnou rovnici spočítat, je třeba vyhodnotit pro čas t pravděpodobnost $L_j(t)$ bytí ve stavu j. To lze efektivně provést pomocí algoritmu zvaného Forward-Backward neboli dopředně-zpětná metoda [5].

$$\alpha_j(t) = \left[\sum_{i=2}^{N-1} \alpha_i(t-1) \cdot a_{ij} \right] b_j(o_t) \tag{3.17}$$

$$\alpha_j(t) = P(o_1, o_2, ..., o_t, x(t) = j|M) \tag{3.18}$$

Dopředná pravděpodobnost $\alpha_j(t)$ vypočítaná z modelu M (o N stavech) se definuje zápisem 3.18. Je vidět, že dopředná pravděpodobnost je určena jako sdružená pravděpodobnost pozorování prvních t vektorů a setrvání ve stavu j v čase t. Z rovnice pro dopřednou pravděpodobnost lze vypočítat také celkový likelihood $P(O|M)$ (tzv. Baum-Welchovu pravděpodobnost), jak ukazuje rovnice 3.19.

$$P(O|M) = \alpha_N(T) \tag{3.19}$$

Zpětná pravděpodobnost, označovaná jako $\beta_j(t)$, je definována zápisem 3.20 a lze ji vypočítat podobně jako dopřednou pravděpodobnost rekurzivně pomocí vzorce 3.21.

$$\beta_j(t) = P(o_{t+1}, o_{t+2}, ..., o_T|x(t) = j, M) \tag{3.20}$$

$$\beta_j(t) = \sum_{j=2}^{N-1} a_{ij} b_j(o_{t+1}) \beta_j(t+1) \tag{3.21}$$

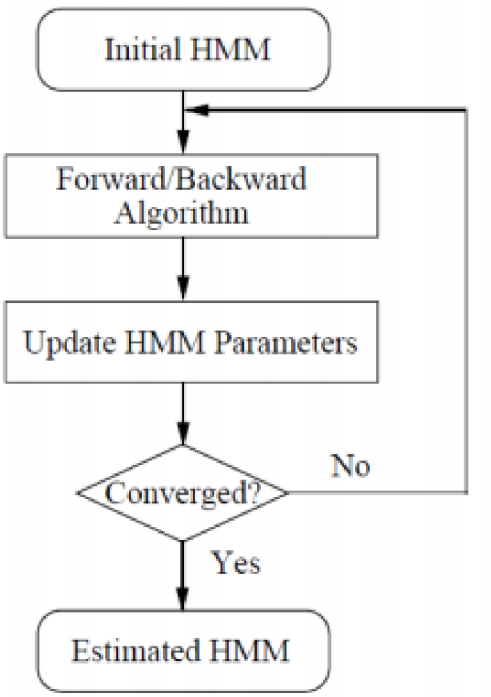
Zpětná pravděpodobnost je (oproti dopředné) podmíněnou pravděpodobností. Tato asymetrie v definicích je záměrná, neboť umožňuje výpočet okupační pravděpodobnosti jako součinu dvou pravděpodobností. Konečný výpočet okupační pravděpodobnosti $L_j(t)$ ilustrují rovnice 3.22 a 3.23, kde $P = P(O|M)$. Pro podrobný postup algoritmu Forward-Backward vizte článek [12] nebo knihy [5, 47].

$$\alpha_j(t)\beta_j(t) = P(O, x(t) = j|M) \tag{3.22}$$

$$L_j(t) = P(x(t) = j|O, M) \equiv \frac{P(O, x(t) = j|M)}{P(O|M)} \equiv \frac{1}{P}\alpha_j(t)\beta_j(t) \tag{3.23}$$

Obecně probíhá trénování HMM iterativně a lze ho provést několika způsoby, např. Baum-Welch algoritmem (jak ukazuje schéma na obrázku 3.16). Následující postup shrnuje důležité kroky trénování.

1. Vstupní data se rovnoměrně rozdělí mezi vnitřní stavy. Z nich se určí rozptyly, střední hodnoty a přechodové pravděpodobnosti.
2. S využitím některého vyhodnocovacího algoritmu se naleznou nová (lepší) rozdělení dat mezi stavy.
3. Pro nové přiřazení dat se určí rozptyly, střední hodnoty a přechodové pravděpodobnosti.
4. Porovnají se nové a předešlé hodnoty rozptylů, středních hodnot a přechodových pravděpodobností. Pokud se liší o více jak zvolený práh, pokračuje se krokem 2., jinak se skončí.



Obrázek 3.16: Schéma trénování HMM pomocí Baum-Welchovy metody, převzato z [47].

Vyhodnocování HMM lze provádět několika způsoby. Často se používá například Viterbiho algoritmus, který prostupuje modelem. Postupně se akumuluje výstupní pravděpodobnost, která se průběžně snižuje. Vždy se vybírá cesta s nejlepší pravděpodobností. Výsledkem je pravděpodobnost (likelihood) s nejlepším ohodnocením [5]. Postup výpočtu částečně maximálního likelihoodu lze vyjádřit rovnicí 3.24,

$$\phi_j(t) = max_i \phi_i(t-1) a_{ij} b_j(o_t) \tag{3.24}$$

kde maximální likelihood[6] je označen jako $\phi_j(t)$ pro stav j v čase t a $o_1 ... o_t$ značí pozorovanou sekvenci. Před započetím výpočtu se definuje výchozí hodnota pravděpodobnosti rovnicemi 3.25 a 3.26 pro $1 < j < N$ [47].

$$\phi_j(1) = 1 \tag{3.25}$$

[6] Tento likelihood je maximální pro danou část výpočtu, pro danou iteraci, nejedná se tedy celkový maximální likelihood.

$$\phi_j(1) = a_{1j} b_j(o_1) \tag{3.26}$$

Celkový maximální likelihood $\hat{P}(O|M)$ je popsán vztahem 3.27.

$$\phi_N(T) = max_i \phi_i(T) a_{iN} \tag{3.27}$$

Neustálé přepočítávání likelihoodu (při trénování i vyhodnocování) vede k tomu, že se toto číslo stále snižuje. Při výpočtech se provádí násobení parciálních likelihoodů. Tyto čísla jsou však malá a násobením se tedy ještě zmenšují. Tento trend vede k překročení kapacitního rozsahu pro uložení čísla v počítačové aritmetice. Aby se zamezilo práci s extrémně malými čísly, převádí se likelihood do logaritmické pravděpodobnosti (na tzv. log-likelihood). Při výpočtu likelihoodu se jeho hodnota zlogaritmuje, čímž lze číslo snadněji uložit. Díky pravidlům pro logaritmování lze také přepsat ve výpočtech součiny logaritmů na součty. Logaritmování se provádí při určování likelihoodu pro každý model, proto lze výsledky poté porovnávat bez újmy na korektnosti výsledku. Pro ukázku slouží rovnice 3.28, kde je uvedena rovnice 3.24 pro výpočet maximálního likelihoodu převedená do logaritmické pravděpodobnosti.

$$\psi_j(t) = max_i \psi_i(t-1) + log(a_{ij}) + log(b_j(o_t)) \tag{3.28}$$

Tento druh klasifikátoru se využívá obecně často. Je vhodný pro zpracování hudby a lidské řeči, například pro rozpoznávání obsahu proslovu a jeho přepisu na text. Dále je tato metoda klasifikace vhodná pro rozpoznávání trajektorií, například pro určování lidských tras chůze v určitém prostředí [26], nebo pro rozpoznávání trasy ruky při gestikulaci (např. při rozpoznávání znakové řeči [24]). Pro detekci gest lze HMM využít s různými druhy příznakových vektorů [46].

3.8 Existující aplikace na rozpoznávání gest

S funkčními aplikacemi na rozpoznávání gest se dnes většina lidí setkává skoro denně. Každý druhý člověk používá například rozpoznávač kreslených gest na chytrém telefonu. Pomocí určených tahů myší u počítače lze gesty ovládat mnohé aplikace (například multimediální program VLC). Berou-li se v úvahu pouze dynamická gesta, jimiž se zabývá tato práce, je možné zmínit následující významné programy.

Aplikace pro překlad znakového jazyka

Po celém světě žijí lidé, kteří jsou neslyšící či nemohou mluvit. Jejich hlavní dorozumívací způsob je znaková řeč. Pro většinu zdravých lidí je však tato řeč neznámá. Proto se s urychlením detekčních metod začaly objevovat aplikace, které dokáží real time[7] rozeznávat gesta znakového jazyka.

Na internetu je spousta dostupných studentských prací na toto téma, které dokáží rozeznat část znakového jazyka. Za zmínku v tomto oboru stojí aplikace od společnosti Microsoft určená pro zařízení Kinect. Tento program dokáže rozeznávat znakovou řeč v aktuálním čase a vypisovat její překlad na obraz [11]. Program obsahuje také modul pro interaktivní komunikaci pomocí znakového jazyka – lze zapsat větu a avatar programu ji převede do znakové řeči a odsimuluje. Ukázka práce s programem je na obrázku 3.17.

[7] Základem znakových jazyků jsou statická gesta určená pro označení často používaných věcí jako jsou předměty denní potřeby nebo čísla [30]. Tyto statická gesta, která jsou definována jen několika obrazy, se rozeznávají poměrně rychle. Jazyk obsahuje také dynamická gesta, která se klasifikují obtížněji a déle. Stejně tak se čas klasifikace prodlužuje, pokud jsou statická gesta spojena ve větě (tzn. dynamická změna polohy ruky).

Obrázek 3.17: Ukázka práce s programem na rozpoznávání jazyka s pomocí Kinectu. Obrázek vlevo ukazuje znakující dívku, kde rozpoznává její řeč. Na obrázku vpravo se provádí překlad zadané věty – avatar programu znakuje vložený text, převzato z [11].

Aplikace pro prohlížení medicínských dat

Pro počítače, tablety i chytré telefony existují programy, kde lze pomocí dotyku posouvat prohlížené fotografie. V oboru lékařství bylo této technologie dříve také využíváno. Na dotykové obrazovce umístěné přímo na operačním sále byly promítány snímky pacienta. Lékař si je mohl prsty posouvat, prohlížet a otáčet. Avšak byl zde problém se sterilizací obrazovek, aby se zabránilo kontaminaci rukavic operatéra [36]. Proto bylo žádoucí vytvořit bezdotykový systém umožňující stejnou funkcionalitu.

Obrázek 3.18: Ukázka z aplikace pro prohlížení medicínských dat. Vlevo je vidět uživatelské rozhraní. Obrázek vpravo ukazuje příklad scény využití – lékař posouvá gestem ruky obraz na monitoru, kamera je označena červeným kruhem, převzato z [36].

Článek [36] popisuje rozhraní používané v nemocnicích v USA. Aplikace rozpoznává naučená dynamická gesta prováděná jednou rukou. Tato aplikace funguje na základě detekce barvy ruky chirurga (viz kapitola 2.1). Lékař po příchodu na operační sál přiblíží ruku ke kameře, aby kalibroval systém. Pro určování pohybu ruky byla využita metoda CAMShift (viz kapitola 2.7). Sada naučených gest je klasifikována pomocí konečného automatu. Příklad aplikace a provádění gesta je na obrázku 3.18.

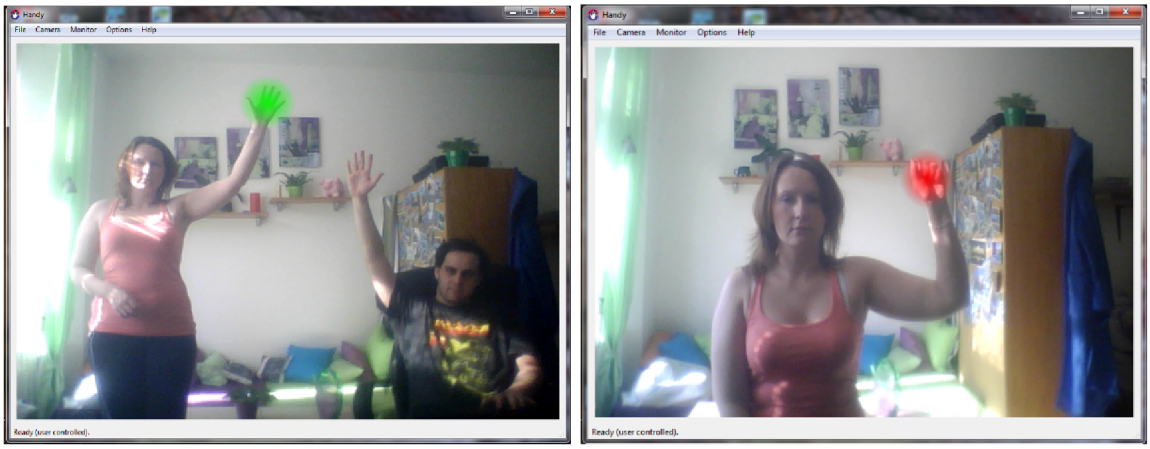
Článek [1] popisuje medicínskou aplikaci s názvem „REALISM“ určenou k rozpoznávání statických gest. Tento program je založený na detekci ruky pomocí Haarových příznaků (viz kapitola 2.3) a rozpoznávání pomocí kaskády klasifikátorů vytvořených metodou AdaBoost

(viz kapitola 3.4).

Aplikace určené na ovládání obrazu na operačním sále by měly splňovat specifické požadavky na ovládání a vlastnosti. Mezi tyto vlastnosti patří především intuitivnost, nenáročnost a rychlost gest, robustnost systému a real time běh aplikace [36]. Tyto parametry mají usnadnit práci s aplikací pro lékaře.

Aplikace pro ovládání počítače

Pro ovládání počítače vzdáleného pár metrů od osoby může posloužit například aplikace s názvem „Handy“ vyvíjená firmou „eyedea recognition“. Na stránkách firmy [40] je volně dostupná omezená testovací verze. Tato aplikace podporující pouze systém Windows běží real time a používá jako vstupní data pouze webkameru.



Obrázek 3.19: Ukázka z aplikace „Handy“, testovaná free trial verze dostupná ke stažení na [40]. Vlevo je příklad uživatelského rozhraní s detekcí jedné ruky (dlaně s roztaženými prsty označené zeleným poloprůhledným bodem). Obrázek vpravo znázorňuje uchycení objektu nebo kliknutí – detekce ruky v pěst, označené červeným poloprůhledným bodem.

Po detekování ruky lze pohybem dlaně ovládat pozici kurzoru (viz obrázek 3.19). Při zatnutí ruky v pěst lze přetahovat aplikace či simulovat kliknutí levým tlačítkem myši. Plná verze slibuje také rozeznávání naučených gest, kterými lze ovládat programy v počítači. Tato verze je však zpoplatněna.

Ovládat kurzor počítače (nebo smart televizi) lze také pomocí přídavných 3D snímacích zařízení, která monitorují prostor kolem sebe. Zařízení (jako například MS Kinect [41] nebo Asus Xtion Pro [39] – viz obrázek 3.20), která kombinují mapovaný trojrozměrný prostor s obrazovými daty, očekávají pro ovládání her nebo chytrých televizí dynamická gesta rukou. Pouhým pohybem prstů lze počítač ovládat pomocí zařízení Leap Motion [42] (viz obrázek 3.21 vlevo). Tento malý přístroj mapuje určitý prostor nad sebou (8 kubických stop) bez získávání obrazových dat. Je velmi přesný, lze pomocí něj pod systémem Windows pracovat například s 3D modelovacím nástrojem Autodesk [42]. Dynamickými gesty lze ovládat PC také pomocí náramku Myo [43] (viz obrázek 3.21 vpravo). Jeho hlavním principem je snímání pohybu svalstva ruky. Obsahuje také bluetooth zařízení, podle kterého určuje svou pozici v prostoru. Všechny přístroje zmíněné v tomto odstavci slouží ke specifickému sběru vstupních dat týkajících se polohy ruky. S koupí těchto zařízení získá

Obrázek 3.20: Ukázka zařízeních pro snímání 3D prostoru. Vyobrazená zařízení kombinují prostorové informace s obrazovými – vlevo MS Kinect, převzato z [41], vpravo Xtion Pro Live, převzato z [39].

Obrázek 3.21: Ukázka zařízeních pro snímání 3D prostoru. Vlelo se nachází zařízení Leap Motion, převzato z [42], vpravo náramek Myo, převzato z [43].

zákazník také speciální knihovny a aplikace, které umožňují práci s daným zařízením[8].

Speciální snímací zařízení i aplikace v počítači ovládané gesty přes obyčejnou kameru většinou fungují na principu rozhraní, přes které lze ovládat další vybrané aplikace (jako například prohlížení fotografií, procházení internetu či přehrávání filmů).

[8] Pro zařízení MS Kinect a Myo existují volně dostupné knihovny podporující systém Linux nebo Android.

Kapitola 4

Zhodnocení stavu a specifikace softwaru

Tato kapitola shrnuje poznatky získané studiem vybraných metod vhodných pro detekci a rozpoznávání gest. Jsou zde zhodnoceny vybrané algoritmy a je uvažováno nad jejich možnou využitelností. Druhá kapitola zmiňuje můj názor na danou problematiku. Je zde popsána motivace k výběru metod, které jsou použity v navržené aplikaci. Závěrem této sekce je přiblížena volba softwaru pro vytvořený program.

4.1 Zhodnocení současných možností

Úkol rozpoznat gesta je možné si rozdělit do několika skupin. Každá tato část pak řeší jeden hlavní problém týkající se zadaného cíle. Úloha je tedy rozdělena na tři sekce. První se zabývá zpracováním obrazu. Druhý segment hodnotí možnosti sledování oblastí zájmu a třetí shrnuje metody klasifikace objektů a akcí.

Zpracování obrazu

První problém, který se řeší při úkolu rozpoznávání gest, je výběr vhodného popisovače obrazu, který by nejlépe a nejrychleji získal požadované informace z obrazu. Před samotným výběrem je však třeba se zamyslet nad povahou obrazu a obrazového snímače, který bude akci monitorovat.

V dnešní době je možné získávat 3D model sledovaného prostoru pomocí vícero dostupných zařízení (například MS Kinect nebo Asus Xtion Pro). Při využití těchto zařízení lze dobře detekovat postavu a její části pomocí kombinace prostorových a obrazových dat (například kombinací HOG a HOD příznaků – vizte [33], nebo přímo pomocí knihoven vytvořených speciálně pro práci s těmito zařízeními – např. OpenNI). Nicméně obyčejné kamery (jako například webkamera na notebooku) jsou stále nejrozšířenější snímací zařízení v domácnostech.

Při výběru popisovače obrazu se musí brát v potaz povaha informací, které je potřeba extrahovat. Dnes je k dispozici široká škála obrazových příznaků, které vychází z různých vlastností hledaného objektu. Dle zaměření lze dělit příznaky do kategorií, jak bylo popsáno v kapitole 2. Významné vlastnosti obrazových příznaků, které jsou popsány v této práci, shrnuje následující tabulka 4.1.

Pro rozpoznání obličeje a jeho částí jsou jedněmi z nejčastěji využívaných Haarovy příznaky [20] (ve spojení s kaskádou klasifikátorů, vizte kapitolu 2.3). Hojně se využívá také

	Lokální	Globální	Hustota detekce – dense	Hustota detekce – sparse	Invariance ke změně velikosti	Invariance ke změně osvětlení	Real time zpracování	Schopnost zachytit pohyb
HOG	ANO	NE	ANO	NE	ANO	ANO	ANO	NE
HAAR	ANO	NE	ANO	NE	ANO	ANO	ANO	NE
MoSIFT	ANO	ANO[1]	ANO	ANO	ANO	ANO*	NE	ANO
Space-time feature	ANO	NE	ANO	ANO	ANO	ANO*	ANO	ANO

* částečně

Tabulka 4.1: Tabulka vlastností vybraných obrazových příznaků.

LBP (Local Binary Pattern). Pro detekci dynamických gest se však Haarových příznaků příliš nevyužívá. Tyto příznaky dávají dobré výsledky při určování jemných vzorů v obraze, proto se hodí pro použití při rozpoznávání statických gest. Pro určení obličeje a rukou se také často využívají metody založené na barvě lidské kůže (viz kapitola 2.1).

V principu jednoduchým popisovačem obrazu je histogram orientovaných gradientů (HOG, vizte kapitolu 2.2). Tento popisovač je vhodný pro detekci kontur v obraze (například při hledání lidské postavy nebo ruky). Samotný se pro detekci gest občas využívá, avšak častěji je využíván v kombinaci se svými modifikacemi (např. s HOD příznaky [33]) nebo s podobnými příznaky založenými na histogramech. Pro detekci různých akcí ve videu včetně gest je HOG kombinován s HOF (histogramy optických toků). S pohybem v obraze pracují také MoSIFT a Space-time deskriptory. První jmenovaný zaznamenává pohyb mezi dvěma následujícími obrazy (viz kapitola 2.4). Pro jeho běh v reálném čase je však třeba paralelizace [10] (kvůli časové náročnosti extrakce HOF příznaků). Space-time features zachycuje pohyb z celého videa (případně z určeného úseku).

Sledování oblastí zájmu

Při rozpoznávání statických gest většinou není třeba detekované oblasti zájmu v následujících obrazech identifikovat. U dynamických gest záleží na výsledné trajektorii určitého bodu zájmu, proto je třeba tento objekt určovat v každém obraze a ztotožnit ho s poslední detekcí. Toto lze provádět pomocí trackerů, viz kapitola 2.6.

Jedním z nejpoužívanějších trackerů je Kalmanův filtr (vizte kap. 2.8), který dokáže dobře určit identitu objektu, pokud systému při inicializaci dodáme dobrý model chování

[1] Lze provádět nad celým obrazem, ale kvůli časové náročnosti zpracování se neprovádí.

hledaného objektu. Další možností je metoda Mean Shift nebo její vylepšení – metoda CAMShift (viz kapitola 2.7). Algoritmus CAMShift je přizpůsoben předpokladu, že se sledovaný objekt bude pohybovat směrem od kamery nebo se k ní přibližovat, proto počítá i s barevným rozložením v prohledávacím okně. Sledování objektů lze provádět i jednoduše metodou Overlapping Boxes, která vychází z částečné znalosti rychlosti pohybu objektu zájmu.

Metody rozpoznávání

K identifikaci konkrétního předmětu nebo akce je třeba vybrat metodu určování, která se pro daný úkol nejlépe hodí. Postupy vhodné pro úkol rozpoznávání gest jsou z kategorie strojového učení – klasifikace. Při výběru konkrétní metody se berou v potaz ty vlastnosti klasifikátoru, které se vztahují k cílovému úkolu (jako je například odolnost k časování akcí při rozpoznávání gest). Důležité vlastnosti klasifikátorů a jejich kombinací s příznaky, které se hodí pro rozpoznávání gest, jsou uvedeny v tabulce 4.2.

	Odolnost vůči časování gesta	**Odolnost vůči šumu/rušení**	**Paralelizovatelnost**	**Běží real time**	**Náchylnost k přetrénování**
Neuronové sítě	NE	ANO	ANO	ANO	ANO
AdaBoost	NE	NE	ANO	ANO	ANO
Clasification forest	NE	ANO	ANO	ANO	NE
SVM	NE	ANO	ANO	ANO	ANO
HMM	ANO	ANO	ANO	ANO	NE
SVM(lineární) + Space-time features	ANO	NE	NE	ANO	ANO
SVM(kvadratický) + MoSIFT	NE	ANO	ANO	ANO[2]	ANO
SVM(lineární) + HOG	NE	ANO	ANO	ANO	ANO
HMM + HOG	ANO	ANO	ANO	ANO	NE
AdaBoost+HAAR	NE	NE	ANO	ANO	ANO
Neuronovcé sítě + detekce kůže	NE	ANO	ANO	ANO	ANO

Tabulka 4.2: Tabulka vlastností vybraných klasifikátorů uvedených v kapitole 3. První část tabulky shrnuje vlastnosti samotných klasifikátorů. Druhá část tabulky ukazuje vlastnosti kombinací klasifikátorů s obrazovými příznaky, které jsou popsány v kapitole 2.

[2] Platí pokud je algoritmus paralelizován, jinak neběží real time.

Pro detekci obličeje v obraze se osvědčilo využití kaskády klasifikátorů, které byly trénovány metodou AdaBoost (viz kapitola 3.4). Tato metoda vytváří silný (primárně) binární klasifikátor kombinací slabých klasifikačních funkcí. Obecně je tato metoda vhodná pro detekci různých objektů [35]. Využívá se pro rozeznávání statických gest [15] v obyčejném obraze či při rozpoznávání dynamických gest ve 3D [31].

Support vector machine (SVM) je silná metoda, jež je v principu podobná kaskádě klasifikátorů. SVM v jeho základní podobě je klasifikátor binární, nicméně s využitím modifikací ho lze natrénovat pro více tříd. Jeho vlastnosti se řídí dle zvolené jádrové funkce [8]. S využitím příznaků popisujících pohyb v obraze ho lze využít pro rozpoznávání gest [10].

Metodě AdaBoost se částečně podobá také postup trénování Klasifikačního lesa. Klasifikační les má srovnatelnou přesnost s metodou AdaBoost, v některých případech dosahuje i lepších výsledků. Je rychlejší než boosting a lze ho snadno paralelizovat [7]. Pro tyto vlastnosti se využívá při rozpoznávání gest [23].

Pro rozpoznání gest se hodí také metoda Neuronových sítí [22]. Tento způsob klasifikace je všestranný, proto se hodí pro rozpoznávání různých objektů i akcí [48]. Často se neuronových sítí využívá například v lékařství, při rozpoznávání poruch spojů nebo pro odhad dynamické stability u energetických systémů [2]. Je zde důležité zvolit správný typ neuronové sítě a optimální počet skrytých vrstev vzhledem k řešenému úkolu.

Skryté Markovovy modely (HMM) jsou jedním z nejčastěji používaných mechanismů pro rozpoznávání gest (viz kapitola 3.7). Je to obecně vhodný způsob pro identifikaci nějaké akce spojené s časovým průběhem. Například se také velmi hojně využívá k rozpoznávání řeči nebo ve verifikačních systémech založených na podpisu osoby [2]. Před trénováním klasifikátoru je nutné zvolit počet vnitřních stavů, které bude automat mít. Tato volba by měla reflektovat složitost vstupních dat, které se budou rozpoznávat [47].

4.2 Motivace k výběru metod

Cílem této práce je prostudovat postupy rozpoznávání objektů a gest v obraze a na základě získaných znalostí navrhnout způsob rozpoznávání gest. V tomto tématu, spadajícím do zpracování obrazu a počítačového vidění, existuje mnoho různých metod pro určování oblastí zájmu. Tyto metody mohou být založené na tvaru hledaného objektu, na jeho barvě, textuře nebo na jiných rozlišovacích vlastnostech daného objektu. Kvůli značné obsáhlosti celého tématu je tato práce zaměřena na detekci a rozpoznávání gest.

Pro rozpoznávání oblastí zájmu je zvolena metoda detekce podle barvy kůže. Tento způsob je vybrán proto, že je jednoduchý a tudíž poměrně rychlý. Pro jeho využití není třeba paralelizace ani žádný přídavný software ani hardware. Identifikace oblastí zájmu s předešlými výskyty je prováděna metodou Overlapping boxes.

Klasifikace zvolených gest je prováděna pomocí Skrytých Markovových modelů. Metoda je vybrána z toho důvodu, že se dokáže dobře vyrovnat s různě dlouhými vstupními vektory (deskriptory obrazu), takže pak částečně nezáleží na dynamice a rychlosti prováděné akce. Proto se hodí pro rozpoznávání gest, neboť každý člověk je při pohybech jedinečný. Ani tentýž člověk neprovede stejné gesto dvakrát stejně (gesto nebude stejně velké ani stejně rychlé a dynamicky provedené).

4.3 Specifikace vlastností a softwaru pro navrženou aplikaci

Na základě studia vybraných metod pro rozpoznávání gest, které jsou uvedené v předchozích kapitolách, jsou pro tuto práci určeny následující postupy a požadavky.

- Aplikace bude rozpoznávat dynamická gesta prováděná lidskou rukou.
- Program bude na základě určených gest ovládat domácí audio-video systém. To bude simulováno ovládáním multimediálního programu v počítači.
- Ovládání multimediálního přehrávače bude probíhat na pozadí, detektor bude běžet paralelně s přehrávačem.
- Vstupní obraz bude získán z obyčejné barevné kamery připojené k počítači.
- Rozpoznávací aplikace bude implementována v jazyce C/C++ a bude podporovat systém Linux.
- Při tvorbě programu (např. pro zpracování obrazu) budou využity pouze volně šiřitelné knihovny (konkrétně knihovna OpenCV).
- V obraze bude probíhat detekce oblastí kůže.
- Aplikace bude pro rozpoznávání gest využívat metodu strojového učení typu učení s učitelem – Skryté Markovovy modely.
- K trénování a testování programu bude vytvořena datová sada. Trénovací datová sada bude opatřena anotacemi.

Výše shrnuté metody a vlastnosti byly určeny ještě před začátkem vývoje aplikace. Při vytváření programu byl využit distribuovaný systém správy verzí jménem Git, což je svobodný software dostupný na stránkách https://about.gitlab.com/.

Kapitola 5

Aplikace na rozpoznávání gest

Následující podkapitoly přibližují návrh aplikace na rozpoznávání gest. Tento návrh se zakládá na specifikacích určených v kapitole 4.3. Jsou zde popsány zvolené postupy, které jsou využity v navržené aplikaci. Všechny postupy jsou detailně rozebrány v kapitolách 2 a 3. Je zde nastíněno propojení jednotlivých částí programu a také celkové využití programu.

Funkčnost je možné demonstrovat buď real time při ovládání počítače nebo na souboru videí vytvořeném v rámci této práce, který je popsán v poslední podkapitole této sekce. Závěrem jsou popsána a ilustrována gesta, která byla zvolena pro ovládání multimediálního programu.

5.1 Celkový pohled na program

Účel tohoto programu je rozpoznat gesta prováděná před kamerou. Jako snímací zařízení je brána obyčejná, snadno dostupná kamera, kterou lze připojit k počítači nebo domácímu kinu (alternativou je také webkamera).

Program rozpoznává skupinu navržených gest. Určená gesta jsou obsažena ve vytvořené trénovací a testovací datové sadě. Rozpoznávají se dynamická gesta prováděná v prostoru kolem člověka, především vedle jeho těla. Více o zvolených gestech je uvedeno v poslední podkapitole této kategorie. Soubory s gesty v datové sadě jsou opatřeny anotací ve formátu „xml“. Gesta mohou být prováděna v různé vzdálenosti od kamery. Primárně je navrhovaný program určen pro použití v místnosti. U některých gest bude možné, aby byl gestikulující člověk otočen ke kameře zády. Tato možnost byla přidána proto, aby bylo možné program ovládat například při vaření, kdy není vždy možné se ke kameře otočit čelem.

Aplikace ovládá multimediální program. Je vybrán program Video LAN Client (VLC), neboť je to široce používaný přehrávač se spoustou možných funkcí. Jsou implementovány pouze vybrané příkazy ovládání VLC. Pro video či hudbu to je spuštění, pauza nebo ukončení přehrávání. Dále je možné gesty ovládat nastavení hlasitosti přehrávaného filmu či přepínat mezi kanály (nebo video-stopami).

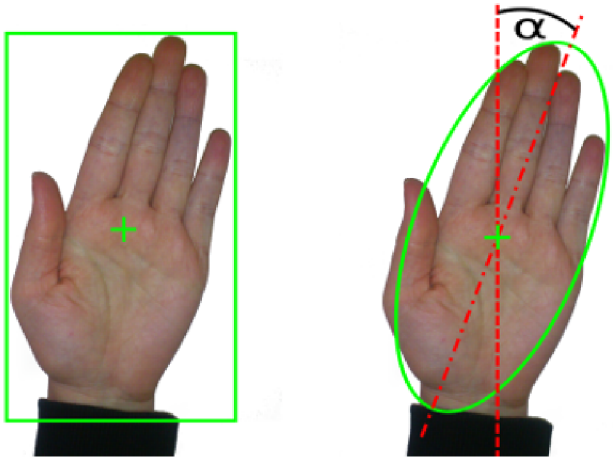
Pro zpracování vstupního obrazu je využita knihovnu OpenCV. Tato knihovna je open-source, pro nekomerční využití zdarma. Program jsem vytvořila v jazyce C/C++ v prostředí Qt neboť s ním mám dobré a bohaté zkušenosti z četných školních projektů.

5.2 Detekce a popis ruky

Rozpoznávají se gesta prováděná rukou. Předpokládá se, že gestikulující člověk má dlaň či hřbet ruky (s nataženými nebo pokrčenými spojenými prsty) proti snímacímu zařízení. Oblastí zájmu je tedy část obrazu, která má barvu pleti.

Kůži lze jednoduše vyhledat pomocí barevného rozložení. Jsou použity explicitně definované rovnice spojené s prahováním obrazu a definice Gaussovského rozložení pro barvu kůže. Vzorky kůže pro tuto metodu jsou brány z vytvořené trénovací datové sady. Obě tyto metody jsou v programu implementovány. Jejich využití se řídí poměrem jasu v obraze.

Při detekci oblastí kůže se vyhledá i obličej člověka, který je natočen čelem ke kameře. Aby se odlišily oblasti rukou od tváře člověka, je v programu využit detektor obličejů. Tento detektor je založen na Haarových příznacích (viz kapitola 2.3) a kaskádě klasifikátorů natrénovaných metodou AdaBoost, tzn. Viola-Jones detektor (vizte kapitolu 3.4). Jelikož tato práce není zaměřená na detekci obličeje, je použit již natrénovaný rozpoznávač tváří z knihovny OpenCV.



Obrázek 5.1: Obrázek vlevo ukazuje obalový obdélník pro ruku. Obrázek vpravo znázorňuje popis ruky elipsou, α značí úhel natočení elipsy.

Z části obrazu s detekovanou oblastí zájmu se získává řada informací. Z těchto vlastností se poté vytváří popisné vektory, které jsou předloženy k rozpoznávání. Vlastnosti určované pro každou oblast zájmu jsou shrnuty v následujících bodech.

- Zjistí se poloha dané oblasti. Uloží se souřadnice jejího středu, které jsou při pohybu ruky použity pro výpočet derivace polohy.
- Uloží se výměra dané oblasti zájmu (v pixelech). Při pohybu se zaznamenává také změna rozlohy dané oblasti.
- Každá oblast je ohraničena obdélníkem (boundingboxem), viz obrázek 5.1 vlevo. Tento obalový box je pak předložen detektoru obličejů. Detekcí pouze nad malou částí obrazu se značně urychlí určování, zda-li v obraze obličej je či není. Dle výsledku detekce je nastaven příznak říkající, jestli je daná oblast zájmu obličej nebo ruka.
- Každá oblast zájmu je popsána elipsou, viz obrázek 5.1 vpravo. Pro určování gest je brán v potaz úhel natočení elipsy při daném gestu.

S kombinací získaných parametrů, které jsou popsány výše, byly prováděny experimenty a bylo sledováno, jak se mění úspěšnost detekce pro jednotlivá gesta. Podrobný popis experimentů a výsledků je k nahlédnutí v kapitole 6.2.

5.3 Sledování pohybu

Pro zjištění pohybu oblastí zájmu je využita metoda s názvem Overlapping boxes (viz kapitola 2.6). Tuto metodu zde lze dobře využít, neboť se neočekává, že člověk dokáže mávat rukou tak rychle, aby pohyb nebyl pro kameru zachytitelný (bereme-li v potaz obyčejnou kameru s průměrně 30fps).

Určitý počet detekovaných objektů zájmu se udržuje v paměti ve vektorech odpovídajících objektům. Po identifikaci nově detekované oblasti s nějakou oblastí, která byla nalezena dříve, se tento nový záznam o posunu objektu zaznamená v příslušné skupině. To znamená, že pro každý objekt zájmu ve scéně je jeden vektor, v němž jsou uloženy záznamy o jeho posledních pozicích a vlastnostech. Díky tomu je možné omezeně předpovídat pohyb objektu.

Při sledování se automaticky ihned dopočítávají derivace polohy a rozlohy objektu. Tyto informace se vypočítávají od poslední předcházející detekce a jsou uloženy jakou vlastnost současného objektu.

5.4 Klasifikace trajektorií

Na rozpoznávání gest je využita metoda Skrytých Markovových modelů. Tento postup je zvolen proto, že je to vhodný nástroj pro rozpoznávání akcí s proměnlivým časem trvání v rámci jedné třídy akcí. Pro zpracování informací z kamery v delším časovém horizontu je využití HMM také výhodné, neboť jsou tyto modely schopny zpracovat potenciálně nekonečnou posloupnost vstupních příznakových vektorů.

Pro vytvoření jednotlivých modelů pro každé zvolené gesto byl použit Hidden Markov Model Toolkit (HTK). Tento soubor knihoven je primárně určen pro trénování rozpoznávačů řeči. Poskytuje sofistikované nástroje pro tvorbu a testování Skrytých Markovových modelů. Počet vnitřních stavů je nastaven dle výsledků experimentů, které jsou popsané v kapitole 6.1. Při tvorbě modelů se zde využívá Baum-Welchovy reestimace. K vyhodnocení Skrytých Markovových modelů je použit Viterbiho algoritmus. Obě zmíněné metody jsou popsány v kapitole 3.7.

Trénování a testování modelů bylo provedeno na vytvořené datové sadě, která je popsána v následující kapitole.

Při rozpoznávání dynamických gest je obtížné určit, kdy gesto začíná či končí. Nelze se úplně spoléhat na detekci pohybu v obraze, neboť na sebe mohou gesta navazovat. Aby nebylo třeba řešit segmentaci vstupních dat, provádí se v aplikaci spouštění modelů cyklicky po určitém časovém intervalu (po zpracování 80 záznamů). Společně s modely pro vybraná gesta byl vytvořen model pozadí (anglicky background model). Tento model byl natrénován nad celou datovou sadou.

Pokud bylo provedeno gesto, má nejvyšší pravděpodobnost model daného gesta. Dle tohoto výsledku se poté provede požadovaná akce s VLC přehrávačem. Pokud gesto provedeno nebylo (největší pravděpodobnost má model pozadí či nebyl překročen práh detekce), neprovede se žádná akce.

5.5 Datová sada

Pro vytvoření funkční robustní aplikace bylo nutné vytvořit sadu trénovacích a testovacích videí, na kterých se předvádí vybraná gesta. Čím větší počet lidí, kteří provádějí zvolená gesta, je v datové sadě, tím by měla výsledná aplikace fungovat obecněji. Proto jsem vytvořila datovou sadu, která obsahuje videozáznamy od dvaceti lidí (dvanácti mužů a osmi žen). Celá datová sada byla pořízena pomocí kamery JVC Everio Hybrid typ GZ-MG155E.

Trénovací sadu bylo potřeba opatřit anotačním souborem, který definuje počátek a konec gesta. Pro vytváření anotací k video souborům existuje spousta placených i volně dostupných nástrojů. Některé neplacené programy byly v rámci tohoto projektu vyzkoušeny (například ViperGT, Anvil nebo ELAN). Většina z nich má však poměrně složité ovládání, práce s nimi je obtížná a většina nepodporuje video formát „.MOD“ (ve kterém je natočena datová sada). Proto byl pro tuto práci vytvořen nový nástroj pro tvorbu anotací k videím (více podrobností o anotátoru se lze dočíst v kapitole 5.6). Vlastní anotační nástroj umožnil vybrat optimální formát anotací k videím. Pro zápis anotace je zvolen formát „xml“, neboť je to strukturovaný přehledný formát a je dobře čitelný pro stroj i pro člověka. Příklad zápisu gesta v anotačním souboru je k nahlédnutí v příloze D.

Vzdálenost kamery a člověka provádějícího gesta je omezena následovně. Minimální vzdálenost je určena tak, aby se člověk a především manipulační prostor jeho rukou vešli do obrazu. Maximální vzdálenost od kamery je omezena dosahem detekčního zařízení, vyhledávajícího ruce člověka. Program je určen pro využití v místnosti, takže maximální vzdálenost se očekává přibližně do 6 metrů.

Pro vytvoření datové sady bylo navrženo osm gest. Některá gesta jsou si velmi podobná (například *sunutí nahoru* nebo *sunutí dolů*), jiná trochu méně (například *kolo* a *čtverec*). Byla tak určena proto, aby se otestovala rozlišitelnost HMM. Zmíněná velmi podobná gesta nejsou určena pro ovládání aplikace (gesta sunutí a další). Mohou však sloužit pro testování aplikace jako šum na pozadí. Pro ovládání multimediálního programu je tedy využita pouze podmnožina gest z datové sady (konkrétně 5 z 8).

Některá gesta z datové sady se dají provádět zády ke kameře, neboť mají být prováděna vedle těla člověka. To usnadňuje ovládání aplikace. Tato gesta jsou využita pro ovládání hlasitosti programu VLC, neboť hlasitost je nastavení, které regulujeme často a v různých situacích. Gesta, u nichž je požadováno, aby člověk stál čelem ke kameře, jsou využita ke spouštění přehrávání nebo k přepínání kanálů. U těchto činností se očekává, že člověk je zaujat přehrávaným videem a tudíž na něj s největší pravděpodobností hledí, tzn. nemusí se kvůli provedení gesta otáčet na kameru – již natočený je.

Navržená gesta z datové sady jsou prováděna jednou rukou. Tato vlastnost vzešla ze zamýšleného uplatnění programu. Pokud člověk používá navrženou aplikaci při vaření či telefonování, nemá často obě ruce volné. Je třeba poznamenat, že také ne každý člověk má to štěstí mít obě ruce zdravé a schopné pohybu.

Gesta obsažená v datová sadě jsou zevrubně popsána v následujících bodech:

1. *kolo* – vykreslení kola rukou směrem od sebe, začátek a konec gesta zhruba na dvou hodinách,
2. *čtverec* – obkreslení čtverce směrem od sebe, počátek v levém horním rohu,
3. *kříž* – obkreslení kříže pootočeného o 45°, počátek nahoře a směr tažení dolů od sebe,
4. *rotované L* – obkreslení rotovaného písmene L o 180° se zpětným tahem,

5. *mávnutí stranou* – ruka v upažení, narovnaná dlaň, ruka se ohne v lokti a posune se do pravého úhle nahoru a zase zpět,

6. *mávnutí dopředu* – ruka v upažení se ohne v lokti do pravého úhle, prsty směřují nahoru, poté dlaň opíše čtvrtinu kruhu směrem dopředu (ruka se překlopí dopředu o 90°) a zase zpět.

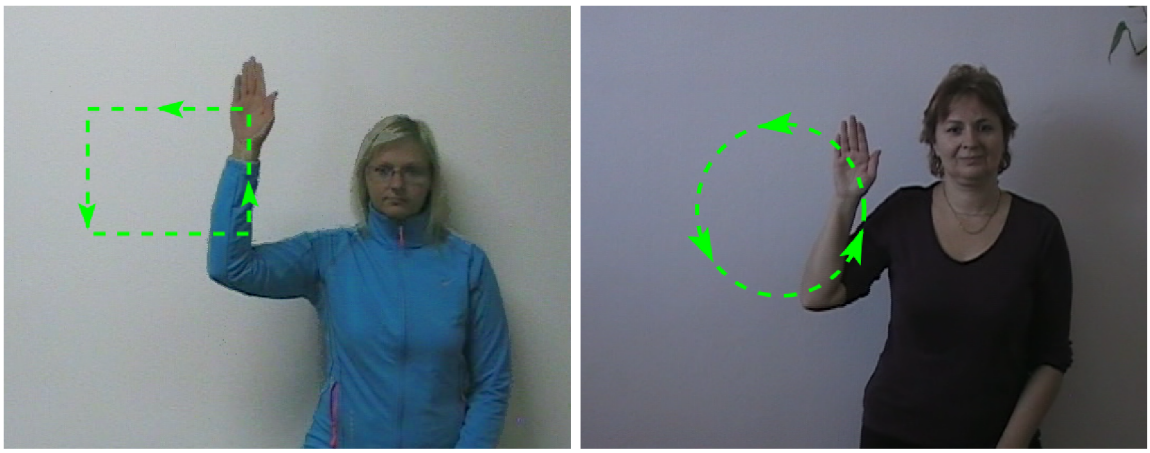
7. *sunutí nahoru* – vytažení ruky nad hlavu (počátek u ramene), dlaň a prsty natažené směřují nahoru, směrem dolů jde ruka zaťatá v pěst,

8. *sunutí dolů* – vysunutí ruky v pěst nahoru (počátek u ramene), dolů jde dlaň a prsty natažené směřující dolů.

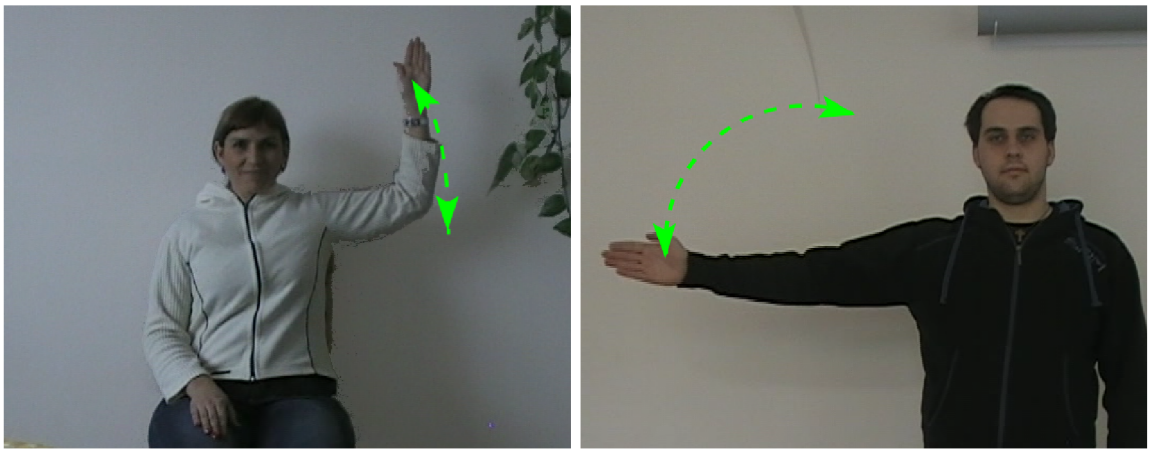
Prvních pět popsaných gest v bodech výše je využito k ovládání aplikace. Následující body určují gesta pro ovládání programu. Zároveň je zde přehled funkcí pro ovládání VLC, které je možné využívat:

- *kolo* – spuštění přehrávání,
- *čtverec* – pauza v přehrávání,
- *mávnutí stranou* – přepnutí na další položku v seznamu pro přehrání,
- *kříž* – ztišení hlasitosti o 10%,
- *rotované L* – zesílení hlasitosti o 10%.

Všechna gesta se musí provádět vedle těla. Ilustrace provádění popsaných gest jsou vidět na obrázcích 5.2 až 5.5. Gesta jsou načrtnuta zelenou přerušovanou čárou. Šipky na čáře naznačují směr provedení.



Obrázek 5.2: Obrázek vlevo přibližuje způsob nakreslení *čtverce* v prostoru vedle těla člověka. Ruka je ve výchozí pozici (gesto pro pozastavení přehrávání). Obrázek vpravo ukazuje gesto *kola*. Ruka je ve výchozí pozici (gesto pro spuštění přehrávání).



Obrázek 5.3: Levý obrázek znázorňuje ženu s rukou ve výchozí pozici pro provedení gesta *mávnutí dopředu*. Pravý obrázek ukazuje muže ve výchozí pozici, kde je naznačen způsob provedení gesta *mávnutí stranou* (gesto pro posunutí v seznamu skladeb).

Obrázek 5.4: Na obrázku vlevo je vidět průběh gesta *sunutí nahoru*. Ruka je zde v polovině provádění gesta, kde bude následovat uzavření v pěst a návrat do výchozí pozice u ramene. Na obrázku vpravo je vidět podobné gesto, které se liší způsobem natočení ruky při provádění. Gesto *sunutí dolů* je zde v polovině provádění, kde následuje návrat rozevřené ruky do výchozí pozice.

Obrázek 5.5: Gesto *kříž* je prováděno na levém vyobrazeném snímku. Ruka je zde ve výchozí pozici (gesto pro ztišení hlasitosti). Gesto *rotované L* je vidět na snímku vpravo. Ruka je také ve výchozí pozici (gesto pro zvýšení hlasitosti).

5.6 Implementace

Hlavní program je rozdělen na tři části mezi nimiž lze přepínat pomocí parametrů při spuštění programu. Manuál k aplikaci je k vidění v příloze B. Manuál k aplikaci na tvorbu anotací je v příloze C.

Detektor gest ovládající VLC

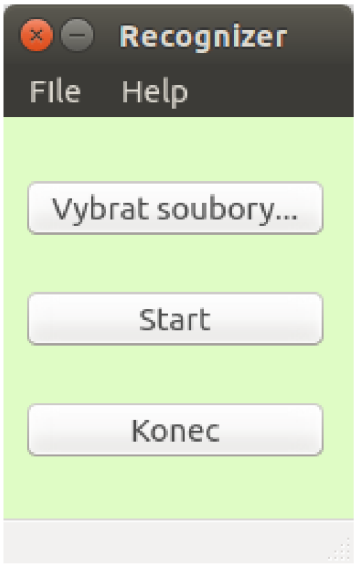
První část aplikace, řízená z třídy `MainWindow`, představuje hlavní funkcionalitu projektu. Jedná se tedy o program, který rozeznává gesta a dle nich provádí příkazy pro program VLC. Po spuštění se objeví jednoduché malé grafické uživatelské rozhraní, jež lze vidět na obrázku 5.6. V něm uživatel vybere soubory, které se mají přehrát. Tlačítko start spustí přehrávání a také detektor gest. Od této chvíle je možné ovládat program VLC pomocí gest, která jsou popsána v kapitole výše. Se spuštěním přehrávání ve VLC programu se otevře také okno ukazující aktuální snímané video z webkamery. Toto okno není spojeno s grafickým uživatelským rozhraním, aby si je mohl uživatel umístit dle libosti. Okno s přenosem kamery je možné skrýt a ponechat pouze ovládací panel.

O ovládání programu VLC se stará třída `movieControll`. Ovládání probíhá pomocí příkazů posílaných přes program telnet a netcast. Je tedy nutné mít na počítači tyto služby dostupné. Spuštění programu VLC lze parametrizovat řadou možností. Prototyp používaného příkazu je vidět v následující ukázce kódu. Pro více informací o možnostech spouštění programu vizte [37].

```
vlc [soubor] -I telnet --telnet-password <heslo>
    [--telnet-host <IP_adresa>] [--telnet-port <port>]
    [--extraintf <nazev_rozhrani>] [--quiet]
```

Používaný spouštěcí příkaz programu je ukázán v následujícím kódu.

```
vlc video.mkv -I telnet --telnet-password xxvlcxx --telnet-host 127.0.0.1
--telnet-port 4321 --extraintf qt4 --quiet &
```



Obrázek 5.6: Ilustrace uživatelského rozhraní vytvořeného programu.

Program VLC je zapnut s klasickým ovládacím menu (tlačítky na spodní liště okna) pomocí přepínače `extraintf qt4`. Tato možnost ovládání byla ponechána z důvodu komfortnosti ovládání programu, neboť nijak nekoliduje s ovládáním pomocí gest. Příkazy určené rozpoznávačem gest se pak zasílají pomocí následujícího příkazu. Příklad ukazuje zaslání požadavku na pozastavení přehrávání.

```
echo  \"xxvlcxx\npause\" | netcat 127.0.0.1 4321 2> /dev/null > /dev/null
```

Práce se vstupním obrazem se provádí v hlavní třídě `MainWindow`. Volá se zde třída `skinDetector`, která obsahuje vyhledávací metody pro nalezení oblastí barvy pleti. Informace o nalezených oblastech jsou udržovány v objektech `skinArea`. Rozpoznávání obličeje ve výseku obrazu, kde byla nalezena kůže, se provádí ve třídě `faceDetector` za pomoci natrénovaného detektoru obličejů převzatého z knihovny OpenCV. Třída `blobTracker` zajišťuje identifikaci nalezených oblastí v předchozím obraze.

Funkce pro manipulaci s natrénovanými Skrytými Markovovými modely obsahuje třída `hmmEval`. Z této třídy jsou volány funkce na vyhodnocení získaných příznaků pomocí Viterbiho algoritmu (třídy `DataClassifier` a `Model` provádějící toto vyhodnocení byly převzaty z ÚPGM).

Pro každý detekovaný objekt zájmu je vytvořen jeden HMM klasifikátor. Tyto klasifikátory se udržují ve vektoru, stejně jako oblasti zájmu, a na stejné pozici ve vektoru (mají tedy stejný index). Vyhodnocení těchto HMM se provádí po určitém časovém úseku (po dosažení určitého počtu záznamů) nebo tehdy, pokud se sledovaný objekt ztratí ze záběru kamery.

Jelikož jsou HMM vyhodnocovány cyklicky po kratších intervalech, udržují se v paměti tři poslední záznamy o vyhodnocení (likelihood každého modelu). Vždy při následujícím určování gesta se v daném HMM vyhodnotí nové příznakové vektory. Nová pravděpodobnost pro každý model gesta se sečte s předchozími dvěma. Výsledná pravděpodobnost pro každé gesto je pak dělena stejným součtem pravděpodobností pozadí. Výsledný poměr gesta ku pozadí se porovná se stanoveným prahem detekce. Tento práh určuje aktivační linii pro zaznamenání gesta. Pokud není překročen, automaticky je vítězem rozpoznávání model pozadí. Určení hodnoty prahu je popsáno v experimentu kapitole 6.1.

Extraktor příznakových vektorů

Druhá část aplikace je zaměřena na získávání popisných vektorů obrazu z videa a jejich ukládání pro trénování. Tato část je řízena ze třídy `InputManager`. Jako vstup z parametru se očekává seznam videí, které se budou zpracovávat.

Po načtení videa (tzn. jedné položky z listu) je dle jména souboru vyhledána anotace k souboru ve formátu „.xml“. Očekává se, že anotační soubory se nachází ve složce s videi. O čtení z anotačních souborů a zápis získaných příznakových vektorů do souboru se stará třída `dataSaver`. Vlastnosti nalezených oblastí zájmu, které se vypisují do souboru s příznakovým vektorem, jsou řízeny pomocí podmíněných maker. Konstanty pro makra jsou zapsány v knihovně `prepconstants`. Příklad zapsaného příznakového vektoru pro jedno gesto je k vidění v příloze E.

Manipulace s obrazem je prováděna pomocí stejných tříd, jak bylo popsáno výše.

Vyhodnocení klasifikátorů

Třetí částí programu je modul pro okamžité vyhodnocování natrénovaných HMM klasifikátorů. Tato část aplikace je řízena z třídy `hmmEval`. Očekává jako vstupní parametr seznam souborů s příznakovými vektory, se kterými se budou provádět testy Skrytých Markovových modelů.

Třída `hmmEval` obsahuje funkce na vyhodnocení HMM klasifikátoru pomocí tabulky záměn (popsáno v následující kapitole). Pomocí konstant lze zapnout různé druhy výpisů při tomto vyhodnocení. Lze vypisovat všechny získané likelihoody s určením nejlepšího výsledku pro každý příznakový vektor nebo zobrazit pouze konečnou tabulku úspěšností, která se po zpracování všech záznamů vypíše na standardní výstup.

Anotační program

Samostatnou aplikací je program na tvorbu anotací k video souborům. Aplikace zpracovává obraz s využitím knihovny OpenCV. Ve vstupním parametru se očekává zadání video souboru či listu souborů, pro které se mají vytvořit anotace.

Program byl vytvořen bez uživatelského rozhraní, neboť bylo záměrem jej především rychle ovládat pomocí klávesnice. Informace o prováděných akcích se vypisují do okna s videem.

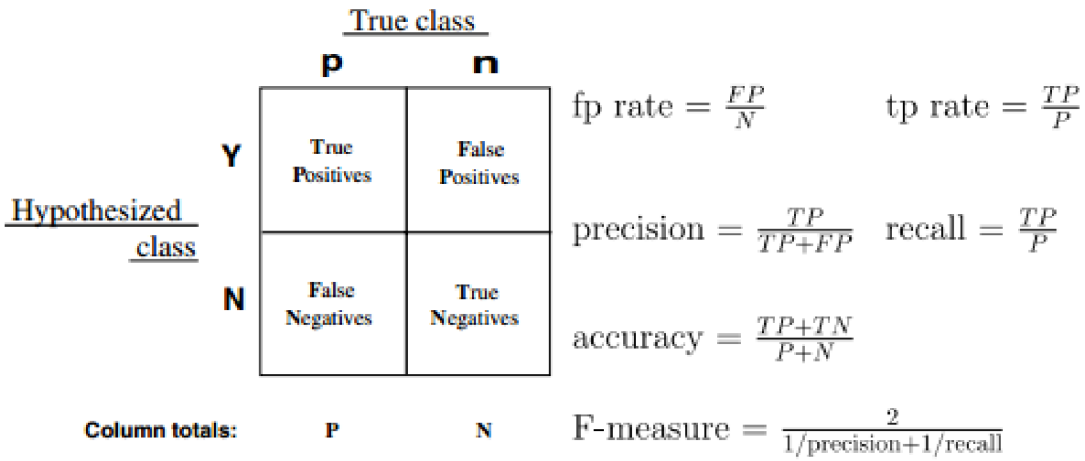
Program obsahuje jednu třídu `Anotator`, která se stará o zpracování obrazu. Aplikace je ovládána pomocí klávesnice. Při stisku definované klávesy se uloží informace o požadavku na vložení anotace. Informace o uložení dat se vypíše do obrazu (viz obrázek 5.7). Po skončení práce se souborem se všechny párové záznamy o vložených anotacích zapíší do anotačního souboru. Tento soubor ve formátu „.xml“ nese stejný název jako soubor s videem. Uložení anotačního souboru se provede do složky, kde se nachází video soubor.

Obrázek 5.7: Ilustrace aplikace na tvorbu anotací.

Kapitola 6

Experimenty a vyhodnocení

Tato kapitola se zabývá experimentováním se základním nastavením HMM za účelem jeho ideálního nastavení pro danou problematiku. Následující kapitoly popisují experimenty s počtem sloupců v příznakovém vektoru a to, jak ovlivňuje jejich druh a počet výsledek detekce.



Obrázek 6.1: Schéma hodnot v matici záměn. Písmena p a n v názvu sloupců značí pozitivní a negativní data dané třídy. Písmena Y a N v názvu řádků zkracují slova „yes“ a „no“, která určují správnost detekce. Celková suma pozitivních vzorků dat správně detekovaných (TP) i chybně neoznačených (FN) je reprezentována písmenem P. Celková suma negativních dat je označena písmenem N, převzato z [16].

Pro zjištění úspěšnosti detekce HMM bylo využito několik způsobů určení vlastností algoritmů. Všechny využité způsoby v základu vychází z tzv. „tabulky záměn“ (angl. confusion matrix) [16]. Příklad tabulky je ilustrován na obrázku 6.1. Sloupce tabulky označují skutečnou třídu dat. Řádky udávají, jak byla data klasifikována. Z hodnot tabulky záměn lze vypočítat metriky zvané „precision“ a „recall“, které se využívají k vyjádření přesnosti a citlivosti detekce. Matici záměn lze dobře upravit pro hodnocení detekce gest. Sloupce jsou pak označeny skutečnou třídou gest a řádky prezentují potenciální třídy gest. V polích matice se uvádí procentuální detekce dané třídy. Aritmetickým průměrem úspěšností detekce z hlavní diagonály pak získáme úspěšnost celého detektoru.

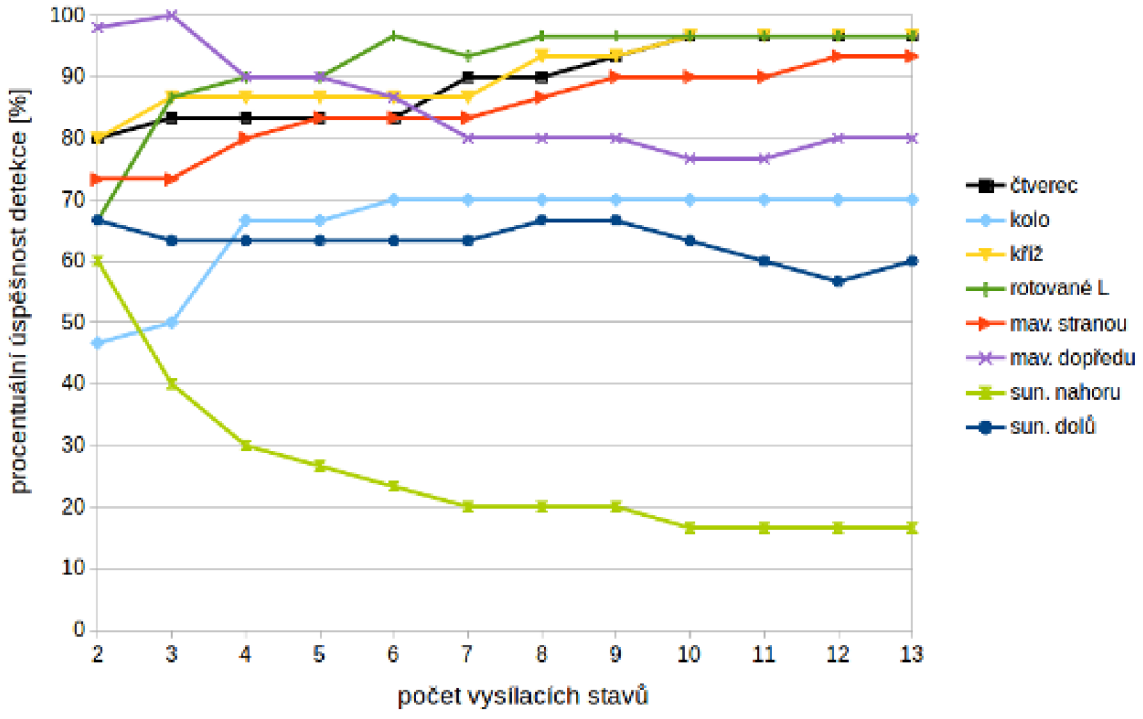
Při nastavení prahu pro určování pozadí byla využita tzv. „ROC křivka“. Jedná se o graf, kde na vodorovné ose je zaznamenán falešně pozitivní počet detekcí (FPR) a na svislé pak správně pozitivní počet detekcí (TPR). Vztahy pro výpočet těchto hodnot jsou vidět na obrázku výše [16].

6.1 Nastavení parametrů modelu

Před vytvářením Skrytých Markovových modelů pomocí HTK je třeba zadat počet vnitřních stavů, které má výsledný model mít. Tento počet stavů nelze nijak matematicky odvodit, lze pouze předpokládat potřebný počet stavů podle složitosti vstupních dat. Ideální počet vysílacích stavů je však třeba nastavit empiricky. Zjištění ideálního počtu vysílacích stavů bylo provedeno v první sérii experimentů popsané v této kapitole.

Pro model pozadí byl třeba nastavit práh detekce, po jehož překročení je akce označena za pozadí. Tento práh byl určen taktéž v této sérii experimentů pomocí ROC křivky.

Byly vytvořeny modely HMM s dvěma až třinácti vnitřními stavy. Pro každý tento model byla sestavena tabulka záměn, z níž byla určena procentuální úspěšnost detekce pro každé gesto. Úspěšnosti detekce jednotlivých gest v závislosti na počtu vysílacích stavů Skrytého Markovova modelu jsou vidět v grafu 6.2. Celková úspěšnost detektorů pro zjištěný optimální počet vnitřních stavů je uvedena v tabulce 6.1. Tabulka 6.2 ukazuje procentuální úspěšnost detekce každého gesta v ideálním případě a při nastavení na optimální počet vnitřních stavů.



Obrázek 6.2: Graf úspěšnosti detekce jednotlivých gest pro různý počet vnitřních stavů Skrytého Markovova modelu (bez nastaveného prahu určující optimální poměr gesta ku pozadí).

Z grafu 6.2 je zřejmé, že menší počet stavů je vhodnější pro jednoduchá a krátká gesta, jako jsou *mávnutí dopředu*, *sunutí dolů* a *sunutí nahoru*. se vzrůstajícím počtem vnitřních stavů klesá úspěšnost detekce těchto gest. U složitějších gest je tomu naopak. S rostoucím počtem vnitřních stavů postupně stoupá úspěšnost detekce. Toto je vidět v grafu například pro gesto *mávnutí stranou* vyobrazené červenou lomenou čárou nebo pro gesto označované jako *rotované L* vyznačené tmavou zelenou čárou.

U posledního zmíněného gesta, tedy u *rotovaného L*, je dobře vidět nárůst úspěšnosti detekce a následné zastavení tohoto růstu. Toto zastavení nárůstu je označením místa dosažení ideálního počtu vysílacích stavů. Přidáváním dalších vysílacích stavů po této hranici se detekce může zlepšit již jen minimálně. Naopak potenciálně může nastat případ, kdy se přidáním vícero vysílacích stavů detekce mírně zhorší. Může totiž dojít k přílišné specifikaci modelu pro konkrétní množinu dat. Tento případ však nastává při malém počtu trénovacích dat.

	úspěšnost detektoru [%]	
počet vnitřních stavů	**8/8**	**5/8**
2	71,67	69,34
3	72,92	76,00
4	73,75	81,34
5	73,75	82,00
6	74,17	84,00
7	73,34	84.67
8	75,42	87,34
9	76,25	88,67
10	75,84	90,00
11	75,42	90,00
12	75,84	90,67
13	76,25	90,67

Tabulka 6.1: Úspěšnost celkového detektoru a detektoru vybrané množiny gest pro jednotlivé počty vysílacích stavů.

Nejlépe ze všech gest se rozeznávají gesta *čtverec*, *kříž* a *rotované L*. To je zapříčiněno pravděpodobně tím, že jsou to gesta vcelku složitá (oproti ostatním trénovaným gestům). Mají poměrně dlouhou trajektorii, kterou musí ruka opsat, aby se gesto identifikovalo. Navíc jejich struktura přispívá ke snadnému a lépe opakovatelnému provedení. Zmíněná gesta jsou totiž souborem lomených čar, při jejichž navazování může člověk zastavit aby si uvědomil následný tah. Také se jistě lépe provádí tah rukou se záměrem předvést rovnou čáru, než například opsat kružnici.

Rozpoznávání zmíněného gesta *kola* dopadlo o něco hůře, než se očekávalo. Chybná identifikace byla v tomto případě především s gestem *čtverec*. Tyto záměny byly nejspíš způsobeny podobným tvarem a stejným směrem provádění daných gest. Zpětnovazební smyčky v každém stavu HMM umožňují vynechat okamžik stagnace pohybu, kdy se navazují dvě lomené čáry při gestu *čtverec*. Tato vlastnost v kombinaci s tím, že lidé obtížně vytvářejí v prostoru oblé tvary a tím pádem i *kolo*, nejspíš byli příčinou znatelně horší úspěšnosti detekce gesta.

gesta	nejlepší výsledek		optimální nastavení	
	počet stavů	úspěšnost [%]	počet stavů	úspěšnost [%]
čtverec	10	96,7	10	96,7
kolo	6	70	10	70
kříž	10	96,7	10	96,7
rotované L	8	96,7	10	96,7
máv. stranou	12	93,3	10	90
máv. dopředu	3	100	9	80
sunutí nahoru	2	60	9	21
sunutí dolů	8	67	9	67

Tabulka 6.2: Úspěšnost detektorů pro jednotlivá gesta.

Ačkoliv má gesto *mávnutí stranou* taktéž oblý tvar, jeho určování dopadlo velice uspokojivě. Důvodem je prostý fakt, že se dobře provádí. Jedná se o opsání přibližně čtvrtiny kruhu s poloměrem délky lokte gestikulujícího. Člověk sám nemusí uvažovat o tvaru trajektorie, pouze využije své kosterní dispozice. Jak je vidět v tabulce 6.2, nejlepší nastavení pro toto gesto bylo při dvanácti stavech, avšak už při 10 stavech dosahoval detektor dobré úspěšnosti (vizte graf 6.2).

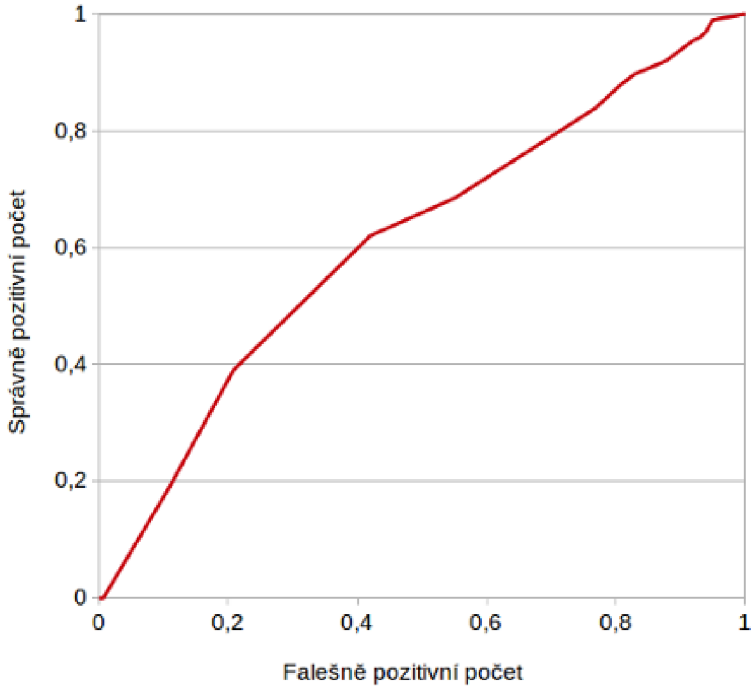
Dobře se rozeznávalo také gesto *mávnutí dopředu*. To bylo důsledkem prostého faktu, že má dané gesto unikátní trajektorii v množině trénovaných gest. Počátek trajektorie má jedinečný směr, takže ho bylo možné dobře rozlišit od ostatních definovaných gest. Rozpoznávání bylo nejúspěšnější při volbě tří vnitřních stavů, kdy byla detekce stoprocentní (viz tabulka 6.2).

Gesta *sunutí nahoru* a *sunutí dolů* byla dle očekávání rozlišována špatně, detekce trpěla především záměnou těchto dvou gest. Jedná se totiž o gesta se shodnou trajektorií u nichž se očekává, že se detekce výrazně zlepší až při použití více prvků v popisném vektoru.

V grafu lze pozorovat, že úspěch detekce přestává růst pro většinu gest po dosažení osmi až deseti vnitřních stavů. Tabulka 6.1 ukazuje přesné úspěšnosti detektorů v závislosti na počtu vnitřních stavů. Pro všechna trénovaná gesta je nejlepší detektor sestaven z devíti stavů, kde se dosahuje úspěšnosti $76,25\%$. Pro detektor vybraných gest je nejvhodnější nastavení na deset vnitřních stavů, které vede k úspěšnosti 90%. Přesnost detektoru vybraných gest se ještě mírně zvyšuje při dosažení dvanácti stavů, avšak tento růst je již zanedbatelný.

Při trénování modelů gest pro HMM byl natrénován také model pro pozadí. Tento model kvůli širokému záběru akcí, které musí určovat, nemá dobrou úspěšnost detekce. Bez nastaveného prahu přesnosti se jeho úspěšnost určování pohybuje kolem 10%. Tato špatná úspěšnost je také ovlivněna faktem, že byl model natrénován na celé trénovací množině dat. Zmíněný práh úspěchu detekce šumu byl určen pomocí experimentu, jež je popsán níže.

Práh detekce byl určen pomocí ROC křivky, kterou lze vidět na obrázku 6.3. Práh byl stanoven na hodnotu 0.74. Při této hodnotě se téměř o polovinu snížil počet chybně detekovaných příznaků, které jsou pozadím. S nastavením prahu je také spjato zhoršení detekce klasifikátoru gest. Při nastaveném prahu dosahuje celkový klasifikátor úspěšnosti 59%. Chybnou detekci pozadí jako gesta však nelze zcela eliminovat. S vyšší hodnotou prahu se hůře detekovalo především gesto *kola*, které i při nastaveném prahu trpí jednosměrnou záměnou převážně s gestem *čtverec*. Při nastaveném prahu na zmíněnou hodnotu se správně



Obrázek 6.3: ROC křivka pro detektor gest s deseti vnitřními stavy.

klasifikuje pouze 1 z 5 gest *kola*. U ostatních gest je tento poměr znatelně lepší. Nejlépe klasifikována byla gesta *rotované L* a *kříž*, pro které se správně určilo 7 z 10 gest.

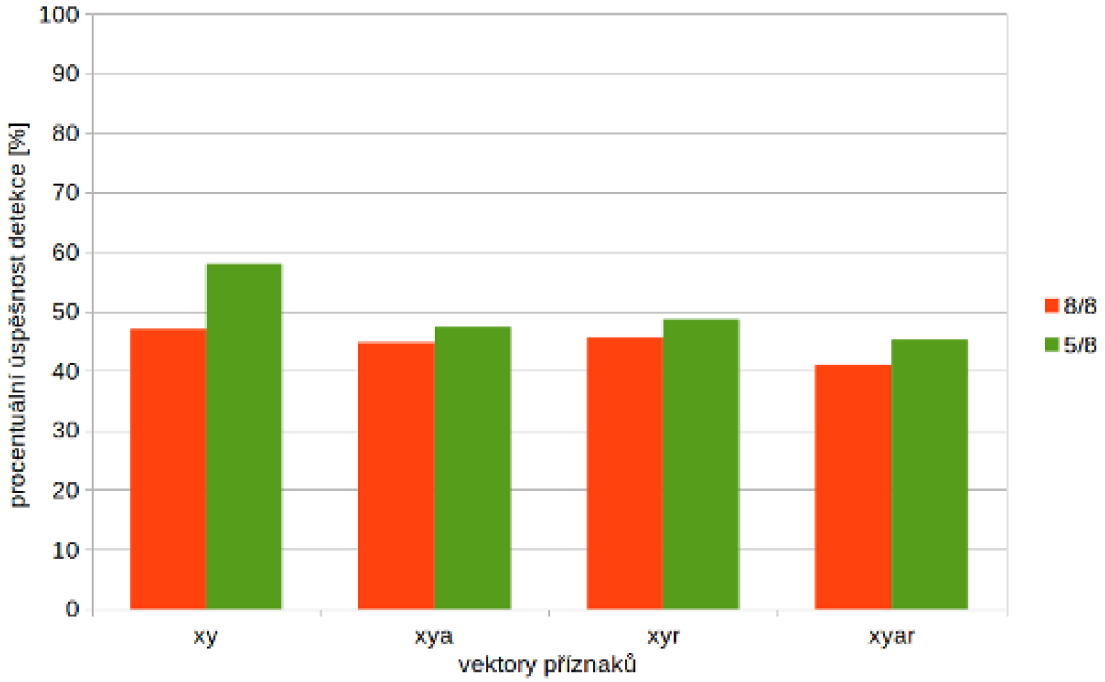
6.2 Změna parametrů příznakového vektoru

Pokud pro popis trajektorie objektu slouží pouze derivace polohy, je možné, že se náhodným pohybem v prostoru vytvoří obdobně vypadající trajektorie ve 2D, která pak může být chybně klasifikována. Proto je vhodné otestovat, jak se změní rozpoznávání při popisu sledovaného objektu pomocí více parametrů. Následující odstavce popisují experimenty s rozšířeným příznakovým vektorem o různé vlastnosti. Základem je pro příznakový vektor vždy derivace polohy objektu.

V programu je použita detekce kůže podle barvy. Implementovaný způsob detekce kůže v každém obraze však vykazuje jisté nepřesnosti. Tyto nedokonalosti se týkají především skokového zvětšování a zmenšování rozlohy určené oblasti zájmu o zhruba 20%. Detekovaná oblast zájmu však stále označuje korektní místo v obraze a její střed zůstává přibližně stejný. Se změnami rozlohy souvisí i změny v natočení oblasti. Z těchto důvodů se očekává, že tento experiment s rozšířeným vektorem příznaků povede ke zhoršení schopnosti detekce gest.

První rozšíření vektoru bylo o derivaci rozlohy sledovaného objektu. Gesta určená k ovládání VLC se provádí v prostoru vedle těla, tzn. prakticky ve dvourozměrném prostoru. Při jejich provádění by se tedy měla rozloha sledované oblasti měnit co nejméně. Naopak zbylá tři gesta z datové sady se provádí v prostoru tak, že při nich ruka mění svoji plochu. Další experiment byl proveden s přidáním položky, která značí změnu natočení sledovaného objektu. Třetí typ příznakového vektoru spojuje předchozí dvě varianty do jedné. Experiment byl tedy proveden s vektorem o čtyřech prvcích.

Výsledky experimentů vyšly přibližně dle očekávání. Bylo zjištěno, že napříč množinou vnitřních stavů se při rozšíření vektoru detekce mírně zhorší. Při menším počtu vnitřních stavů se úspěšnost detekce všech rozpoznávačů přibližuje stejné hodnotě. S větším počtem stavů se zvětšuje rozdíl v úspěšnosti detekce gest.



Obrázek 6.4: Graf úspěšností detekce při použití různých příznakových vektorů. Písmena x a y značí derivaci polohy, a je derivace rozlohy, r je derivace rotace objektu. Výsledky grafu jsou pro 10 vnitřních stavů pro gesta z množiny pro ovládání VLC (5/8) a pro 9 stavů pro všechna gesta datové sady (8/8).

Při rozšíření vektoru o jeden parametr se detekce mírně zhorší (přibližně o 12%) oproti rozpoznávání pouze pomocí polohy. S rozšířením o dva parametry dojde k dalšímu mírnému zhoršení detekce – přibližně ještě o dalších 5%. Porovnání celkových úspěšností detektorů je vidět na obrázku 6.4, kde je vidět i základní vektor obsahující pouze informace o poloze ruky. Nejlépe v experimentu vyšlo rozšíření vektoru o rotaci, které vede k již zmíněnému deseti procentnímu poklesu úspěchu detekce. Srovnatelně fungovalo také rozšíření o parametr rozlohy, které mělo konstantně ve všech testech horší detekci přibližně o 2% od úspěchu klasifikátoru, který používal vektor s rotací.

Úspěšnost detekce každého gesta pro jednotlivé vektory je vidět v tabulce 6.3. Některá gesta se rozpoznávala se srovnatelnou úspěšností jako při využití vektoru (pouze) s polohou objektu, jehož úspěšnosti jsou v tabulce uvedeny pro srovnání v prvním sloupci. Nicméně většina gest měla úspěšnost znatelně horší než při pouhém využití derivace polohy.

Dle očekávání se v těchto experimentech nejlépe detekovala gesta *kříž* a *rotované L*. Velké rozdíly byly při rozpoznávání gest *kola* a *čtverce* s pomocí různých vektorů. Reflektování rotace přineslo dobrou detekci *čtverce*, avšak tato úspěšnost je stále horší než při pouhém využití derivace polohy. Zohlednění rozlohy přispělo výrazně lepší detekci pro gesto *kola*, které se jako jediné detekovalo výrazně lépe než při využití dvouprvkového vektoru.

gesta	úspěšnost pro rozšířené vektory příznaků [%] derivace polohy	+ rozloha	+ rotace	+ kombinace
čtverec	55.7	15.3	49,7	14.7
kolo	23	51.7	23	32
kříž	82.7	59.7	65.7	71.7
rotované L	74.7	68	66	66
máv. stranou	59.3	42.7	39	42.3
máv. dopředu	64	52	60.7	54
sunutí nahoru	19	28	31	21
sunutí dolů	46.3	41.7	31	27

Tabulka 6.3: Úspěšnost detekce jednotlivých gest pro různé příznakové vektory.

Při detekci pomocí rozšířených vektorů se vyskytovalo znatelně více záměn mezi gesty. Nejvíce záměn bylo mezi podobnými gesty, poměrně hodně záměn bylo u většiny gest také s gesty *rotované L* a *kolo*.

Z experimentů vyplynulo, že testovaná rozšíření vektoru příznaků nejsou vhodná k použití při této konfiguraci programu. Dle očekávání se detekce gest mírně zhoršila, nicméně nad očekávání se lehce zlepšila funkčnost modelu pozadí (přibližně o 15%). To bylo nejspíše zapříčiněno více rozličnými informacemi o pozadí, které pak vyústily v lepší rozlišovací schopnost modelu.

6.3 Zhodnocení funkčnosti

Aplikace na rozpoznávání gest funguje real time. Nevýhoda programu spočívá v malé variabilitě vzhledem k osvětlovacím podmínkám v prostoru. Kvůli špatnému modelu pozadí dochází během provádění gesta k mylnému určování kategorie prováděné akce. Je zde také stále vyšší procento detekce šumu jako gest, především jako gesta na ztišení hlasitosti (*kříž*). Nastavení větší hodnoty pro práh detekce by však vedlo k znatelné eliminaci možnosti detekce gesta.

Model pozadí používaný programem má poměrně špatné výsledky. Jiný způsob vytvoření modelu pozadí (pouze na trénovacích datech obsahujících šumu) byl testován, ale přinesl značně horší výsledky.

Kvůli funkčnosti modelu pozadí byla omezena četnost vyhodnocování získaných příznakových vektorů. Neprovádí se tedy vyhodnocování po každém snímku, ale pouze přibližně jednou za dvě sekundy (určeno empiricky). Při tomto způsobu vyhodnocování dochází k duplicitní identifikaci gesta spojené s aktivací ovládací funkce. To je zapříčiněno rozdělením gesta do dvou vyhodnocovacích cyklů. Při prvním cyklu model pozadí nezíská takovou váhu aby potlačil stoupající pravděpodobnost detekce gesta. Při dalším cyklu vyhodnocení se připočítá předchozí pravděpodobnost gesta k nové a je tedy korektně detekováno provedené gesto.

Popsanou duplicitu detekce by bylo možné vyřešit programovým ošetřením, které by zamezilo detekci dvou stejných gest po sobě. Případně by bylo možné určit časový interval mezi dvěma stejnými gesty. Nicméně toto řešení jsem vyhodnotila vzhledem k programu jako nevhodné. Důvody jsou následující. Pokud se vícekrát detekuje gesto *kola*, jenž provede spuštění přehrávání, program VLC druhý příkaz ignoruje. Tudíž duplicitní detekce nijak ne-

vadí. V případě provádění gest *kříž*, *rotovaného L* a *mávnutí stranou* by bylo z principu nevhodné neumožnit opakované provedení gesta. Tato gesta jsou určena k přepínání programu a ovládání hlasitosti, kde je možné, že uživatel bude žádat akci opakovaně.

Pro detekci gesta je kvůli modelu pozadí nastaven práh pro poměr detekce gesta ku pozadí, který musí být menší než hodnota 0.74. Tím se snižuje úspěšnost detekce. Nejvíce daným prahem trpí gesto *kola*, které je poté detekováno přibližně v jednom z pěti případů provedení. Detekce při tomto gestu může vyjít kladně vícekrát, je zde však velké procento záměn za gesto *čtverce*. Nicméně tuto záměnu uživatel nijak nerozpozná, neboť pokud uživatel provádí gesto *kola*, snaží se zapnout přehrávání. V tom případě program VLC nijak neovlivní zaslání příkazu na pozastavení přehrávání, jelikož se nic nepřehrává. Samotná gestikulace tvaru *čtverce* se rozezná v pěti případech z deseti. Srovnatelně je rozpoznávané gesto *mávnutí stranou*. Nejlépe se rozeznávají gesta na ovládání hlasitosti aplikace.

6.4 Možnosti pokračování práce

Z experimentů vyplynulo, že nejslabší stránkou programu je model pozadí. Tento model špatně odlišuje běžný šum (pohyb) ve videu od vybraných gest. Možností, jak ho vylepšit, je rozšíření datové sady řádově o tisíce záznamů. Toto rozšíření by však vyžadovalo další experimentování k ověření funkčnosti.

Při rozšiřování množiny dat pro trénování bych také doporučila rozšíření množiny gest, která ovládá aplikaci. V současné době je implementováno ovládání pro pět gest. Při samotném ovládání programu VLC lze však používat vícero zajímavých funkcí, jako jsou například možnost okamžitého absolutního ztlumení zvuku nebo zvětšení okna přehrávače na celou obrazovku. Tyto funkce by přispěly k většímu komfortu při používání aplikace.

Dle experimentů popsaných v kapitole 6.1 je vidět, že plošné optimální nastavení není zcela ideální pro všechny modely gest. Jako další možnost pokračování práce by bylo možné nastavit každý model na vlastní optimální nastavení a dále testovat funkčnost tohoto celku.

Při rozšiřování práce by bylo vhodné vyměnit metodu detekce ruky, přičemž by se sledovala úspěšnost detektoru. To by mohlo být provedeno změnou způsobu detekce podle barvy kůže nebo výměnou příznakového vektoru, například za popsaný MoSIFT příznak. Polohu ruky by bylo možné také snadno detekovat a sledovat pomocí zařízení Kinect, k němuž jsou dostupné knihovny obsahující moduly na detekci objektů (částí těla). Zařízení Kinect v této práci nebylo použito z důvodu, že aplikace je cílena pro běžné použití. Kinect je poměrně nové zařízení, jehož cena je zatím o dost větší než cena běžných webkamer. S postupem času je však možné, že dojde ke zlevnění tohoto produktu, takže by se mohl více dostat mezi běžné uživatele. Tím by experimentování s tímto bodem vylepšení práce nabylo na významu.

Další možné vylepšení stran používaných metod je zlepšení softwaru pro trénování Skrytých Markovových modelů. V práci jsou použity modely, jež popisují každý stav jednou vícerozměrnou Gaussovou funkcí. Bylo by vhodné zkusit vytvořit složitější trénovací software, který by byl schopen vytvářet modely, které popisují stavy pomocí více Gaussových funkcí.

Kapitola 7

Závěr

Hlavním cílem této diplomové práce bylo prostudovat metody týkající se detekce ruky a klasifikace trajektorií, a na základě toho navrhnout a implementovat aplikaci zabývající se danou problematikou. Cíl práce byl splněn.

Prostudovala jsem literaturu, která se zabývá zpracováním obrazu a metodami strojového učení, které jsou vhodné pro rozpoznávání gest. Vybrané poznatky jsou shrnuty v kapitolách 2 a 3. Zhodnocení současných možností v dané oblasti je provedeno v kapitole 4. Na základě nastudovaných informací jsem navrhla aplikaci na rozpoznávání gest prováděných rukou, která je popsána ve kapitole 5. V této kapitole je uveden také popis vytvořené datové sady, která vznikla v rámci práce pro trénování a testování modelů. Pro optimální nastavení programu jsem provedla řadu experimentů, které jsou popsány v kapitole 6. Koncem této kapitoly je uvedeno také zhodnocení práce a jsou diskutovány možná vylepšení programu.

Vytvořený program využívá obyčejnou webkameru ke sledování prováděných gest. Provádí se detekce ruky na základě barvy kůže. Nalezené objekty zájmu se sledují a zjišťuje se jejich směr pohybu. Získané trajektorie se poté klasifikují pomocí Skrytých Markovových modelů, které byly natrénovány pomocí HTK na vytvořené datové sadě. Podle určené třídy gesta program ovládá multimediální přehrávač VLC. Experimenty bylo zjištěno, že nejlépe se gesta určená k ovládání VLC klasifikují pomocí modelů o 10 stavech. Pro odlišení gest od okolí vznikl také model pozadí, který byl natrénován na celé datové sadě. Pro tento model byl určen poměr detekce gesta ku okolí na hodnotu 0.74. Při této hodnotě dosahuje detektor gest úspěšnosti přibližně 60%.

Další možnosti pokračování práce spočívají v testování různých způsobů detekce ruky tak, aby bylo rozpoznávání nezávislé na osvětlení. Následně by bylo vhodné rozšířit datovou sadu. Širší datový základ při trénování by mohl přispět k lepší detekci modelu pozadí. Implementace ovládání VLC pomocí více gest by také vedlo ke zjednodušení práce s aplikací. Při dalším experimentování by bylo vhodné otestovat celkový úspěch detektoru pro specifické ideální nastavení pro každý model gesta. V budoucnu by se dala detekce ruky pomocí kamery nahradit použitím Kinectu a jeho knihoven. Toto rozšíření by však mělo být podmíněno finanční dostupností zařízení a také dalším testováním aplikace.

Literatura

[1] Achacon, D. L. M.; Carlos, D. M.; Puyaoan, M. K.; aj.: REALISM: Real-Time Hand Gesture Interface for Surgeons and Medical Experts. In *9th Philippine Computing Science Congress [online]*, Citeseer, 2009 [cit. 2015-04-28].
URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.170.4498&rep=rep1&type=pdf

[2] Alpaydin, E.: *Introduction to machine learning*. London: MIT Press, druhé vydání, 2010, ISBN 978-0-262-01243-0.

[3] Amari, S.; Wu, S.:: Improving support vector machine classifiers by modifying kernel functions. *Neural Networks [online]*, ročník 12, č. 6, 1999 [cit. 2015-01-10]: s. 783–789, ISSN 0893-6080, doi:10.1016/s0893-6080(99)00032-5.
URL http://www.sciencedirect.com/science/article/pii/S0893608099000325

[4] Basilio, J. A. M.; Torres, G. A.; Pérez, G. S.; aj.: Explicit Image Detection using YCbCr Space Color Model as Skin Detection. Puerto Morelos, Mexico: Proceedings of the 2011 American conference on applied mathematics and the 5th WSEAS international conference on Computer engineering and applications, January 2011, ISBN 978-960-474-270-7, s. 123–128.

[5] Bishop, C. M.: *Pattern Recognition and Machine Learning*. New York, USA: Springer, 2006, ISBN 0-387-31073-8.

[6] Bradski, G. R.: Real time face and object tracking as a component of a perceptual user interface. In *Proceedings Fourth IEEE Workshop on Applications of Computer Vision. WACV'98*, October 1998, s. 214–219, doi:10.1109/ACV.1998.732882.

[7] Breiman, L.: Random Forests. *Machine Learning*, ročník 45, č. 1, 2001: s. 5–32, ISSN 0885-6125, doi:10.1023/A:1010933404324.

[8] Burges, C.: A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, ročník 2, č. 2, June 1998: s. 121–167, ISSN 13845810, doi:10.1023/a:1009715923555.

[9] Chen, M.-Y.; Hauptmann, A.: MoSIFT: Recognizing Human Actions in Surveillance Videos [online]. Technická zpráva, School of Computer Science, Carnegie Mellon University, Pittsburg, USA, September 2009 [cit. 2014-12-24].
URL http://www.cs.cmu.edu/~mychen/publication/ChenMoSIFTCMU09.pdf

[10] Chen, M.-Y.; Mummert, L.; Pillai, P.; aj.: Controlling Your TV with Gestures. In *Proceedings of the internation conference on Multimedia information retrieval*,

Philadelphia, Pennsylvania, USA: Association for Computing Machinery, March 2010, s. 405–408, iSBN 978-1-60558-815-5.

[11] Chen, X.; Li, H.; Pan, T.; aj.: Kinect Sign Language Translator expands communication possibilities [online]. 2013 [cit. 2015-05-01].
URL http://research.microsoft.com/en-us/collaboration/stories/kinectforsignlanguage_cs.pdf

[12] Collins, M.: The Forward-Backward Algorithm [online]. [cit. 2015-03-25].
URL http://www.cs.columbia.edu/~mcollins/fb.pdf

[13] Dalal, N.; Triggs, B.: Histograms of Oriented Gradients for Human Detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, ročník 1, Los Alamitos, California: IEEE Computer Society, June 2005, ISBN 0-7695-2372-2, ISSN 1063-6919, s. 886–893, doi:10.1109/CVPR.2005.177.

[14] Dalal, N.; Triggs, B.; Schmid, C.: Histogram of Oriented Gradients (HOG) for Object Detection [online]. [cit. 2014-12-08].
URL http://people.cs.missouri.edu/~duanye/cs8690/lecture-notes/HoG.pdf

[15] Fang, Y.; Wang, K.; Cheng, J.; aj.: A real-time hand gesture recognition method. In *2007 IEEE International Conference on Multimedia and Expo*, IEEE, June 2007, ISBN 1-4244-1016-9, s. 995–998, doi:10.1109/ICME.2007.4284820.

[16] Fawcett, T.: An introduction to ROC analysis. *Pattern Recognition Letters*, ročník 27, č. 8, 2006: s. 861 – 874, ISSN 0167-8655, doi:http://dx.doi.org/10.1016/j.patrec.2005.10.010.

[17] Hlaváč, V.; Šonka, M.: *Počítačové vidění*. Praha, CZ: GRADA a.s., 1992, ISBN 80-85424-67-3.

[18] Huson, D.: SVMs and Kernel Functions [online]. Algorithms in Bioinformatics II, Eberhard Kalrs Universität Tübingen, 2007 [cit. 2015-01-02].
URL http://ab.inf.uni-tuebingen.de/teaching/ss07/albi2/script/svm.pdf

[19] Keskin, C.; Kirac, F.; Kara, T. E.; aj.: Randomized decision forests for static and dynamic hand shape classification. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, June 2012, ISSN 2160-7508, s. 31–36, doi:10.1109/CVPRW.2012.6239183.

[20] Kölsch, M.; Turk, M.: Robust Hand Detection. In *Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. Proceedings [online]*, May 2004 [cit. 2015-04-16], ISBN 0-7695-2122-3, s. 614–619, doi:10.1109/AFGR.2004.1301601.

[21] Lienhart, R.; Maydt, J.: An extended set of Haar-like features for rapid object detection. In *2002 International conference on image processing proceedings*, ročník 1, Rochester, New York, USA: IEEE, 2002, ISBN 0-7803-7622-6, ISSN 1522-4880, s. 900–903, doi:10.1109/ICIP.2002.1038171.

[22] Lin, H.-I.; Hsu, M.-H.; Chen, W.-K.: Human hand gesture recognition using a convolution neural network. In *2014 IEEE International Conference on*

Automation Science and Engineering (CASE), August 2014, s. 1038–1043, doi:10.1109/CoASE.2014.6899454.

[23] López-Méndez, A.; Casas, J. R.: Can our TV robustly understand human gestures?: Real-time gesture localization in range data. In *Proceedings of the 9th European Conference on Visual Media Production CVMP 2012, London*, New York, NY: ACM, December 2012, ISBN 978-1-4503-1311-7, s. 18–25.

[24] Mayer, J.; Breda, V.: Continuous gesture recognition using hidden markov models. In *The 2014 International Conference on Image Processing, Computer Vision and Pattern Recognition [online]*, Florianopolis, Brazil, 2014 [cit. 2014-12-01].
URL http://world-comp.org/preproc2014/IPC3527.pdf

[25] Mlích, J.; Zemčík, P.; Jiřík, L.: Trajectory classification using HMMs. In *WSCG 2009 Communication Papers*, Plzeň, CZ: University of West Bohemia in Pilsen, 2009, ISBN 978-80-86943-94-7, s. 67–72.

[26] Mlích, J.; Chmelař, P.: Trajectory classification based on Hidden Markov Models. In *Proceedings of 18th International Conference on Computer Graphics and Vision*, Lomonosov Moscow State University, 2008, ISBN 978-5-9556-0112-0, s. 101–105.

[27] Nixon, M. S.; Aguado, A. S.: *Feature Extraction and Image Processing*. Hungary: Academic Press, druhé vydání, 2008, ISBN 978-0-12372-538-7.

[28] Novák, M.; Faber, J.; Kufudaki, O.: *Neuronové sítě a informační systémy živých organismů*. Praha: Grada a.s., první vydání, 1992, ISBN 80-85424-95-9.

[29] Přinosil, J.; Krolikowski, M.: Využití detektoru Viola-Jones pro lokalizaci obličeje a očí v barevných obrazech. *Elektrorevue [online]*, 21. srpen 2008 [cit. 2015-03-25], ISSN 1213-1539,
url http://www.elektrorevue.cz/cz/clanky/zpracovani-signalu/0/vyuziti-detektoru-viola-jones-pro-lokalizaci-obliceje-a-oci-v-barevnych-obrazech/.

[30] Rajam, P. S.; Balakrishnan, G.: Real time Indian Sign Language Recognition System to aid deaf-dumb people. In *2011 IEEE 13th International Conference on Communication Technology*, September 2011, ISBN 978-1-61284-307-0, s. 737–742, doi:10.1109/ICCT.2011.6157974.

[31] Sheng, J.: A Study of Adaboost in 3D Gesture Recognition [online]. Technická zpráva, Department of Computer Science, University of Toronto, Toronto, Ontario, December 2003 [cit. 2015-04-10].
URL http://www.dgp.toronto.edu/~jsheng/doc/CSC2515/Report.pdf

[32] Singh, S. K.; Chauhan, D. S.; Vatsa, M.; aj.: A Robust Skin Color Based Face Detection Algorithm. *Tamkang Journal of Science and Engineering [online]*, ročník 6, č. 4, 2003 [cit. 2014-12-20]: s. 227–234.
URL http://www2.tku.edu.tw/~tkjse/6-4/6-4-6.pdf

[33] Spinello, L.; Arras, K. O.: People Detection in RGB-D Data. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Piscataway: IEEE, September 2011, ISBN 978-1-61284-454-1, ISSN 2153-0858, s. 3838–3843, doi:10.1109/IROS.2011.6095074.

[34] Vezhnevets, V.; Sazonov, V.; Andreeva, A.: A Survey on Pixel-Based Skin Color Detection Techniques. In *GraphiCon'2003*, ročník 3, Moscow, Russia, January 2003, s. 85–92.

[35] Viola, P.; Jones, M.: Robust Real-time Object Detection [online]. In *International Journal of Computer Vision*, Vancouver, Canada, 13. July 2001 [cit. 2015-03-25]. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.110.4868&rep=rep1&type=pdf

[36] Wachs, J.; Stern, H.; Edan, Y.; aj.: Real-Time Hand Gesture Interface for Browsing Medical Images. *TSI Press*, ročník 1, č. 3, 2007: s. 175–185.

[37] WWW stránky: *VideoLAN's Wiki: VLC command-line help*. 2008 [cit. 2015-05-01]. URL https://wiki.videolan.org/VLC_command-line_help/

[38] WWW stránky: OpenCV documentation - Introduction to Support Vector Machines. April 2014 [cit. 2015-01-01]. URL http://docs.opencv.org/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html?highlight=svm\example

[39] WWW stránky: Asus Xtion Pro Live. [cit. 2015-04-28]. URL http://www.asus.com/Multimedia/Xtion_PRO_LIVE/

[40] WWW stránky: Handy - hand gesture recognition. [cit. 2015-04-28]. URL http://www.eyedea.cz/hand-gesture-recognition/

[41] WWW stránky: Kinect for Windows. [cit. 2015-04-28]. URL https://www.microsoft.com/en-us/kinectforwindows/meetkinect/features.aspx

[42] WWW stránky: Leap Motion. [cit. 2015-04-28]. URL https://www.leapmotion.com/product

[43] WWW stránky: Thalmiclabs: Myo. [cit. 2015-04-28]. URL https://www.thalmic.com/en/myo/techspecs

[44] Yang, D.; Xia, J.: Face Tracking Based on Camshift Algorithm and Motion Prediction. In *2009 International Workshop on Intelligent Systems and Applications [online]*, May 2009 [cit. 2015-01-09], s. 1–4, doi:10.1109/IWISA.2009.5072863.

[45] Yilmaz, A.; Javed, O.; Shah, M.: Object Tracking: A Survey. *ACM Computing Surveys*, ročník 38, č. 4, December 2006, ISSN 0360-0300, doi:10.1145/1177352.1177355.

[46] Yin, Y.; Davis, R.: Real-time continuous gesture recognition for natural human-computer interaction. In *2014 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC) [online]*, Juli 2014, s. 113–120, doi:10.1109/VLHCC.2014.6883032.

[47] Young, S.; Evermann, G.; Gales, M.; aj.: *The HTK Book*. Cambridge, United Kingdom: Cambridge University Engineering Department, December 2006.

[48] Řezanková, H.; Húsek, D.; Snášel, V.: *Shluková analýza dat.* Praha: Professional Publishing, druhé vydání, 2009, ISBN 978-80-86946-81-8.

[49] Řezníček, I.: *Human action recognition in video.* Dizertační práce, FIT VUT v Brně, Brno, 2014.

[50] Řezníček, I.; Zemčík, P.: On-line human action detection using space-time interest points. In *Zborník príspevkov prezentovaných na konferencii ITAT, september 2011*, Praha, SK: Faculty of Mathematics and Physics, 2011 [cit. 2014-12-23], ISBN 978-80-89557-01-1, s. 39–45.

[51] Řezníček, I.; Zemčík, P.: Action recognition using combined local features. In *Proceedings of the IADIS Computer graphics, Visulisation, Coputer Vision and Image Processing 2013*, Prague, CZ: IADIS, 2013, ISBN 978-972-8939-89-2, s. 111–118.

[52] Řezníček, I.; Zemčík, P.: Human action recognition for real-time applications. In *Proceedings of the 3rd International Conference on Pattern Recognition Applications and Methods*, 2014, ISBN 978-989-758-018-5, s. 646–653.

Příloha A

Obsah DVD

Přehled souborů umístěných na přiloženém DVD:

- technická zpráva ve formátu „.pdf“
- složka doc - zdrojové soubory k technické zprávě pro program Latex
- složka VideoManager - zdrojové kódy aplikace k diplomové práci
- složka anotator - zdrojové kódy pomocného programu k anotaci videí
- složka data - trénovací datová sada a demonstrační videa, také dostupné na: http://uloz.to/xJdtS4kr/dp-data-test-demo-zip
  - testovací datová sada je dostupná na: http://uloz.to/xJLyRSZw/dp-data-trein-zip
- README - soubor s návodem na instalaci potřebných knihoven a programů

Příloha B

Manuál aplikace na ovládání VLC

Hlavní aplikaci lze spustit přímo (dvojklikem na spustitelný soubor) nebo z příkazové řádky s následujícími parametry:

```
./VideoManager [ -l <list> | -f <videoFile> | -e <list>]
               [ -h | --help | <videoFile> ]
```

Kód výše představuje možnosti spustit program bez parametru či s pěti přepínači. Je možné zadat vždy pouze jeden z těchto přepínačů. Popis parametrů shrnuje následující seznam.

- `-l <list>` - Spustí se program na získání příznakových vektorů z obrazu. Za parametrem následuje seznam souborů s videi ke zpracování. Seznam obsahuje vždy jedno jméno video souboru na jednom řádku.
- `-f <videoFile>` - Parametr podobný přepínači `-l`, avšak získání příznakového vektoru se provede pouze pro jeden zadaný soubor s videem.
- `-e <list>` - Spustí se část programu, která vyhodnocuje činnost Skrytých Markovových modelů. Za přepínačem následuje seznam souborů určených k testování, které obsahují příznakové vektory. Seznam musí obsahovat vždy jeden název testovacího souboru na jednom řádku.
- `-h` | `--help` - Vypíše se nápověda k programu s příklady spuštění.
- `<videoFile>` - Spustí se detekční část aplikace, kde vstup není brán z kamery ale ze zadaného video souboru (pro snazší testování aplikace).
- Při spuštění programu bez parametru se spustí detekční část aplikace. Zobrazí se GUI a po spuštění programu VLC bude možné jej ovládat pomocí gest.

Příklady spuštění aplikace:

```
./VideoManager -l list.txt
./VideoManager -f MOV001.MOD
./VideoManager -e featureFileList.txt
./VideoManager -h
./VideoManager MOV001.MOD
./VideoManager
```

Příloha C

Manuál aplikace k anotaci souborů

Aplikace na vytváření anotačních souborů k videím se spouští z příkazové řádky. Následující deklarace uvádí možné parametry:

```
./anotator -l <list> | -f <videoFile> | -h
```

Příklad parametrů v příkazu spuštění aplikace uvedený výše ukazuje možnost zvolit ze tří parametrů. Vyžadován je vždy právě jeden z těchto parametrů. Následující body přibližují význam přepínačů.

- `-f <videoFile>` - Spustí se anotační program pro jeden zadaný video soubor.
- `-l <list>` - Program bude postupně načítat a zpracovávat video soubory ze zadaného seznamu. Očekává se, že jeden název video souboru bude na jednom řádku.
- `-h` - Vypíše se krátká nápověda o spuštění a ovládání programu.

Příklady spuštění aplikace:

```
./anotator -l list.txt
./anotator -f MOV001.MOD
./anotator -h
```

Samotný program se ovládá pomocí vybraných tlačítek klávesnice:

- `q` | `Q` zastavení přehrávání videa,
- `e` | `E` pokračování v přehrávání,
- `x` | `X` ukončení přehrávání videa,
- `a` | `A` vložení anotace,
- `p` | `P` posunutí o jeden frame zpět,
- `n` | `N` posunutí o jeden frame dopředu.

Příloha D

Ukázka anotačního souboru

Následující kód ve formátu „xml“ ukazuje anotační záznam pro jeden trénovací video soubor, ve kterém se nacházejí čtyři gesta.

```
<?xml version="1.0" encoding="UTF-8"?>
<ANNOTATION_DOCUMENT AUTHOR="Daniela Johanova" DATE="14-3-2015">
<ANNOTATION_DOCUMENT FILE="a024.MOD">
<gesture>
   <startframe>12</startframe>
   <endframe>69</endframe>
</gesture>
<gesture>
   <startframe>104</startframe>
   <endframe>206</endframe>
</gesture>
<gesture>
   <startframe>258</startframe>
   <endframe>325</endframe>
</gesture>
<gesture>
   <startframe>390</startframe>
   <endframe>468</endframe>
</gesture>
<allframes>508</allframes>
```

Příloha E

Příklad zápisu příznakového vektoru v souboru

Příklad obsahu souboru s příznakovým vektorem pro jedno gesto. První řádek souboru udává informace o vektoru. První číslo vyjadřuje počet prvků (sloupců) ve vektoru, druhé číslo udává počet záznamů (řádků).

```
4 24
2 -13 214 6
3 -13 -38 9
6 -12 421 13
8 -11 -33 10
10 -7 19 -6
11 -4 -426 -10
13 3 268 -7
16 4 -154 -5
9 5 283 -4
7 3 327 -2
4 2 96 -1
1 1 184 1
0 2 -24 0
-1 -1 -231 -3
-2 -2 -122 1
-8 -4 48 3
-12 -4 -74 4
-10 1 253 3
-15 5 -200 -6
-12 7 -14 1
-9 18 -2 6
-6 9 -91 4
-4 10 42 2
-2 8 -73 1
```