Prohlášení

Předkládám tímto k posouzení a obhajobě diplomovou práci zpracovanou na závěr studia na Fakultě aplikovaných věd Západočeské univerzity v Plzni.

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím odborné literatury a pramenů, jejichž úplný seznam je její součástí.

V Plzni dne 30. srpna 2013. .....

vlastnoruční podpis

Anotace

Práce se zaměřuje na rozpoznávání gest z webové kamery pro ovládání PC. Řešení úlohy se skládá z klasifikace barvy kůže trénované z obličeje, sledování shluku a klasifikace tvaru ruky pomocí Histogram of oriented gradient deskriptoru. Jsou porovnávány různé volby deskriptoru a různé volby SVM klasifikátoru. Součástí práce je vytvořený program pro sběr gest, jejich natrénování a klasifikací gest z kamery.

The work focuses on hand gesture recognition from webcam to control the PC. Solution of problem consists of a skin color detection trained in facial, blob tracking and classification of hand gesture with histogram of oriented gradient descriptor. There is comparation of different options of desriptors and different choices of SVM classifier. The work contents software for collection gestures, training gestures and classification hand gestures from camera.

Poděkování

Rád bych touto cestou poděkoval svému vedoucímu práce Ing. Marku Hrúzovi za odborné konzultace a cenné rady při zpracování této práce. Také bych rád poděkoval svému zaměstnavateli za vstřícnost, pochopení a poskytnutí času pro studijní aktivity.

Obsah

Seznam obrázků

Kapitola 1

Současný stav řešení

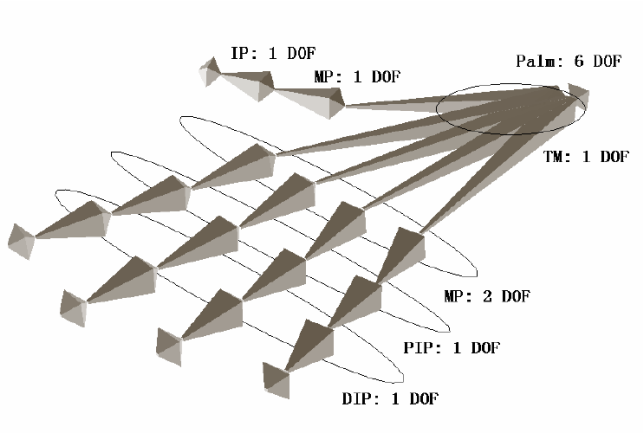
Rozpoznávání obrazu se rozšiřuje i do komerční sféry osobních počítačů, mobilních telefonů, automobilů a dalších přístrojů vybavených kamerovým systémem. Jedna z takto využívaných možností je vytvoření periférie pro neverbální komunikaci mezi uživatelem a technikou v podobě rozpoznávání gestikulace od uživatele. Množství gest při komunikaci závisí na entropii informace a rychlosti komunikace. Může se jednat jen o pár gest rozhodčího ve sportu, přes gesta dokreslující sdílenou informaci u herce či řečníka, až po několik tisíc gest ve znakové řeči.

Tato práce se zaměřuje na ovládání gest s běžně dostupným hardware osobních počítačů které jsou vybaveny jednou webovou kamerou. Výsledkem má být software, který rozpozná gesto jedné ruky. Úvodní kapitola má shrnout některá dnes používaná řešení rozpoznávání gest. Zaměří se na projekty, z kterých tato práce vychází nebo řeší podobný úkol. Nejprve si určíme, jak lze gesto chápat, jak lze gesto ruky popsat v počítačovém vidění. V dalších částech jsou popsány detekce oblasti ruky na scéně, rozpoznání tvaru ruky a její trajektorie řešené v jiných pracích.

1.1 Popis gesta

Tvar ruky je dán různým postavením prstů vůči sobě s fyziologickými omezeními danými klouby, šlachami a svaly. Gesto rukou je dáno nejenom jejím tvarem, ale i natočením ruky vůči pozorovateli. Například gesto, kdy ukazováček ukazuje na konkrétní věc či osobu nebo gesto pozor, kdy ukazováček míří vzhůru, mají zcela odlišný význam, ačkoli prsty jsou vzájemně stejně postavené. Dále význam gesta může být rozšířen i o trajektorii pohybu ruky, a spolu s tvarem a natočením ruky lze získat astronomické množství možných gest.

Realistický 3D model lidské ruky na Obrázku 1.4 má podle Článku [HDC] 26 stupňů volnosti. Jeden stupeň volnosti (protažení / ohnutí) je pro každý mezičlánkový (interfalangeální) kloub mezi články prstů a nebo kloub na zápěstí (interfalangeální). Dva stupně volnosti (protažení / ohnutí a přitažení / odtažení) pro každý metakarpofalangetální kloub kromě palce, (u kterého je jeden stupeň volnosti) a jeden stupeň volnosti (krut) pro každý trapeziometakarpální klouby kromě palce (který má dva stupně volnosti pro přitažení / odtažení a krut). Dlaň má šest stupňů volnosti pro zápěstní pohyby a rotaci.



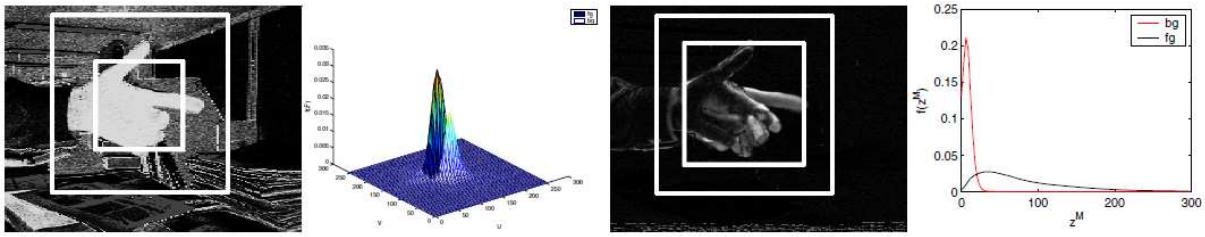
Obrázek 1.1: Virtuální 3D model ruky z Článku [HDC] s popisem stupňů volnosti.

1.2 Detekce oblasti ruky

Při rozpoznávání gest nebývá k dispozici jenom obraz s gestikulovanou rukou, ale scéna bývá mnohem složitější a ruka je jenom malá část obrazu. Proto je zapotřebí ruku od zbytku scény odlišit, určit tak oblast, kde se ruka nachází, a pak použít klasifikátor tvaru ruky. Kromě problémů se složitou scénou ovlivňuje řešení úlohy i kvalita záznamu. Z fyzikálního hlediska se musí splnit pro senzory podmínky minimální světlost pro digitalizaci obrazu a nastavení ohniskové vzdálenosti. Z matematického hlediska tu jsou podmínky, kde je zapotřebí splnit věty jako je vzorkovací teorém nebo míra šumu a ruchů vzniklých při digitalizaci. Úloha se dá usnadnit rozšířením hardware, kde se používá například více kamer pro stereometrii, infra kamera, gyroskopické senzory, nebo barevnými terči na scéně. Bez použití výše zmíněného hardware a pomůcek se může oblast určit jen z barvy, pohybu a nebo tvaru.

Oblast ruky z barvy získáme klasifikátorem oddělující barvu kůže od zbytku scény. Jeden z klasifikátorů vychází z převodů do barevných modelů s jasovou složkou. Tím se oddělí barevná složka od složky šedotónové a barevná složka pak není závislá na intenzitě světla či vzniklých rušivých stínů na scéně. Tyto Články [Soo] a [Ste06] vycházejí z vlastnosti, že barevná složka pigmentu kůže má v histogramu normální rozdělení četnosti. Liší se však výchozím barevným modelem, kde se v [Soo] používá YCbCr model a v [Ste06] YUV model. V obou případech se referenční barva určí z obličeje, který se detekuje snadněji, než ruka. V Článku [Ste06] se upřesňuje barva z nalezené oblasti ruky a i jejího okolí zobrazené na 1. a 2. Obrázku 1.2. Výsledkem je pak podmíněná pravděpodobnost výskytu barvy kůže a barvy okolí.

Zcela jiný přístup k řešení klasifikace barvy kůže je napsán v Článku [Bue]. V něm se vychází z myšlenky, že většina jasových barevných modelů (jako je HSV, HSL, YCbCr, LUV) funguje za předpokladu, že scéna je osvětlena bílým a konstantním světlem. Výpočet barvy kůže ze snímané scény vychází z rovnice obsahující dvě složky. Jednou z nich je zdrojové světlo s určenou vlnovou délkou, a druhou je odraz od osvícených předmětů. Na základě tohoto modelu se řeší rozložení RGB modelu v gaussovských směsí, které se určují pomocí *Expectation–maximization* (EM) algoritmu. Z gaussovské směsi je dána pravděpodobnost, která určuje, zda se jedná o pokožku či o pozadí scény v závislosti na barvě v RGB prostoru. Podíl těchto dvou pravděpodobností se porovná s hodnotou rizika a tím se klasifikuje, zda daná barva je či není barva kůže.



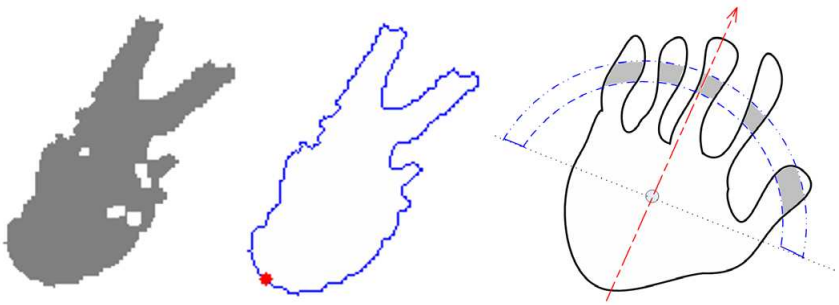
Obrázek 1.2: Tento obrázek je z Článku [Ste06], který ukazuje plochy oblasti ruky a okolí (1. a 3. obrázek, kde 1. je z barevných kanálů $[U, V]$ a 3. je z rozdílového obrázku.) a pravděpodobnostní rozložení těchto ploch (2. a 4. obrázek).

Za předpokladu, že na scéně nejsou žádné pohyblivé objekty a že při gestikulaci rukou pohybujeme, lze použít tyto pohyby k detekci oblasti výskytu ruky. Této vlastnosti využívá [Ste06] k upřesnění již dříve popsaného modelu pro určení barvy kůže. Pohyb pak určuje z rozdílového obrazu, kde určí pravděpodobnost pohybu ruky a pravděpodobnost pohybu pozadí zobrazené na 3. a 4. Obrázku 1.2. Z pravděpodobností pak určí hodnotu prahu v rozdílovém obrazu.

1.3 Klasifikace tvaru ruky

Objekt získaný z videozáznamu je pro strojové klasifikování nevhodný. Tvar potřebujeme popsat do vektoru příznaků nebo syntaktického popisu, který co nejvěrohodněji reprezentuje tvarové vlastnosti objektu, a tento popis je potom předán klasifikátoru k rozpoznání.

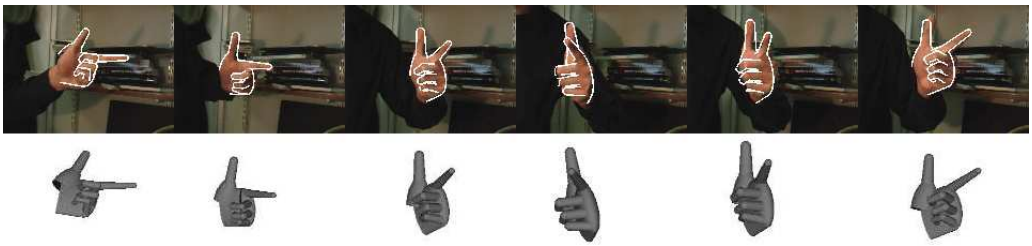
Jednoduchou aplikací rozpoznávání tvaru ruky je klasifikování jejího obrysu. U použití detekce oblasti máme k dispozici dvourozměrný binární obraz, kde jedna barva značí pozadí a druhá vyznačuje ruku. Ve Článku [WZD07] určí centrální moment vybrané oblasti a vypočítá křivku v polárních souřadnicích se středem centrálního bodu. Na této křivce potom pomocí *K-Means* klasifikátoru určují, které prsty jsou natažené.



Obrázek 1.3: Ukázka obrysu ruky a modelu ruky pro klasifikaci natažených prstů z Článku [WZD07].

U rozpoznávání složitějších gest je zapotřebí sledovat více vlastností, než jen obrys ruky. Ve Článku [Bue] se pro rozpoznávání tvaru používá metoda *template matching*, která porovnává části obrazu se vzorem hledaného objektu. Jako referenční vzor je používána textura ruky o velikosti 30x30 pixelů. Pro každý tvar ruky daného gesta se používá dvou stranově obrácených vzorů pro

levou a pravou ruku. Další metoda popisu tvaru je například *Histogram of oriented gradients (HoG)*, která byla prezentována ve Článku [Dal] v roce 2005. Technika rozpoznávání spočívá na základě četností gradientů v lokálních částech obrazu. Objekt se klasifikuje pomocí distribuce intenzity gradientů nebo směrů hran. *HoG* je úspěšně používaný na rozpoznávání osob, zvířat či automobilů z videa nebo statických obrazů. Další možnosti, jak popsat a klasifikovat tvary, můžete nalézt například v [BHS93].



Obrázek 1.4: Ukázka snímané ruky s odpovídajícím 3D modelem ruky [STTC06]. Horní řada obrázků je snímaná ruka s obrysem z modelu a spodní řada je 3D model ruky, který koresponduje s horní řadou.

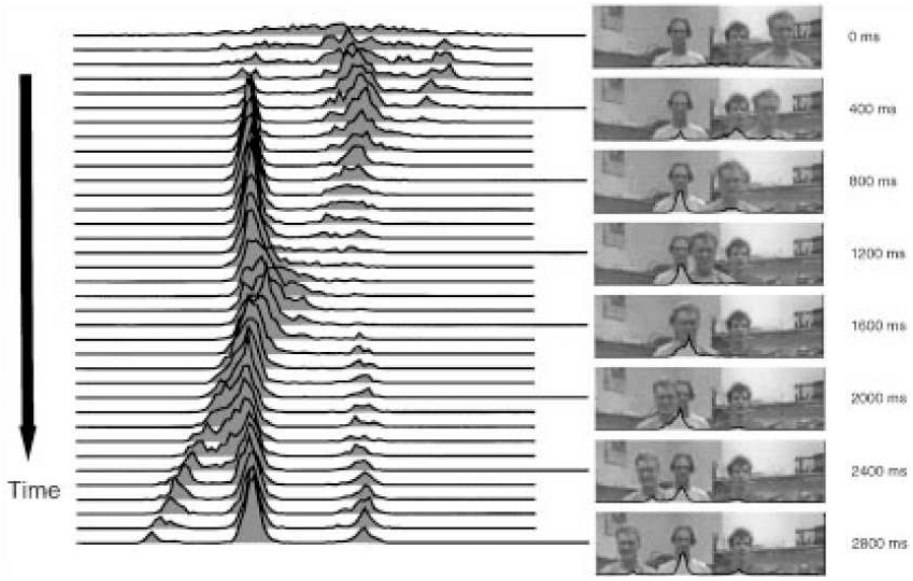
Jeden z nejlepších způsobů rozpoznávání tvaru ruky, založeném na virtuálním 3D modelu zobrazen na Obrázku 1.4. Tento model se používá především pro rozpoznávání znakové řeči. Daný model má stejnou pohyblivost jako má lidská ruka a různé varianty postavení prstů vůči sobě nevyžadují žádný vzorek nebo referenční texturu. Během rozpoznávání se postupně mění tvar a natočení modelu ruky, jež se postupně porovnává s nalezenou oblastí ruky a určuje se největší shoda. Více o rozpoznávání pomocí virtuálního modelu 3D ruky je popsán například ve Článku [STTC06]

1.4 Trajektorie pohybu ruky

Další význam gesta dodává trajektorie ruky, stejně jak je tomu i u tvaru ruky, složitost trajektorie se mění v závislosti na okolnostech. Například ve znakové řeči je potřeba sledovat složitou trojrozměrnou trajektorii, rychlost ruky a zároveň změnu jejího tvaru během trajektorie. V případě gestikulace během přímé řeči si vystačíme jenom se směrem pohybu.

Jeden jednoduchý popis pohybu ruky je v již zmiňovaném Článku [Soo], kde z dvou posledních známých poloh se vypočítá 1. a 2. derivace pohybu a z tohoto popisu odhaduje, kde je ruka na dalším snímku. Ve Článku [Bue] používají kalmanův filtr, který rekurzivním odhadem určuje vnitřní stav a dokáže odhadovat polohu při zašuměném měření. Dále používaná metoda, která se používá při sledování objektů v reálném čase, je *Camshift*. Byl vyvinut pro ovládání her pomocí pohybů hlavy. Tato metoda byla uvedena v roce 1998 ve Článku [Bra] a jedná se o adaptivní verzi metody *Mean-shift*. Na rozdíl od *Mean-Shiftu Camshift* dokáže měnit velikost okna během konvergence a úhel natočení sledovaného shluku.

Velmi úspěšnou metodou pro sledování pohybu je *Condensation algorithm*, popisovaný ve Článku [IB98] který umožňuje velmi obecné vyjádření pravděpodobnostního popisu. Experimentální výsledky ukazují, že *Condensation algorithm*, umožňuje velmi kvalitní sledování při křížení objektů a umožňuje použít nelineárních složitějších modelu než se používá například v kalmánově filtru. Na Obrázku 1.5 je ukázka použití *Condensation algorithm* pro sledování osob.



Obrázek 1.5: Ukázka sledování tří osob pomocí *Condensation algoritmu*. Vpravo na scéně jsou tři osoby, kde se jedna pohybuje a dvě stojí. Na levé části obrazu je pravděpodobnostní odhad osob. V inicializaci má pravděpodobnost zhruba normální rozdělení, které se rychle vyvijí ve vrcholky korespondující na každou ze tří osob na scéně.

Kapitola 2

Teoretický základ

V úvodním textu již bylo nastíněno pár příkladů řešení této úlohy. Pro rozpoznávání gest potřebujeme získat příznaky opisující tvar ruky a klasifikátor, který pomocí příznaku dané gesto zařadí. Obraz můžeme segmentovat z klasifikované barvy a sledovat tak pozici ruky, která gestikuluje. Tato kapitola je zaměřena na teoretický základ a popis procesů, který jsou v práci použity. Rozděluje se na klasifikaci barvy, trajektorii objektu a klasifikaci tvaru.

2.1 Klasifikace barvy

Klasifikátor barvy se trénuje z histogramu jednoho nebo více barevných kanálů. Výsledkem je scéna rozdělená na oblasti s pravděpodobným výskytem hledané barvy. Pro přesnější určení pravděpodobnosti se histogram počítá i z jiných barevných modelů, než který nám poskytuje hardware.

2.1.1 Barevné modely

Barva je v přírodě dána směsí světla různých vlnových délek. Různé barevné modely se snaží napodobit barvu co nejvěrněji. Rozdílnými pohledy na popis barvy (umělecký, grafický, fyzikální) a různou zobrazovací či záznamovou techniku zde vznikla celá řada barevných standardů. Většinou jde ale o podobné principy, které se dají shrnout do následujících rodin:

CIE [1] (*XYZ, Lab, Luv. Yuv ...*) – popisující barvu na základě fyzikálních vlastností, jako je vlnová délka světla a používají se jako referenční prostory.

YUV (*YUV, YDbDr, YIQ, YCbCr ...*) – byli vyvinutý pro přenos videosignálu standardy *PAL, SECAM, NTSC* a komprimaci videa.

RGB (*RGB, Adobe, scRGB, CMYK ...*) – pomocí míchání základních třech barev jsou používané na veškeré zobrazovací technice jako je monitor, projektory, tiskárny.

HSV (*HSV, HSL, RYB ...*) – odpovídají lidskému popisu vnímání barev a barvy jak jí člověk vnímá. Tyto modely využívají hlavně umělci a grafici.

V následující části textu si více přiblížíme některé zástupce těchto rodin, které jsou v práci používané.

[1] Mezinárodní komise pro osvětlování (Commission internationale de l'éclairage, CIE)

RGB je dominantní ve video elektronice jako součást obrazového signálu a je nejpoužívanějším modelem pro datové struktury v programovacích jazycích a datech v souborech. Pro náš případ potřebujeme pracovat s modelem, který rozdělí barvu na jasovou a chromatickou složku. Tuto vlastnost mají modely z kategorie *CIE* a *YUV*, kde si u vybraných modelů blíže popíšeme jejich fyzikální či matematický popis a převod z *RGB* modelu.

YCbCr a YUV *YUV* se používá ve standardech kódování barevného signálu *PAL*. Tento standard byl navržen pro zpětnou kompatibilitu černobílých přijímačů, které využívaly pouze složky Y. Chromatické složky se používají v rozsahu od -0.5 do +0.5, jasová složka má rozsah od 0 do 1. *YCbCr* obsahuje navíc váhové koeficienty $K_R - K_B$, a je vhodnější pro použití digitálního přenosu a komprese *JPEG* nebo *MPEG*. Y' je řazen do luminační (jasové) složky a *Cb* s *Cr* do modrých a červených chrominačních složek. (někdy krácen na *YCC*)

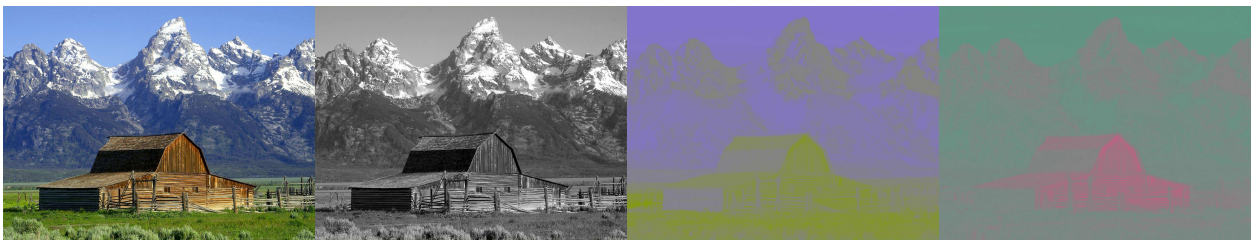
$$Y' = K_R \cdot R' + (1 - K_R - K_B) \cdot G' + K_B \cdot B' \tag{2.1}$$

$$P_B = \frac{1}{2} \cdot \frac{B' - Y'}{1 - K_B} \tag{2.2}$$

$$P_R = \frac{1}{2} \cdot \frac{R' - Y'}{1 - K_R} \tag{2.3}$$

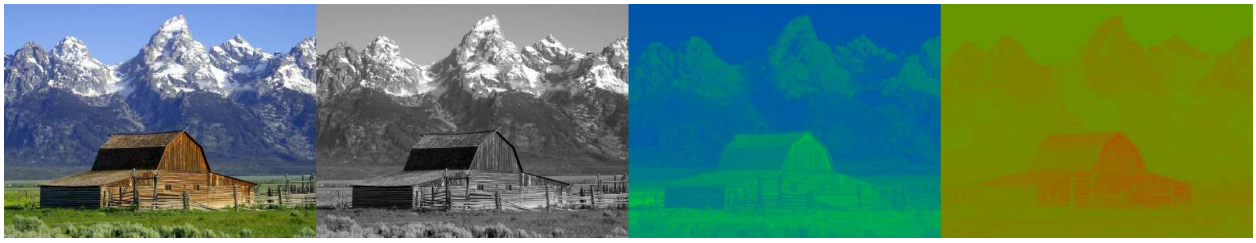
Kde *Kb* a *Kr* jsou obyčejně odvozeny od odpovídajícího *RGB* prostoru. Například pro standard *ITU-R BT.601 (SDTV)* má koeficienty $Kb = 0.114, Kr = 0.299$ a pro *ITU-R BT.709* standard (*HDTV*) má koeficienty $Kb = 0.0722, Kr = 0.2126$. Tento rozdíl je dán odlišností zobrazení mezi staršími *CRT* monitory a novějšími monitory či display.

$$\begin{bmatrix} Y' \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.14713 & -0.28886 & 0.436 \\ 0.615 & -0.51499 & -0.10001 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{2.4}$$



Obrázek 2.1: Barevný obrázek a jeho prvky Y, Cb a Cr. (CC [2] BY-SA 2.5)

Lab, CIE XYZ Barevný prostor CIE *XYZ* a *Lab* (který je z *XYZ* odvozen) vychází ze studie fyziologie člověka. Vnímání barev může být rozděleno na 2 části – jas (vnímají tyčinky) a barvu (vnímají čípky). Prostor *CIE XYZ* byl záměrně navržen tak, že parametr *Y* vyjadřuje jas a vlastní barva je specifikována dvěma odvozenými parametry *X* a *Z*. *Lab* je navržen tak, že má nelineární vztah mezi složkami *L*, *a* a *b*, které mají napodobovat nelineární vlastnost oka. To umožní, že



Obrázek 2.2: Barevný obrázek a jeho prvky Y, U a V. (CC BY-SA 2.5)

změna hodnoty v prostoru barvy způsobuje přibližně stejnou vizuální změnu barvy. Tato vlastnost umožní, že podobné barevné odstíny jsou v chromatických složkách zobrazeny v kruhu. V ostatních barevných prostorech jsou tyto barevné odstíny zobrazeny v jiných strukturách jako je elipsa nebo obdélník. Neexistuje jednoznačně daný vzorec pro převod mezi *RGB* nebo *CMYK* hodnoty a *Lab*, protože *RGB* a *CMYK* barevné modely jsou závislé na hardwaru. *RGB* nebo *CMYK* hodnoty je nutno nejprve transformovat na konkrétní absolutní barevný prostor, jako *CIE XYZ* nebo *sRGB*. Tato úprava bude závislá na zařízení, ale výsledná data z transformace budou na přístroji nezávislá a potom převeditelná do *Lab*.

Obrázek 2.3: Barevný obrázek a jeho prvky L, a a b. (CC BY-SA 2.5)

HSV, HSL HSL i HSV popisují barvy jako body ve válci nebo kuželu. Centrální osa sahá od černé k bílé a udává hodnotu jasu nebo množství bílého světla (*Value* nebo *Lightness*). Úhel kolem osy odpovídá odstínu nebo barevnému tónu (*Hue*) a vzdálenost od osy odpovídá sytosti barvy, představující množství šedi v poměru k odstínu *Saturation*.

$$h = \begin{cases} \text{nedefinován}, & \text{jestliže } max = min \\ 60° \times \frac{g-b}{max-min} + 0°, & \text{jestliže } max = r \text{ a } g \geq b \\ 60° \times \frac{g-b}{max-min} + 360°, & \text{jestliže } max = r \text{ a } g < b \\ 60° \times \frac{b-r}{max-min} + 120°, & \text{jestliže } max = g \\ 60° \times \frac{r-g}{max-min} + 240°, & \text{jestliže } max = b \end{cases} \tag{2.5}$$

$$l = \tfrac{1}{2}(max + min) \tag{2.6}$$

$$s = \begin{cases} 0, & \text{jestlize } l = 0 \text{ nebo } max = min \\ \frac{max-min}{max+min} = \frac{max-min}{2l}, & \text{jestliže } 0 < l \leq \frac{1}{2} \\ \frac{max-min}{2-(max+min)} = \frac{max-min}{2-2l}, & \text{jestliže } l > \frac{1}{2} \end{cases} \tag{2.7}$$

2.1.2 Pravděpodobnostní popis histogramu

Histogram obrázku vyjadřuje poměrné zastoupení počtu pixelů s odpovídající hodnotou jasu či barvy. Vícerozměrný histogram vyjadřuje poměrné zastoupení pixelů popsaný vektorem nebo více složkami barvy. Nyní pracujeme s barevnými modely s jasovou složkou, kde pro počítání dvourozměrného histogramu používáme vektor z chromatických složek. Vlivem nepřesnosti snímání a vlivu nekonzistentní barvy je v histogramu reprezentována jako shluk bodů, které mohou být aproximováni normálním rozdělení.

Pro určení dvourozměrného normálního rozdělení potřebujeme znát střední hodnotu a kovarianční matici. Tyto parametry můžeme získat pomocí statistického popisu *Moment*. Obecný tvar momentu je vyjádřen pomocí následujícího vztahu a z něho získáme střední hodnotu μ.

$$M_{ij} = \sum_x \sum_y x^i y^j I(x,y) \qquad E(x,y) = \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix} = \left[\frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}} \right]^T \tag{2.8}$$

Pro kovarianční matici potřebujeme ještě centrální moment, který je ve tvaru

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q f(x,y) \tag{2.9}$$

ze kterého vypočteme druhé centrální momenty.

$$\mu'_{20} = \frac{\mu_{20}}{\mu_{00}} = \frac{M_{20}}{M_{00}} - \bar{x}^2 \qquad \mu'_{02} = \frac{\mu_{02}}{\mu_{00}} = \frac{M_{02}}{M_{00}} - \bar{y}^2 \qquad \mu'_{11} = \frac{\mu_{11}}{\mu_{00}} = \frac{M_{11}}{M_{00}} - \bar{x}\bar{y} \tag{2.10}$$

Hodnoty μ'_{20} a μ'_{02} vyjadrují rozptyly jednotlivých veličin (pro x a y). Nyní můžeme určit kovarianční matici, která je ve tvaru

$$\operatorname{cov}(x,y) = \begin{bmatrix} \mu'_{20} & \mu'_{11} \\ \mu'_{11} & \mu'_{02} \end{bmatrix} \tag{2.11}$$

$$\tag{2.12}$$

Tyto vztahy nám popisují dvourozměrné normální rozdělení a můžeme pomocí Gaussovy funkce určit hustotu pravděpodobnosti:

$$f_X(x) = \frac{1}{\sqrt{(2\pi)^2 |\Sigma|}} \mathrm{e}^{\left(-\frac{1}{2}(x-\mu)' \operatorname{cov}(x,y)^{-1} (x-\mu) \right)} \tag{2.13}$$

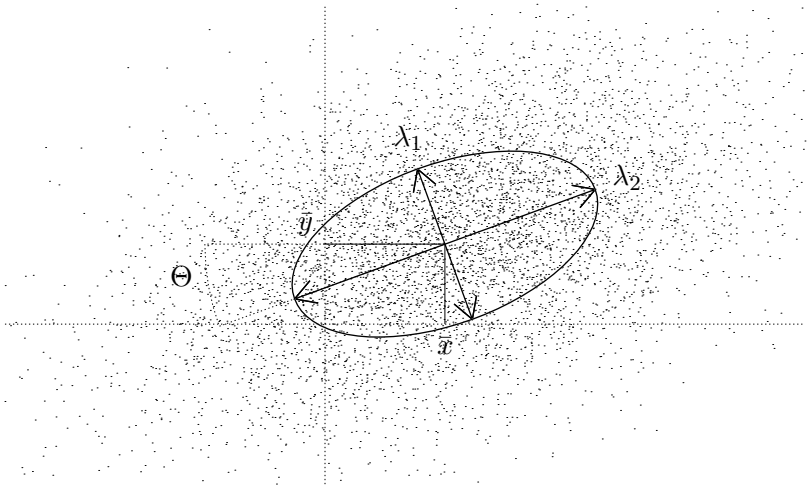
Kde Σ je kovarianční matice a $|\Sigma|$ je determinant kovarianční matice. V případě jednorozměrného normálního rozdělení nám stačí znán hodnotu rozptylu σ.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \mathrm{e}^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2.14}$$

Dále z momentů můžeme určit ještě další vlastnosti dvourozměrného normálního rozdělení jako jsou poloosy elipsy λ a jejich orientace Θ.

$$\lambda_i = \sqrt{\frac{\mu'_{20} + \mu'_{02} \pm \sqrt{4\mu'^2_{11} + (\mu'_{20} - \mu'_{02})^2}}{2}} \qquad \Theta = \frac{1}{2} \arctan\left(\frac{2\mu'_{11}}{\mu'_{20} - \mu'_{02}}\right) \tag{2.15}$$

Tyto parametry jsou názorně zobrazeny na Obrázku 2.4.



Obrázek 2.4: Příklad dvourozměrného normálního rozdělení a popis veličin

Podobně, jako se z obrázku vypočítá histogram, lze zpětně z obrázku a histogramu vypočítat obrázek, který pro každý pixel přidělí hodnotu z histogramu. Pokud použijeme histogram aproximovaný normálním rozdělením pro hledanou barvu, vyjadřuje tento obraz pro každý pixel s jakou pravděpodobností se shoduje jeho barva s hledanou barvou. Takto sestavený obrázek se v odborné literatuře nazývá *back projection*.

2.1.3 Získání oblastí

Filtrovaná barva z obrázku nám vyjadřuje pro každý pixel pravděpodobnost barevné shody pixelu s barvou objektu, který chceme rozpoznávat, ale nepodává přesnou informaci o jeho oblasti. Na tomto pravděpodobnostním obrazu se mohou objevovat různé rušivé pixely, které mají buď vysokou pravděpodobnost mimo oblast objektu na scéně, nebo naopak malou pravděpodobnost v oblasti objektu. Tyto chyby mohou být způsobeny již na scéně, kde je barva mimo objekt zaměnitelná s hledaným objektem nebo vlivem nevhodného osvětlení a odrazů vzniknou na objektu. Dále chyby

mohou vznikat při digitalizaci v podobě šumu, nebo při přechodu hrany vznikají nepřesnosti vlivem rozlišení kamery.

Filtrovaný obraz segmentujeme jednoduchou metodou prahování, která rozdělí binárně pixely na hodnotu 1 s pravděpodobností vetší než práh a opačném případě 0. Matematický zápis transformace obrazu f na binární obraz g je

$$g(i,j) = \begin{cases} 1 & \text{jestliže } f(i,j) \geq \text{práh} \\ 0 & \text{jestliže } f(i,j) < \text{práh} \end{cases} \tag{2.16}$$

Binární obraz lze vyjádřit jako 2D bodovou množinu. Díky tomu můžeme na binární obraz použít morfologické transformace, která je založena na teorii množin, integrální algebře a algebře svazů.

Morfologické transformace mění tvar a velikost geometrických objektů v obraze, a cílem je odstranění šumu, a zjednodušení tvaru objektů. Morfologická transformace je dána relací mezi obrazem (bodová množina) a jinou bodovou množinou, které se říká strukturní element. Základními morfologickými transformacemi jsou dilatace $\oplus$, eroze $\ominus$, otevření $\circ$ a uzavření $\bullet$. Dilataci můžeme vyjádřit jako sjednocení posunutých bodových množin (obraz a strukturní element) a erozi jako jejich průnik.

$$A \ominus B = \bigcap_{b \in B} A_{-b} \qquad A \oplus B = \bigcup_{b \in B} A_b \tag{2.17}$$

Příklad operací dilatace $\oplus$, eroze $\ominus$ je graficky znázorněn na Obrázku 2.5 Kombinací těchto dvou

(a) Original (b) Dilatace (c) Eroze

Obrázek 2.5: Dilatace zvětšuje objekty o strukturní element a zaplňuje díry menší než strukturní element. Eroze naopak zmenšuje objekty, rozděluje oblasti o strukturní element a objekty menší než strukturní element vymizí.

operací získáme operace otevření a uzavření. Otevření můžeme vyjádřit jako erozi následovanou dilatací a uzavření naopak.

$$A \circ B = (A \ominus B) \oplus B \qquad A \bullet B = (A \oplus B) \ominus B \tag{2.18}$$

Příklad operací otevření $\circ$ a uzavření $\bullet$ je graficky znázorněna na Obrázku 2.6

Posloupnost těchto kroků (filtrace barvy, prahování a morfologické transformace) nám umožňují objekty popisovat pomocí velikosti plochy – nebo-li počet pixelů obsahující souvislou plochu, můžeme určit výšku a šířku objektu, rozměry opsaného obdélníku a jiné vlastnosti. Díky tomu můžeme objekty porovnávat a odstranit ty, co nám velikostně nevyhovují. Příklad této segmentace je zobrazen na Obrázku 2.7.

(a) Original (b) Otevření (c) Uzavření

Obrázek 2.6: Otevření a uzavření zachovávají původní rozměr. Otevření oddělí objekty spojené užší šíjí než je velikost strukturního elementu a odstraní malé detaily. Uzavření spojí objekty, které jsou od sebe blíže, než je velikost strukturního elementu, zaplní malé díry a úzké zálivy.

(a) Výchozí obrázek (b) Pravděpodobnostní obrázek (c) Výběr souvislých oblastí

Obrázek 2.7: Posloupnost obrázků, které znázorňují postupně kroky při rozpoznávání barvy pigmentu. Obrázky (a) a (b) jsou okomentovány v předchozích sekcích. Obrázek (c) vznikl z obrázku (b) prahováním na hodnotě 64 (kdy pravděpodobnost je převedena do intervalu od 0 do 255), morfologickou operací uzavření s jádrem 4x4, vyplnění děr a výběru oblastí, které mají plochu opsaného obdélníku větší než 100 pixelů.

2.2 Trajektorie objektu

V této práci se jedná o sledování shluku v obrazu filtrované barvy z Podkapitoly 2.1 a určujeme tak jeho polohu, změnu polohy a velikost. Tuto polohu pak použijeme pro vymezení oblasti v jasové složce barevného modelu a získáme tím sledovaný objekt v šedotónovém obrázku.

2.2.1 Mean-shift a Camshift

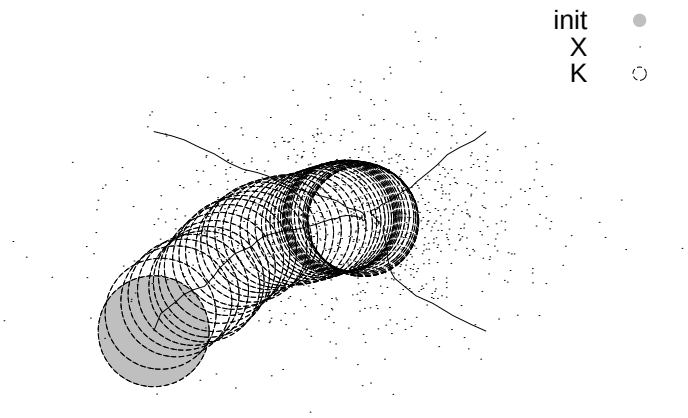
Mean shift je procedura pro nalezení maximální hustoty v diskrétních funkcích. Jedná se o interační algoritmus, kdy se každým krokem posune odhad směrem k nejbližší vyšší hustotě. Jednoduše řečeno, algoritmus postupně šplhá na vrcholek. *Mean shift* využívá okénka pro váhu okolních bodů pro nový odhad střední hodnoty. Vážený průměr hustoty z okénka K se vypočte jako

$$m(x) = \frac{\sum_{x_i \in N(x)} K(x_i - x) x_i}{\sum_{x_i \in N(x)} K(x_i - x)} \tag{2.19}$$

kde $N(x)$ je okolí vektoru x, pro které platí $K(x) \neq 0$. Okénko $K(x)$ může být například čtvercové, gaussovské či epanechnikovské

$$K(x) = \begin{cases} 1 & \text{jestliže } a \geq x \geq b \\ 0 & \text{jinak} \end{cases} \qquad K(x) = e^{-c||x||^2} \qquad K(x) = \begin{cases} \frac{3}{4}(1 - x^2) & \text{jestliže } |x| \geq 1 \\ 0 & \text{jinak} \end{cases} \tag{2.20}$$

V každém kroku algoritmu se potom hodnota vektoru x nastaví z funkce $m(x)$ a ukončuje se, pokud je posun menší než stanovený práh. Grafické znázornění algoritmu Mean-shift je zobrazeno na Obrázku 2.8



Obrázek 2.8: Příklad nalezení maximální hustoty X algoritmu Mean-shift pro různé počáteční souřadnice. Šedé kolečko vyjadřuje inicializační souřadnici. Kružnice znázorňují velikost okénka K a jednotlivé kroky algoritmu.

Camshift (Continuously Adaptive Mean Shift) prezentovaný v roce 1998 je navržen na sledování obličeje pro ovládání počítačových her. Algoritmus rozšiřuje *Mean shift* o adaptivní změnu okolí a

úhel natočení hustoty. Výslednou velikost a pozici okénka použije pro další snímek jako počáteční nastavení algoritmu. *Camshift* tak dokáže sledovat přibližující se nebo vzdalující se objekty ve video sekvenci. Adaptivní část úhlu natočení a velikost okénka se určuje z momentů popsaných v Podkapitole 2.1.2. Podrobenější popis algoritmu přesahuje rámec toho textu a více informací můžete nalézt ve článku [Bra].

2.2.2 Kalmanův filtr

Kalmanův filtr lze shrnout jako rekurzivní algoritmus pro odhad stavu lineárního gausovského pozorovaného systému s jednokrokovou predikcí. Bývá aplikován v navigacích, ovládání vozidel či letadlech. Je široce používaný v analýze časových řad, zpracování signálu nebo ekonometrii.

Lineární diskrétní systém je popsán následujícími vztahy:

$$\mathbf{x}_k = \mathbf{F}_k\mathbf{x}_{k-1} + \mathbf{u}_k + \mathbf{w}_k \tag{2.21}$$

$$\mathbf{z}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k \tag{2.22}$$

Rovnice 2.21 popisuje vývoj stavu $\mathbf{x}$, vstupní signál $\mathbf{u}_k$ a neměřitelný šum systému $\mathbf{w}_k \sim N(0, \mathbf{Q}_k)$. Rovnice měření 3.2.2 definuje vztah mezi výstupem systému $\mathbf{y}_k = \mathbf{H}_k\mathbf{x}_k$ šumem $\mathbf{v}_k \sim N(0, \mathbf{R}_k)$ a měřením $\mathbf{z}_k$.

Algoritmus pracuje ve dvou krocích. Krok predikce, kdy Kalmanův filtr vytváří odhady aktuálního stavu systému, a po tomto kroku je krok korekce, kdy z výsledků měření určí lepší aposteriorní odhad. Predikce je popsána následujícími vztahy.

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_k\hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}_{k-1}\mathbf{u}_{k-1} \tag{2.23}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k\mathbf{P}_{k-1|k-1}\mathbf{F}_k^{\mathrm{T}} + \mathbf{Q}_k \tag{2.24}$$

Rovnice 2.23 je odhadovaný stav systému a Rovnice 2.24 je odhad kovarianční matice. Korekce je popsána následujícími vztahy.

$$\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k|k-1} \tag{2.25}$$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}_k^{\mathrm{T}}(\mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_k)^{-1} \tag{2.26}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k\tilde{\mathbf{y}}_k \tag{2.27}$$

$$\mathbf{P}_{k|k} = (I - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k|k-1} \tag{2.28}$$

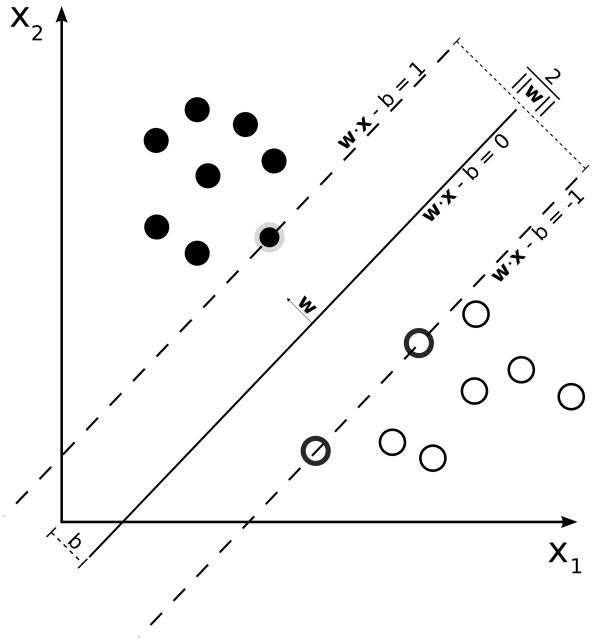
Rovnice 2.25 je měření výstupu, 2.26 je Kalmanův zisk, 2.27 aposteriorní odhad a 2.28 je aktualizovaná kovarianční matice.

2.3 Klasifikace tvaru

Klasifikace nám umožní zařazovat pozorované objekty do tříd a nebo nalézt objekt dané třídy na obrázku. V našem případě se pracuje se šedotónovou složky a potřebujeme rozhodnout, jaký tvar ruky specifikovaná část obsahuje. Pokud chceme objekt klasifikovat do jednoho z vybraných tvarů, musíme objekt formálně popsat. V následující části textu je popsáno několik metod, jak objekt popsat pomocí vektoru příznaků a jak ho následně zařadit do konkrétní třídy.

2.3.1 Support vector machines

Support vector machines (SVM) je metoda strojového učení s učitelem pro analýzu dat, která se používá pro klasifikaci a regresní analýzu. *SVM* umožňuje klasifikovat data do dvou tříd $\vec{x_i}, y_i$, kde $y_i \in -1, 1$ a $i = 1, 2, ..., n$. Cílem klasifikátoru je nalezení rozdělujícího pásu maximální šířky tak, aby se při klasifikaci do třídy minimalizovalo riziko klasifikace. Výsledkem je rozdělující optimální nadrovina $\vec{w} \cdot \vec{x} - b = 0$ (příklad na Obrázku 2.9), kde $\vec{w}$ je normálový vektor rozdělující nadrovinu a hodnota $b/||\vec{w}||$ je vzdálenost nadroviny k počátku souřadnicového systému. Pro lineární *SVM*



Obrázek 2.9: Příklad lineární rozdělující nadroviny pro dvou dimenzionální prostor

Trénovací data vyhovují následujícím podmínkám.

$$\vec{w} \cdot \vec{x} - b \geq +1 \qquad \text{pro } y_i = +1 \tag{2.29}$$

$$\vec{w} \cdot \vec{x} - b \leq -1 \qquad \text{pro } y_i = -1 \tag{2.30}$$

Tyto dvě podminky lze formulovat

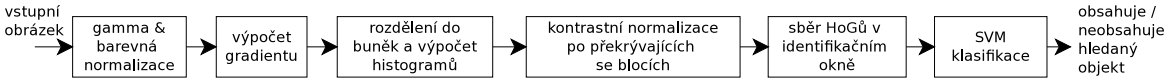
$$y_i \cdot (\vec{w} \cdot \vec{x} - b) - 1 \geq 0 \forall i \tag{2.31}$$

U nelineární *SVM* se trénovací data transformují do prostoru s vyšší dimenzi pomocí zvoleného jádra a postup klasifikace je pak stejný jako v případě lineárního *SVM*.

2.3.2 Histogram of oriented gradient

Histogram of Oriented Gradient descriptors, nebo-li *HoG* descriptors je technika rozpoznávání objektů, která byla vyvinuta pro rozpoznávání osob, zvířat či automobilů z videa nebo statických

obrazů. Tato technika vychází z myšlenky, že objekt lze popsat distribucí intenzity gradientů nebo směry hran. Posloupnost operací při použití *HoG* je znázorněno na schématu 2.10, kde jednotlivé kroky budu dále popisovat.



Obrázek 2.10: Diagram kroků výpočtu a klasifikace objektů

Na šedotónovém nebo barevném obraze provedeme gamma korekci, která srovná světelné intenzity objektů na scéně a zíkáme tak normalizovaný obraz I. Z obrazu I vypočteme gradientní obraz G, který vyjadřuje vektorové pole směru a velikosti největší změny intenzity. Diferenci obrazu v horizontálním a vertikálním směru získáme konvolucí jednoduché 1–D masky $[-1, 0, 1]$

$$I: \quad \text{  } \tag{2.32}$$

$$I_x = I * [-1, 0, 1] \quad \Rightarrow \quad I_x: \text{  } \qquad I_y = I * [-1, 0, 1]^T \quad \Rightarrow \quad I_y: \text{  } \tag{2.33}$$

Z takto získané diference I_x a I_y vypočteme velikost a orientace gradientu

$$|G| = \sqrt{I_x^2 + I_y^2} \quad \Rightarrow \quad |G|: \text{  } \qquad \theta = \arctan \frac{I_x}{I_y} \quad \Rightarrow \quad \theta: \text{  } \tag{2.34}$$

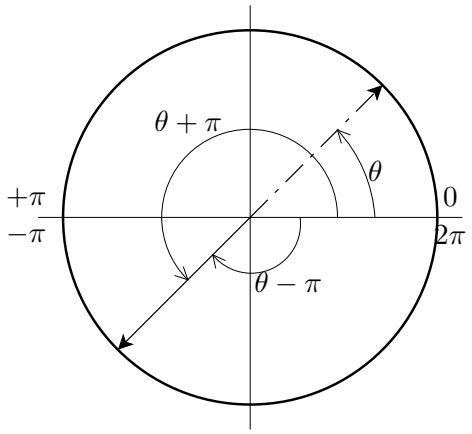
$$\tag{2.35}$$

Ve článku [Dal] jsou zmíněny i jiné masky pro výpočet gradientu, ale nedosahovaly takového výkonu. Pro vícekanálový obraz vypočteme odděleně gradienty pro každý kanál zvlášť a výsledný gradientní obraz získáme normováním gradientů ze všech kanálů.

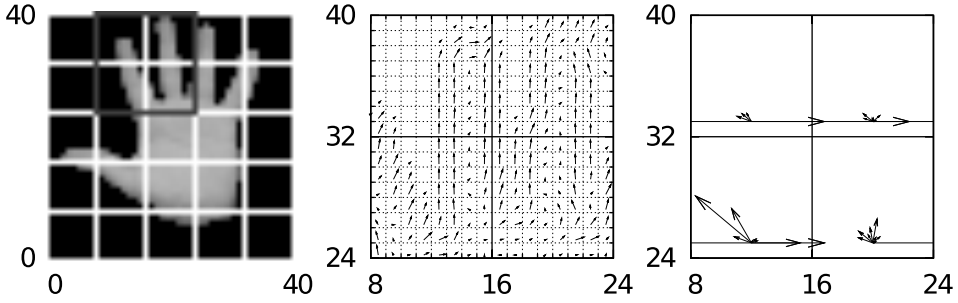
Dalším krokem v metodě *HoG* je rozdělit obraz G do buněk, ve kterých se budou počítat histogramy orientací gradientu. Buňky mohou být obdélníkového či radiálního tvaru. Pro výpočet histogramu potřebujeme určit interval na kterém se bude histogram počítat a počet tříd, na který se interval rozdělí. Úhel natočení gradientu θ je v intervalu $[-\pi - \pi]$ nebo $[0 - 2\pi]$ při použití znaménkového gradientu a interval $(0 - \pi]$ při použití neznaménkového gradientu. Převod intervalu ze znaménkového gradientu na interval neznaménkový je znázorněn na Obrázku 2.11. Ve článku [Dal] experimentálně ověřili, že pro klasifikaci osob je histogram nejvhodnější použít interval $(0-\pi]$ dělený do devíti tříd. Ilustrační příklad rozdělení obrazu do buněk a vypočtené histogramy v jednotlivích buňkách je znázorněn na následujícím Obrázku 2.12, kde velikost buňky je použita 8x8 pixelu a histogramy s četností v devíti třídách.

Kvůli změnám jasu a kontrastu musí být gradienty jednotlivých buněk lokálně normalizovány, což nás vede k seskupování jednotlivých buněk do bloků. *HoG* deskriptor je potom vektor normalizovaných buněk histogramu ze všech blokových oblastí. Bloky se mohou překrývat a pak některé buňky do výsledného deskriptoru mohou přispívat i vícekrát. 2.13 Pro normalizaci bloků lze použít několik různých normalizačních schémat:

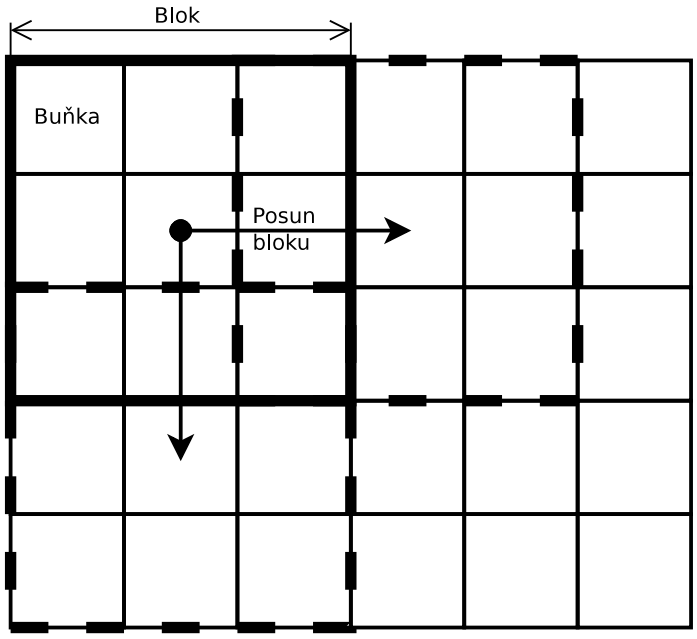
$$L^2 - \text{norm}, \; v = \frac{v}{\sqrt{||v||_2^2 + e^2}} \qquad L^1 - \text{norm}, \; v = \frac{v}{||v||_1 + e} \qquad L^1 - \text{sqrt}, \; v = \sqrt{\frac{v}{||v||_1 + e}} \tag{2.36}$$



Obrázek 2.11: Převod orientace gradientu do mezí $(0 - \pi]$



Obrázek 2.12: První graf znázorňuje rozdělení obrazu do buňek o velikosti 8x8 pixelů. Druhý znázorňuje gradienty pro každý pixel a na tředím grafu jsou hystogrami gradientů pro jednotlivé buňky.

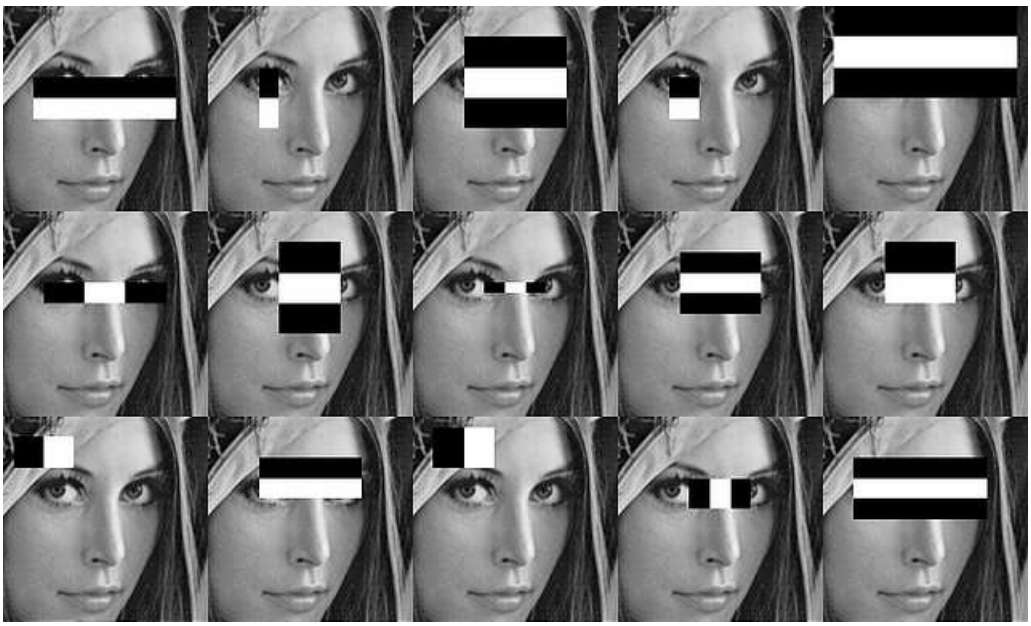


Obrázek 2.13: Na obrázku jsou znázorněny buňky, normalizační bloky a šipky znázorňují posuv normalizačního bloku po obrázku.

kde v je neznormalizovaný vektor obsahující všechny histogramy v daném bloku, $||v||_k$ jeho $k-$sqrt pro $k = 1, 2$ a e je malá konstanta. L^2-hys lze spočítat jako L^2-sqrt, kde je výsledek oříznut (limitován maximální hodnotou v na 0, 2) a normalizován. Finální deskriptor je tvořen všemi buňkami uvnitř všech bloků v detekčním okně. Konečnou fází při rozpoznávání objektů pomocí *HoG* deskriptorů je předání deskriptorů klasifikátoru. Ve článku [Dal] se pro klasifikaci používá lineární *SVM* klasifikátor popsaný v části 2.3.1

2.3.3 Haar-like příznaky

Detekce objektů pomocí *haar-like* příznaků se používá na objekty, které mají tvarově dané vlastnosti jako je například kolo, auto, postava, obličej ze předu, obličej z profilu a jiné objekty. Například u obličeje jsou tmavší oblasti kolem očí než oblasti v místě čela. Tyto vlastnosti se určují v detekčním okénku kde se vyhodnotí jestli daný objekt je či není v okénku obsažen. Tvary se popisují pomocí *haar-like* vlny, která se skládá z obdélníkových oblastí kde část obdélníku se k obrazu přičítá a část odečítá. Používá se asi 15 základních typů *haar-like* vln a její pozicí a velikostí v rámci detekčního okénka je příznak definován. Příklad detekčního okénka a postup vkládání *haar-like* vlny v okénku je na obrázku 2.14 Pro každou *haar-like* vlnu se vyhodnocuje jestli výsledný součet ploch je pod či nad stanoveným prahem a přidělí se jí hodnota +1 nebo -1. Klasifikátor pak vyhodnotí vážený součet všech těchto hodnot v rámci detekčního okénka. Kladná nebo záporná hodnota součtu určuje jestli je či není objekt v okénku obsažen. Postupném posouvání a změnou velikosti detekčního okénka v rámci obrazu se vyhodnotí kde všude je hledaný objekt v obrazu nalezen.

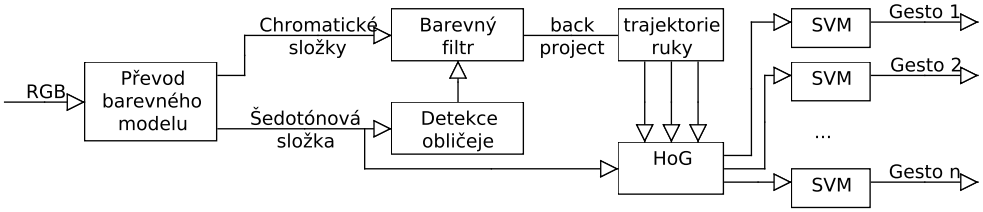


Obrázek 2.14: Příklad detekčního okénka a 15 ukázek přiložených *haar-like* vln pro detekci obličeje. Tento příklad je získaný z klasifikace obličeje knihovny OpenCV a celý proces obsahuje přes 200 příznaků. (CC BY-NC 3.0, autor Adam Harvey)

Kapitola 3

Systémové řešení

Řešení úlohy se skládá z rozpoznání barvy kůže na scéně, rozdělení na oblasti a jejich sledování, popsání tvaru v oblastech do příznaků, a klasifikace získaných příznaků. Zjednodušené schéma tohoto procesu je zobrazeno na obrázku 3.1 V této kapitole jsou rozepsány výše zmíněné kroky, jsou



Obrázek 3.1: Schéma řešené úlohy pro rozpoznávání gest. Vstupem je sekvence obrázků z kamery, videa nebo seznamu souborů. Výstupem je pro každé natrénované gesto jestli je či není na obrázku obsaženo.

doplněny ilustračními příklady z programu. Časová náročnost udávaná u příkladů je získána při výpočtu na procesoru Athlon 64 X2 2.6 GHz.

3.1 Rozpoznávání barvy kůže

Barva kůže je hlavně určena množstvím melaninu[1] v pokožce, který jí chrání proti poškození světlem. Různé populace jsou vyvíjeny s různou intenzitou slunečního záření, které způsobuje, že různé lidské rasy mají geneticky zakódované odlišné barvy pleti. Dále barva kůže se může změnit chemickou nebo operativní cestou jako je líčení či tetování, které často způsobují nerovnoměrné rozdělení barvy a nesplňuje předpoklad stejného odstínu.

Vstupní obrázek může být předzpracován geometrickou transformací přetočení nebo zrcadlení aby výstup byl pro člověka přirozenější. Dále zmenšení vstupního obrázku sníží výpočetní a paměťovou

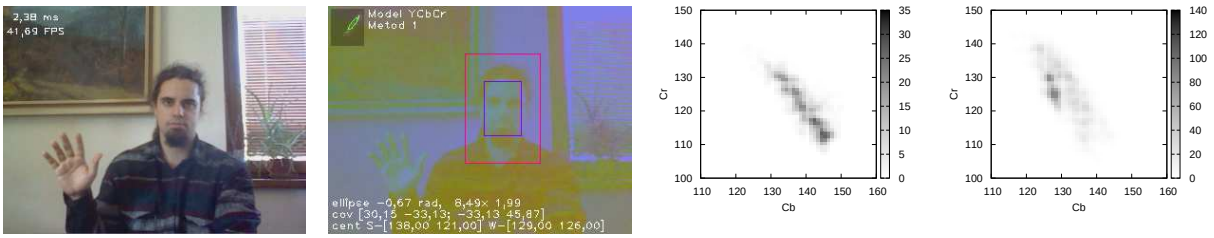
[1]Melanin je označení pro hnědý až černý pigment, který se vyskytují v tělech živočichů i prvoků.

náročnost úlohy, která dokáže pracovat s rozlišením již od 320x240 pixelů. Pro potlačení šumu typu sůl a pepř lze použít mediánový filtr.

Rozpoznávání barvy kůže je založeno na klasifikaci barvy chromatických složek popsaný v části 2.1. Jsou zde použity dvě metody z článků [Ste06, Soo] kterými se budeme blíže zabývat. V obou případech rozpoznávání barvy probíhá ve dvou fázích. Fáze trénovací, která se pouští jen na jeden snímek na začátku nebo při změně světelných podmínek, a fáze klasifikace barvy, kdy se z obrázku vypočte pravděpodobnostní obraz výskytu barvy kůže.

3.1.1 Trénování

V obou metodách se používá jako referenční oblast obličej, který detekujeme pomocí *Haar-like příznaků* (popsané v 2.3.3). V nalezené oblasti se vypočte z chromatických složek dvourozměrný histogram. V tomto histogramu je četnost pixelů barvy kůže dominantní a může se pomocí momentového popisu, vyjádřený v Rovnicích 2.10 až 2.11, spočítat parametry normálního rozdělení. Ve článku [Soo] se dvourozměrný histogram chromatických složek aproximuje na dva jednorozměrné histogramy v podprostoru hlavní a vedlejší poloosy elipsy $\lambda_{1,2}$ vyjádřené v Rovnici 2.15. Ve článku [Ste06] se kromě histogramu z oblasti obličeje počítá navíc i histogram z jeho okolí, který popisuje četnost barvy v pozadí. Příklad histogramů je zobrazen na obrázcích 3.2. Z výsledných histogramů



(a) Výchozí obrázek (b) Chromatické složky (c) 2D histogram obličeje (d) 2D histogram okolí obličeje

Obrázek 3.2: Ukázka jednotlivých kroků při výpočtu dvourozměrných histogramů. Obrázek (a) se převede do barevného modelu s chromatickými složkami, dále na obrázku (b) se určí oblast obličeje a jeho okolí. Obrázky (c) a (d) jsou již vypočtené histogramy.

se určí normální rozdělení které určuje pravděpodobnost výskytu kůže ($P(fg)$) a pravděpodobnost pozadí ($P(bg)$). Výsledná pravděpodobnost je pak rozdíl pravděpodobností $P(fg) - P(bg)$.

Parametry pravděpodobností se počítají jenom během trénování a můžeme si hodnoty předpočítat pro každý barevný vektor zvlášť. Získáme tak *look up* tabulku, která každému vektoru barvy z chromatické složky přidělí danou hodnotu pravděpodobnosti. Výpočetní náročnost jednotlivých kroků při trénování barvy je v tabulce 3.1. Generování 2D *look up* tabulky a 1D *look up* tabulky se pouští

	detekce obličeje	výpočet histogramu	výpočet momentů	generování 2D look up	generování 1D look up
čas[ms]	407	0.8	0.6	386	0.07

Tabulka 3.1: Časová náročnost trénování barvy. Vypočteno na obrázku o velikosti 640x480 pixelů

pro každou metodu zvlášť. Vysoká hodnota generování 2D *look up* tabulky je způsobena neefektivní implementací vypočtu dvourozměrného normálního rozdělení. Při správné implementaci by hodnota měla být o řád až dva menší.

3.1.2 Klasifikace

V metodě ve článku [Ste06] se klasifikace vypočte z chromatických složek a z *look up* tabulky vypočtenou při trénování. Příklad pravděpodobnostního obrazu vypočtený touto metodou je zobrazen na obrázku 3.3. Z příkladu je vidět že při použití pravděpodobnosti barvy pozadí se můžou

(a) Originální obrázek (b) $P(fg)$ (c) $P(bg)$ (d) $P(fg) - P(bg)$

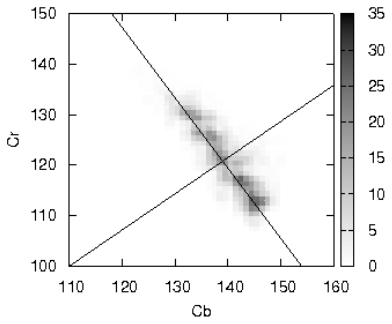
Obrázek 3.3: Ukázka klasifikace barvy kůže podle článku [Ste06]. Obrázek (a) je originální obrázek kde je zeleným čtvercem vyznačena oblast obličeje a červeným čtvercem jeho okolí z kterých je klasifikátor trénován. Obrázek (b) je klasifikovaná barva z oblasti obličeje. Obrázek (c) je klasifikovaná barva z okolí oblasti obličeje nebo–li barva pozadí. Výsledná pravděpodobnost barvy kůže je zobrazená na obrázku (d).

potlačit rušivé barvy (v tomto případě obraz) a zlepšit tak klasifikaci barvy. Barva pozadí nebývá konzistentní jako je tomu v případě kůže a při použití směsi normálních rozdělení pro barvu pozadí by se proces mohl ještě zlepšit.

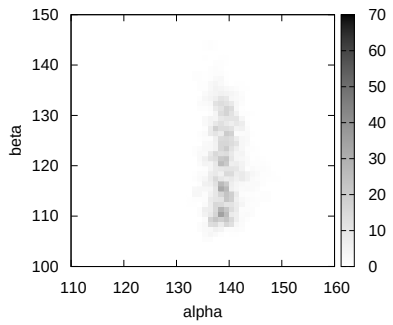
V metodě ze [Soo] nejprve musíme transformovat vektor chromatických složek o úhel Θ natočení elipsy do vektoru $[\alpha, \beta]$, na který lze použít dva jednorozměrné histogramy. Transformace kanálů se dá vyjádřit pomocí následujícího vztahu

$$\begin{bmatrix}\alpha\\ \beta\end{bmatrix} = \begin{bmatrix}\cos\Theta & -\sin\Theta\\ \sin\Theta & \cos\Theta\end{bmatrix}\begin{bmatrix}C1-\bar{x}\\ C2-\bar{y}\end{bmatrix} - \begin{bmatrix}\bar{x}\\ \bar{y}\end{bmatrix} \tag{3.1}$$

kde $C1$ a $C2$ jsou chromatické kanály barevného modelu a α a β jsou transformované kanály. Tato transformace umožní, že pravděpodobnost kanálů α a β je nezávislá, a pro výpočet barvy kůže je můžeme vynásobit. Nezávislost kanálů α a β je zřetelná z ukázky na Obrázku 3.4, kde jsou histogramy chromatických kanálů a transformovaných kanálů. Dva pravděpodobnostní obrazy pro každý kanál zvlášť (α a β) vypočteme z jednorozměrných *look up* tabulek. Jejich vynásobením získáme pravděpodobnostní obraz barvy kůže. Příklad pravděpodobnostního obrazu vypočtený touto metodou je zobrazen na obrázku 3.5. Z obrázků je patrné, že klasifikované barvy kanálů α a β nepopisují požadovanou barvu a obsahují velké množství rušivých barev. Jejich vynásobením získáme oblasti které mají společné, ostatní oblasti se vzájemně vyruší a výsledek je potom podobný s klasifikací na obrázku 3.3d v předchozí metodě. Výhody této metody je menší výpočetní a paměťová náročnost. Výpočetní náročnost jednotlivých kroků při klasifikaci barvy je v tabulce 3.2. Při klasifikaci kanálů α a β je započten i jejich transformace, která dobu výpočtu značně zvyšuje. Pokud by tato transformace vynechala a počítalo by se přímo s chromatickými kanály, výsledný čas bude menší než u výpočtu z 2D tabulky.

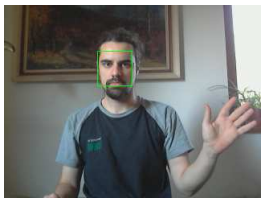


(a) Hisotram chromatických složek



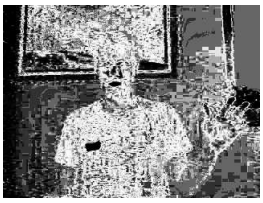
(b) Histogram transformovaných chromatických složek

Obrázek 3.4: Příklad histogramu chromatických složek na obrázku (a) a transformované chromatické složky $\alpha\beta$ na obrázku (b). Na obrázku (a) je vyznačena hlavní a vedlejší poloosy elipsy, které zobrazují podprostor histogramu pro kanál α a β.



(a) Originální obrázek

(b) $p(\alpha|x_\alpha)$



(c) $p(\beta|x_\beta)$

(d) $p(\alpha|x_\alpha)p(\beta|x_\beta)$

Obrázek 3.5: Ukázka klasifikace barvy kůže podle článku [Soo]. Obrázek (a) je originální obrázek kde je zeleným čtvercem vyznačena oblast obličeje z které je klasifikátor trénován. Obrázky (b) a (c) jsou klasifikované barvy pro kanály α a β, kde x vyjadřuje pixely s barvou kůže. Obrázek (d) je výsledný pravděpodobnostní obraz, který vznikl vynásobením obrázků (b) a (c)

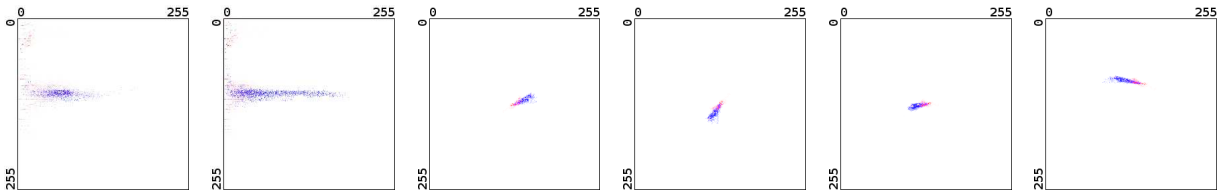
	transformace barvy	klasifikace barvy chromatických složek	klasifikace barvy α a β
čas[ms]	5.3	2.1	30

Tabulka 3.2: Časová náročnost klasifikace barvy. Vypočteno na obrázku o velikosti 640x480 pixelů.

Během filtrace jsou parametry pravděpodobností konstantní a při změně světelných podmínek se musí opět pustit trénování barvy. Pokud by se trénování pouštělo každý snímek včetně detekce obličeje, rychlost filtrace by se zpomalila tak 50x (viz tabulky 3.1 a 3.2). Tento nedostatek by šel řešit, jako je ve Článku [Ste06], adaptivní změnou klasifikátoru barvy trénované z ruky.

3.1.3 Výběr barevného modelu

Pro rozpoznávání barvy kůže byly vyzkoušeny barevné modely *HSV*, *HLS*, *YCbCr*, *YUV*, *Lab* a *Luv*. Příklad histogramů těchto barevných modelů je zobrazen na obrázku 3.6.



(a) HSV (b) HLS (c) YCbCr (d) YUV (e) Lab (f) Luv

Obrázek 3.6: Ukázka histogramů chromatických složek výše zmíněných barevných modelů. Na vodorovné a svislé ose jsou hodnoty chromatických kanálů převedené do 8–bit neznaménkové stupnici. Modré body znázorňují barvu kůže a červené barvu pozadí.

Barevné modely *HSV*, *HLS* mají četnost barvy v histogramu spíše obdélníkový charakter, což je způsobeno že v chromatické složce jedna osa charakterizuje barevný tón a druhá sytost barvy. Dále poloosy elipsy u normálního rozdělení této četnosti jsou téměř rovnoběžné s osami histogramu a nemusí se provádět transformace do kanálů α a β. Tyto barevné modely fungují za ideálních podmínek, kdy v pozadí je bílá nebo klíčovací barva.

Zbylé modely mají četnost barvy v histogramu elipsovitý charakter. Barevné modely *YCbCr* a *YUV* mají při filtraci shodné výsledky a tato vlastnost je patrná i z histogramů, kde jejich shluky jsou si podobné akorát jinak orientované. Model *Lab* má shluk četností o něco menší než je u *YCbCr* a model *Luv* má tvar podlouhlé elipsy. Při experimentování se všemi modely barvu kůže nejlépe reprezentuje model *Lab*, nicméně za specifických podmínek má mnohem horší výsledky než *YCbCr* který má robustnější charakter.

3.2 Sledování shluku

Z klasifikace barvy máme k dispozici pravděpodobnostní obraz výskytu barvy kůže v podobě na obrázku 3.3d nebo 3.5d. Na něm jsou vidět shluky, které reprezentují především obličej a ruce. Tyto shluky se dají lokalizovat pomocí metody *Camshift* popsané v 2.2.1 a opětovnému použití této metody pro další snímky můžeme získat trajektorii objektu. Shluky, které chceme sledovat, musíme v prvním kroku odhadnout jejich oblast. V následující části textu je popsán postup, jak shluky v inicializaci vybrat a jak se dále sledují.

3.2.1 Výběr oblastí

Pravděpodobnostní obraz rozdělíme pomocí prahování popsané v části 2.1.3. Pro různé barevné modely je volba prahu empiricky určena v rozmezí od 0.12 do 0.36. V případě programu je to

hodnota od 30 do 92, protože obrázek je v paměti uložen do 8–bit datového typu a obor hodnot pravděpodobnosti je v intervalu od 0 do 255. Volba prahu je volitelná a je přednastavena na hodnotu 64. Tím získáme binární obraz na který je použitá morfologická operace uzavření pro spojení nekonzistentních částí ploch. Na takto zpracovaném obraze často zůstávají malé plochy z rušivých barev, které můžeme zanedbat. Velikost plochy, které se mají zanedbat, je empiricky určena na velikost 100 pixelů. Zbylé plochy již můžeme použít jako počáteční oblasti pro sledování shluků. Celý tento proces segmentování trvá méně než 4 ms na obrázku o velikosti 640x480 pixelů.

3.2.2 Sledování

Vstup metody *Camshift* je pravděpodobnostní obraz z klasifikace barvy, a obdélníková oblast popsána souřadnicemi a rozměrem obdélníku. Tato oblast je buď určená z předchozího kroku a nebo jako počáteční oblast získaná ze segmentace obrazu. Výstupem jsou souřadnice, rozměry a úhel natočení obdélníkové oblasti. Předpokládáme, že oblast pro gesto prováděné dlaní a prsty má čtvercový tvar a tak sledovanou oblast můžeme popsat pomocí souřadnic a velikostí strany čtverce.

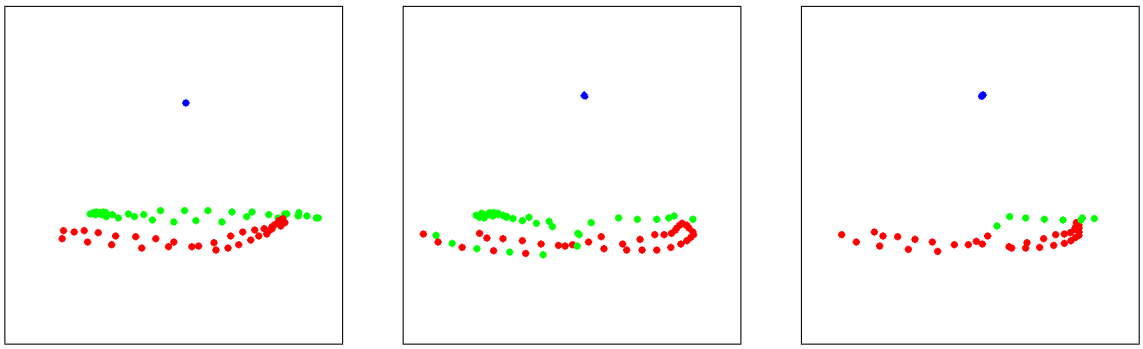
Měření oblasti shluku je dáno vektorem $\mathbf{z}_k = [x_k, y_k, s_k]^T$, kde x a y jsou souřadnice shluku a s je jeho velikost. Dále měřenou oblast můžeme zpřesnit *Kalmanovým filtrem*. Měřenou oblast rozšíříme stavem o rychlost a zrychlení jejího pohybu, kde stavový vektor má tvar $\mathbf{x}_k = [x_k, y_k, s_k, \dot{x}_k, \dot{y}_k, \dot{s}_k, \ddot{x}_k, \ddot{y}_k]^T$. Zrychlení velikosti $\ddot{s}$ zanedbáme, protože velikost oblasti se v průběhu pozorování mění minimálně. Stavová rovnice *Kalmanva filtru* z 2.21 má pak následující tvar:

$$\begin{bmatrix} x_k \\ y_k \\ s_k \\ \dot{x}_k \\ \dot{y}_k \\ \dot{s}_k \\ \ddot{x}_k \\ \ddot{y}_k \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & dt & 0 & 0 & \frac{1}{2}dt^2 & 0 \\ 0 & 1 & 0 & 0 & dt & 0 & 0 & \frac{1}{2}dt^2 \\ 0 & 0 & 1 & 0 & 0 & dt & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & dt & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & dt \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ s_{k-1} \\ \dot{x}_{k-1} \\ \dot{y}_{k-1} \\ \dot{s}_{k-1} \\ \ddot{x}_{k-1} \\ \ddot{y}_{k-1} \end{bmatrix} + 0 + w_{k-1} \tag{3.2}$$

Vstupní signál $\mathbf{u}_k$ je nulový, protože pohyb není řízen žádnou veličinou. Rovnice měření má následující tvar:

$$\begin{bmatrix} x_k \\ y_k \\ s_k \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ s_k \\ \dot{x}_k \\ \dot{y}_k \\ \dot{s}_k \\ \ddot{x}_k \\ \ddot{y}_k \end{bmatrix} + v_k \tag{3.3}$$

Při sledování více shluků můžeme stavy porovnávat normou vektoru a detekovat tak, jestli dva a více funkcí začne sledovat jeden shluk. Rozšíření *Camshiftu* o *Kalmanuv filtr* nám vylepší sledování více objektů, které svojí trajektorii kříží. Tato vlastnost je zobrazena na obrázku 3.7 kde je porovnání skutečné trajektorie, trajektorie bez *Kalmana* a trajektorie s *Kalmanem*. Z příkladu je vidět, že trajektorie bez *Kalmana* se sloučí při prvním překřížení rukou. Trajektorie s *Kalmanem* je spolehlivější, nicméně při křížení objektů také selhává. Ukázka je puštěna na video sekvenci,



(a) skutečný pohyb (b) sledování s Kalmanem (c) sledování bez Kalmana

Obrázek 3.7: Horní řada obrázků je výběr snímků z video sekvence, z které sledujeme pohyb rukou a obličeje. Obrázky (a), (b) a (c) zobrazují trajektorii rukou a obličeje během video sekvence. Zelené body jsou pro levou ruku, červené pro pravou a modré pro obličej.

kde máme zaručený konstantní čas mezi snímky a predikce stavu je dobře odhadnutá. Záznam používaný z webové kamery nemá garantovaný konstantní čas mezi snímky a trajektorie se tím může zhoršit. Pro spolehlivější trajektorii je nutné zvolit jinou metodu pro sledování jako je například *Condensation algorithm* a nebo *Particle Filter*. Časová náročnost sledování dvou shluků (obličeje a ruky) pomocí *Camshift + Kalmana* trvá méně než 0.5 ms.

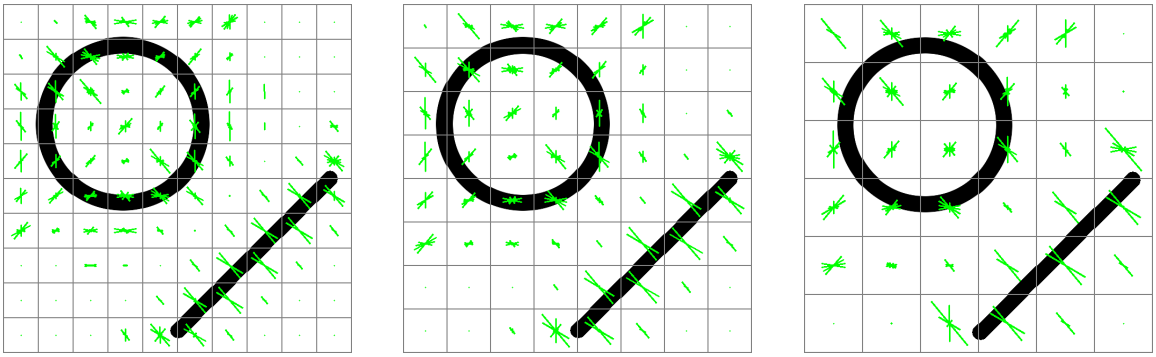
3.3 Klasifikace tvaru

Z kroku sledování shluků máme pro každý snímek několik čtvercových oblastí, které kopírují trajektorii předmětů natrénované barvy. Při sledování trajektorie ruky je sledovaná oblast zaměřena hlavně na dlaň a prsty u ruky mohou být mimo sledovanou oblast. Pro klasifikace gesta jsou prsty na ruce nezbytné a proto je potřeba tuto oblast rozšířit. Velikost čtvercové oblasti je empiricky určena na 1.5 násobku strany čtverce. Dále již pracujeme pouze se šedotónovou složkou barevného modelu z které vybereme výše zmíněné rozšířené oblasti.

3.3.1 Volba příznaků

Ze šedotónového obrázku se příznaky objektu počítají metodou *Histogram of oriented gradient* popsanou v části 2.3.2. Ve článku [Dal] doporučují používat buňky ve velikosti od 4x4 do 12x12 pixelů. Pro detekci postav používají velikost okna 48x96 nebo 64x128 pixelů. V našem případě máme čtvercový obrázek objektu, který chceme klasifikovat a pro klasifikaci jsou navrženy tři typy deskriptorů. Liší se velikostí buňky a velikostí okna a zbylé parametry *HoG* jsou převzaty ze článku

[Dal]. Příklady navržených *HoG* deskriptorů vypočteném na ilustračním obrázku jsou zobrazeny na 3.8. Z ukázky je patrné, jak poměr velikosti okna a buňky ovlivňuje počet příznaků.



(a) Okno 60x60 pixelů a buňka 6x6 pixelů

(b) Okno 64x64 pixelů a buňka 8x8 pixelů

(c) Okno 60x60 pixelů a buňka 10x10 pixelů

Obrázek 3.8: Příklad tří *HoG* deskriptorů vypočtené na černobílém obrázku kolečka a čáry. Šedá mřížka znázorňuje jednotlivé buňky HoGu a zelené čáry jsou histogramy gradientů vyznačující směr a jejich velikost. Všechny příklady jsou počítány pro histogramy v intervalu $(0 - \pi]$ dělený do devíti tříd, a jsou průměrovány na blok o velikosti 2x2 buňky s posuvem bloku 1x1 buňka.

HoG deskriptor počítá příznaky z obrázků o velikosti okna deskriptoru. Pokud chceme spočítat příznaky z obrázků libovolné velikosti, musíme změnit velikost obrázku na velikost okna deskriptoru.

3.3.2 Trénování

Pro klasifikaci *HoG* deskriptoru dle Článku [Dal] se používá *SVM* klasifikátor, o kterém jsme již psali v Podkapitole 2.3.1. Pro trénování klasifikátoru je zapotřebí mít dva seznamy. Pozitivní seznam obrázků, který trénovaný objekt obsahují a negativní seznam obrázků, který obsahují jiný či nespecifikovaný objekt. Pro každý obrázek z obou seznamů se vypočte příznak z *HoG* deskriptoru, a všechny příznaky se předají klasifikátoru s ohodnocením $+1$, v případě kladného, a -1 záporného objektu.

Proces trénování je navržen podle dokumentace [ope13], kde klasifikátor trénují pomocí software *SVMlight*[2]. Obsahuje dva programy, `svm_learn` pro trénování, z kterého získáme natrénovaný model, a `svm_classify` pro klasifikaci natrénovaného modelu. V našem případě používáme klasifikátor v knihovně *OpenCV* a tak posledním krokem je převod modelů do příznakového vektoru pro použití klasifikátoru této knihovny.

Pro trénování a následnou klasifikaci je zvoleno pět tvarů ruky zobrazené na obrázku 3.9. Pro každý tvar bylo pořízeno cca 1500 snímků v několika různých prostředí. Ve Článku [Dal] standardně používají *lineární SVM* a pro zvýšení výkonu klasifikátoru používají *Gaussova jádro* na úkor doby zpracování. Gaussovo jádro se nastavuje hodnotou γ, kde pro detekci lidí určili jako nejvhodnější hodnotu $3e - 2$. Další nastavení klasifikátoru je citlivosti okraje C. Ve výše zmíněném Článku jí používají ve velikosti 0.01. Program *SVMlight* tuto hodnotu nastavuje na $avg(x^2)^{-1}$ vektoru, pokud jí uživatel nenastaví sám.

[2] http://svmlight.joachims.org/

(a) tvar 1 (b) tvar 2 (c) tvar 3 (d) tvar 4 (e) tvar 5

Obrázek 3.9: Zvolená gesta ruky pro ovládání.

V tabulce 3.3 jsou porovnány vlastnosti a výpočetní náročnosti navržených *HoG* deskriptorů. Časová náročnost je vzhledem velikosti příznakového vektoru přibližně lineární. V případě lineárního

Typ deskriptoru	60x60–6x6	64x64–8x8	60x60–10x10
Počet buněk	100	64	36
Velikost příznakového vektoru	2917	1765	901
Doba výpočtu 7800 HoG příznaků	290 s	223 s	180 s
Doba výpočtu modelu z příznaků lineární SVM	244	105 s	41 s
Doba výpočtu modelu z příznaků Gaussova jádra	1480 s	950 s	530 s
Doba převodu modelu	46 s	22 s	14 s

Tabulka 3.3: Porovnání HoG deskriptorů. Typ deskriptoru je ve formátu velikost_okna–velikost_buňky v pixelech.

klasifikátoru, výpočet pěti příznakových vektorů ze 7800 snímků trvá řádově desítky minut. V případě nelineárního Gaussova jádra, se stejnými příznaky a snímky je doba trénování hodina až dvě.

Výsledkem celého trénování je 5 navržených příznakových vektorů pro *SVM* klasifikátory, kde každý z nich dokáže vyhodnotit jestli testovaný obrázek obsahuje či neobsahuje natrénovaný tvar.

3.3.3 Klasifikace

Máme navrženo pět tvarů ruky, tři modifikace *HoG* deskriptorů, dva typy *SVM* klasifikátorů s volitelnou hodnotu C. V tabulce 3.4 je porovnána kvalita lineárního *SVM* se všemi *HoG* deskriptory a v tabulce 3.5 je porovnána kvalita *SVM* s Gaussovým jádrem.

Z Tabulky 3.4 lineárního *SVM* je patrné, že vysokou úspěšnost mají deskriptory 60x60–6x6 a 64x64 –8x8. Nižší úspěšnost deskriptoru 60x60– 10x10 může být způsobena již malou velikostí příznakového vektoru. Různé hodnoty C mají na klasifikaci vliv, ale z hodnot se nedá jednoznačně určit, která by měla být vhodnější. Na zvolení vhodné hodnoty C je potřeba udělat více testů s její různou hodnotou. V tabulce jsou velmi špatné hodnoty tvaru 3, který se spíš klasifikuje jako tvar 4. Jedno z možných vysvětlení je, že při nepřesnosti klasifikace barvy kůže, může u prstů tvaru 3 vznikat příliš malé shluky na to, aby je metoda *Camshift* zaregistrovala. Tím se následně sledovaná oblast posouvá mimo střed sledované ruky, a prsty u tvaru se pak dostávají mimo sledovanou oblast. Pak je tvar 3 zaměnitelný s tvarem 4.

V případě tabulky 3.5 *SVM* s Gaussovým jádrem, mají vysokou úspěšnost všechny deskriptory. Hodnota C vychází lepe pro 0.01. Test byl prováděn pouze s gesty, na které byly klasifikátory

60w, 6b 64w, 8b 60w, 10b					
	28.5 /70.2 49.00/88.1 31.1 /70.9	0 /0 0 /1.3 0 /0	0 / 0 0 / 0 0 / 0	0 / 0 0/ 0 0/ 0	0 / 0 0 / 0 0 / 0
	0 / 0 0 / 0 0 / 0	93.4 / 71.5 86.7 / 74.8 66.9 / 78.8	0 / 0 0 / 0 0 / 0	0.6 / 10.3 0.6 / 16.8 0 / 18.1	0 / 0 0 / 0 0 / 0
	0 / 0 0 / 0 0 / 0	0 / 0 0 / 0 0 / 0	1.3 / 11.2 2 / 5.3 0 / 4.6	3.2 / 18.7 0.6/ 32.9 0 / 16.8	0 / 0 0 / 0 0 / 0
	0 / 0 0 / 0 0 / 0	0 / 0 0 / 0 0 / 0	0 / 0 0 / 0 0 / 0	97.4 / 100 98.1 / 100 77.4 / 100	0 / 0 0 / 0 0 / 0
	0 / 0 0 / 0 0 / 02.6	0 / 0 0 / 0 0 / 0	0 / 0 0 / 0 0 / 0	3.2 / 0 01.3 / 0 0 / 0	93.8 / 68.1 76.4 / 65.3 51.4 / 40.3

Tabulka 3.4: Hodnoty klasifikace *lineárního SVM* klasifikátoru s hodnotou $C = 0.01$ a $C = avg(x^2)^{-1}$. Řádky znázorňují vstupní texturu pro klasifikátor. Sloupce znázorňují gesto, které klasifikátor rozpoznal. Textury, které nerozpoznal ani jeden klasifikátor nejsou v tabulce uvedeny. V každé buňce jsou hodnoty ze všech třech *HoG* deskriptorů v pořadí uvedeném v levé horní buňce. Dvě různě hodnoty pro každý deskriptor vyjadřují různou hodnotu C *SVM* klasifikátoru ve formátu $C = 0.01$ / $C = avg(x^2)^{-1}$. Hodnoty v tabulce jsou uvedeny v %. Test byl proveden cca na 150 texturách pro každé gesto.

60w, 6b 64w, 8b 60w, 10b					
	100 / 100 100 / 100 100 / 100	0 / 0 0 / 0 0 / 0	0 / 0 0 / 0 0 / 0	0 / 0 0 / 0 0 / 0	0 / 0 0 / 0 0 / 0
	0 / 0 0 / 0 0 / 31.8	100 / 98.7 100 / 99.3 100 / 60.3	0 / 0 0 / 0 0 / 0	0 / 1.3 0 / 0.6 0 / 7.7	0 / 0 0 / 0 0 / 0
	6.6 / 19.2 11.9/ 35.1 7.3/ 41.7	0.7 / 0 3.3 / 4 2 / 0	39.5 / 36.8 50.7 / 23.7 49.3 / 11.2	52.3 / 43.2 33.5 / 36.8 0.4 / 46.5	0 / 0 0 / 0 0/ 0
	0 / 0 0 / 0 0 / 0	0 / 0 0 / 0 0 / 0	0 / 0 0 / 0 0 / 0	100 / 100 100 / 100 100 / 100	0 / 0 0 / 0 0 / 0
	1.3 / 21.2 2 / 25.8 2.6/ 72.2	0 / 0 0 / 0 0 / 0	0 / 0 0 / 0 0 / 0	0 / 0 0 / 0 0 / 0	98.6 / 77.8 97.9 / 72.9 97.2 / 24.3

Tabulka 3.5: Hodnoty klasifikace *SVM* s Gaussovým jádrem $\gamma = 3e - 2$ klasifikátoru s hodnotou $C = 0.01$ a $C = avg(x^2)^{-1}$. Řádky znázorňují vstupní texturu pro klasifikátor. Sloupce znázorňují gesto, které klasifikátor rozpoznal. Textury, které nerozpoznal ani jeden klasifikátor nejsou v tabulce uvedeny. V každé buňce jsou hodnoty ze všech třech *HoG* deskriptorů v pořadí uvedeném v levé horní buňce. Dvě různě hodnoty pro každý deskriptor vyjadřují různou hodnotu C *SVM* klasifikátoru ve formátu $C = 0.01$ / $C = avg(x^2)^{-1}$. Hodnoty v tabulce jsou uvedeny v %. Test byl proveden cca na 150 texturách pro každé gesto.

trénované, a klasifikace je velmi přesná. Nicméně klasifikátory s Gaussovým jádrem vyhodnotí téměř libovolné gesto, které nebylo trénované, do jedné z pěti tříd. Tato vlastnost je patrná na tvaru 3, který klasifikátor v podstatě hádá. Takto natrénované klasifikátory by se nemohli použít pro vyhledávání ruky v rámci celého vstupního obrazu, jak je tomu použito při detekci osob na fotce. Tato "nepřesnost"je pravděpodobně způsobena nevhodnou volbou textur do negativního seznamu, který obsahoval převážně tvary ruky ostatních klasifikátorů. Dále výsledek můžeme zpřesnit lepšími volbami C a γ, protože hodnoty byly navrženy pro detekci osob.

Celý krok klasifikace od změny velikosti obrázku, vypočtení *HoG* příznaku až po vyhodnocení 5–ti *SVM* klasifikátorů trvá 4 ms pro všechny tři deskriptory. Stejný čas pro různé deskriptory naznačuje tomu, že klasifikace samotná má o proti změně velikosti obrázku zanedbatelný čas.

Kapitola 4

Softwarová implementace

Softwarová část práce je implementována v jazyce *C++* a byla vyvíjena na překladačích `MinGW` pod platformou Win32 a `GCC` pod platformou Linux. Program používá pro zpracování obrazu knihovny `OpenCV`, dále řadu nástrojů knihovny `boost`. Klasifikátory jsou trénované v programu `SvmLight`.

Tato kapitola je zaměřena na stručný přehled zdrojových kódů a ukázek programu. Veškeré metody a proměnné jsou komentovány v hlavičkových souborech. Kompletní výčet metod a popis přesahuje rámec tohoto textu.

4.1 Třídy a soubory

Zdrojové kódy práce se dají rozdělit na pomocné a základní třídy. V základních jsou implementovány hlavní funkce programu a pomocných třídy tvoří mód programu, kde jen sestavují objekty ze základních tříd Základní třídy programu jsou:

enums.h číselniky pro typ zdroje, barevné modely, metody barev a mód programu.

Arguments je pro čtení argumentů z příkazové řádky. Načítá výchozí hodnoty ze souboru configfile.dat. Kompletní výčet voleb získáte parametrem `-h`, který je vožen do Přílohy A.

VideoSource zpracovává vstup z video souboru, seznamu obrázků nebo webové kamery. Zdroj se nastavuje parametry `-c -i -v`, u obrázku lze použít regulární výraz pro skupinu obrázků.

SaveData ukládá obrázky nebo textová data do souboru.

Gui řídí vizualizace a nastavuje události myši.

ColorClassifier zpracovává barevné modely a klasifikaci barvy `--color-model` a `--color-method`

MotionObject zpracovává sledování shluku. Každý nalezený shluk je reprezentován objektem této třídy.

GestDescriptor výpočet *HoGů* a klasifikace. Každý používaný klasifikátor je reprezentován objektem této třídy.

GestRecognition zpracovává data mezi ColorClassifier a několika objekty MotionObject a GestDescriptor. Společně tak poisují pohyb a tvar všech nalezených objektů.

Některé pomocné třídy si popíšeme v Podkapitole 4.3.

CollectData funguje jako sběr nových gest a tvarů ruky.

LearnGest je určena pro získání z obrázků soubor příznaků používané v *SvmLight*. Tato třída také převádí model z *SvmLight* do příznakového vektoru.

GestTest je konečná podoba rozpoznávání gest s vizualizací.

HeadLess funguje stejně jako *GestTest* akorát nepouští žádné grafické prvky.

4.2 Sestaveni programu

Program byl vyvíjen v `GCC` ve verzi 4.8, `MinGW` ve verzi 4.7.2, `OpenCV` ve verzi 2.4.6 a `boost` ve verzi 1.54. Pro sestavení programu potřebujete následující soubory a adresáře.

makefile skript pro sestaveni programu.

objectsUnix.mk nastavení knihoven a adresářů pro platformu Linux.

objectsWin32.mk nastavení knihoven a adresářů pro platformu Windows.

src zdrojové kódy programu

subdir.mk pomocný skript k *makefile*

K sestavení programu se použije příkaz `make` a příkaz `make clean` smaže sestavený program a všechny vytvořené závislosti.

4.3 Ukázky programu

Po správném spuštění začne program fungovat autonomně. Po dalších desíti vteřinách program spustí sledování a rozpoznávání gest.

Následující část textu popisuje jednotlivé módy programu, který lze vybrat parametrem `--mode`. Většina módů funguje autonomně, což v našem případě znamená po prvních 10–ti vteřinách se vyhledá obličej na scéně a natrénuje barvy. Tato prodleva je z důvodu vyvážení bíle a kontrastu u kamery, která se při startu provádí. a po dalších 10–ti vteřinách spustí sledování shluků. Obě funkce jdou opětovně vyvolat levým (v případě kalibrace barvy) a prostředním tlačítkem u myši. Pravé tlačítko bývá pro ukládání dat dle modu.

Program potřebuje následující soubory nutné pro běh programu:

configfile.cfg konfigurační soubor výchozích nastavení programu. Většina parametrů lze přenastavit volbou v příkazové řádce.

hog*.yaml nastavení *HoG* deskriptoru.

classGest adresář s příznaky jednotlivých gest. Přidávají se parametrem `-d`. POZOR! Příznak musí odpovídat vlastnostem HoG.

haarcascades adresář či soubor pro detekci obličeje

4.3.1 Sběr textur

Určená k vytvoření seznamu gest. Spouští se parametrem `-m cd` nebo `--mode collectData`. Příklad spuštění:

```
rozpoznavani_gest -c 1 -F -m cd --color-model lab --color-method win --flip
```

Ovládání: Po stisknutí klávesy *0* až *9* ukládá textury do souboru s číslem třídy zvolené klávesy. Klávesa z abecedy ukládání zastaví. *Esc* program ukončí.

4.3.2 Trénování SVM

Proces trénování je navržen podle dokumentace [ope13], kde deskriptor trénují pomocí *SVMlight*. Spouští se parametrem `-m lg` nebo `--mode learnGest`. Funguje dvou krokově. První krok: vytvoří z textur soubor příznaků formátovaného pro příkaz `svm_learn`. Příklad spuštění:

```
rozpoznavani_gest -m lg -P train/template0/.*png -P train/template1/.*png
-N train/template_invalid/.*png --HOG hog_10b_60w.yaml
```

Na vzniklé soubory se pustí trénování pomocí `svm_learn`
Příklad použití `svm_learn` pro trénování *lineárního SVM*

```
svm_learn -c 0.01 -z r -t 0 features_hog.dat model_hog.dat
```

Příklad použití `svm_learn` pro trénování *SVM* s Gaussovým jádrem

```
svm_learn -c 0.01 -z r -t 2 -gamma 0.03 features_hog.dat model_hog.dat
```

Druhý krok: převedení modelu do příznakového vektoru

```
rozpoznavani_gest -m lg -M train/linearAuto/model_features_hog_10b_60w_class_0.dat
-M train/linearAuto/model_features_hog_10b_60w_class_1.dat --HOG hog_10b_60w.yaml
```

4.3.3 Testování SVM

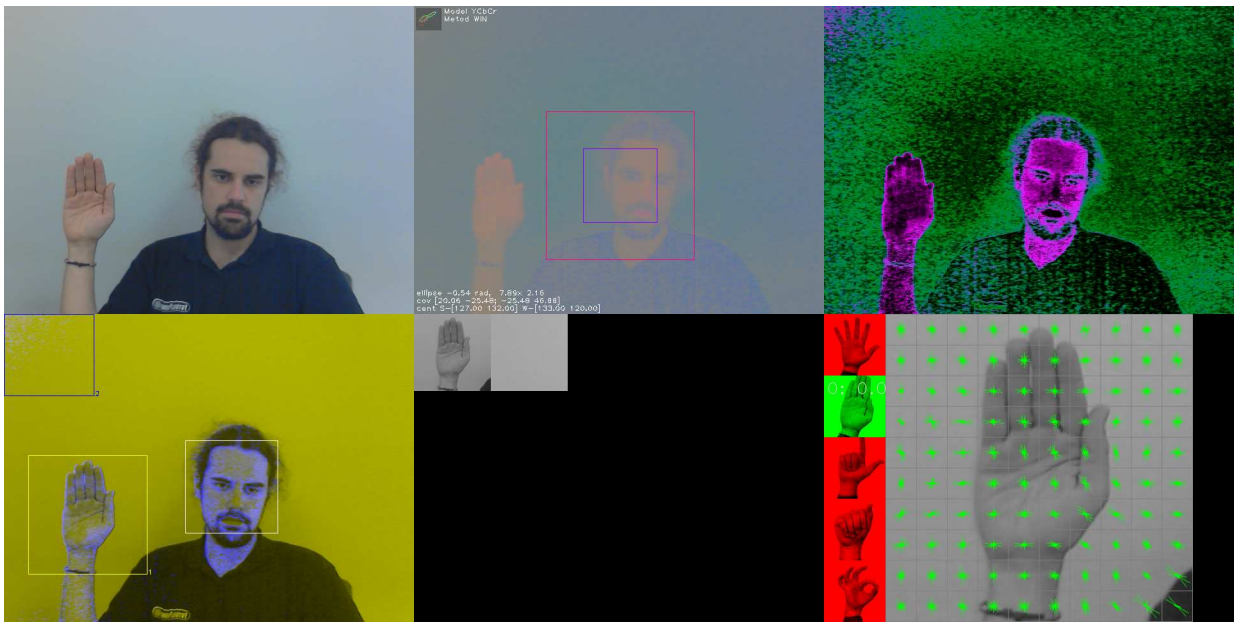
Porovnává různé typy SVM klasifikátorů. Spouští se parametrem `-m st` nebo `--mode svmTest`. Příklad spuštění:

```
rozpoznavani_gest -m st -c 1 --color-model hsv --color-method win --flip
```

Ovládání: Po stisknutí klávesy *1* až *5* porovnává gesto se zadanou číslovkou. Klávesa z abecedy ukládání zastaví. *Esc* program ukončí a vypíše se procentuální úspěšnost jednotlivých gest.

4.3.4 Rozpoznávání gest

Příklad který demonstruje finální podobu programu: Rozpoznává gesta rukou od jednoho uživatele. Vizuální ukázka programu je zobrazena na obrázku 4.1. Nalezená gesta se vypisují do konzole a vizuálně se zobrazují na šestém políčku grafické vizualizace. Spouští se parametrem `-m gt` nebo `--mode gestTest`. Příklad spuštění:



Obrázek 4.1: Screenshot programu. Na šesti částech se zobrazují jednotlivé kroky při zpracování. První je snímaný obraz. Navazují chromatické kanály, kde jsou vyznačený oblasti trénování barvy a histogramy barev. Třetí je pravděpodobnostní obraz, kde zelená je pravděpodobnost pozadí, modrá pravděpodobnost trénované barvy a červená je barva pigmentu. Čtvrtý je sledování objektů. Pátý výběr textur pro klasifikaci a šesté je rozpoznané gesto.

```
rozpoznavani_gest -m gt -c --flip --HOG hog_6b_60w.yaml
   -d classGest/svm_detector_hog_6b_60w_class_0.dat
   -d classGest/svm_detector_hog_6b_60w_class_1.dat
   -d classGest/svm_detector_hog_6b_60w_class_2.dat
   -d classGest/svm_detector_hog_6b_60w_class_3.dat
   -d classGest/svm_detector_hog_6b_60w_class_4.dat
```

Klávesa *Esc* program ukončí.

Tento příklad lze spustit i v HeadLess modu (`-m hl` nebo `--mode headLess`), který běží bez grafických komponent. funguje s rychlostí 16 snímků za vteřinu s rozlišením 640x480 snímaného obrazu. HeadLess mod se ukončuje rozpoznání tvaru 5.

Kapitola 5

Závěr

Práce vychází z vědeckých článků, které se podobnou problematikou zabývaly. Řešení úlohy se skládá z filtrace pigmentu, trajektorie objektů a klasifikace tvaru ze sledovaných oblastí.

Pro klasifikaci barvy kůže byly v práci vyzkoušeny dvě metody klasifikací. V obou metodách se používá jako referenční oblast obličej. Dále se pracuje jen s barevnými modely s jasovou složkou, kde se z chromatických komponent počítají histogramy obličeje a jeho okolí. V jednom přístupu se dvourozměrný histogram chromatických složek aproximuje na dva jednorozměrné histogramy v podprostoru hlavní a vedlejší poloosy elipsy. Chromatické kanály barevného modelu transformuje do kanálů α a β, které jsou nezávislé a jejich vynásobením získá pravděpodobnost barvy kůže. V druhém přístupu používá dvourozměrné histogramy obličeje a jeho okolí, a odečtením jejich pravděpodobnostních modelů získá lepší výsledky než v předchozí metodě. V rámci práce bylo experimentováno s více barevnými modely konkrétně *HSV*, *HLS*, *Lab*, *YCbCr*, *Luv YUV*. Barevné modely *HSV*, *HLS* fungují za ideálních podmínek, kdy v pozadí je bílá nebo klíčovací barva. Barvu pigmentu nejlépe reprezentuje model *Lab*, ale za specifických podmínek má mnohem horší výsledky než *YCbCr*, který má robustnější charakter.

Trajektorie objektu se sleduje z klasifikované barvy kůže, kde se v podobě shluků sledují pomocí metody *Camshift* rozšířenou o *Kalmanuv filtr*. Trajektorie s Kalmanem je spolehlivější, nicméně při křížení objektů také selhává. Při sledování trajektorie ruky je sledovaná oblast rozšířena o 1.5 násobku její velikosti z důvodu nekompletního obrazu ruky.

V klasifikaci tvaru počítáme příznaky pomocí metody *Histogram of oriented gradient*. Byly navrženy tři její variace v různém poměru velikosti okna a buňky. Pro klasifikaci gesta bylo navrženo pět tvarů ruky a pro každé gesto bylo pořízeno cca 1500 vzorků pro trénování. Jako klasifikátory byly vyzkoušen lineární *SVM* a *SVM* s Gaussovým jádrem. Test klasifikátorů byl proveden na cca na 150 texturách pro každé gesto, kde pro lineární *SVM* jsou vhodné *HoG* deskriptory s okénkem 60x60 a buňkou 6x6 pixelů, a také s okénkem 64x64 a buňkou 8x8 pixelů. U *SVM* s Gaussovým jádrem mají vysokou úspěšnost všechny deskriptory, nicméně vyhodnotí téměř libovolné jiné gesto, které nebylo trénované, do jedné z pěti tříd. Tento nedostatek by se měl zlepšit volbou rozmanitějšího negativního seznamu při trénování klasifikátoru.

Program pro rozpoznávání gest byl implementován v jazyce *C++* a lze ho sestavit pod platformami Linux i Windows. Výsledný program funguje s rychlostí 16 snímků za vteřinu s rozlišením 640x480 snímaného obrazu.

Literatura

[BHS93] R. Boyle, V. Hlavac, and M. Sonka. Image processing, analysis, and machine vision. In *Chapman and Hall*, 1993.

[Bra] G. R. Bradski. Real time face and object tracking as a component of a perceptual user interface. In *Applications of Computer Vision, 1998. WACV '98. Proceedings., Fourth IEEE Workshop on*, pages 214–219, Princeton, NJ.

[Bue] D. Bueno, J. I. Kragic. Integration of tracking and adaptive gaussian mixture models for posture recognition. In *Robot and Human Interactive Communication, 2006. ROMAN 2006. The 15th IEEE International Symposium on*, pages 623–628, Hatfield.

[Dal] B. Dalal, N. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893, San Diego, CA, USA.

[HDC] E. Huan Du Charbon. 3d hand model fitting for virtual keyboard system. In *Applications of Computer Vision, 2007. WACV '07. IEEE Workshop on*, pages 31–31, Austin, TX.

[IB98] Michael Isard and Andrew Blake. Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29:5–28, 1998.

[ope13] OpenCVWiki: trainHOG. `http://opencv.willowgarage.com/wiki/trainHOG`, 2013. [Online; 30. srpna 2013].

[Soo] S. Chalidabhongse T. H. Soontranon, N. Aramvith. Improved face and hand tracking for sign language recognition. In *Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on*, volume 2, pages 141–146.

[Ste06] Björn Stenger. Template-based hand pose recognition using multiple cues. In *In Asian Conference on Computer Vision*, pages 551–560, 2006.

[STTC06] Björn Stenger, Arasanathan Thayananthan, Philip H. S. Torr, and Roberto Cipolla. Model-based hand tracking using a hierarchical bayesian filter. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28:1372–1384, 2006.

[WZD07] Xiying Wang, Xiwen Zhang, and Guozhong Dai. Tracking of deformable human hand in real time as continuous input for gesture-based interaction. In *IUI '07: Proceedings of the 12th international conference on Intelligent user interfaces*, pages 235–242, New York, NY, USA, 2007. ACM.

Příloha A

Volby programu

```
Zakladni volby:
  -h [ --help ]                         produce help message
  -c [ --camera ] [=arg(=0)]            ID kamery
  -v [ --video ] arg                    video soubor
  -i [ --image ] arg                    obrazek soubor. Pro skupinu obrazku lze
                                        pouzit regularni vyraz
  -f [ --fps ] arg (=15)                pocet snimku za vterinu videovstupu
  -D [ --delay ] arg (=100)             Zpozdeni v ms pro prepinani vlaken.
                                        Hodnota 0 ceka na stisk klavesnice.
                                        (cv2.waitKey([delay]))
  -p [ --prefix ] arg                   prefix pro ukladani souboru
  --color-model arg (=win)              barevny model.
                                        Moznosti:hsl, hsv, lab, luv, ycbcr, yuv
  --color-method arg (=lab)             metoda barevneho filtru.
                                        Moznosti: win - trenuje se z obliceje a
                                                  okoli 2D histogram, sub -
                                                  trenuje se jen z obliceje 1D
                                                  histogramy
  -F [ --flip ] [=arg(=VERTICAL)] (=NONE)
                                        zrcadlovy pretoceni vstupu
                                        Moznosti:VERTICAL, HORIZONTAL, BOTH,
                                                 NONE
  -m [ --mode ] arg                     mod programu
                                        Moznosti: colorTest (ct), motionTest
                                                 (mt), collectData (cd),
                                                 hogTest (ht), learnGest (lg),
                                                 svmTest (st), gestTest (gt),
                                                 headLess (hl)
  -H [ --height ] arg (=0)              Vyska zaznamu
  -W [ --width ] arg (=0)               Sirka zaznamu
  -b [ --median-blur ] [=arg(=1)]       Velikost jadra medianoveho filtru.
                                        Odstranuje sum ze zaznamu
```

```
-s [ --close-size ] [=arg(=5)]        Velikost jadra morfologicke operace
                                      uzavreni.
-S [ --scale ] arg (=1.5)             Sledovany objekt muze mit jinou
                                      velikost nez je urcen z barevneho
                                      modelu. Tato hodnota velikost upravuje.
--HOG arg (=hog.yaml)                 Konfiguracni soubor HOG descriptoru. Ve
                                      formatu YAML
--HAAR arg (=haarcascades/haarcascade_frontalface_alt2.xml)
                                      haar cascade. Pro detekci obliceje
-P [ --positive-classes ] arg         Seznam obrazku pro jednotlive tridy.
                                      Argument lze zadat vicekrat a kazdy
                                      reprezentuje jednu tridu!. Zadava se
                                      pomoci regularnich vyrazu.
-N [ --negative-class ] arg           Seznam obrazku ktere nepatri do zadne
                                      tridy. Zadava se pomoci regularnich
                                      vyrazu.
-M [ --svm-model ] arg                Seznam modelu z programu svm_learn pro
                                      jednotlive tridy.
                                      Argument lze zadat vicekrat a kazdy
                                      reprezentuje jednu tridu!.
-d [ --svm-detectors ] arg            Seznam detektoru pro SVM ve tride
                                      HOGDescriptor.
                                      Argument lze zadat vicekrat a kazdy
                                      reprezentuje jednu tridu!.
```

Evidenční list

Souhlasím s tím, aby moje diplomová/bakalářská práce byla půjčována k prezenčnímu studiu v Univerzitní knihovně ZČU v Plzni.

Datum: Podpis:

Uživatel stvrzuje svým čitelným podpisem, že tuto diplomovou/bakalářskou práci použil ke studijním účelům a prohlašuje, že ji uvede mezi použitými prameny.

Jméno	Fakulta/katedra	Datum	Podpis