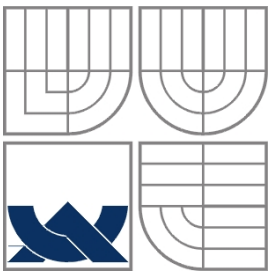
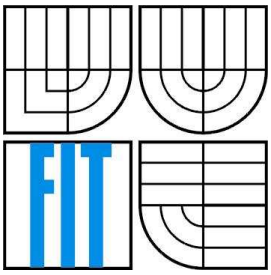


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

OVLÁDÁNÍ POČÍTAČE POMOCÍ GEST

HUMAN-MACHINE INTERFACE BASED ON GESTURES

DIPLOMOVÁ PRÁCE
DIPLOMA THESIS

AUTOR PRÁCE
AUTHOR

Bc. Jaroslav Charvát

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Radek Bartoň

BRNO 2011

Abstrakt

Diplomová práce „Ovládání počítače pomocí gest“ zpracovává teoretické poznatky z oblasti problematiky počítačového vidění, rozpoznávání gest a podrobněji popisuje jednotlivé metody, použité k vývoji programu. Praktickou část této práce tvoří popis funkčnosti vyvinuté aplikace, pomocí které je možné ovládat počítač gesty pravé i levé ruky a hlavy. Program funguje na základě prvotní detekce kůže, následně pracuje na vyhledávání gest dlaní a hlavy. Pro tyto činnosti byly mimo jiné využity metody AdaBoost a PCA.

Abstract

Master’s thesis „Human-machine interface based on gestures“ depicts the theoretical background of the computer vision and gesture recognition. It describes more in detail different methods that were used to create the application. Practical part of this thesis consists of the description of the developed program and its functionality. Using this application, user should be able to control computer by gestures of both right and left hands and also his head. The program is primarily based on the skin detection that is followed by the recognition of palms and head gestures. There were used two essential methods for these actions, AdaBoost and PCA.

Klíčová slova

Počítačové vidění, detekce obličeje, rozpoznávání gest, AdaBoost, PCA.

Keywords

Computer vision, face detection, gesture recognition, AdaBoost, PCA.

Citace

Charvát Jaroslav: Ovládání počítače pomocí gest, diplomová práce, Brno, FIT VUT v Brně, 2011.

Ovládání počítače pomocí gest

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Radka Bartoně.
Další informace mi poskytl konzultant Ing. Miroslav Švub.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Bc. Jaroslav Charvát
23.5.2011

Poděkování

Tímto bych rád poděkoval vedoucímu této práce, panu Ing. Radku Bartoňovi a konzultantovi, panu Ing. Miroslavu Švubovi, za cenné rady a připomínky, jimiž přispěli k vypracování této diplomové práce.
Na tomto místě chci také poděkovat celé své rodině za podporu, kterou mi poskytovali po dobu celého studia.

Obsah

1 Úvod

Počítačové vidění (anglicky computer vision) je jedním z poměrně progresivně se vyvíjejících vědních odvětví. Spolu s robotikou a dalšími podobnými obory se snaží v mnohém pomoci lidem a umožňuje automatizaci procesů, které by byly pro člověka zdlouhavé nebo náročné. Síla počítačů a výpočetní techniky obecně spočívá v její schopnosti rychle a bez vyčerpání provádět složité a časově náročné úkony, výpočty a podobně.

Donedávna byla velkou slabinou počítačů jejich neschopnost vidět a vnímat svět tak, jako lidé. Věci tak zřejmé, jako slučování obrazců podle tvarů nebo velikosti, natož pak jejich rozpoznání, byly dříve nemyslitelné. Nyní již existují rozmanité funkce, kterých lze pro správné nebo alespoň přibližné vnímání počítačů použít.

Jelikož mě uvedená problematika zajímá, vybral jsem si pro svou diplomovou práci téma „Ovládání počítače pomocí gest“. Při zpracovávání tohoto tématu se chci zabývat situacemi, kdy bude počítač ovládán gesty snímanými prostřednictvím běžné webové kamery. Úkolem, který jsem si pro další práci stanovil, je prostřednictvím vývoje nové aplikace zajistit, aby počítač na základě videosekvence získané z webové kamery dokázal porozumět požadavkům, které jsou na něj uživatelem kladeny. Praktické využití zamýšlené aplikace je možné zejména pro lidi handicapované, pro které je klasický způsob ovládání počítače komplikovaný. Místem vhodným pro uplatnění této aplikace jsou i továrny a podobné provozy, v nichž není možné, například z důvodu prašnosti, využít standardních vstupů.

V návaznosti na předchozí úvahu o budoucí podobě plánované aplikace jsem formuloval cíl této diplomové práce. **Cílem** mé diplomové práce je navrhnout a implementovat systém, který bude rozpoznávat gesta hlavy a obou dlaní a následně vykoná akce, jež budou s daným gestem asociovány.

Text práce je tematicky rozdělen do několika kapitol. V kapitole nazvané „Teoretická část“ shrnu a poukáži na nejdůležitější metody a principy z oblasti počítačového vidění, které podstatnějším způsobem zasáhnou do realizace mé práce. V části „Návrh řešení“ pak navážu na představené metody a nastíním předpoklad toho, jak by měla aplikace fungovat. Do „Realizace“ zahrnu všechny své poznatky z implementace systému a obeznámím čtenáře s funkčností a ovládáním aplikace. Kapitola „Zhodnocení výsledků“ bude obsahovat výkonová měření a měření procentuálních úspěchů detekce gest. Jednotlivá měření budou doplněna vysvětlujícími komentáři. V poslední kapitole, kterou jsem nazval „Závěr“, shrnu výsledky práce.

2 Teoretická část

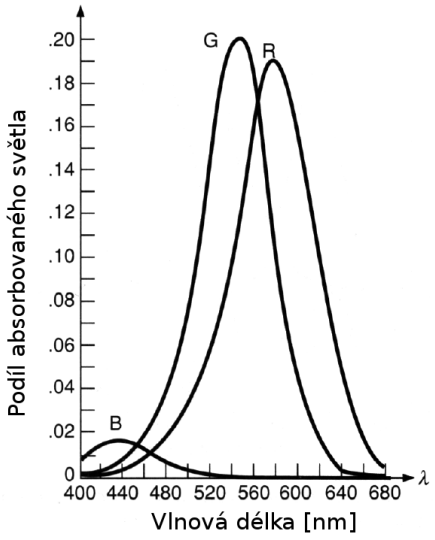
Následující kapitola bude obsahovat základní informace a teoretické podklady, jež poslouží k navazující tvorbě návrhu a budou také sloužit při realizaci záměru této diplomové práce. Cílem kapitoly je vytvořit obecný rámec dané problematiky a podrobněji popsat ty oblasti teorie, které mají větší význam pro zpracování tématu této diplomové práce.

Kapitola bude dále rozčleněna do podkapitol, z nichž první bude sumarizovat obecné poznatky týkající se vnímání světa a vidění, druhá se bude podrobněji zabývat teoretickými poznatky týkajícími se detekce kůže, třetí bude mapovat zkoumání z oblasti rozpoznávání obličeje a ve čtvrté podkapitole sloučím přístupy k hledání dlaní v obraze a v této části uvedu souhrn poznatků z teorie PCA (Analýza hlavních komponent).

2.1 Úvod do problematiky

V této kapitole rozeberu obecné postupy, vztahy a principy, běžně se objevující v oboru počítačového vidění a zpracování obrazu. Nejprve vysvětlím, jak je svět člověkem vnímán, dále vysvětlím co je světlo, jakou část spektra budu zachycovat a co k tomu použiji. Následně provedu rozbor několika formátů pro uložení barev a bude se zabývat otázkou, zda a případně jakých způsobem je možné provést převod z jednoho na druhý.

K vnímání okolního světa člověk využívá různé smyslové orgány. Pro záměry této diplomové práce je nejdůležitějším z nich lidské oko, které umožňuje vidění. Lidské oko je velmi složitý orgán. Zde se pokusím nastínit jen základní vlastnosti, které budou v rámci mé práce zohledněny. Světlo proniká do oka skrze čočku, která, podobně jako lupa, slouží k přiblížení resp. oddálení obrazu a jeho zaostření resp. rozostření. Dále pak pokračuje na sítnici, kde dopadá na světlocitlivé buňky. Tyto buňky se nazývají čípky a tyčinky. Tyčinky jsou schopny reagovat na intenzitu dopadajícího světla, nejsou však schopny rozlišit jejich barvu (určit vlnovou délku, více viz kapitola „Světlo, světelné spektrum“). Jsou přibližně desektrát citlivější. Oproti tomu čípky rozlišit barvu světla dokážou. V oku jsou obsaženy tři druhy čípků, z nichž každý typ reaguje na světelné paprsky z různých částí spektra jinak. Poté co tyčinky a čípky přijmou různá kvanta energie, vyšlou do mozku informaci o tom, kde a s jakou intenzitou se ta či ona barva nachází. Reakce buněk (čípků) není pro každou barvu shodná. Na následujícím grafu je možné vidět, jak se vnímání liší.

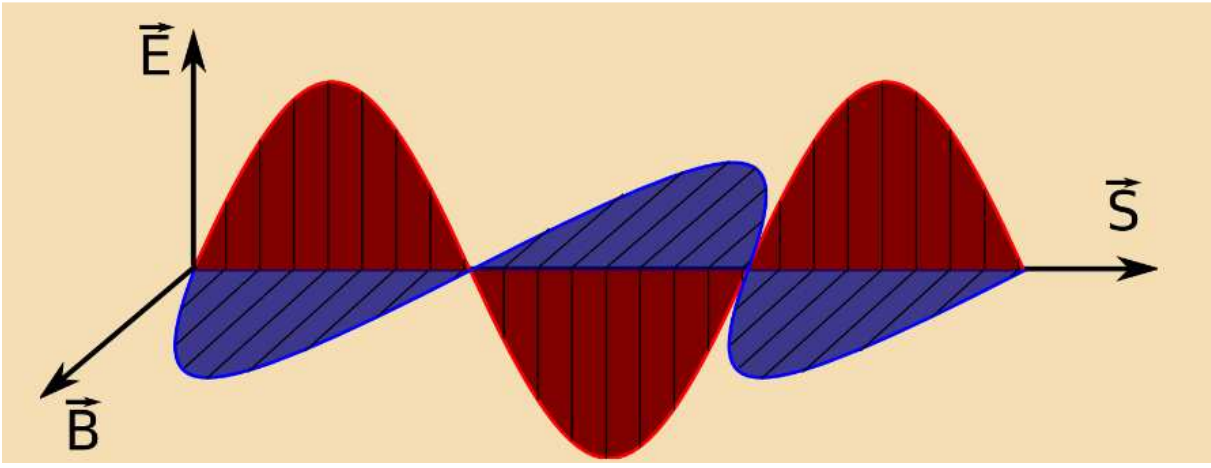


Obrázek 1: Absorbované světlo různých vlnových délek [3]

Toto rozložení budu muset zohlednit později i ve svém návrhu a implementaci, neboť například při převodu z barevného schématu na úrovni šedi je třeba využít nelineárního podílu jednotlivých barevných složek na výsledném jasu (úrovni šedi).

2.1.1 Světlo, světelné spektrum

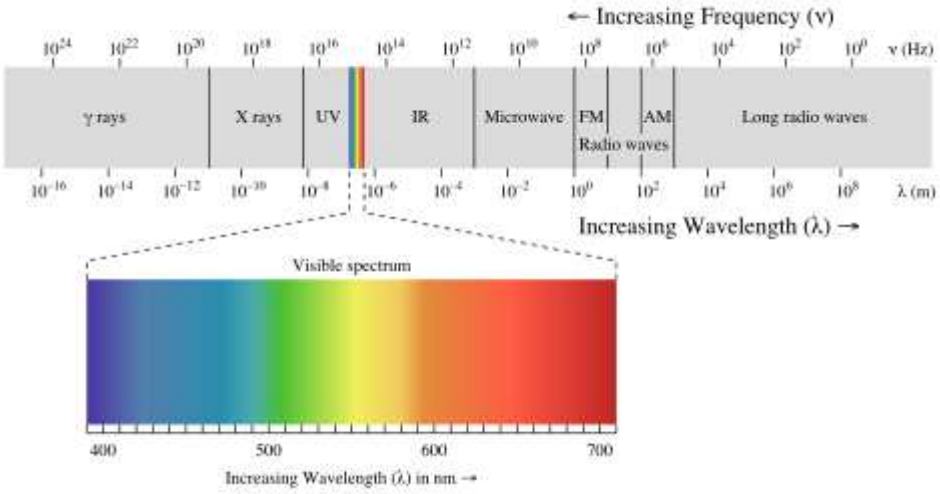
Světlo je elektromagnetické vlnění. Vlna osciluje vždy kolmo ke směru šíření světla, a pokud není polarizované, pak může působit v nekonečné množině směrů, které jsou kolmé ke směru šíření. Elektrická a magnetická složka jsou navzájem kolmé. Bližší vysvětlení je nad rámec této práce a je možné jej hledat v literatuře zabývající se vlněním. Průběh postupu světelné vlny vysvětlí následující obrázek.



Obrázek 2: Průběh elektrické a magnetické části světla [5]

Paprsků elektromagnetického vlnění existuje nepřeberné množství. Od těch s větší frekvencí kmitání jako je Gama nebo Rentgenové záření, přes ultrafialovou, barvy ve viditelném spektru až po světlo

s podstatně větší vlnovou délkou, např. infračervená, rádiové vlny, mikrovlny a dlouhé rádiové vlny. Následující obrázek znázorňuje vlnovou délku, frekvenci i typy vln.



Obrázek 3: Spektrum elektromagnetického vlnění [6]

Světelné spektrum je rozmezí vlnových délek, jichž může světlo nabývat. Jsou různé metody, jak zachytit intenzitu dopadajících paprsků v různých částech spektra. Pro účely této práce však plně postačí viditelná část spektra, což bude popsáno v následující podkapitole.

2.1.2 Zachycení obrazu

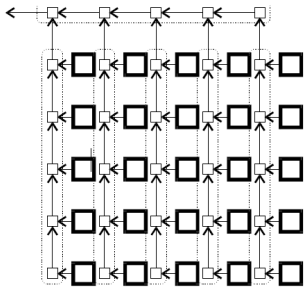
Vyfotografování scény je dnes podstatně jednodušší, než tomu bylo dříve. Ať už se jedná o zachycení zážitků z dovolené nebo snímání mikročipu ve výrobě, základní princip je ve většině případů stejný. Pořízením snímku zachycujeme projekci trojrozměrného světa na dvourozměrný, světlo-citlivý materiál. Pokud tedy nepořizujeme hologram nebo nesnímáme více kamerami (alespoň dvěma) z různých úhlů, vždy zaznamenáme pouze část informace. Nenávratně se vytratí informace o třetím rozměru a zbudou pouze perspektivně zaznamenané objekty ve scéně.

Každý fotoaparát nebo kamera zachycuje informaci v podobě světla, které dopadá na světlo-citlivý materiál. Tím může být např. fotografický film (tenký plast potažený vrstvou halogenidů stříbra vázaných do želatiny) do obyčejných fotoaparátů. Dále to mohou být digitální přístroje, kde se k zachycení dopadajícího světla používají především technologie CMOS a CCD. Pro automatické zpracování obrazu bude použito výhradně digitálních technologií, proto se v následujících řádcích zaměřím na zmíněné CCD a CMOS.

CCD

CCD neboli Charged Couple Device jsou polovodičové senzory, jejichž název je odvozen ze způsobu transportu obrazové informace. Princip spočívá v následujícím: „*Nejprve je pořízen obraz v elektrické podobě. Na čipu jsou vytvořeny struktury podobné kapacitorům (kondenzátorům), které se před expozicí nabijí na konstantní náboj (je na nich konstantní napětí). V průběhu expozice se na kapacitory přestane přivádět napětí, vlivem obrazu (světla) promítaného na plochu čipu se kapacitory vybíjejí díky svodu struktury, který je úměrný světelné energii absorbované příslušnou částí čipu (pixelem).*“ [2, str. 22]

Takto získaná informace se následně přesune přes transportní registry k dalšímu zpracování (je možné pozorovat na následujícím obrázku). V souvislosti s přesouváním náboje a zásahem do náboje sousedního pixelu mohou vznikat parazitní jevy. Je tedy těžké získat v některých případech ostré hrany nezvětšené, respektive nezmenšené, např. o jeden pixel. Na druhou stranu je však velmi přínosné, že CCD čipy mohou snímat scénu najednou (na rozdíl od CMOS) a nedochází tedy k zabrání dvou mírně odlišných scén vlivem pohybu (kamery nebo scény).

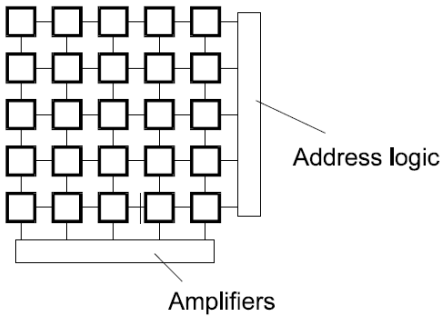


Obrázek 4: Ideové schéma transportních registrů na CCD [2]

Technologie, i přes své četné nevýhody, je často využívána a patří prozatím mezi nejlepší běžně dostupné.

CMOS

CMOS je zkratka pro anglický název Complementary Metal Oxid Semiconductor. Stejně jako CCD obsahuje i CMOS technologie světlo-citlivé buňky. Hodnota náboje je však na napětí převáděna přímo v nich a teprve poté je výsledek přesouván pomocí kaskády analogových multiplexorů na okraj čipu. Tento proces není možné provádět současně pro všechny pixely, proto je nutno předávat informace a tedy i snímat po částech. Právě kvůli tomuto dochází k velmi nepříznivému efektu a to sice, že při pohyblivé scéně nebo při pohybu s kamerou během expozice, může být pro část snímaného obrazu scéna mírně změněná. Na obrázku níže je vyobrazena struktura CMOS čipu.



Obrázek 5: Ideové zobrazení CMOS čipu [2]

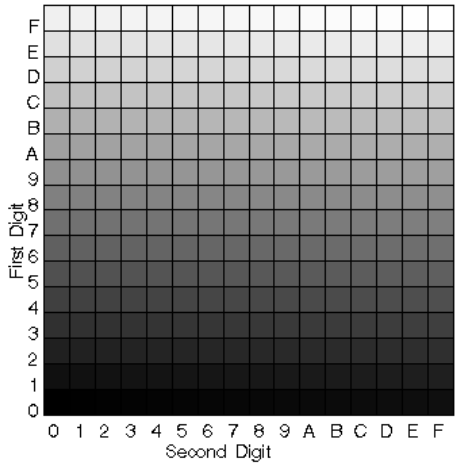
Při porovnání metody CCD a metody CMOS je možné uvést jako výhody CMOS technologie zejména podstatně nižší spotřebu energie, možnost využití pouze jednoho napájecího napětí a další výhodou této technologie je také možnost integrace na jeden čip.

2.1.3 Uložení obrazu

Obraz může být v počítači uložen několika způsoby. V jádru se však tyto způsoby v mnohém shodují a jejich odlišnost není na první pohled tolik patrná. Lze konstatovat, že se vždy jedná o určité body zařazené do mřížky, přičemž každému bodu je přiřazena jistá digitální informace. Popis možných hodnot, jichž mohou jednotlivé body nabývat, uvedu v podkapitole „Barevná schémata“. V části nazvané „Vzorkování a uložení do mřížky“ pak představím možnosti řazení bodů (pixelů) a uvedu, který a proč budu používat. V závěru prezentuji formáty pro ukládání obrázků.

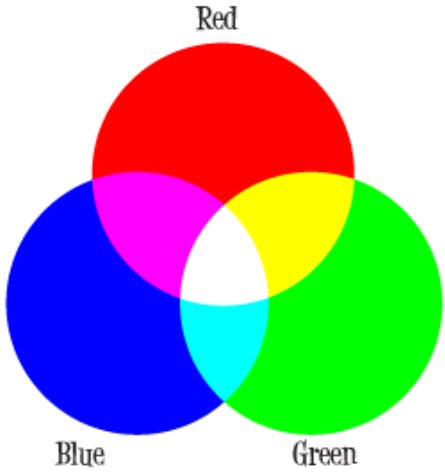
Barevná schémata

Nejjednodušší barevné schéma je tzv. **Grayscale**, neboli schéma, kde každý pixel sestává z hodnot popisujících odstín šedi. Nejčastěji je to reprezentováno jedním bajtem v paměti, tudíž je možné uložit 256 odlišných odstínů. Jelikož je *člověk schopen rozlišit pouze 100 úrovní jasu v jeden moment* [25], tato hodnota je plně dostačující.



Obrázek 6: 256 úrovní šedi [10]

Mezi nejzákladnější a pravděpodobně nejčastěji používaný barevný model patří **RGB** (Red, Green, Blue). Tento aditivní model zachycuje tři uvedené barvy a jejich mícháním, respektive skládáním jejich různých odstínů, odvozuje barvy ostatní, a tím je možné docílit vytvoření rozmanité palety barev. Při zastoupení všech barev najednou vzniká bílá barva, jak je vidět na následujícím obrázku.

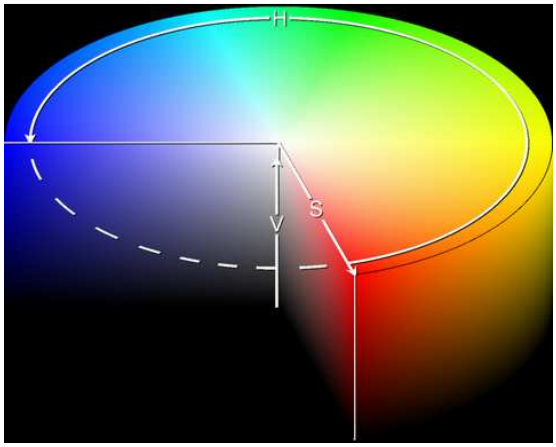


Obrázek 7: Schéma barevného modelu RGB [9]

Jelikož lidské oko vnímá barvy různě, není možné pro získání jasové hodnoty odstínu šedé (neboli Grayscalu) použít průměr všech RGB složek. Ve snaze o převod z RGB schématu do Grayscalu je proto zapotřebí zohlednit poměr intenzit jasu vnímaných lidským okem. Pro zmiňovaný převod byl vytvořen vzorec, který by měl zajistit správné vyvážení podílů jednotlivých složek na výsledné hodnotě jasu. Vzorec, který následuje za tímto textem, popisuje způsob, jakým jas pro daný pixel vypočítat. Y popisuje jas, R, G a B jednotlivé barevné složky. Číselné hodnoty byly odvozeny na základě empirického testování.

$$Y = 0{,}299 * R + 0{,}587 * G + 0{,}114 * B \quad [1]$$

Posledním, neméně důležitým barevným schématem použitým v mé diplomové práci bude **HSV**. Tento název je zkratkou vyjadřující způsob uchování informace, neboli Hue (odstín), Saturation (sytost) a Value (jas – anglicky někdy také brightness). HSV barevné schéma je názorně představeno na následujícím obrázku.

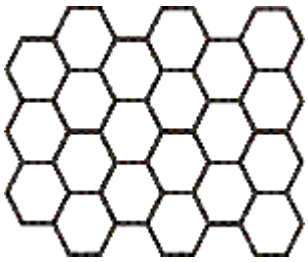


Obrázek 8: Barevné schéma HSV[11]

Tento model bude pro mou práci důležitý tím, že je možno oddělit informaci o barvě a informaci o jasu. Uvažme například, že rozpoznáváme barvu kůže v místnosti, kde je určité osvětlení. Pokud poté budeme rozpoznávat barvu kůže stejného člověka jen v místnosti s jinou intenzitou světla, je potřeba, aby obě rozpoznávání proběhla stejně dobře. Proto je nutné, aby se intenzita světla neuvažovala. V barevném schématu RGB toto není snadné, neboť jen těžko můžeme ze schématu vyčíst, které dvě barvy si odpovídají a liší se jen ve své intenzitě.

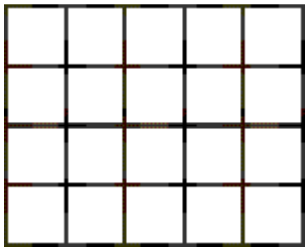
Vzorkování a uložení do mřížky

Jsou známy dva hlavní způsoby uložení obrazové informace. Jedním z nich je uložení do dvojrozměrné čtvercové mřížky, dalším je uložení do hexadecimální mřížky (pro ilustraci je hexidecimální mřížka zobrazena na obrázku pod tímto odstavcem). Právě hexadecimální rozložení má jednu podstatnou výhodu a to, že střed každé buňky je od všech sousedních buněk vzdálen stejně. Oproti tomu výpočetní nároky na zpracování takové struktury dat jsou neúměrně větší než přínos z toho plynoucí. Proto tento model pro zpracování této práce nebudu využívat.



Obrázek 9: Hexagonální mřížka [12]

Výše zmíněná pravidelná dvojrozměrná čtvercová mřížka je pro účely mé práce výhodnější, proto se dále zaměřím na její podrobnější popis. Pro zjednodušení budu mřížku dále pojmenovávat jako rastr, což je běžně používané označení. V rastru sice nemá každý pixel optimální stejnou vzdálenost od středu svých sousedů, avšak zpracování takové struktury dat je mnohem jednodušší. Postupy měření vzdálenosti jsou uvedeny v kapitole „Metriky měření vzdálenosti“ o několik řádků níže.



Obrázek 10: Čtvercová mřížka [12]

Pro účely využití obrazové informace je třeba tyto informace vlastnit v digitální podobě. To znamená, že každý bod námi zvoleného rastru bude obsahovat jednu nebo více hodnot. Tyto hodnoty získáme tzv. digitalizací vstupního obrazu. U procesu digitalizace je nejpodstatnější dodržení tzv. vzorkovacího teorému.

Pokud dojde k porušení podmínky vzorkovacího teorému, nastane stav, v němž pořízené vzorky nebudou přesně odpovídat původnímu signálu a informace bude ztracena. Tento jev bývá označován pojmem aliasing. Opětovná rekonstrukce signálu v takovém případě nebude možná.

„Půvab vzorkovacího teorému spočívá v tom, že umožňuje vyjádřit spojité funkce jistého typu hodnotami těchto funkcí (vzorky) v určitých izolovaných bodech. Přitom nejde o nějakou aproximaci, nýbrž o přesné vyjádření funkce.“ [13, str. 1]

To, že jde o přesné vyjádření je mimo jiné dáno tím, že lze bez ztráty signál opět zrekonstruovat. Přesná definice vzorkovacího teorému pak říká: “*Přesná rekonstrukce spojitého, frekvenčně omezeného, signálu z jeho vzorků je možná tehdy, pokud byl vzorkován frekvencí alespoň dvakrát vyšší, než je maximální frekvence rekonstruovaného signálu*.“ [14]

Formáty uložení

Obrazovou informaci je možné zachovat několika způsoby. Známe nekomprimované a komprimované formáty, ztrátovou a bezeztrátovou kompresi apod. Zmiňovat zde formáty, jako je JPEG nebo jiné podobného druhu nepovažuji pro účely této práce za důležité. V další práci budu pracovat zejména s bitmapou neboli rastrem. Tato bitmapa bude mít bitovou hloubku zpravidla 1 nebo 3 bajty. 1 bajt výhradně pro uložení barvy v různých odstínech šedi a 3 pro jedno ze dvou barevných schémat.

2.2 Přístupy k rozpoznání gest

Existují dvě základní skupiny gest. V první skupině jsou gesta statická, ve druhé gesta dynamická. *Je možné nalézt i jejich kombinaci* [28].

K rozpoznávání **statických gest** je možné využít například Template matching, Neuronové sítě, Skryté Markovské modely (HMM), metodu RBM (Restricted Boltzmann Machines) či jiné.

Princip Template matchingu spočívá v porovnávání zkoumaného signálu s určitým vzorem.

„Skryté Markovské modely jsou zvláštním případem stochastických konečných automatů. Tyto modely dovolují vyjádřit statistické závislosti dané pořadím pozorování (stavů), jako například v časových řadách“. [30, slajd 6]

RBM není tak známý přístup k zpracování statických gest, jako tomu bylo v předchozích případech. *Jedná se o stochastické neuronové sítě.* [31]

Dynamické gesto je obecně zaměřeno na pohyb sledované části těla po vymezeném prostoru, který je snímán jednou nebo více kamerami. Tím je možné rozlišit, jestli bude zachycovaný pohyb jen ve dvou rozměrech, či ve 3D (za použití dvou a více kamer).

Běžnou technikou používanou pro sledování pohybu ruky je nasazení rukavice, na kterou je umístěno několik senzorů. Ty poskytnou informace o poloze ruky, její orientaci a případné ohnutí prstů.[29, str. 14.4]

Další možnou technikou pro hledání dynamických gest je tzv. sledování optického toku. Principem metody sledování optického toku je zeměření se na význačné body v obraze a následná detekce těchto bodů v obraze následujícím. Rozdílem pozic těchto bodů lze zjistit, jak se celá scéna změnila.

2.3 Detekce kůže

V této části teoretického rozboru se budu zabývat detekcí kůže, která tvoří důležitý krok při předzpracování obrazu před vlastní detekcí gest a jejich rozpoznáváním. Z toho důvodu považuji podrobné seznámení se s danou problematikou za prvotní pilíř práce. Jednotlivé odstavce této podkapitoly budou popisovat kroky, jichž je zapotřebí k získání informace o pozici pixelů, jejichž barva by se dala klasifikovat jako barva kůže. Metod, jak barvu pokožky v obraze nalézt, existuje několik. Každá metoda má jak své silné, tak i slabé stránky, a proto se může ta či ona hodit v různých situacích, scénách, za určitých podmínek. Jednotlivé metody budou také popsány níže, stejně jako jejich nezbytné předchozí předzpracování.

2.3.1 Metody detekce kůže

Jak jsem popisoval o několik odstavců výše, pro detekci kůže není příliš vhodné používat RGB model uložení barev. Jelikož se jako výstup z kamery dá očekávat rastr v RGB formátu, bude žádoucí tento obrázek převést do HSV barevného schématu.

Popis principu fungování jednotlivých metod pro rozpoznávání kůže bude uveden níže. Pro všechny metody zůstává jeden společný rys a to, že každý režim bude zpracovávat údaje z HSV. Z obrázku v tomto formátu následně odstraníme (nebo jen nepoužijeme) část obsahující V (Value - jas).

Metoda porovnání H a S

První metoda, kterou je možné ke zjištění kůže využít, vychází z HSV barevného schématu. Využívá porovnání hodnot H a S každého pixelu, a pokud obě splňují určité podmínky, pak se prohlásí zkoumaný pixel za bod s barvou kůže. V opačném případě je považován za pozadí. Přesná implementace, kterou jsem v rámci své diplomové práce vytvořil, nastavení parametrů apod. bude vysvětleno a představeno v části „Realizace“.

Metoda založená na histogramu

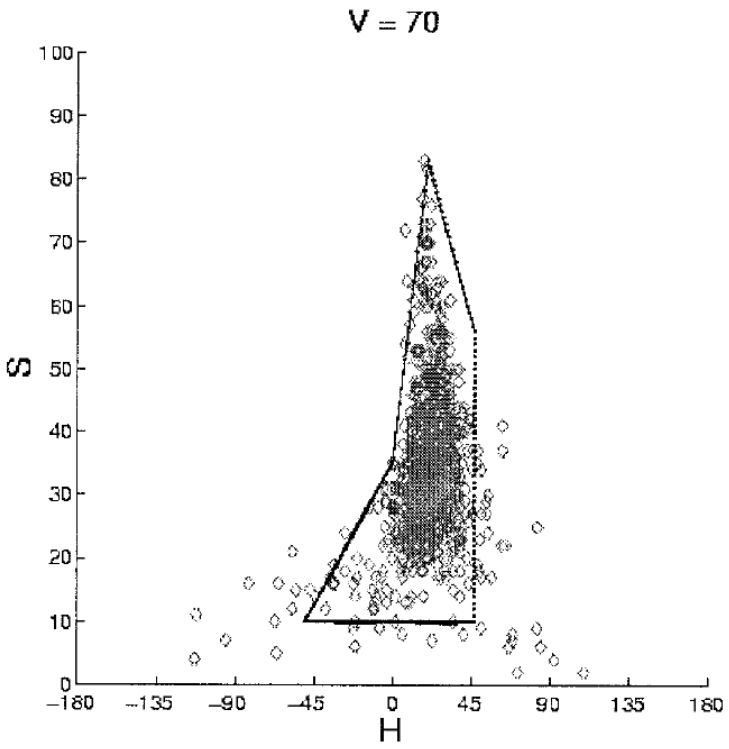
Postup využitý pro účely této metody má do značné míry obdobné prvky jako u předchozí metody. Rozdíl oproti předchozí metodě spočívá v tom, že v tomto případě nejsou předem dány hranice, které by přesně definovaly, co je a naopak co už není považováno za kůži. V rámci metody založené na histogramu si tyto hranice stanoví program sám. Počítačové zpracování tohoto úkonu je poměrně jednoduché. Jako vstupní data zde figuruje sada pixelů, které uživatel dodá a které považuje za shodné s barvou kůže.

K těmto pixelům se následně vytvoří histogram popisující jejich četnost. Bližší definování histogramu bude součástí dalšího textu této práce. Histogram tedy bude obsahovat četnost hodnot H a S, které odpovídaly pixelům s barvou kůže. Při vyhodnocování se poté ověří H a S v histogramu a zjistí se, kolik takovýchto pixelů, které kůží určitě byly, se v histogramu nachází. Pokud počet překročí stanovenou mez, je pixel prohlášen za bod kůže.

Ukázka histogramu, který obsahuje „natrénované“ informace o kůži, je graficky znázorněna v kapitole Realizace, konkrétně v části Hledání barvy kůže na základě histogramu.

Metoda využívající obalových přímek

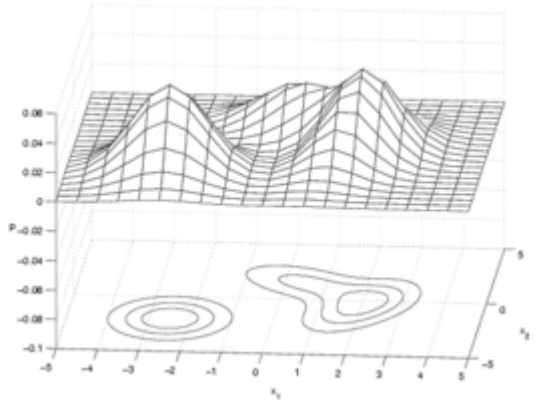
Princip této metody opět není výrazně odlišný od metod předchozích. Podobně jako u první metody porovnáváme barevnou část H a S, ale tentokrát se zjišťuje, zda je barva zkoumaného pixelu umístěna mezi přímkami, které určují hranice regionu kůže. Není to tedy jako u první metody pouze porovnání hodnoty se dvěma konstantami, ale mírně složitější vyhodnocování. Na obrázku, který následuje za tímto odstavcem, nalezne čtenář grafické vyjádření vzorků kůže (znázorněných malými kroužky), přičemž oblast jejich nejhojnějšího výskytu je vymezena obalovými přímkami.



Obrázek 11: Vymezení barvy kůže obalovými přímkami [27]

Metoda využívající GMM

GMMs se používá ke klasifikaci signálů. Algoritmus se vyznačuje tím, že *„každá třída je popsána jednou Gaussovkou. Neznáme přiřazení vektorů y_k do tříd, ani parametry Gaussovek“* [22].



Obrázek 12: Gaussian Mixture Models [24]

Předchozí obrázek představuje modelování tříd pomocí GMM. Ke klasifikaci pomocí GMMs dojde tím, že se algoritmus pokusí co nejlépe namodelovat vstupní data kolekcí vícerozměrných Gaussovek (v našem případě to bude 2D).

Formálně zapsáno, předpokládá se přítomnost určitých n-dimensionálních dat $X = \{x_1, x_2, \ldots, x_n\}$. Hledáme K Gaussových rozložení $\Gamma_1, \Gamma_2, \ldots, \Gamma_K$, která nejlépe reprezentují X, kde Γ_i je normála se středem μ_k a kovariancí Σ_k.[1] Pak:

$$\Gamma_\kappa = N(\mu_\kappa, \Sigma_\kappa) \qquad [2]$$

Dohromady s K odpovídajícími příspěvky váhy π_k, kde suma všech π_k pro k od 1 do K je 1, dostaneme následující rozložení hustoty pravděpodobnosti:

$$p(x_j) = \sum_{k=1}^{K} \pi_k \, p(x_j|\Gamma_k) \qquad [3]$$

V předchozí rovnici symbolizuje vektor x_j dvě složky barvy zkoumaného pixelu (H a S) a $p(x_j)$ pravděpodobnost, s jakou se tento vektor (pixel) považuje za kůži. Hodnotu pravděpodobnosti je vhodné porovnat s předem stanoveným prahem. Hodnoty, jejichž výsledky se nachází nad úrovní zvoleného prahu, budou považovány za kůži a hodnoty pod tímto prahem za pozadí. V praxi se tento model nejčastěji využívá pro výpočet algoritmu Expectation-maximization. Podrobnější informace k této problematice je možné dohledat v [1] nebo v [19].

2.3.2 Výsledky klasifikace

Na tomto místě představím čtenáři čtyři možné výsledky klasifikace. Tyto výsledky nekorespondují pouze s využitím metod pro detekci kůže, ale lze jich je identifikovat pro jakýkoliv klasifikátor. Zařazení do této podkapitoly vyplývá z důležitosti určení chybovosti při detekování kůže.

True positive matching
V tomto případě dochází ke správnému zařazení do třídy.

True negative matching
Pokud je výsledkem zkoumání zařazení do této skupiny, potom se jedná o správné nezařazení do třídy. Zkoumaný příznak nepatřil do určité třídy, a proto do ní zařazen nebyl.

False positive matching
Jedná se o chybně určenou příslušnost do třídy, čili příznak, který do určité třídy nepatří, byl identifikován, jako její součást.

False negative matching
V této kategorii chyb se nachází situace, kdy daný příznak do určité třídy patří, ale nebyl do ní zařazen.

2.4 Identifikace obličeje

Pro další postup práce v rámci ovládání počítače pomocí gest je zapotřebí navázat na detekci kůže rozpoznáváním obličeje. Za využití jedné z metod pro nalezení kůže uvedených v předchozím textu je možné vymezit pravděpodobnou lokaci obličeje. Takto lokalizovaný obličej je nutné dále upravit, aby jej bylo možné později nalézt. Další nutnou úpravou je převod z RGB do úrovní šedi a následné vyrovnání histogramu, které pomůže upravit globální vlastnosti snímku. Postup končí detekcí obličeje pomocí algoritmu AdaBoost.

2.4.1 Histogram

Histogram lze definovat jako statistické vyhodnocení dat. Pro využití v analýze obrazu se jedná o rozložení jednotlivých barevných úrovní. Pro každý pixel dojde k zapamatování jeho jasové hodnoty a ve výsledku se pro každou možnou jasovou hodnotu (obvykle 0-255) zjistí počet výskytů. To pak

vypoví o nasvícení celé scény a umožní další úpravy, jako je kvalitnější prahování (viz dále), změna rozložení barev, vyrovnání histogramu apod.

Vyrovnání histogramu je operace, v jejímž rámci se přepočítávají jasové informace o pixelu tak, aby bylo využito celého spektra hodnot a to při zachování přibližně stejné četnosti zastoupení.

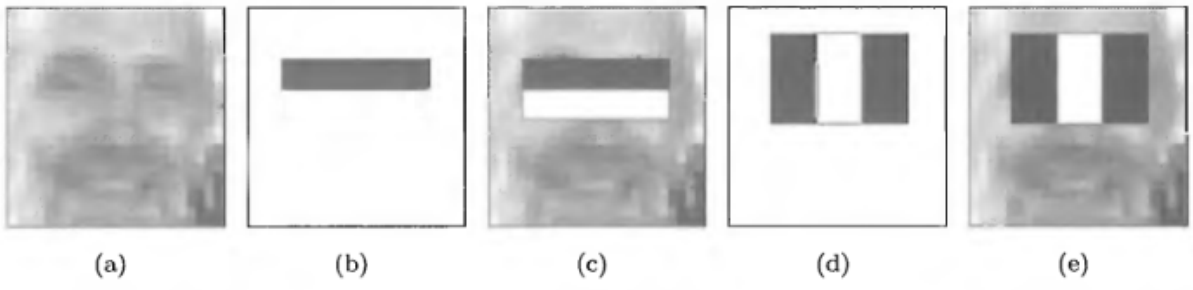
2.4.2 Haarovy příznaky

Základem pro odvození Haarových příznaků je vlnková transformace. *Haarovy vlnky jsou poměrně jednoduchými vlnkami, které neumožňují hladkou rekonstrukci signálu.*[32]

Haarova vlnková transformace byla historicky jednou z prvních, a to právě z důvodu jednoduchosti Haarových vlnek. Vlnkovou transformaci je obecně možné chápat jako konvoluci určité vlnky se signálem, který je v daném okamžiku zkoumán.

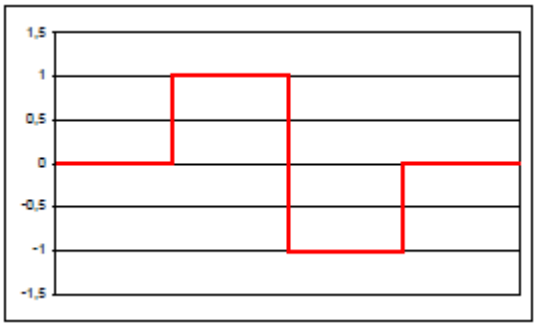
Haarovy příznaky jsou nejčastěji používanými příznaky pro metodu AdaBoost, která bude popsána v následujícím textu. Jejich význam pro detekci obličeje spočívá v tom, že na obličeji je možné rozlišit tmavší a světlejší místa. Příkladem je vlnka, která má v horní polovině černou barvu a v dolní polovině barvu bílou. V tomto případě se dá najít nápadná podobnost v části obličeje, kde oči tvoří tmavší část a oblast nosu a tváří je výrazně světlejší.

Podstatou obrázku pod tímto odstavcem je ukázat, jak se Haarovy vlnky prakticky využívají. Paralelně předchozímu popisu je možné detekovat oblast nosu. Jelikož oblast očí působí jako tmavší než nos, je možné ji pomocí Haarových příznaků jednoduše vyznačit.



Obrázek 13: Využití Haarových vlnek pro detekci obličeje [1, str. 495]

V návaznosti na předchozí definici považuji za vhodné zmínit, že jednou z vlastností Haarových vlnek je jejich pravoúhlost (ortogonalita). Další důležitou vlastností těchto vlnek je jejich nespojitost a s tím související skokové změny. Názorný příklad toho, jak Haarovy vlnky vypadají, je uveden na následujícím obrázku.



Obrázek 14: Průběh Haarových vlnek [33, slajd 24]

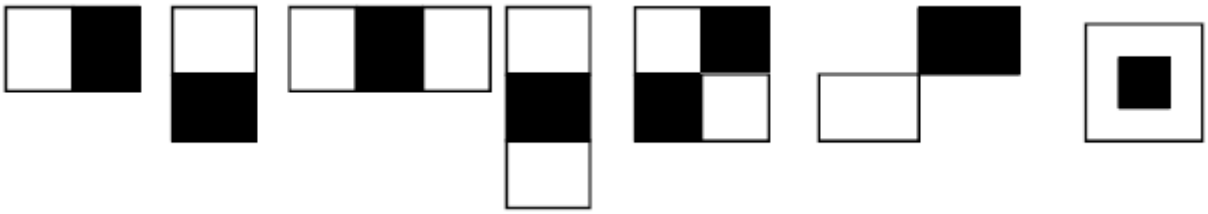
Jednorozměrné Haarovy vlnky jsou popsány následující rovnicí:

$$\Psi_{ji}(x) = 2^{\frac{j}{2}}\Psi\left(2^j x - i\right), \qquad i = 0, \dots, 2^j - 1, kde \tag{4}$$

$$\Psi(x) = \begin{cases} 1\ pro\ 0 \le x \le \frac{1}{2}, \\ -1\ pro\ \frac{1}{2} \le x \le 1, \\ 0\ pro\ ostatní\ případy, \end{cases}$$

kde $2^{\frac{j}{2}}$ je normalizační faktor.

Haarovy vlnky umožňují detekci jak obličejů, tak i automobilů či chodců. Pro účely rozpoznání obličeje je významná práce Violy a Jonese z roku 2001, kteří Haarovy vlnky vhodně seskupili do obdélníku, čímž byla umožněna rychlá a účinná detekce obličeje. [34, str. 230, vlastní překlad]



Obrázek 15: Ukázka tvarů Haarových vlnek [33]

2.4.3 AdaBoost

Historie AdaBoostu začíná u metody Boosting (anglický název se do češtiny nepřekládá), který byl rozpracován Kearnsem a Valiantem. Tito vědci si pokládali otázku, zda lze vylepšit tzv. slabé učící algoritmy takovým způsobem, aby z nich vznikl tzv. silný učící algoristmus, který by dával spolehlivé a přesné výsledky. Přes řadu dalších postupných kroků došli v roce 1997 Freund a Shapire k metodě AdaBoost. AdaBoost umožňoval odstranit řadu problémů, s nimiž si přechozí teorie poradit nedokázaly. [35, str. 2, vlastní překlad]

Metoda AdaBoost má velmi široké uplatnění. Jejím cílem je obvykle *lokalizace a rozpoznání všech smysluplných objektů v obraze* [23]. Myšlenka, která je základem AdaBoostu, je využita v řadě aplikací a to jak komerčních, tak i volně dostupných. Pro účely této diplomové práce se zaměřím na využití AdaBoostu pro detekci obličeje v obraze, což je běžně poskytovaná funkcionalita knihovny OpenCV.

Obecně je AdaBoost metoda, která umožňuje vhodné spojení mnoha slabých klasifikátorů a společně tak utvoří jeden silný. Slabým klasikátorem se rozumí například odezva Haarovy vlnky, prahování neboli treshold apod. Vstupem metody jsou trénovací data a výstupem silný klasifikátor. Slabým klasifikátorem může být testování jakýchkoliv dat, z nichž je možné extrahovat tzv. příznaky. Přitom musí však být splněna podmínka, že úspěšnost tohoto slabého klasifikátoru bude větší než 50%. Pokud dojde k jejímu splnění, může algoritmus dosáhnout klasifikační přesnosti blížící se 100%.

„Výsledný (silný) klasifikátor je lineární kombinací jednoduchých (slabých) klasifikátorů, ale není lineárním klasifikátorem. Není příliš odolný vůči šumu.“ [23]

Jako nejsilnější vlastnost AdaBoostu se dá charakerizovat fakt, že pro velké množství dat chyba na trénovacích datech klesá k nule a na většině dat jej není možné přetrénovat. Přetrénování můžeme chápat jako stav, kdy se klasifikátor až příliš přizpůsobil trénovacím datům a to natolik, že pro jiná data vykazuje výraznou chybovost.

Trénovací data jsou sadou obrazových informací, na nichž se trénují klasifikátory. Příklad jedné z možných sad trénovacích dat uvádím na následujícím obrázku.

Obrázek 16: Trénovací data pro AdaBoost [23]

Konkrétní algoritmus, který je pro metodu AdaBoost využíván, uvádím v následujícím textu [podle 1, str. 476, vlastní překlad]:

Při tvorbě algoritmu se využívá množina příznaků X, trénovací sada s m příznaky x_i, dále jim korespondující klasifikátory ω_i a předpokládá se rozdělení do dvou tříd ($\omega_i \in \{-1,1\}$).

1. Inicializuj K, počet slabých klasifikátorů, které budou využity.
2. Nastav $K = 1$ a inicializuj $D_1(i) = 1/m$.
3. Pro každý krok k natrénuj slabý klasifikátor W_k při využití trénovacího setu se sadou vah $D_k(i)$ tak, aby bylo určité reálné číslo přiřazeno každému příznaku x_i; W_k: $X \rightarrow R$.
4. Vyber $\alpha_k > 0 \in R$
5. Aktualizuj $D_{k+1}(i) = \frac{D_k(i) e^{-\alpha_k \omega_i W_k(x_i)}}{Z_k}$,
   kde Z_k je normalizační faktor vybraný tak, aby $\sum_{i=1}^{m} D_{k+1}(i) = 1$.
6. Nastav $k = k + 1$.
7. Pokud $k \leq K$, vrať se ke kroku číslo 3.
8. Finální silný klasifikátor S je definován jako:

$$S_{(x_i)} = sign\left(\sum_{k=1}^{K} \alpha_k W_k(x_i)\right) \quad [5]$$

Využití kaskádového klasifikátoru je vázáno na několik prvotních předpokladů. *Je zapotřebí mít nízkou False Alarm Rate, což znamená, že je důležité pracovat s řádově miliony pozic. Dále je nezbytná vysoká rychlost a také je nutné mít k dispozici neomezenou zásobu trénovacích dat pro třídu pozadí. Klasifikátor pracuje na tom principu, že v každém stupni kaskády eliminuje výřezy, jež zhodnotí tak, že nejsou hledaným objektem. Potom pokračuje dalším stupněm. Složitost jednotlivých stupňů kaskády se postupně zvyšuje. K eliminaci (nebo také zahazování dat) dochází již ve fázi trénování.*[33]

V současnosti existuje již řada různých verzí, které metodu AdaBoost určitým způsobem modifikují. Pro účely této práce bude dostačující základní metoda, jejíž popis jsem uvedl v předchozím textu.

2.5 Rozpoznání dlaní

Podkapitola zabývající se rozpoznáváním dlaní shrnuje popisy jednotlivých metod, které jsou pro nalezení dlaně zapotřebí. Jedná se o metodu podvzorkování, jednotlivé metriky pro měření vzdáleností, erozi a dilataci a další. Hlavní částí této sekce je popis Analýzy hlavních komponent.

2.5.1 Podvzorkování

Za podvzorkování je považován postup, při kterém se snižuje velikost rozlišení (počet pixelů snímku), čehož lze dosáhnout sloučením několika pixelů vstupního obrazu na menší počet pixelů obrazu výstupního. Při realizaci musí být dodržován vzorkovací teorém. Proto jsou vstupní pixely nejprve filtrovány.

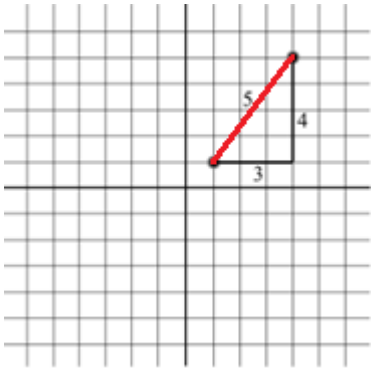
2.5.2 Metriky měření vzdálenosti

Z různých metrik pro měření vzdálenosti se zaměřím na tři nejdůležitější způsoby měření vzdálenosti dvou bodů. Pro zjednodušení budu uvažovat tuto vzdálenost ve 2D prostoru, není však obtížné najít zobecnění i do více rozměrů.

Euklidova vzdálenost je nejkratší možnou vzdáleností dvou bodů. Její výpočet je popsán vztahem:

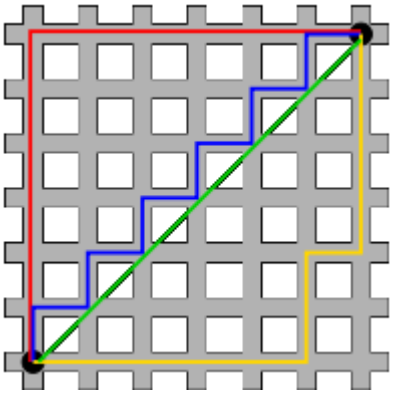
$$d_e = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \quad [6]$$

a následující obrázek lépe popisuje tento vzorec.



Obrázek 17: Znázornění Euklidovy vzdálenosti [16]

Druhou možnou metrikou je **Manhattan vzdálenost** (originální název Manhattan distance). Existují i další pojmenování téže metody jako třeba Taxi cab distance, City block distance apod. Na obrázku pod tímto textem je nakresleno porovnání s Euklidovou vzdáleností a je uveden výpočet.

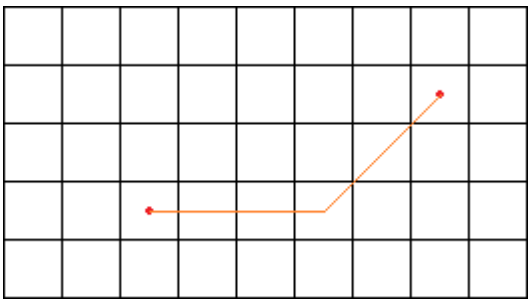


Obrázek 18: Manhattan vzdálenost [17]

$$d = |x_1 - x_2| + |y_1 - y_2| \qquad [7]$$

Uvedená rovnice je pro 2D, obecně by byla sumou pro všechny rozměry absolutních hodnot rozdílů souřadnice dvou měřených bodů. Pro uvedenou rovnici platí, že první resp. druhý bod je dán souřadnicemi (x_1, y_1) resp. (x_2, y_2).

Poslední metrikou vzdálenosti, kterou zde zmíním, je tzv. **šachová vzdálenost** (chessboard distance). Je ji možné vysvětlit jako počet tahů králem, které musí být vykonány k dosažení nejkratší cesty z bodu 1 do bodu 2. Jelikož je možno s králem hýbat i diagonálně a pouze o jedno pole, je zřejmé, že tato vzdálenost bude rovna maximální vzdálenosti každého z párů souřadnic.



Obrázek 19: Šachová vzdálenost [vlastní zpracování]

Vztah, který tento postup popisuje je následující:

$$d = max\{|x_1 - x_2|, |y_1 - y_2|\} \quad [8]$$

2.5.3 Druhy šumu

Málokterý obrázek je ušetřen šumu. Šumem nazýváme ponížení kvality obrázku vlivem některé nedokonalosti. *Šum se může objevit při zachycení obrázku, přenosu nebo zpracování a může nebo nemusí být závislý na obsahu obrázku.* [1, str. 29, vlastní překlad]

Bílý šum (nadbytečné světlé pixely) se vyskytuje velmi často. Jeho výskyt ani intenzita není závislá na frekvenci. Speciálním případem bílého šumu je tzv. Gaussovský šum. Jeho odlišnost spočívá v pravděpodobnosti výskytu, kde se tato pravděpodobnost mění podle Gaussova rozložení.

Aditivní šum je šumem, kdy barva každého pixelu je změněna přičtením drobné šumové hodnoty. Šum není na obrázku závislý.

Kvantizační šum (nedostatečné zastoupení možných hodnot) *se objeví, pokud používáme nedostatečný počet úrovní, např. 50 úrovní pro jednobarevný kanál. V tomto případě je možno tento druh objevit.* [1, str. 31, vlastní překlad]

Impulsní šum je dalším z druhů šumu, je jím např. velmi známý „sůl a pepř“. Tento se projeví tím, že se v obrázku objeví zcela bílé a černé pixely.

2.5.4 Prahování

Prahování je jednou z nejjednodušších metod předzpracování obrazu. Můžeme jej považovat za částečný hranový detektor a slouží ke zvýraznění některých geometrických vlastností. Výstupem metody obvykle bývá obraz stejných rozměrů (ve vztahu k původnímu obrazu), který však obsahuje výrazně nižší počet úrovní barvy. Často se obrázek rozčlení jen na bílé a černé oblasti. Princip spočívá v tom, že ověříme hodnotu každého pixelu a pokud je větší než zvolený práh, změníme jeho hodnotu na 255, pokud je menší, pak na 0. Obdobným způsobem za použití více prahů můžeme obraz rozdělit do více úrovní. Situace je ilustrována na obrázcích níže.

Obrázek 20: Prahování, před úpravou [18]

Obrázek 21: Prahování, po úpravě [18]

Prahování samotné nemusí vždy dávat dobré výsledky. Vlivem různého osvícení snímané scény na obrázku nebo stíny apod. dochází k tomu, že na obrázku jsou některá místa tmavější a některá světlejší. Pokud bychom takový snímek prahovali jednou konstantní hodnotou, výsledek by nebyl vyhovující. Výrazného zlepšení můžeme dosáhnout, když obrázek rozdělíme do částí a každou z nich prahujeme zvlášť. Hodnotu prahu této sekce určíme např. pomocí průměrné hodnoty jasu pixelů v této oblasti se nacházejících, histogramu apod.

2.5.5 Zjemnění, rozmazání

Zjemnění případně rozmazání (v angl. literatuře „Smooth“) je využíváno ve zpracování obrazu zejména k odstranění šumu. Obvykle se tato činnost provádí diskrétní 2D konvolucí. Vytvoříme obdélníkovou masku (často čtvercová, s lichým počtem řádků a sloupců). Tato maska poslouží k výpočtu příspěvků jednotlivých pixelů pro právě počítaný pixel z jeho okolí. Příklad obyčejné konvoluční masky je následující:

$$h = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad [9]$$

Použitím 2D diskrétní konvoluce můžeme tedy obraz vyhladit, čímž se odstraní vysoké frekvence. Úroveň toho, jak moc velké rozmazání se použije, je závislá na velikosti i parametrech masky. Bližší informace je možné dohledat v [19] nebo [1].

Obrázek 22: Originál [19]

Obrázek 23: S přidaným šumem [19]

Obrázek 24: Vyhlazeno 3x3 [19]

Obrázek 25: Vyhlazeno 7x7 [19]

Předchozí obrázky názorně dokreslují, jak se jednotlivé zásahy v praxi projeví.

2.5.6 Eroze a dilatace

Operace eroze a dilatace, které slouží ke zpracování obrazu, jsou metodami matematické morfologie. **Eroze** slouží v obraze k tomu, aby byly eliminovány oblasti, které jsou ve své podstatě chybně detekovanými pixely. Účelem **dilatace** je navrátit obraz do tvarů, jež se co nejvíce shodují s původním tvarem snímaného objektu. Zároveň dilatace slouží k zaplnění volných míst v obraze, jinými slovy jde o vyplnění děr, které v obrazu vznikly chybnou detekcí pixelů.

2.5.7 Mean-shift algoritmus

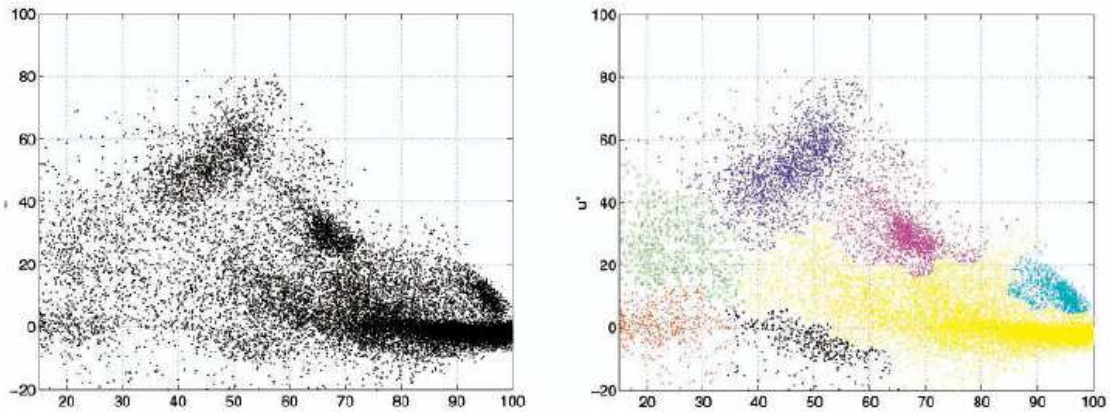
„Mean-shift je obecný algoritmus pro iterativní nalezení lokálního maxima hustoty vzorků.“ [20] *Tato metoda předpokládá, že hustota roste směrem ke středu shluku*. [22] Metoda se navíc vyznačuje tím, že nevyžaduje předchozí informaci o počtu nebo tvaru tříd. Hustotu f můžeme lokálně určit jako:

$$f(x) = \frac{1}{Nh^d} \sum_{i=1}^{N} k\left(\frac{\|x - x_i\|}{h}\right) \quad [10]$$

N označuje počet bodů (vektorů) v d rozměrném prostoru, h je poloměr okna a k je Gaussovské jádro a je možné jej vypočítat:

$$k(x) = \frac{1}{\sigma\sqrt{2\pi}} exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right] \quad [11]$$

Na následujícím obrázku je vidět, jak je možné metodu využít. Obrázek vlevo prezentuje získaná data. Čtenář může pozorovat shluky dat, které s největší pravděpodobností představují samostatné třídy. Metoda bez vnějšího zásahu tyto třídy po několika iteracích rozpozná. Nalezené třídy jsou vyobrazeny různými barvami na obrázku vpravo.



Obrázek 26: Využití Mean-shift algoritmu [22]

2.5.8 Wienerova filtrace

Wienerova filtrace se používá zejména k restaurování poškozeného obrazu. Je velmi důležité mít k dispozici statistické informace o výskytu šumu v obraze. Smyslem metody je počítání inverzní konvoluce s přihlédnutím k přítomnosti nenulového šumu. Algoritmus dosahuje minimální střední kvadratické chyby (*minimal mean square error*):

$$e^2 = \varepsilon\left\{\left(f(i,j) - \hat{f}(i,j)\right)^2\right\} \quad [12]$$

2.5.9 PCA – analýza hlavních komponent

PCA neboli Principal Component Analysis česky též „analýza hlavních komponent“ se využívá k různým účelům. V oblasti informačních technologií slouží například ke zpracování signálu, obrázků apod. Lze ji však využít i pro odvětví zcela vzdálená, kupříkladu pro botaniku, kde slouží k statistickému zpracování dat ze vzorkování. Setkat se s touto metodou můžeme i jako s Karhunen-Loéve transform (KLT) nebo Hotelling transform (později Harold Hotelling). [1, str. 111]

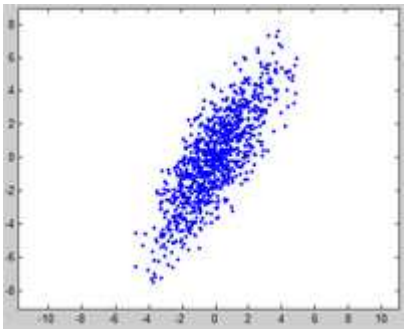
Analýza hlavních komponent spočívá v redukci velkého množství informací, které se v signálu nachází, na mnohdy podstatně menší datovou sadu, avšak s co možná nejmenší ztrátou důležitých charakteristik. *Ve statistice je metodou pro zjednodušení mnohorozměrného datasetu do méně početného prostoru a to zejména pro analýzu a vizualizaci. PCA je lineární transformace, která reprezentuje data v novém souřadném systému, ve kterém bázové vektory odpovídají módům největších odchylek v datech. Je to optimální lineární transformace, která rozděluje sledovaný prostor do kolmého podprostoru s největší odchylkou.* [1, str. 111, vlastní překlad]

Jako zdroj informací se předpokládá soubor dat, který obsahuje mnoho bodů v M rozměrném prostoru. Každý takovýto bod pak reprezentuje jedno pozorování a všechny informace, které se pro PCA použijí. Počet pozorování můžeme označit jako N. Vztah, který by následně měl platit je:

$$N \gg M \quad [13]$$

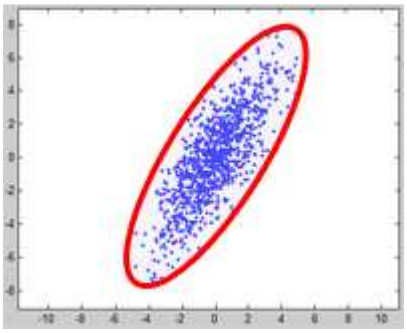
Nyní ve stručnosti představím jednotlivé kroky PCA analýzy a následně se budu v jednotlivých odstavcích těmito kroky zabývat a blíže je popisovat.

Prvním krokem je sběr dat. Data mohou vypadat tak, jak je to znázorněno na následujícím obrázku. Musím však zdůraznit, že téměř vždy jsou tato data jen těžko na obrázku zobrazitelná, neboť je nepopisují pouze dvě veličiny, tak jak je možné sledovat níže, avšak několik (desítky, stovky i více), řekněme *n* veličin. Poté by jejich zobrazení mělo také *n* dimenzí.



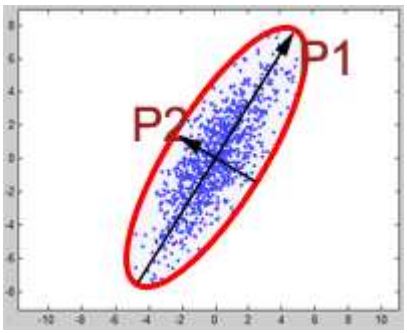
Obrázek 27: Ukázka dat pro PCA ve 2D [26]

Pokud jsou data k dispozici, je nutné se zaměřit na jejich statistické vlastnosti. Pro každou osu (dimenzi) se spočítá průměrná hodnota a následně pro každý vzorek standardní odchylka od těchto průměrných hodnot.



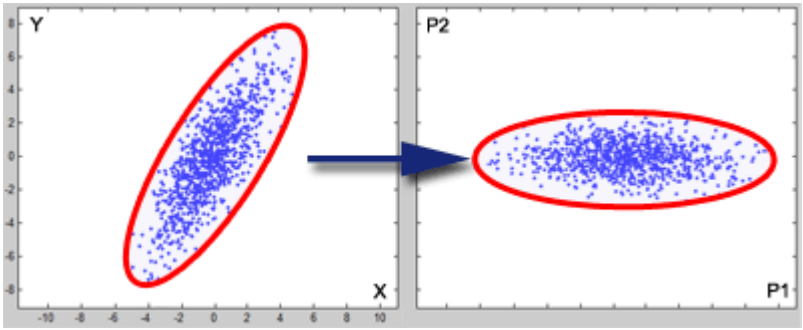
Obrázek 28: Rozptyl dat [26]

Na obrázku výše je možné pozorovat, jak jsou data rozptýlena a jakou charakteristiku má jejich tvar. Pomocí operací, které budou vysvětleny v navazujících kapitolách, je možné z takovýchto dat vyčíst tzv. vlastní vektory a vlastní hodnoty (angl. eigen vectors a eigen values). Vektory P_1 a P_2, které jsou na obrázku níže, jsou složením právě těchto dvou matic (vlastních vektorů a vlastních hodnot), kde směr P_1 a P_2 je udáván směrem vlastních vektorů normalizovaných na délku 1 a délka P_1 a P_2 je tvořena vlastními hodnotami.



Obrázek 29: Vektory P_1 a P_2, které reprezentují eigen vektory a eigen hodnoty [26]

Vektory P_1 a P_2 mohou být nazvány jako komponenty dat. Jak jsem uvedl výše, obecně platí, že data jsou závislá na více veličinách, proto i vlastní vektory a vlastní hodnoty budou mít stejnou mohutnost, jako je počet dimenzí. PCA analýza poté využívá toho, že vlastní vektory, jejichž odpovídající vlastní hodnota je menší (odpovídající P_x je kratší, např. P_2 na našem obrázku), jsou pro výslednou informaci o datech méně podstatné, a proto je možné tuto informaci odebrat.



Obrázek 30: Transformace dat pro PCA [26]

Nejprve je nutné provést transformaci dat, jak je vidět na obrázku výše. Poté se odebere informace o P_2, neboli hodnota se nastaví rovna nule. Následně se provede zpětná transformace, výsledkem které se všechny vzorky objeví položené na vektoru P_1.

Tímto jsem popsal základní princip metody PCA. Následující odstavce budou obsahovat podrobněji popsaný matematický základ jednotlivých kroků této analýzy.

Statistický základ

V této podkapitole se zaměřím na základní statistické metriky pro vyhodnocení získaných dat. Mezi tyto řadím zejména **průměrnou hodnotu, směrodatnou odchylku, rozptyl a kovarianci.**

Průměrná hodnota je vlastnost množiny dat, která specifikuje střed, kolem něhož se ostatní hodnoty nacházejí. Nejlépe je toto tvrzení možné vysvětlit na vztahu:

$$\overline{X} = \frac{\sum_{i=1}^{n} X_i}{n} \qquad [14]$$

Informace, která se tímto získá, však nemůže být dostačující. *Průměrná hodnota toho o datech příliš nepoví, pouze poukáže na nějaký střední bod. Například následující dvě množiny mají naprosto stejný průměr, ale očividně se velmi liší.* [36, str. 3, vlastní překlad]

[0 8 12 20] a [8 9 11 12] [15]

Prvkem, který tyto množiny rozděluje, nebude střední hodnota, nýbrž rozptyl hodnot od této průměrné hodnoty. ***Směrodatnou odchylkou*** *nazveme vlastnost, která popisuje, jak jsou data rozptýlena. Směrodatná odchylka je definovaná jako průměrná vzdálenost prvků zkoumané množiny od průměru.* [36, str. 3, vlastní překlad]

Směrodatná odchylka je poté vyjádřena vztahem:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{(n-1)}} \quad [16]$$

Ve jmenovateli se použije (n – 1) namísto n, protože se jedná pouze o podmnožinu všech možných vzorků. [36, str. 3, vlastní překlad]

Další možnou metrikou porovnávání množin dat je **odchylka**. *Metoda je téměř identická směrodatné odchylce a je dána vztahem:* [36, str. 4, vlastní překlad]

$$\sigma^2 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})^2}{(n-1)} \quad [17]$$

Tyto metriky jsou vhodné pro porovnávání dat v 1D. Pokud však máme data ve dvou a více dimenzích, je třeba použít tzv. kovarianci. *Kovariance popisuje vždy vztah mezi právě dvěma dimenzemi. Pokud se vypočítá kovariance mezi jednou dimenzí a jí samou, výsledkem bude již zmíněná odchylka. Oproti tomu, počítá-li se kovariance např. pro 3D (x, y, z), bude výsledkem kovariance pro (x, y), (x, z) a (y,z). Vztah pro určení kovariance je zobrazen níže.* [36, str. 5, vlastní překlad]

$$cov(x,y) = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{(n-1)} \quad [18]$$

Matematický základ

V této části vysvětlím význam pojmu *matice*, popíšu, jaké operace se pro PCA využívá a definuji pojem *kovarianční matice*.

Za **matici** budu ve své práci považovat seskupení hodnot obdélníkového charakteru o *m* řádcích a *n* sloupcích. **Vektorem** poté nazvu matici o jednom sloupci a počtu řádků odpovídající dimenzi prostoru, ve kterém je vektor definován.

Kovarianční matice je taková matice, jejíž každý prvek je výsledkem výpočtu kovariance svých souřadnic v matici. Pro vysvětlení ukáži na příkladu pro cov(x,y,z).

$$C = \begin{pmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{pmatrix} \quad [19]$$

Obecně lze tedy říci, že pro výpočet n-dimenzionálních dat je zapotřebí spočítat $\frac{n!}{(n-2)! * 2}$ *různých kovariančních hodnot.* [36, str. 6, vlastní překlad]

Pro další práci ještě zmíním postup nejdůležitější operace s maticemi, kterou budu potřebovat a tou je nasobení matice a vektoru.

Násobení matice a vektoru je spojeno s kontrolou, jejímž cílem je zajistit, aby matice měla stejný počet sloupců jako vektor řádků. Není-li tato podmínka splněna, pak nemůže být ani matice tímto vektorem vynásobena.

Postup násobení je znázorněn zde:

$$\begin{pmatrix} X_{1,1} & X_{1,2} \\ X_{2,1} & X_{2,2} \end{pmatrix} \times \begin{pmatrix} Y_1 \\ Y_2 \end{pmatrix} = \begin{pmatrix} X_{1,1} * Y_1 + X_{1,2} * Y_2 \\ X_{2,1} * Y_1 + X_{2,2} * Y_2 \end{pmatrix} \quad [20]$$

Vlastní vektor, vlastní hodnota

Pokud se bude na matici ve vztahu nahoře nahlížet jako na transformační matici a vektor bude považován za vektor, který je transformován, po vynásobení vznikne jiný vektor, který je transformován ze své původní pozice. Toto je povaha transformace, z níž vznikly vlastní vektory. [36, str. 9, vlastní překlad]

Za vlastní vektor lze považovat takový nenulový vektor, který je spojen s transformační maticí a po provedení transformace se jeho směr nezmění. Vlastní hodnotou (vlastním číslem) je nazýván koeficient, který udává, jak se změnila velikost vektoru.

Další důležitá věc je, že pro další zpracování je velmi vhodné nalezený vlastní vektor upravit, aby měl stejný směr, ale jeho velikost byla normalizovaná na velikost délky 1. [36, str. 10, vlastní překlad]

Princip PCA

Analýza hlavních komponent se provádí v několika krocích, kde každý krok je v podstatě využitím postupů, které jsem popsal v předchozích podkapitolách.

Nejprve se získají data a určí se jejích průměrné hodnoty. Následně se pro tyto data vypočítá kovarianční matice a k ní náležící vlastní vektory a vlastní hodnoty.

Každý vlastní vektor a jeho vlastní hodnota má určitou důležitost. Důležitostí myslím váhu, která udává, jak moc se daný prvek podílí na formování (utváření) dat (do jaké míry budou data změněna, pokud dojde k jejich odstranění). Tato důležitost je vyjádřena vlastní hodnotou a to v přímé závislosti, přičemž čím větší vlastní hodnota je, tím podstatnější je pro charakteristiku dat onen vlastní vektor.

Na konec analýzy se tedy vyberou vlastní hodnoty a vektory takové, které jsou pro data nejpodstatnější, nazveme je hlavní komponenty a sloučíme dohromady do tzv. vektoru příznaků.

V závěru této části chci uvést, že data je možné zpětně rekonstruovat beze ztrát jen v případě, že jsme žádné vlastní vektory ani hodnoty neodstranili. V opačném případě však ztráta informace nebude příliš významná, neboť právě odstraněné příznaky nebyly pro data příliš významné. Pro svou práci nebudu zpětnou rekonstrukci využívat, proto zde nebudu uvádět podrobný postup, je však možné jej najít v [36].

2.6 Dílčí shrnutí

Při studiu literatury týkající se zvolené tematiky jsem se seznámil s řadou postupů, které by mohly být využity pro zpracování této diplomové práce. Pokusil jsem se vybrat takové, u nichž jsem očekával, že budou poskytovat nástroje pro co nejefektivnější praktické zpracování. V předchozím textu jsem proto věnoval více pozornosti zejména metodám AdaBoost a PCA, které pro vývoj aplikace měly velký přínos.

3 Návrh řešení

V této kapitole představím čtenáři návrh toho, jak by měl výsledný systém pracovat a z jakých částí by se měl dle mého názoru skládat. Nejprve popíši specifikaci a omezení budoucí aplikace jako celku, následně ukážu schéma součástí systému a v závěru kapitoly vysvětlím každou součást samostatně.

3.1 Předpokládaná podoba budoucí aplikace

Vytvořím aplikaci, která nebude pracovat pouze s jednou dlaní, rukou nebo obličejem, jako tomu bývá u jiných prací. Mým cílem bude spojit informace z obou dlaní a současně detekci polohy/pohybu obličeje, a to proto, aby se rozšířila rozmanitost kombinací gest. Pro správné rozpoznávání zvolených gest a pro vyvarování se jejich chybné detekce budu volit taková gesta, která se od sebe výrazně liší. To, že gesta musí být volena dostatečně odlišně, aby nedocházelo k jejich chybné detekci, má za následek, že množství gest splňující tuto podmínku bude výrazně nižší, než kolik by jich bylo možné najít bez tohoto omezení.

Běžně používaný soubor gest jedné dlaně bude v rámci této diplomové práce rozšířen, jak již bylo zmíněno, o druhou dlaň a také obličej. Gesta budou rozdělena do dvou kategorií. V první kategorii se budou nacházet gesta vytvořená jako samostatná, tedy taková, která budou prováděna pouze jednou dlaní či obličejem a to samostatně bez součinnosti s ostatními. Druhá kategorie bude obsahovat různé kombinace vytvořené jak dlaněmi společně, tak jednou dlaní a hlavou či oběma dlaněmi a hlavou. Tímto způsobem bude aplikace schopna získat rozsáhlou sadu nástrojů pro ovládání počítače.

Přesná gesta využitá pro tuto diplomovou práci budou součástí konečného návrhu řešení popsaného v kapitole „Realizace“. Následující odstavce obsahují původní gesta, s kterými jsem pro realizaci počítal ve fázi návrhu celého projektu.

3.1.1 Gesta dlaní a obličeje

Jak bylo uvedeno výše, následující gesta byla vybrána jako základ pro obecnou představu o budoucí podobě diplomové práce. Již na začátku práce jsem si byl vědom toho, že pro finální detekci budu volit i jiná, nová gesta, případně dojde ke změnám původně zvolených. Předpokládaná změna gest je z velké části dána zejména tím, že v průběhu vývoje programu získám řadu zkušeností a dalších znalostí, jež umožní zlepšit detekci a přispějí tak ke zkvalitnění práce.

a) Gesta pravé ruky
Posun vertikální, horizontální a diagonální.

Obrázek 31: Posun doleva

Obrázek 32: Posun nahoru

b) Gesta levé ruky (pouze levá ruka)
Zejména náhrada kláves Alt, Shift, Ctrl, Caps Lock, tabulátor

Obrázek 33: Volně asociovatelná gesta

c) Gesta obličejem
Potvrzení/zrušení akce (Enter, Escape)

d) Levá i pravá ruka (obě dlaně současně)
Pro určení číslic

e) Levá ruka a obličej
Dosud nepřiřazena žádná gesta

f) Pravá ruka a obličej
Scroll myši (nahoru, dolů i do stran), prostřední tlačítko myši

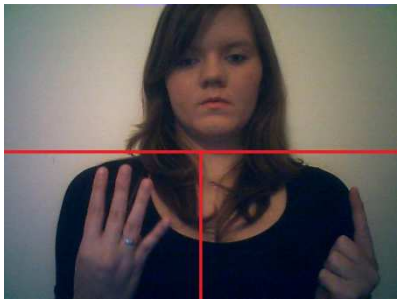
g) Gesta všech tří částí
Dosud nepřiřazena žádná gesta

3.1.2 Využití gest

Cílem zpracování předchozího seznamu gest bylo nastínit, jakým směrem se má práce bude dále ubírat. Výběr těchto gest není samoúčelný, navazuje na něj plánovaná vazba na vyvíjenou aplikaci. Zamýšlená podoba výsledné aplikace počítá s tím, že program bude běžet na pozadí a na základě určitého gesta zašle aktivnímu oknu prostřednictvím operačního systému odpovídající zprávu. Přesné uplatnění této myšlenky bude dopodrobna popsáno v kapitole „Realizace“.

3.1.3 Omezení

Předpokládám, že má snaha o současné získání a zpracování gest ze třech možných zdrojů povede k výraznému zvýšení výpočetního času, rozhodl jsem se proto rozdělit obraz na 3 segmenty. První segment bude tvořit horní polovina snímaného obrazu. V této části obrazu se program pokusí nalézt obličej. Dolní polovina obrazu bude dále rozdělena vertikálně, přičemž v každé z částí se aplikace pokusí najít dlaň ruky. Předpokladem je, že každá ruka bude zachycena zvlášť, tzn. pro pravou ruku pravá část obrazu, pro levou ruku levá část.

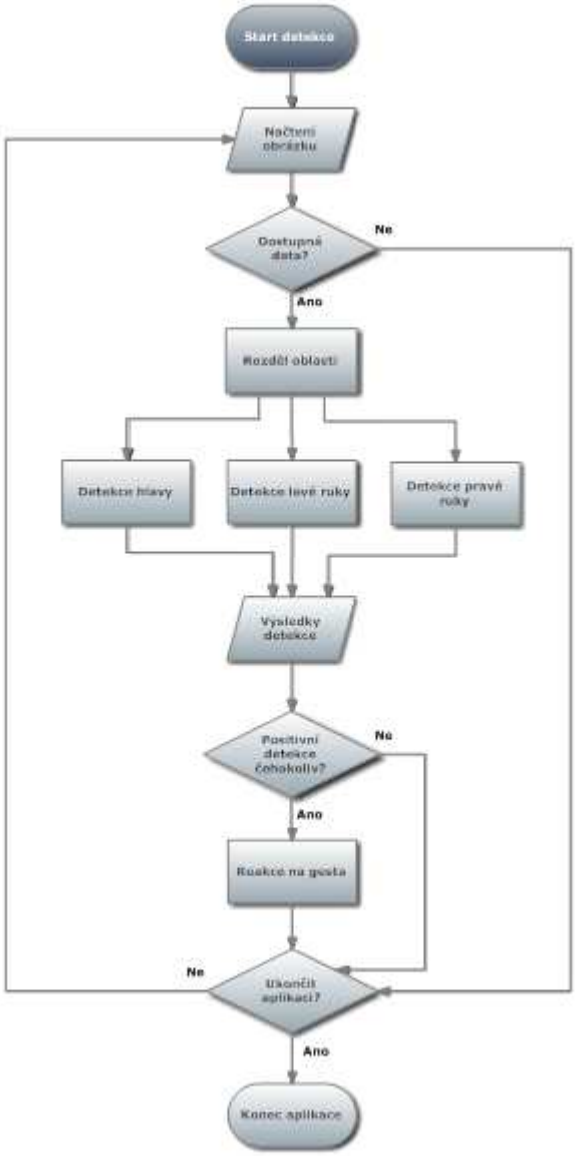


Obrázek 34: Rozdělení snímku

Program se zároveň nebude snažit zohlednit situace, kdy se ruka objeví v oblasti určené hlavě a naopak nebo na druhé straně, než je jí vyhrazena. Pokud bude levá ruka předvádět gesto v pravé spodní části obrazu, bude tato situace zkoumána jako gesto pravé ruky.

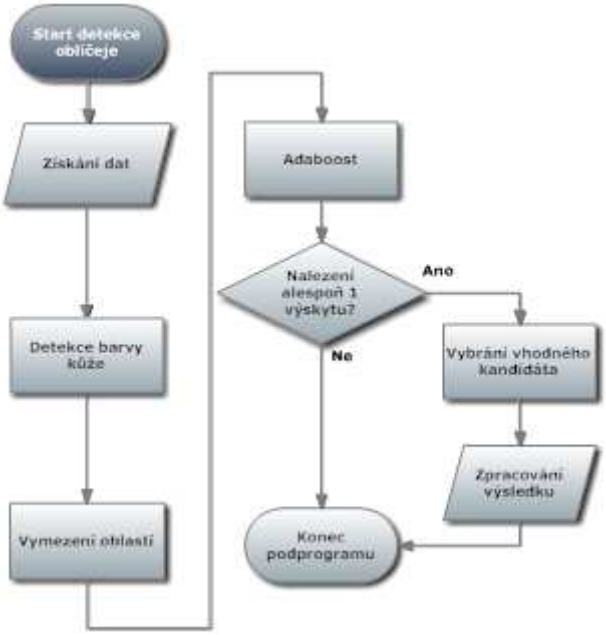
3.2 Průběh detekce gest

V této části popíši, jak bude probíhat detekce jednotlivých gest. V části aplikace zaměřené na detekci bude docházet k rozdělení obrazu a následnému sekvenčnímu volání funkcí, které obstarají detekci konkrétní části těla v odpovídající části obrazu. Hlavní smyčka je vidět na následujícím obrázku.

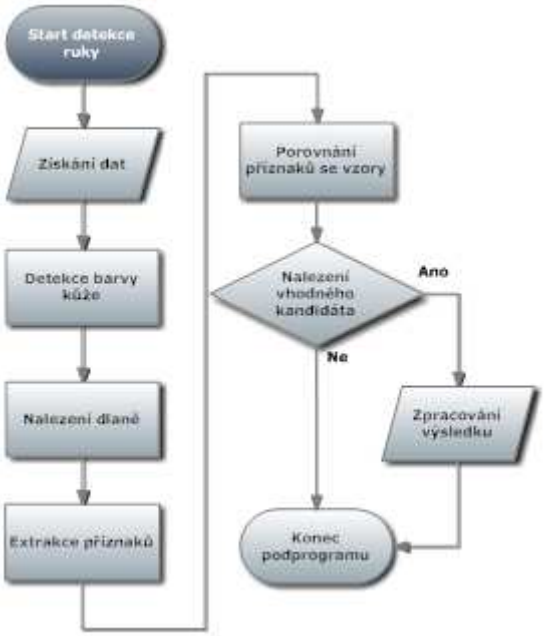


Obrázek 35: Hlavní smyčka detekce gest

Hlavní smyčka detekce gest znázorněná na předchozím obrázku popisuje, jaké procesy jsou v rámci algoritmu využity. K primární inicializaci snímacího zařízení dojde mimo tento algoritmus. Prvním krokem v rámci popisovaného algoritmu je tedy získání obrázku. V následujícím kroku dojde ke kontrole toho, zda skutečně došlo k načtení nějakého obrázku a v případě, že nedošlo, je možné rozhodnout o pokračování či ukončení běhu aplikace. V případě úspěšného načtení obrázku dojde k jeho rozdělení na tři části dle popisu uvedeného výše. Nejprve bude zkoumána horní polovina snímaného prostoru. Tato část bude předána k dalšímu zpracování detektoru obličeje. Jeho princip je znázorněn na následujícím schématu vlevo. Následně dojde ke zkoumání dolní části snímku, nejprve jeho pravé části a potom levé. Jednotlivé části obrazu budou dále předány ke zpracování podprogramu, který slouží k nalezení gesta ruky. Diagram popisující algoritmus, na jehož základě podprogram pro nalezení gest dlaní funguje, je graficky znázorněn na níže uvedeném obrázku vpravo.



Obrázek 37: Detekce obličeje



Obrázek 36: Detekce ruky

Schéma vlevo nahoře vyznačuje, jak by mělo probíhat vyhodnocování gesta ruky. Nejprve si je možné všimnout části, kde podprogram získá data. Aby nebylo nutné hledat obličej v celém vymezeném prostoru, neboť je tato operace poměrně náročná z hlediska výpočetního času, provede se nejprve detekce kůže. Na základě lokalizovaných oblastí touto metodou se nasměruje v části programu nazvané jako „Vymezení oblasti“ budoucí pokračování detekce obličeje právě na tuto část obrazu.

Po určení oblasti zájmu (nalezená kůže) se tato část obrázku předá metodě AdaBoost, která začne na tomto vzorku dat hledat obličej. V případě pozitivního nálezu se výsledky zpracují. Poté funkce končí.

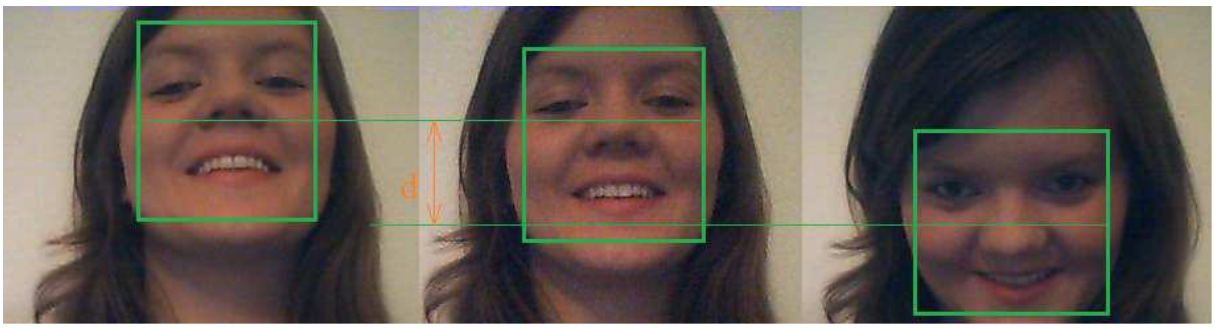
Na obrázku nahoře vpravo je možné sledovat postup detekce a zpracování ruky. Nejdříve se opět provede získání dat a následuje detekování barvy kůže v obraze. Nad nalezenou kůží se provede hledání maxima, jakožto princip „Nalezení dlaně“. Poté proběhne extrakce příznaků a jejich následné následné porovnání s již známými (natrénovanými).

Pokud je porovnání se vzorem úspěšné a dojde-li k dostatečné míře shody, bude obrázek prohlášen za konkrétní gesto ruky, s nímž byla shoda nejlepší. Jsou-li díky pozitivnímu nálezu gesta dostupné výsledky, jsou závěrem této metody zpracovány a podprogram se ukončí.

Získání dat v obou diagramech představuje zpřístupnění oblasti obrázku, která se má použít pro detekování. Detekce barvy kůže provede vyznačení těch oblastí, kde se ve značném množství vyskytuje barva, která by mohla představovat barvu kůže. Aby došlo ke zlepšení detekčních vlastností, bude oblast ze všeho nejdříve převedena do barevného schématu HSV (více viz kapitola „Barevná schémata“) a poté dojde k ověření, které části svou barvou spadají do rozsahu barev určených pro kůži.

Jako způsob detekce kůže je možné využít tzv. GMM, který po natrénování dvou tříd (kůže versus pozadí) poslouží pro klasifikaci, zda se o kůži jedná, či nikoliv.

Plánované provedení souhlasného gesta hlavy je pro čtenáře k dispozici na následujícím obrázku. Pro vyhodnocení pohybu hlavy bude pravděpodobně využit střed nalezeného obličeje a jeho rozdílná poloha na po sobě jdoucích snímcích.



Obrázek 38: Detekce souhlasného kývnutí hlavou

Obdobným způsobem, jako je vidět na obrázku výše, bude prováděna detekce i u nesouhlasného pootočení hlavou ze strany na stranu. U gesta hlavou se tedy změří posun středu obličeje v horizontálním a vertikálním směru a na základě těchto vzdáleností se rozhodne, zda bylo kývnuto, nesouhlasně zavrtěno, či zda se hlava nepohnula dostatečně, tudíž se pravděpodobně o gesto vůbec nejedná.

3.3 Skládání gest

Než bude možné považovat nalezené gesto za hledané, je potřeba, aby program prověřil, zda není součástí gesta složeného. V práci jsem si stanovil za cíl, že by má aplikace měla detekovat větší množství gest. Toho chci docílit mimo jiné i tím, že některá budou složena ze dvou nebo tří. Otázkou, která se nabízí na tomto místě, je, jak program zjistí, které z gest upřednostnit. V rámci zjednodušení jsem se rozhodl, že program vytvořím tak, aby došlo vždy k upřednostnění složeného gesta před gestem jednoduchým, jinak by bylo obtížné složeného gesta dosáhnout.

Nejprve se tedy detekují možná gesta ve všech třech částech obrazu. Až po dokončení poslední dílčí detekce se vyhodnotí, zda se jedná o gesto složené či nikoliv. Plánované pořadí rozpoznávání složených/jednoduchých gest je vidět v tabulce níže.

		Složené z…		
		Pravá dlaň	Levá dlaň	Obličej
Priorita	1	Ano	Ano	Ano
	2	Ano		Ano
	3		Ano	Ano
	4	Ano	Ano	
	5	Ano		
	6		Ano	
	7			Ano

Tabulka 1: Priorita skládání gest

3.4 Reakce na gesta

Po úspěšném rozpoznání je potřeba, aby navrhovaný systém dal uživateli najevo, jaké gesto se mu podařilo rozpoznat. Vhodný je dle mého názoru vizuální vjem. Aplikace bude pracovat tak, že po rozpoznání ukáže ikonu připomínající gesto v místě, které pro tento účel bude vyhrazeno. Zároveň se pod tímto obrázkem objeví textový popis gesta.

Domnívám se také, že by bylo vhodné ukázat uživateli původní obrázek a vyznačit v něm, jak se systém dopracoval právě k tomuto gestu. K tomuto účelu poslouží jiná část okna. Ovládání počítače provedu pomocí zasílání WM zpráv do systému Windows. Systém by měl tyto zprávy přeposílat aktuálnímu oknu, které se bude tímto způsobem řídit.

3.5 Popis nástrojů pro zpracování diplomové práce

V této podkapitole se zaměřím na podrobnější popis knihovny OpenCV, kterou plánuji využít při tvorbě aplikace. Dále popíšu princip ukládání v XML souboru.

3.5.1 Knihovna OpenCV

Knihovna OpenCV je volně dostupná knihovna, která obsahuje algoritmy umožňující práci s obrazem. Tato knihovna obsahuje přibližně 500 algoritmů, jež jsou optimalizovány pro využití v oblasti počítačového vidění. Knihovna je využitelná při práci s operačními systémy Windows, Mac a Linux. Jednotlivé funkce, které jsem využíval pro praktickou část této diplomové práce, budou představeny v příslušných odstavcích kapitoly „Realizace“.

3.5.2 XML soubor

XML je zkratkou pro anglické pojmenování Extensible Markup Language. Hlavní funkcí XML souboru je, jednoduše řečeno, přenos dat mezi jednotlivými aplikacemi.

Principem ukládání v souboru XML je uvození všech ukládaných dat tagy. Účelem tagů je popsat význam jednotlivých uložených dat či hodnot. Toto pravidlo ukládání umožní zmiňovanou přenositelnost informací mezi řadou aplikací.

3.6 Dílčí shrnutí

Celá kapitola Návrh řešení slouží k představení základních myšlenek, na kterých byl výsledný program vystavěn. Návrh řešení prezentuje zejména zamýšlený způsob realizace jednotlivých pokynů, ať již gest rukou či pohybů hlavy. Uvádí seznam metod, s nimiž jsem pro implementaci počítal.

4 Realizace

V této části diplomové práce popíšu konkrétní postupy, které jsem využil pro tvorbu finální aplikace. Zaměřím se na jednotlivé metody, jež jsem do práce implementoval. Představím všechny části rozpoznávacího procesu a uvedu podmínky, jejichž splnění je nezbytné pro fungování programu. Součástí této kapitoly budou i obrázky dokreslující postupné kroky.

4.1 Výběr programovacího jazyka, využité knihovny

Pro vývoj aplikace jsem se rozhodl využít některou z knihoven, která je vhodná pro práci s obrazovými daty. Po průzkumu dostupných alternativ jsem zvolil knihovnu OpenCV, neboť právě tato knihovna je volně šiřitelná, podává dobré výsledky a má velmi dobře zpracovanou dokumentaci, z níž je možné čerpat řadu informací, které mohou dále přispět ke zkvalitnění práce.

Výběr jazyka se tak zúžil na podstatně menší podmnožinu. Jazyk C# jsem byl nucen zavrhnout, protože OpenCV je psáno v jazycích C a C++, což je obecně nekontrolovaný kód (unmanaged), kdežto C# vyžaduje kód zabezpečený (managed). Našel jsem řadu řešení jak toto „navrapovat“ (nabalit) na kód psaný v jazyce C a tudíž tak, aby bylo i přesto možné knihovny OpenCV využít, ale zejména z obav o výkonový dopad jsem od tohoto řešení ustoupil.

V závěru jsem se rozhodoval mezi jazyky Python a C++. Právě s druhým jmenovaným mám o mnoho více zkušeností, proto je má aplikace napsaná právě v jazyce C++.

Kromě již zmíněné knihovny od firmy Intel jsem potřeboval již pouze několik funkcí z windows.h pro ovládání operačního systému.

I když je aplikace primárně určena pro využití na počítačích s operačním systémem Windows, přesto je na tomto zvoleném systému téměř nezávislá. Knihovna OpenCV je použitelná na mnoha jiných operačních systémech, jako jsou například různé distribuce Linuxu. Jedinou úpravou v případě zájmu o migraci například na Linux by muselo být nahrazení nejnižší vrstvy aplikace zajišťující zasílání zpráv do systému na takovou, která lépe odpovídá danému OS. Jiná změna by neměla být nutná.

4.2 Obecné rozdělení aplikace

Zvažoval jsem také, jakou podobu bude mít výsledná aplikace, a rozhodl jsem se pro spouštění z příkazové řádky. Jelikož v řadě případů využívám knihoven OpenCV, považoval jsem za chybu nevyužít jejich velmi dobře optimalizovaných kódů např. na zobrazování obrázku, je-li uživatelem požadováno. Odstoupil jsem tedy od původního návrhu na jednoduché GUI s využitím WinForms nebo jako Win32 aplikaci.

Aplikace, kterou je zároveň možné nazývat rozhraním, pracuje ve dvou základních módech. Prvním z nich je učení. Na otázku, proč již nemá systém vše naučeno, lze odpovědět velmi snadno. Jelikož si každý může pod statickým gestem představit řadu různých alternativ, považoval jsem za zbytečné omezovat uživatele na několik mnou vytvořených gest a vnutit mu i způsob, jakým by je měl zadávat. Z předchozího je patrné, že aplikace dává uživateli prostor k tomu, aby systém naučil v podstatě jakékoliv gesto a mohl jej následně využívat. Přesný popis procesu učení, stejně jako seznámení čtenáře s tím, co vše je v aplikaci možné vytvořit, upravit či změnit, bude zakomponováno do textu v kapitole nazvané „**Mód učení“.**

Druhou hlavní sekcí programu je část rozpoznávání. Zde se program snaží hledat hlavu a ruce v obraze a na základě toho a dalších konfigurací vykonávat nebo nevykonávat různé akce. Podrobněji se tímto budu zabývat v kapitole „**Mód rozpoznávání“.**

Dle mého názoru je jednou z hlavních předností této aplikace její variabilita. Aplikace se flexibilně přízpůsobuje změnám, umožňuje konfigurovat řadu vlastností a chování nejen pro vlastní rozpoznávání, ale i pro aplikaci jako takovou. Zároveň tím není uživatel obtěžován, a pokud si nepřeje cokoliv upravovat nebo si není jistý tím, jak změny provést, program je nastaven do vyzkoušené a dobře fungující konfigurace. Podrobněji budu kanály nastavení diskutovat v části „Konfigurace“.

4.3 Mód učení

Zde nejprve uvedu, jak probíhá hlavní proces celého trénování, a po uvedení tohoto postupu se zaměřím na dílčí důležité části, které podrobnéji popíši v podkapitolách zařazených v závěru této kapitoly.

Režim učení je určen zejména pro vytvoření sady gest, které uživatel od systému vyžaduje a které bude následně používat pro ovládání svého počítače. Spuštění nastane přidáním parametru –L v příkazové řádce za jméno spustitelného souboru. Po zadání tohoto příkazu si program načte ze souborového systému konfigurační soubory, jsou-li dostupné, nebo si do paměti uloží nastavení defaultní.

Poté začne aplikace příjmat snímky z webové kamery a čekat na příkaz „p“ pravá nebo „l“ levá. Zadáním jednoho z těchto písmen uživatel sdělí programu, kterou ruku by si přál systém naučit a kde na snímku ji má hledat. Aplikace obrazovou informaci zpracuje a následně ji zobrazí, aby bylo možné zkontrolovat, jestli gesto odpovídá požadavku. Zde program vypíše instrukce na standardní výstup a bude očekávat stisk klávesy malé „a“ nebo velké „A“ pro souhlas s dalším zpracováním nebo stisk jakékoliv jiné klávesy pro zrušení tohoto snímku a pokračování v učení.

Pokud je tedy zadáno cokoliv jiného, obrazová informace se vymaže a pokračuje se dál. Pokud ne, je požadována další informace. Systém teď zná gesto, ale neví, co znamená. Proto je třeba jej pojmenovat. Jelikož je možné zadat libovolné gesto, je zbytečné pro něj vytvářet složitá jména. Bude tak očekávána jen číselná hodnota. Aby program uživateli práci co nejvíce usnadnil, je přednastaveno 7 gest a jejich číselných hodnot (1 = směr vlevo, 2 = směr vpravo, 3 = směr vzhůru, 4 = směr dolů, 5 = sevření v pěst, 6 = otevřená dlaň a 7 = gesto pojmenované jako gurmán). Tyto hodnoty však záměrně nejsou povinné, ale spíše orientační, aby si uživatel udělal lepší představu o tom, jak by mohlo gesto

vypadat. Uživatel tedy může popisky hodnot ignorovat a zadat je takovým způsobem, který mu nejvíce vyhovuje. Není ani limmitován počtem 7.

Limit pro počet možných gest jedné ruky jsem stanovil na 19 (1 až 19). Dle mého názoru je velmi obtížné vymyslet množství gest převyšující tento počet tak, aby byla od sebe dostatečně odlišná a tudíž i dobře rozpoznatelná, proto je zakázáno zadat pro název hodnotu větší než právě číslo 19. Navíc, jak vysvětlím později, není ani více než 15 gest potřeba. Po zadání pojmenování gesta je obraz vložen do speciálního bufferu a inkrementován čítač pro odpovídající ruku a spuštěno další kolo učení.

V momentě, kdy již uživatel vyhodnotí, že zadal dostatek gest, stiskne klávesu Esc a učící se smyčka se ukončí. Ověří se hodnoty čítačů a pro každou ruku se zjistí, zda je počet větší než 2 (PCA analýza vyžaduje nejméně dva vzorky, vysvětlení je možné nalézt v teoretické části) a pokud ano, provede se PCA analýza vložených vzorků a výsledek se uloží do nově vytvořeného nebo tímto obsahem změněného XML souboru.

XML soubor jsem si zvolil především proto, že je s ním velmi jednoduchá práce a data jsou přehledně uložena. Uložením do onoho souboru proces učení končí. Pokud uživatel vybídl program k uložení natrénovaných dat do souborů podle vlastního výběru, je potřeba, aby při následném spouštění v režimu detekce a ovládání opět sdělil, kde tento soubor (případně více souborů) hledat

4.3.1 Spuštění učení

K naučení gest se užije webové kamery podobně jako pro následné rozpoznávání. Doporučuji, aby tuto operaci provedla stejná osoba, která bude následně program využívat a to proto, aby se gesta při trénování co nejvíce podobala gestům zadávaným při rozpoznávání. Dále doporučuji trénovat gesta při podobných světelných podmínkách a se zapnutím stejné metody pro detekování kůže.

Různé světelné podmínky mohou způsobit různé zastoupení v nalezených bodech barvy kůže podobně, jako jiná metoda může nalézt barvu kůže na některých částech dlaně, které jiná metoda přehlédla a opačně. Jelikož je i tvar regionů s kůží pro další zpracování významný, může se právě stejnou metodou nalezení kůže předejít řadě chybných detekcí a dosáhnout lepších výsledků v detekci požadovaných shod.

Při vlastním trénování není nutné dodržovat jakékoliv pořadí gest ani rukou. Je možné toto pořadí libovolně měnit nebo prohazovat. Pro pozdější kvalitnější detekci je vhodné každé gesto trénovat několikrát, optimálně 2x – 4x. Každé další trénování sice stále zvyšuje pravděpodobnost správné detekce a tím pádem i kvalitu detektoru, avšak i čas pro výpočet detekce.

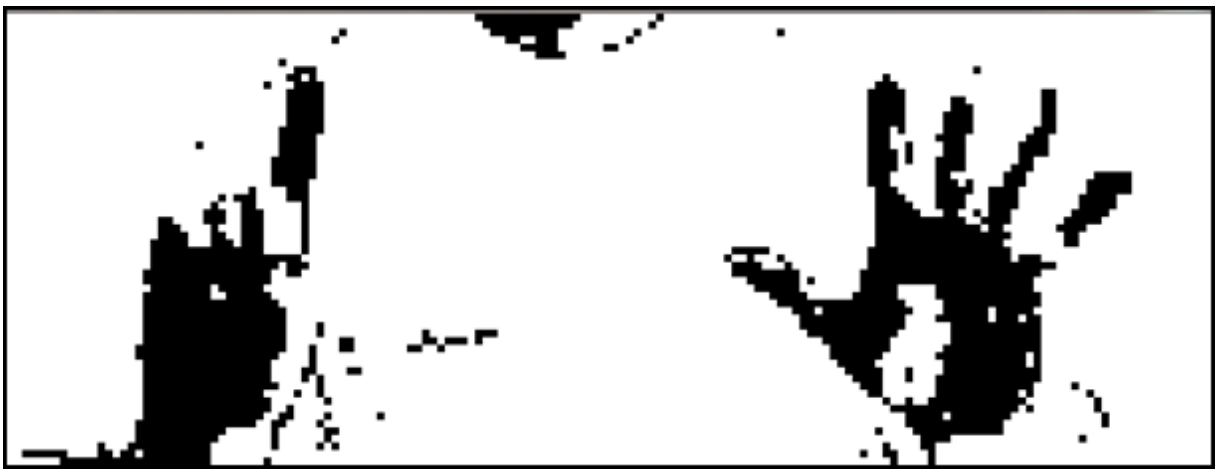
Pokud je tedy každé gesto zastoupeno např. 10 vzory a jen gest jedné ruky je např. 10, je tento výpočet podobnosti pro toto jednoduché gesto roven 100 * výpočet Manhattan vzdálenosti pro každý vlastní vektor (eigen vector) zkoumaného vzoru. Jak je patrné, s počtem vzorů pro jedno gesto roste lineárně i složitost jeho výpočtu.

Manhattan porovnání vzdáleností vektorů jsem si zvolil, protože jejich výsledky nejsou o mnoho horší než třeba Euklidovská vzdálenost a pro jejich výpočet není třeba použít výpočet odmocniny, který patří z hlediska výpočetního času mezi časově náročnější.

4.3.2 Analýza dat pro učení

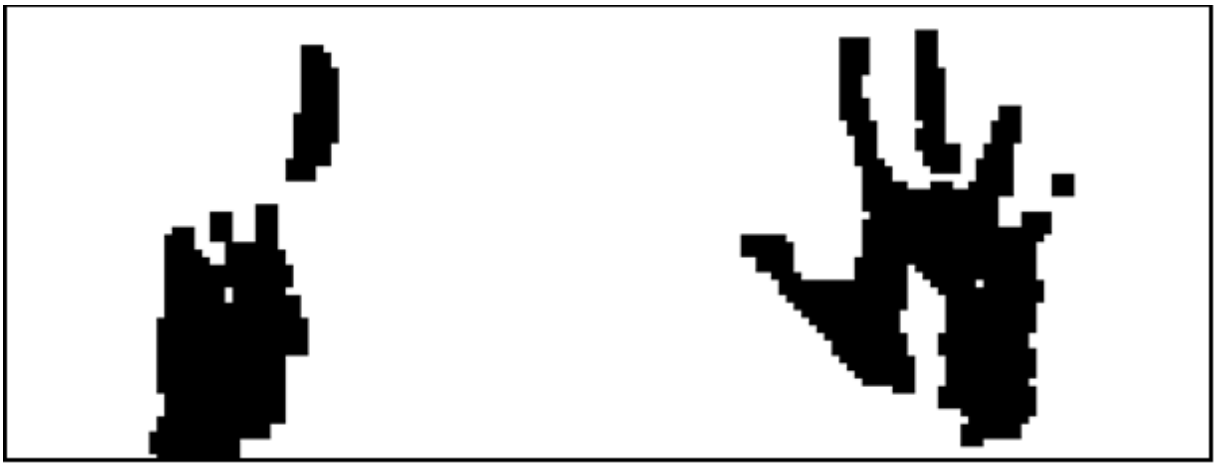
Jak již bylo zmíněno, učení probíhá v cyklu, který ukončí uživatel stiskem klávesy „Esc“. Do té doby bude program v nekonečné smyčce čekat na vstupy. Nezávisle na uživateli bude aplikace příjmat obrazová data z kamery a ty převádět pomocí detekce kůže na obrázek, představující masku, kde se kůže nachází a kde nikoliv.

Pokud bude uživatel využívat možnosti zobrazení oken, bude tuto masku vidět. V momentě, kdy určí, že již zadává gesto, stiskne na klávesnici zmíněné klávesy pro určení, jaká ruka gesto vykonává.



Obrázek 39: Detekce barvy kůže

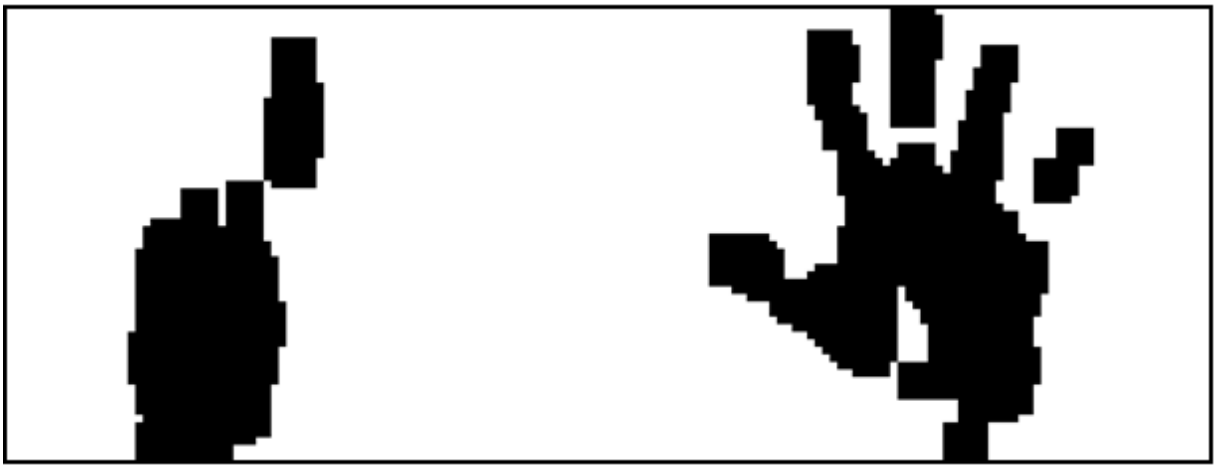
Následuje operace analýzy, která nalezne okno konfigurovatelné velikosti s maximálním zastoupením kůže. Dalším krokem analýzy bude odstranění chybně detekovaných pixelů pomocí použití „eroze“ a následné „dilatace“, které by měly odstranit většinu takovýchto chyb a zakrýt část naopak nenalezených pixelů (děr v útvaru).



Obrázek 40: Kůže s provedením 1x eroze a 1x dilatace

Na obrázku nad předchozím odstavcem je vyobrazena detekce kůže pro natrénování gest. Je zde možné zároveň sledovat, že obrázek obsahuje několik chybně detekovaných pixelů kůže. Oproti tomu obrázek pod ním, přímo nad tímto textem již prošel jednou iterací eroze a jednou iterací dilatace.

Došlo k mírné změně tvaru, avšak hlavní rysy zůstaly nezměněny. Chybné pixely vymizely a některé díry v dlani, zejména u pravé ruky, byly doplněny. Na snímku pod tímto textem jsem provedl dilataci dvakrát. Zde již metoda výrazné zlepšení nepřinesla a spíše došlo ke změně a tím pádem i ke zkreslení tvaru dlaní a tudíž i gest.



Obrázek 41: Kůže s provedením 1x eroze a 2x dilatace

U obrázků byly invertovány barvy z důvodu šetření životního prostředí a úspory toneru.

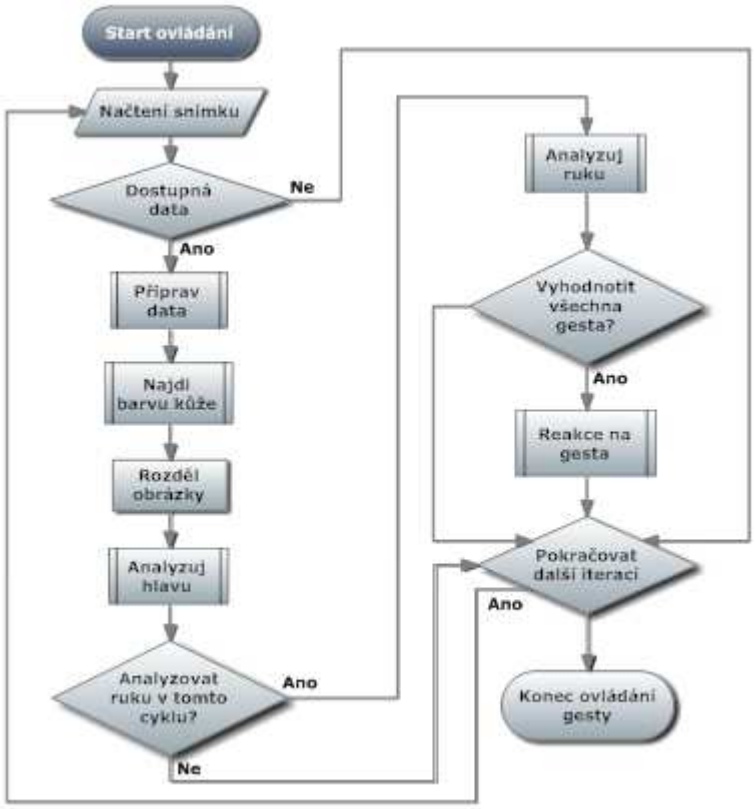
Takto upravenou část dat je možné ještě vyhladit pomocí Gaussovského 2D filtru implementovaného v knihovně OpenCV. Po provedení všech těchto úprav se obraz uloží do paměti a navýší jeho čítač. Ostatní postup je již popsán v úvodu kapitoly, jíž je tato součástí.

4.4 Mód rozpoznávání a ovládání

Tento režim je již komplikovanější, proto jej popíši v několika podkapitolách. V následujících odstavcích nastíním hlavní smyčku, následně se zaměřím na dílčí části.

4.4.1 Hlavní smyčka

Jak je patrné z obrázku níže, nejprve dojde k výběru snímku z mezipaměti pomocí cvQueryFrame (funkce knihovny open CV pro získávání obrázků). Je-li obrázek přetočený nebo přeje-li si to uživatel, obrázek se přetočí podle vertikální osy (v proceduře v obrázku pojmenovaná jako „Připrav data“). Dále je velmi důležitým krokem obrázek podvzorkovat, je-li to v konfiguraci nastaveno, aby se tím urychlily a zlepšily výsledky. Podvzorkování dosahovalo nejlepších výsledku pro hodnotu 4, kde již čas strávený nad zpracováním jednoho snímku je přibližně o řád rychlejší, přesto je však možné bezpečně rozpoznat předváděná gesta.



Obrázek 42: Finální podoba hlavního cyklu programu

Pro podvzorkování byla použita funkce cvPyrDown (Funkce knihovny OpenCV pro podvzorkování obrázku, kde je každý pixel výstupního obrazu složen ze čtverce 2x2 pixelu vstupního obrazu. Jako parametr funkce jsem využil IPL_GAUSSIAN_5x5, který vyvolá filtraci vstupních dat pomocí Gaussova filtru s jádrem o velikosti 5 x 5). Tato funkce po jednom zavolání vyplní cílovou strukturu IplImage a každý jeden pixel vypočítá ze 4 navzájem sousedních bodů (čtverec 2 x 2). Pro hodnotu podvzorkování 4 (myšleno jeden pixel složen ze čtverce 4 x 4) je tedy nutné provést podvzorkování hned dvakrát. Filtr jsem využil proto, abych se vyhnul chybě, která může nastat porušením vzorkovacího teorému (frekvence pořizování vzorků musí být alespoň 2x větší než maximální frekvence zpracovaného signálu).

Dalším krokem v diagramu je detekce kůže. Tuto operaci jsem popsal podrobně v kapitole nazvané „Hledání barvy kůže“, kde může čtenář najít podrobné vysvětlení mého postupu.

Část zajišťující rozdělení obrázků pracuje tak, že podle nastavených parametrů zavolá před každým dalším zpracováním části těla, pokud se v daném cyklu provádí, odpovídající funkci setImageROI. Tato vytváří nad obrazovými daty instanci takzvané ROI, což je oblast zájmu (Rectangle Of Interest).

Největší přínos této oblasti zájmu je v tom, že pokud uživatel zamýšlí zpracovat pouze část dat, nemusí složitě kopírovat tuto část, upravit ji a poté ji vkládat zpět, ale po nastavení ROI je možné provádět operace přímo nad zdrojovým obrazem a změny se provedou jen v takto ohraničené oblasti. Po skončení dané operace dojde k uvolnění ROI zavoláním metody OpenCV cvResetImageROI.

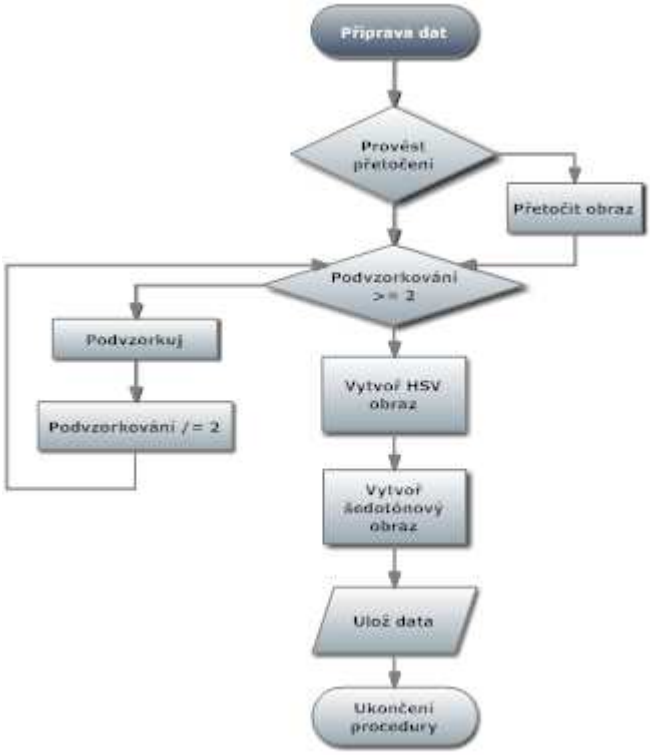
Celé vymezení a uvolnění oblasti, jakož i práce nad takto určenou částí dat jsou v této knihovně optimalizované, a proto má zcela minimální výkonový dopad ve srovnání s variantou, v níž bychom pro takové úpravy data kopírovali a poté vkládali zpět.

Jak je možné z předchozího diagramu pozorovat, dalším krokem se provede analýza hlavy. Tento proces je proveden v každé iteraci chodu programu a jeho výsledky se ukládají do speciálního bufferu. Blíže je metoda popsaná v kapitole nazvané „Analýza hlavy“.

V hlavní smyčce je implementován čítač průchodu smyčkou, který bude zdrojem rozhodnutí, ve kterém cyklu se bude provádět analýza dlaní a ve které se provede celkové zhodnocení provedených gest.

Vždy v každém třetím, šestém a desátém cyklu z deseti (čítač modulo 10 je 3, 6 nebo 0) dojde k provedení zpracování dlaní rukou. Popis přesného provedení této procedury může čtenář najít níže v kapitole nazvané „Zpracování dlaně“.

Pokud je hodnota čítače dělitelná 10 (každý desátý cyklus), dojde ke zhodnocení nasbíraných nedetekovaných gest a pokud byly tyto nalezeny, dojde k zavolání funkce, která se postará o provedení správného gesta. Blíže viz kapitola „Reakce na gesta“.



Obrázek 43: Příprava dat, podvzorkování

Na předchozím diagramu je znázorněno provádění přípravy dat. Postup podvzorkování jsem již popsal dříve, proto se nyní zaměřím na zbývající části tohoto schématu.

V další fázi následuje vytvoření obrázků v HSV formátu a v černobílé verzi. HSV následně putuje ke zpracování na hledání barvy kůže. O té bude pojednávat samostatná podkapitola v dalším textu. Za tímto odstavcem je možné pozorovat obraz převedený do formátu HSV. Když se čtenář zaměří na pozadí, může si povšimnout významné podobnosti barvy kůže a zdi situované za postavou.

Obrázek 44: Obraz převedený do HSV

Jak je patrné, mimo požadovaných oblastí se na snímku objevují i drobné oblasti kůže vlivem tzv. false positive matchingu, které by bylo dobré odstranit, aby dále nezpůsobovaly komplikace. Pro tento účel jsem do programu zakomponoval *n* krát provedenou erozi a poté *n* krát dilataci. Počet provedení *n* je opět konfigurovatelný, defaultně zvolený na 1 iteraci. Popis provedení eroze a dilatace je možné nalézt výše v části práce zabývající se teorií v kapitole „Eroze a dilatace“. K provedení eroze byla volána funkce cvErode a k provedení dilatace byla volána funkce cvDilate.

Pokud použijeme tyto úpravy několikrát po sobě, zbavíme se tak většiny chybně detekovaných pixelů, což je velmi přínosné, na druhou stranu však ztratíme poměrně velké množství informací o velikosti a především tvaru většího, pravděpodobně správně nalezeného objektu složeného z pixelů barvy kůže. Z mých pokusů bych nedoporučil vyšší iteraci než 2, je však nutné přihlédnout k úrovni podvzorkování. Je-li podvzorkování 1(originální velikost), nebo 2(2x2), bude možné použít i hodnotu vyšší.

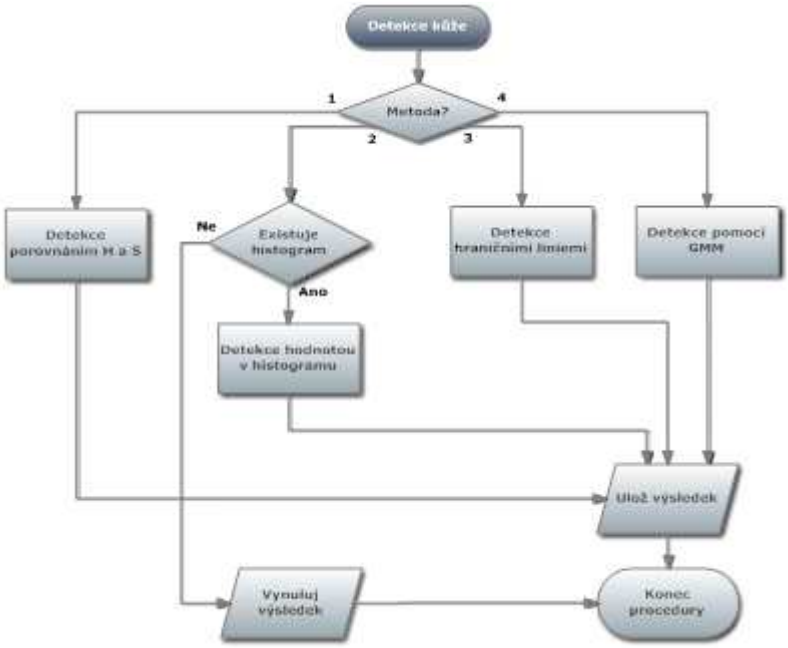
V této části již aplikace obsahuje dvě struktury IplImage. První z nich je označená jako pColorMask. Obsahuje jeden kanál o bitové hloubce 8 s hodnotami 0 a 255. 0 představuje barvu, kde se nachází pozadí, oblečení postavy apod. 255 je hodnota, která vyznačuje místa, v nichž se podařilo nalézt kůži, a tudíž kde se s velkou pravděpodobností nachází nějaká část těla. Druhá ze struktur se stejnou bitovou hloubkou i počtem kanálů je nazvána pCamImgGray. Jedná se o tzv. grayscale, neboli plnou paletu všech 256-ti odstínů šedi.

První zmiňovaná poslouží zejména k nalezení dlaně a k vybrání části obrázku jako gesta pro následné zpracování gesta jedné z rukou (nebo obou) a druhotné využití nalezne v pomoci a urychlení nalezení hlavy (tváře) na snímku. Podrobnosti k jednotlivým postupům budou popsány v následujících kapitolách.

Druhý obrázek bude posléze použit jako vstup pro metodu pro hledání obličeje pomocí AdaBoost implementovanou v knihovně OpenCV. Podrobnosti blíže objasňující tyto úkony budou čtenáři k dispozici níže.

4.4.2 Hledání barvy kůže

Hledání a zejména správné nalezení barvy kůže považuji za naprostý základ funkčnosti celé aplikace. Sestrojit takový algoritmus, který bude spolehlivě nacházet části lidského těla a přitom minimalizuje chybně detekovaná místa, bylo proto pro splnění cíle této diplomové práce velmi důležité.



Obrázek 45: Detekce kůže

Na obrázku výše je možné pozorovat, jakým způsobem probíhá detekce kůže v obraze a jakým způsobem se vybírá zvolená metoda. Aplikace načte z nastavení hodnotu 1 až 4, které specifikují metodu, která se pro detekci kůže má použít a na základě tohoto čísla zavolá příslušnou funkci. V závěru zpracuje výsledky a vrátí vykonávání programu do hlavní smyčky. Ostatní schémata dalších dílčích částí aplikace je možné nalézt v závěru práce v přílohách. Jedná se o schémata pro hledání hlavy a ruky.

Pro účely této práce jsem vytvořil tři vlastní způsoby detekce a použil jeden již existující. Je na uživateli, který způsob si zvolí. Každý přístup má své výhody a nevýhody a podává kvalitnější nebo méně kvalitní výsledky pro různá prostředí, úrovně a typy osvětlení, množství pigmentu v pokožce a podobně.

Každý algoritmus popíši ve zvláštní podkapitole a pokusím se shrnout jeho hlavní výhody a nevýhody.

Jednoduché nalezení pomocí H a S z HSV barevného formátu

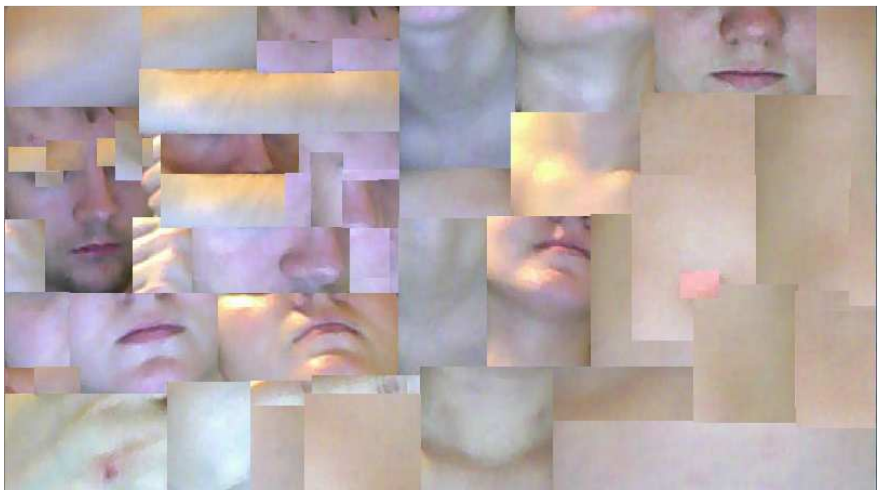
První a zároveň nejjednodušší metodou je přístup, v rámci kterého dochází k převedení obrazu z RGB (resp. ve skutečnosti je pořadí složek BGR) do barevného formátu HSV. Jak je již nastíněno v teoretické části této diplomové práce, tento převod se provádí s tím záměrem, aby byla oddělena informace o barvě a úrovni osvětlení. Jako mezní hodnoty jsem zvolil (na základě vlastních testů a také různých internetových článků a diskuzí):

$$0 \leq H \leq 19 \qquad [21]$$
$$48 \leq S \leq 255$$

Postup je tedy takový, že se ověří hodnota každého pixelu a jsou-li obě hodnoty v daném rozmezí, považuje se tento pixel za pixel barvy kůže a hodnota masky v tomto bodě se nastaví na 255, v opačném případě dojde k jejímu nastavení na 0.

Hledání barvy kůže na základě histogramu

Ve srovnání s ostatními metodami, které jsem v této práci použil, hodnotím tuto jako velmi adaptivní. Tato metoda podávala poměrně dobré výsledky, pro její uplatnění je však nezbytná vyšší míra interakce s uživatelem aplikace. Je zapotřebí, aby uživatel vytvořil libovolnou koláž, obrázek, který bude obsahovat pouze barvu jeho kůže. Může tedy spojit různé fotografie (nejlépe ze stejné kamery, kterou bude následně používat pro ovládání, čímž se provede i kalibrace) a z těchto fotografií vybrat a poskládat pouze ty části, kde je vyobrazena část lidského těla. Programu pak zadá cestu k tomuto souboru a ten se již postará o zbývající.



Obrázek 46: Sada dat pro sestavení histogramu kůže

Na obrázku výše jsem uvedl názornou ukázku koláže, kterou jsem vytvořil pro sestavení histogramu barvy kůže. Vyobrazení histogramu následuje za tímto textem. Na histogramu je možné pozorovat nechtěné zkreslení vpravo nahoře, které bylo způsobeno objekty koláže, které kůží nejsou (rty, vousy, vlasy apod.).

Obrázek 47: Histogram pro hodnoty H a S kůže

Nechce-li uživatel obrázek tvořit, nebo neví-li jak, je součástí programu jedna implicitní koláž a program k ní zná cestu, tudíž pokud není specifikováno, jaký soubor použít, použije se tento.

Vložený obrázek se vyhodnotí a převede do HSV. Následně se vytvoří 2D histogram pro složky H a S. Tento histogram si program vytvoří po spuštění a vloží si jej do své paměti, aby již za běhu programu nemusel brzdit svůj chod přístupem k souboru.

Poté, co je funkce tohoto druhého typu požádána o vyhodnocení barvy pixelu, porovná hodnoty pro H a S, a pokud se v dostatečném množství nacházely právě tyto hodnoty ve trénovacím obrázku, vypovídá to o tom, že pixel by měl být označen jako shodující se s barvou kůže, tudíž odpovídající pixel v colorMask se nastaví na hodnotu 255.

Metoda detekce kůže s využitím obalových linií

Smysl tohoto přístupu spočívá v tom, že oblast, kterou označíme za kůži, není vymezena pouze jednoduchým rozsahem hodnot, jako je tomu v prvním příkladu, ale tato oblast se určí pomocí několika přímek. Tyto parametry jsem vyčetl z již zmíněného zdroje. Přesný popis metody je možné nalézt v teorietické části práce.

Pro tuto metodu jsem využil hodnot obalových přímek ze zdroje [27]. Metodu jsem do své práce zahrnul jen pro doplnění a možné porovnání. Hodnoty jsou voleny tak, že rozpoznávají spíše tmavší pleť, proto pro mé odzkoušení nepodávala dostačující výsledky.

Metoda založená na GMM

Tento přístup se výrazně odišuje od předchozích metod pro detekci kůže, neboť nehledí pouze na samostatné pixely a neporovnává je jedna ku jedné, avšak snaží se vytvořit několik barevných tříd ve snímku. Tyto třídy jsou obvykle dvě nebo tři. Jedna až dvě třídy odpovídají různým pozadím a jedna třída je určena pro další nejvíce zastoupené rozmezí barev (v mém případě se jedná o kůži). Pokud tedy není program předučen pro dvě třídy – kůži a pozadí, může se i bez učitele naučit rozdělit obraz na tyto třídy sám.

Naučení vyžaduje výrazně více výpočetního času, protože natrénování (vytvoření tříd) je velmi náročné. Je také možné využít naučená data z minulé iterace a použít je jako vstupní data i pro nově probíhající učební proces. Tímto krokem se však výpočetní čas výrazně nezkrátí, spíše dojde ke

zpřesnění výsledků. Zároveň se eliminuje situace, kdy na dvou po sobě jdoucích snímcích bude stejná třída pojmenovaná opačným názvem, tedy třída jedna pojmenovaná jako třída nula a naopak.

Závěr k této metodě je takový, že může podávat velmi dobré výsledky, ale je dobré mít již třídy natrénované a pro každý obrázek pouze provádět jeho klasifikaci určující, do které třídy jeho pixely patří. Pro mou realizaci se proto na začátku provede naučení těchto tříd a v části detekce se již provádí pouze predikce hodnot a klasifikace na pozadí a kůži.

4.4.3 Zpracování dlaně

Po vytvoření struktury IplImage, kterou je v podstatě možné považovat za binární masku (obrázek s hodnotami 0 a 255), je program připraven nalézt a vyhodnotit případné gesto levé nebo pravé ruky.

Je důležité zmínit, že program pro rozpoznání očekává zdrojová natrénovaná data. Způsob pořizování těchto dat byl popsán v předchozím textu, v podkapitole „Mód učení“. Pokud uživatel nezasahoval do vytváření souboru s naučenými gesty levé resp. pravé ruky, nebude mít program problém tyto soubory (nebo soubor, bude-li se detekovat jen jedna ruka) nalézt. V opačném případě je třeba cestu k těmto souborům explicitně zadat.

Jestliže aplikace nenajde vhodný soubor, nastaví si příznak určující, že nebude možné tuto ruku použít a o této vzniklé situaci informuje uživatele na standardním výstupu. Po vyhledání souboru se provede kontrola na rozměry obrázku, který byl pro natrénování použit. Rozměry se musí shodovat, pro aplikaci je toto velmi podstatné. Pokud tomu tak není, není umožněno pokračovat s tímto souborem a aplikace se zachová obdobně, jako když jej ani nenalezne.

Po úspěšném načtení natrénovaných dat jsou tato uložena do paměti. Rozpoznávání gest pouze přistupuje k takto načteným datům na začátku spuštění aplikace a za běhu tak aplikace není zpomalována přístupem na disk. V případě, že vše proběhlo v pořádku a data jsou v paměti, je po spuštění funkce pro detekci gest rukou postupně pátráno po levé a následně po pravé ruce. Situace pro obě ruce je shodná, liší se jen v tom, jaký natrénovaný soubor se použije, proto budu dále popisovat jen jednu z nich.

Oblast zájmu obrázku se nastaví na odpovídající podle ruky, kterou hledáme. V této části metodou nazvanou „Mean shift“ algoritmus hledá lokální maximum nálezů kůže o proměnlivé (nastavitelné) velikosti okna. Toto okno odpovídá velikosti natrénovaného gesta. Po zaměření maxima dojde (je-li to uživatelem požadováno) k vyhlazení obrázku pomocí cvSmooth (2D filtrace obrazu z knihovny OpenCV) nastaveného jako lineární Gaussův filtr s proměnnými parametry dle uživatelské konfigurace.

Obrázek 48: Detekce gesta jedné ruky

Obrázek 49: Nalezená kůže jedné ruky

Na předchozím obrázku vlevo je vidět gesto, které jsem pojmenoval „otevření dlaně“, vpravo od tohoto obrázku je detekovaná kůže, která koresponduje s tvarem ruky.

Vyhlazený obrázek se použije pro PCA (viz teorie, kapitola „PCA – analýza hlavních komponent“) a výsledky této analýzy se posléze porovnají s natrénovanými daty. K obrázku, který byl natrénován a nejvíce se podobá právě pořízenému snímku gesta, je vyhledán název (číselné vyjádření gesta) a to se považuje za gesto, které bylo právě provedeno. Výsledky obou dlaní se uloží do struktury a po zpracování hlavy se budou všechny tři (byly-li všechny zadány) vyhodnocovat.

Výpočetní náročnost tohoto vyhodnocení roste lineárně s počtem (rozmanitostí) natrénovaných gest. Navíc velké množství různých gest nepříznivě ovlivňuje i úspěšnost správného rozpoznání. Čím více bude gest, tím více si budou jednotlivá gesta podobná a tím vyšší bude hodnota chybných rozpoznání.

Naopak pro zvýšení počtu správných detekcí bych doporučil každé gesto zadat 2x – 4x, čímž se sice zvýší čas strávený nad tímto výpočtem, ale zároveň to napomůže správnému vyhodnocení. Aplikace nemůže předpokládat, že uživatel vždy zadá gesto úplně stejně, jeho ruka bude stejně osvětlena apod. tudíž ani natrénovaný tvar gesta nebude totožný. Opakování gest proto může výrazně pomoci. Opět je třeba volit vhodný kompromis, aby výsledný počet porovnání nebyl příliš velký.

Další neméně důležitá část, kterou může uživatel ovlivnit kvalitu výsledků, je vložení několika „chybných“ či prázdných gest a jejich následné označení jako gesto 0. To výrazně sníží úroveň nalezených gest, když se o žádné gesto nejedná a k nalezení shody by nemělo dojít.

K porovnání podobnosti gest se po cvEigenDecomposite (funkce OpenCV pro výpočet vlastních vektorů a vlastních hodnot pro danou matici/obrázek) provede tzv. Manhattan měření vzdálenosti vektorů, ze kterého vyplyne blízkost gest. Podrobnosti k tomuto způsobu měření vzdáleností mezi vektory příznaků jsou popsáno v teoretické části v kapitole „Metriky měření vzdálenosti“.

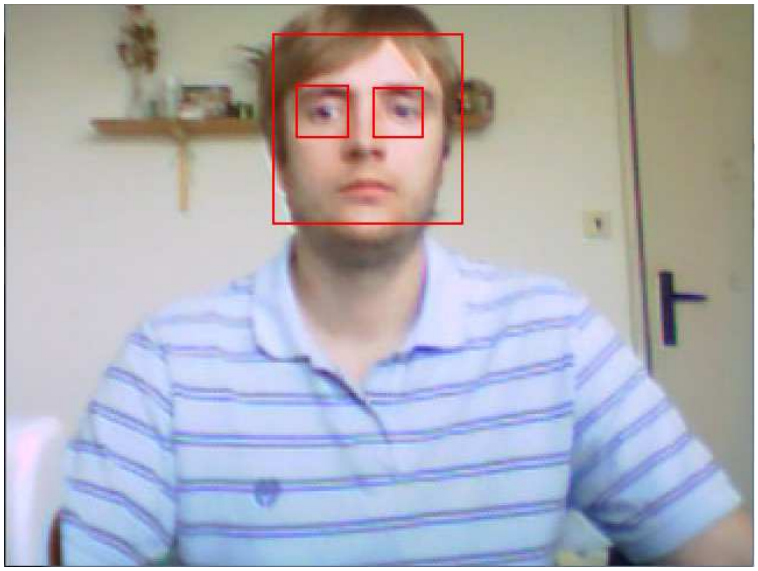
4.4.4 Zpracování hlavy

Pokud se jedná o první spuštění smyčky programu, nebo nebyla-li v předchozím průchodu nalezena tvář, vyhledá nejprve aplikace kůži v oblasti, kde by se hlava mohla nacházet. Obdobně jako u dlaní se v této oblasti pokusí nalézt lokální maximum kůže a tuto oblast si poznačit.

Následuje pokus o nalezení obličeje pomocí AdaBoost z knihovny OpenCV. K hledání je použit vstupní obraz převedený do 256 úrovní šedi. Byla-li v předchozím průchodu tvář nalezena, aplikace se pokusí mírně zvětšit tuto část a při hledání se na ni zaměřit, pokud ne, použije se oblast popsaná v předchozím odstavci. Tyto postupy mají velmi přínosné výkonové zrychlení, je však nutné mít vysoké procento nálezů tváře v takto předpočítané části snímku. Takového stavu je možné docílit např. tím, že se vhodně zvětší oblast posledního nálezu tváře před tím, než se oblast pro nové hledání použije. Jak moc se oblast bude zvětšovat je uživateli umožněno konfigurovat.

V případě, že AdaBoost nenalezne tvář v námi vyznačené oblasti, nezbývá, než hledat v celém snímku. Pro vyhodnocení, zda se o tvář jedná, či nikoliv jsem využil Haarovy kaskádové klasifikátory, které jsou součástí knihovny OpenCV. Pokud se ani takto nepodaří hlavu nalézt, pokračuje se dalším cyklem.

Nalezne-li program tvář, jsou v této obdobným způsobem vyhledány oči, nos a ústa. Nyní je vhodné zmínit dva mé přístupy k detekci gesta hlavy. Prvním, původním způsobem detekce gesta bylo porovnání rozdílu vzdáleností očí, jako princip pohybu hlavou ze strany na stranu a rozdílu vzdáleností mezi nosem a očima a ústy a očima pro zjištění pohybu nahoru a dolů ve smyslu souhlasného přikývnutí.



Obrázek 50: Detekce obličeje a očí pomocí metody Adaboost

Na obrázku výše je možné pozorovat praktickou ukázku, jak moje aplikace detekuje obličej a oči. Oproti tomu pod tímto textem je umístěn snímek zachycující nalezenou kůži v obraze, která byla použita pro rychlejší lokalizaci hlavy pro metodu Adaboost. Čtenář si rovněž může povšimnout, že detekce kůže chybně rozpoznala béžovou poličku v pozadí. To je způsobeno velkou podobností barvy této police s rozsahem barev běžných pro kůži.

Obrázek 51: Detekce kůže v oblasti hlavy

Jelikož jsem dosahoval nízkého pozitivního nálezu všech obličejových částí, rozhodl jsem se tento přístup přehodnotit. V druhém přístup bude aplikaci postačovat pouze oblast hlavy, ze které spočítám střed a pozice středu jednoho z očí a z těchto dvou bodů (jedno oko a hlava nebo druhé oko a hlava) budou vypočteny rozdíly ve směru horizontálním a vertikálním (dx, dy). Pokud některý z rozdílů přesáhne zvolený práh, bude podle toho aplikace považovat pohyb za gesto „Ano“ či „Ne“.

4.4.5 Vyhodnocování složitých a jednoduchých gest

Nejprve je nutné popsat, kdy se nalezené gesto považuje za správné a je tak zaznamenané do porovnávací struktury. Hlava se hledá v každém framu a to z toho důvodu, abychom měli maximum pozitivních nálezů. Po nalezení hlavy a oka (nebo hlavy, očí, nosu a úst viz výše), se tyto pozice, resp. středy útvarů, zapamatují ve zvláštní datové struktuře.

Při každém desátém průchodu hlavní smyčkou se vyhodnotí tento buffer (pole) struktur, a pokud nalezne dostatečné množství změn dx nebo dy (podrobnosti viz výše) bude do již zmíněné porovnávací struktury gest zaznamenáno gesto 1 pro nesouhlasné zavrtění hlavou (opakovaná velká změna dx) nebo gesto 2 pro souhlasné gesto přikývnutí nahoru a dolů (velká opakující se změna dy). V případě, že ani jedna z těchto možností dominantně nepřevažuje, je situace zhodnocena tak, že nebylo zadáno gesto žádné, tudíž hodnota bude rovna 0.

Rozpoznání gesta ruky má větší pravděpodobnost nalezení shody, proto postačí, když se během 10-ti framů pokusí aplikace rozpoznat gesto ruky třikrát a pokud je alespoň dvakrát vyhodnoceno stejně, považuje se toto jako správné, a proto se uloží do porovnávací struktury jako gesto dané ruky.

V rámci předchozích odstavců jsem popsal prerekvizity této funkce. Další část aplikace bude spuštěna pouze za situace, kdy bude zadáno jedno nebo více gest a jejím úkolem bude zavolat vhodnou reakci na gesta. Kterou akci bude aplikace považovat za „vhodnou“ je umožněno plně ovlivnit uživatelem. Tím je mu předána absolutní volnost nad celým chodem aplikace. Možnost konfigurace popíši v kapitole „Konfigurace“, která se tímto zabývá.

Předpokládejme, že došlo k zadání gesta, tudíž struktura gest obsahuje alespoň jedno gesto. Nejprve se ověří, zda byly zadány všechna tři gesta. Toto vyhodnocení je vždy prioritní, neboť jinak by k němu nemohlo nikdy dojít (pokud by bylo vyhledáno „správné“ gesto složené ze dvou, již by se další shoda nehledala). Jsou-li tedy nalezeny tři hodnoty v této struktuře, bude volána reakce na gesto složená z kombinací tří gest a program bude pokračovat další iterací nekonečné smyčky.

Za situace, kdy došlo k identifikování všech ří gest, ale pro danou kombinaci není nastavena žádná akce, se systém pokusí nalézt vhodnou reakci na jeden ze tří možných párů gest. Pořadí pokusů o nalezení vhodné reakce na ten či onen pár se řídí výhradně prioritami zadanými v konfiguračním souboru priority.xml.

Pokud ani jeden z párů neuspěl, pak se provede gesto jednoduché, opět podle priorit uvedených v již zmíněném souboru nastavení.

V případě, kdy byly přijaty pouze dvě gesta, se systém chová velmi podobně. Nejprve se pokusí toto gesto složené ze dvou gest vykonat, nenajde-li pro složené gesto odpovídající akci, pak se podle priorit pokouší vykonat gesto jednoduché.

Vždy platí, že pokud se nepovede k sadě gest nebo k jednoduchému gestu najít vhodnou akci, neprovede se akce žádná. Aplikace v takovém případě počítá s tím, že se pravděpodobně jedná o nechtěné gesto.

4.4.6 Reakce na gesta

Realizace reakcí na gesta je rozdělena tak, že každá kombinace reakce (7 kombinací) má vlastní obslužnou funkci. Podle toho, jestli se má zavolat reakce na gesto složené nebo jednoduché a z jakých částí je složeno, dojde k volání odpovídající ze sedmi funkcí. Ta se podívá do pole reakcí náležících tomuto složenému / jednoduchému gestu a zjistí, jestli se v tomto n rozměrném poli (n odpovídá gestu, pro gesto složené z obou dlaní i hlavy bude n = 3, apod.) pro tuto kombinaci nachází číslo reakce.

Pokud je reakce rovna nule, funkce vrátí neúspěch a podle schématu popsaného v předchozí kapitole bude zavolána jiná obslužná funkce. V opačném případě se toto číslo předá funkci, která vykonává všechna gesta. Zmíněná funkce je nejnižší vrstvou mé práce, je to v podstatě rozhraní zajišťující komunikaci mezi aplikací a operačním systémem.

4.5 Konfigurace

Celá aplikace obsahuje velké množství volitelných nastavení. Kladl jsem důraz na to, aby bylo možné aplikaci maximálně přizpůsobit jejímu uživateli. Prvním možným přizpůsobením bude variabilní postup detekce kůže. Uživatel tak nebude limitován například barvou své kůže. Existují velmi dobře se prodávající řešení jako je např. systém Kinect od firmy Microsoft, která mají problém s detekcí kůže lidí s tmavší barvou pleti. Věřím, že například toto jsem minimalizoval právě tím, že jsem implementoval čtyři různé detekční funkce, přičemž jedna je založena na histogramu, čímž v podstatě umožní nastavit libovolnou barvu kůže.

Další velmi podstatná konfigurace se týká úrovně podvzorkování. Tuto možnost využijí zejména uživatelé pomalejších počítačů, kterým tímto není znemožněno aplikaci využít. Většinu takovýchto nastavení je možné provést přímo z příkazové řádky. Přesná syntaxe se objeví při chybném spuštění nebo po zadání argumentu --help.

Pokud uživatel usoudí, že má zájem si tento program více přizpůsobit a tím dosahovat kvalitnějších výsledků, nežli při defaultním nastavení, není nucen vše zadávat pomocí příkazové řádky. Pro tento případ jsem vytvořil možnost vygenerovat konfigurační soubor. Ten aplikace sama vygeneruje a uživatel už pouze upraví údaje podle vlastní potřeby a soubor uloží. Při dalším spuštění se již toto nastavení načte a použije na místo přednastaveného.

V tomto konfiguračním souboru se dají nastavit parametry, jako je např. maska určující, která gesta si uživatel přeje hledat, jestli potřebuje všechny tři typy, nebo jestli si vystačí pouze s rukama, jen levou rukou, pouze hlavou a podobně. Maska má tvar integerového čísla, to se poté programově binárně násobí (AND) s hodnotami 1 pro hlavu, 2 pro levou ruku a 4 pro pravou ruku. Přeje-li si uživatel například používat jen ruce, zadá 6, pokud pouze hlavu pak 1 apod.

Další, již zmíněnou možností je detekce kůže. Akceptované hodnoty jsou od jedné do čtyř. Pořadí odpovídá popisu, který je uveden o několik stránek výše.

Z dalších možností chci na tomto místě zmínit ještě variantu nastavení minimálního a maximálního pohybu hlavy a dále například úroveň vyhlazení.

Cílem této kapitoly není poskytnout čtenáři vyčerpávající seznam všech dostupných modifikací. Úplný seznam všech možností nastavení je součástí souboru settings.xml, priority.xml a gesta.xml, které jsou přiloženy k této diplomové práci.

Minimální pohyb hlavy určuje, o kolik pixelů se musí změnit dx nebo dy (vysvětlení výše), aby to bylo považováno za pohyb a ne za „chybu měření“. Jelikož jsem zjistil, že motoda hledání obličeje a zejména částí obličeje není vždy stoprocentně přesnát, může nastat situace, kdy je za oko vyhodnocený jiný objekt. V takové situaci dx a dy budou velmi vysoké (nereálně oproti ostatním). Pro tyto případy slouží možnost nastavit práh určující, jak velký pohyb bude ještě považován za regulérní, byť rychlé natočení / posun hlavy a co už bude považováno za chybnou detekci. Tato proměnná se jmenuje maximální pohyb hlavy.

Poslední zmiňovanou možností konfigurace bude úroveň vyhlazení. Tato hodnota specifikuje, jestli se má provádět vyhlazení obrázku pomocí cvSmooth po provedení dilatace, nebo zda se má vyhlazení vynechat. Například pokud provedeme dvě iterace eroze a dilatace, vyhlazení již není téměř potřeba, pokud však je eroze a dilatace vypnuta, je velmi vhodné vyhlazení provézt. Nulová hodnota této proměnné zakáže vyhlazení, zatímco kladné číslo provede vyhlazení Gaussovým filtrem o velikosti jádra $n \times n$, kde n je právě zadaná kladná hodnota. Doporučené jsou hodnoty 11 – 15 (musí být vždy liché).

Tímto jsem popsal část možností příkazové řádky a soubor nastavení settings.xml. Aplikace má však, jak již bylo zmíněno, další dva soubory, kterými je možné ji ovládat. Prvním je soubor primárně nazývaný jako priority.xml. Tento soubor obsahuje 6 tagů nastavení. Každý tag odpovídá jedné proměnné. Tyto proměnné pak obsahují binární masku v podobě jednoho integer čísla. Jak tuto hodnotu správně nastavit vysvětlím na příkladu. Do značné míry se podobá postupům, které jsem využil i v jiných částech práce.

Uvažujme tedy situaci, kdy budu chtít nastavit první možnost, která se vyzkouší jako párové gesto hlavy a levé ruky. Hodnotu tagu <Maska_hledanych_gest> nastavím na číslo 3, kde 1 představuje hlavu binárně sečtenou (OR) s hodnotou 2 pro levou ruku. Obdobným způsobem uživatel může nastavit ostatní kombinace. Program pak opačným postupem čísla rozloží a tím získá priority, s kterými se má pokoušet hledat složená nebo jednoduchá gesta ve scéně.

Posledním souborem, kterého je možno pro kontrolu aplikace použít je samotné nastavení reakcí na gesta. Nastavení opět probíhá pomocí tagů souboru XML. Pro správné pochopení je třeba si představit, že je-li gesto složené z obou dlaní i hlavy, pak pro popsání všech kombinací je nutné trojrozměrné pole (kvádr) o velikosti [2][19][19]. Program implicitně nastaví všechny hodnoty na 0 a pouze explicitně určené pozice v poli nalezené v souboru budou nastaveny na patřičné hodnoty tagu. Postup nastavení je podrobně popsán v komentáři tohoto XML souboru.

4.6 Dílčí shrnutí

Hlavním záměrem kapitoly „Realizace“ bylo seznámit čtenáře s naprogramovanou aplikací. V předchozím textu jsem proto představil jednotlivé její části, způsoby ovládání a zaměřil jsem se také na vysvětlení různých možností pro volitelnou konfiguraci řady prvků.

5 Zhodnocení dosažených výsledků

V rámci této kapitoly uvedu, jaké výsledky vytvořená aplikace podává a tedy, jak efektivně funguje. Nejprve se zaměřím na hodnocení předzpracování prováděné pro zkvalitnění dalších operací, následně uvedu úspěšnost detekce gest.

Celé toto hodnocení bude rozděleno do tří podkapitol. První popíše míru správného rozpoznávání gest dlaní rukou, ve druhé pak budou uvedeny výsledky pro detekci pohybu hlavy a ve třetí podkapitole shrnu dosažený výkon (rychlost výpočtu gest).

5.1 Úspěšnost detekce gest provedených rukou

Na tomto místě popíši kvalitu výstupů metod, které jsem použil pro předzpracování snímku a v závěru uvedu hodnocení pro celé rozpoznání rukou.

Detekování kůže představovalo první zásadní krok této práce. Proces nalezení kůže je závislý na mnoha okolních vlivech (jako je úroveň a typ osvětlení, předměty v pozadí, jež jsou barvy podobné kůži, kvality kamery), a proto dosahuje nestálých výsledků. Tato skutečnost se následně promítne i do kvality navazujících procesů, které jsou na detekci přímo závislé. Pro zachování co nejvyšší objektivity výsledků jsem simuloval různé situace, využíval jsem jak denního světla, tak světla umělého. Zároveň jsem testoval dvě různé webkamery. První kamera měla maximální rozlišení udávané jako 640 x 480 px, druhá pak 320 x 240 px. Po využití druhé webkamery se výrazně projevilo zlepšení v rozpoznávání kůže, tudíž je možné konstatovat, že kvalita webkamery je pro výkonnost aplikace podstatná.

I přes prvotní problémy s detekcí se mi nakonec podařilo sestrojit takové čtyři metody, které se vzájemně vhodně doplňují a jsou funkční. Za běžného denního světla nebo vhodné intenzity bílého světla byla kůže v dostatečné míře nalezena (za dostatečnou považuji například takovou, jakou je možné spatřit u popisu učení gest v kapitole „Realizace“).

Mean-shift algoritmus, který jsem používal pro lokalizaci dlaně, fungoval poměrně spolehlivě. Při každém pokusu nalezl po maximálně třech iteracích lokální maximum kůže ve zkoumané oblasti a tím určil dlaň.

Eroze a dilatace měly pro aplikaci velký přínos. Jejich využitím došlo k odstranění většiny chybně detekovaných pixelů kůže a navíc byly kraje dlaně zbaveny většiny nerovností, což umožnilo snížit případnou rozdílnost dvou shodných gest. Výsledky zakomponování těchto metod lze pozorovat na obrázcích prezentovaných v podkapitole „Analýza dat pro učení“. Úroveň (počet iterací) eroze a následné dilatace je možné přiloženým konfiguračním souborem měnit, pokud však neexistují pádné důvody pro jejich užití, doporučuji používat pouze jednu.

PCA neboli výběr hlavních komponent a následné porovnání tohoto vektoru příznaků s naučenými vektory známých gest probíhalo spolehlivě. Gesto bylo správně rozpoznáno v 90 – 95%, přičemž řada chybných detekcí proběhla za situace, kdy se ruka pohybovala změnou z jednoho gesta na druhé, nebo

když nebylo zadané gesto žádné a systém se snažil i přes to najít shodu. První problém jsem vyřešil podmínkou, která požaduje, aby se čekalo na zadání alespoň dvou shodných gest ze tří. Druhý pak tím, že jsem vložil více gest označených jako žádné gesto, což představovalo například pozadí, nesmyslné tvary apod. Po těchto úpravách již úspěšnost detekce překročila 95%.

5.2 Správnost detekce pohybu hlavy

Detekce gest obličeje v závislosti na pohybu hlavy nepodávala ve srovnání s předchozí kapitolou tak dobré výsledky. V následujících odstavcích představím jednotlivé kroky a jejich charakteristiky a v závěru opět shrnu dosažené výsledky.

Hledání barvy kůže má pro detekci hlavy pouze informativní význam (její vliv na výsledky tedy není velký) a ve většině případů není ani zapotřebí. K detekci kůže pro oblast hlavy se přistoupí pouze v případě, kdy v předchozím průběhu hlava nebyla nalezena (osoba vystoupí ze záběru, nebo aplikace začíná) a v takovémto případě se provede. Pro tento účel se nezjišťuje tvar, tak jako tomu bylo u rozpoznání ruky, ale pouze umístění, jež se snadno nalezne pomocí **Mean-shift** algoritmu, který opět podával velmi dobré výsledky.

Převod obrázků, provedení vyrovnání histogramu a další podobné metody nebudu hodnotit, jelikož čas potřebný k jejich výpočtu je optimalizován na poměrně nízké hodnoty. Oblastí, kterou budu hodnotit, je **Detekování obličeje**. Tento krok má dobré výsledky, dochází k poměrně rychlému nalezení obličeje, a co se týká úspěšnosti detekce, pak mohu konstatovat, že obličej byl nalezen ve většině realizovaných pokusů. Pro spočítání pohybu hlavy, což je další postupný krok, potřebuji mít k dispozici informace o pozici očí.

Nalezení umístění očí v oblasti je částí aplikace, která nedává očekávané výsledky. Problém není v metodě, ale v tom, jaký obraz se zpracovává. Zde je totiž velmi podstatná kvalita obrazu, s kterou metoda pracuje. O kvalitě v tomto případě rozhoduje především rozlišovací schopnost kamery, úroveň okolního osvětlení a velikost podvzorkování.

Pokud by **funkce rozpoznání pohybu hlavy** dostala na svůj vstup vždy správně detekovaný obličej a oči, proběhlo by vyhodnocení vždy správně, neboť se jedná pouze o porovnání změn rozdílů pozic středů těchto objektů. K tomuto však nedochází tak často, jak bych očekával. Jakým způsobem se podařilo detekovat jednotlivé objekty v obrazech pro testované kamery v závislosti na zvoleném podvzorkování je možné vidět v následující tabulce.

	kamera 1			kamera 2		
Podvzorkování	1	2	4	1	2	4
rozlišení šířka x výška v pixelech	640 x 480	320 x 240	160 x 120	320 x 240	160 x 120	80 x 60
pravděpodobnost detekce hlavy	100%	100%	98%	100%	95%	90%
pravděpodobnost detekce očí	95%	80%	20%	40%	15%	5%

Tabulka 2: Porovnání nálezu hlavy a očí v obrazech pro 2 testované kamery

5.3 Výkonové testy

Z předchozí tabulky by se čtenář mohl domnívat, že kamera označená jako „kamera 1“ je výrazně kvalitnější a spolu s vyšším rozlišením dosahuje příznivějších výsledků. Toto však není zcela pravda. Při testování, kde jsem si zaznamenával, kolik času je věnováno zpracování obrázků z obou kamer, jsem došel k zajímavému zjištění. Zatímco průběh zpracování jednoho snímku z druhé kamery byl zpomalován v podstatě jen hledáním obličeje (ostatní úpravy si vyžádaly okolo 50ms v závislosti na přesných podmínkách), snímek z kamery číslo 1 trval o mnoho déle. Je pochopitelné, že pokud se neprovede podvzorkování obrazu, je obrázek z první kamery složen z více pixelů, a tudíž dojde i k více výpočtům.

Zaznamenal jsem, že kromě času nutného pro výpočet je výrazně vyšší i jiný časový údaj a to ten, který vyjadřuje dobu, po kterou funkce cvQueryFrame (funkce OpenCV sloužící pro získání nového obrázku z kamery) čeká na obrazová data od systému. Nepodařilo se mi zjistit, jestli je to vnitřní funkcí kamery, komunikací s operačním systémem, špatnými ovladačem, nekompatibilitou s OpenCV nebo z úplně jiného důvodu.

Celkový čas pro provedení jednoho kroku se tak pohybuje v rozmezí 40ms – 800ms podle zvolené kamery a velikosti obrázku.

5.4 Propojení gest a ovládacích prvků

Pokud se aplikaci podařilo vyhledat a správně rozpoznat zobrazované gesto, došlo k jeho vykonání ve 100% případů. Toto konstatování umožňuje potvrdit, že byl splněn stanovený cíl diplomové práce.

5.5 Dílčí shrnutí

Výsledky provedených testů jsou důležitou informací, která podává zprávu o tom, jak byla aplikace ve svých dílčích částech úspěšná. Jak jsem uvedl, většina testování dopadla dobře, rozpoznávání rukou i obličeje fungovalo bez větších problémů. Detekce pohybu obličeje má rezervy, které mohou být zdrojem pro další rozvoj programu. Propojení gest a ovládacích prvků počítače probíhalo se stoprocetní úspěšností.

6 Závěr

V rámci této diplomové práce jsem se rozhodl zpracovat téma Ovládání počítače pomocí gest. Důvody výběru jsem již vysvětlil v úvodu práce a na jejím konci mohu konstatovat, že svou volbu považuji za správnou. Podrobné seznámení se s touto problematikou bylo nejen zajímavé, ale považuji je i za přínosné pro mé budoucí pracovní uplatnění.

Diplomová práce byla rozdělena tematicky do několika kapitol, z nichž první se zabývala teorií, jejíž pochopení bylo nezbytné pro praktické zpracování daného tématu. V „Teoretické části“ jsem začal s obecným popisem zvolené problematiky, abych vymezil plánovaný rámec celé práce. Následně jsem se zabýval jednotlivými přístupy k detekci kůže a obličeje. Z jednotlivých metod jsem se podrobněji zaměřil na AdaBoost a dále na popis metody PCA, kterých jsem nejvíce využíval při vývoji aplikace.

V kapitole „Návrh řešení“ jsem chtěl čtenáře seznámit se základními premisami, s nimiž jsem začal na aplikaci pracovat. Předpokládal jsem, že ne vše, co si vytyčím, budu schopen přesně splnit, ale každá změna oproti původnímu plánu měla své opodstatnění.

K vysvětlení funkčnosti výsledného programu slouží kapitola „Realizace“, v níž jsem průběžně uváděl, z jakých důvodů jsem se odchýlil od prvotních záměrů. Aplikace, kterou jsem naprogramoval pro využití v operačním systému Windows s využitím jazyka C++ má uživateli poskytnout flexibilní řešení pro ovládání počítače pomocí gest, ať už defaultně nastavených, či volitelně nakonfigurovaných uživatelem. Rozhraní dokáže detekovat kůži pomocí čtyř různých metod. Pro ovládání počítače je po vyhledání oblasti kůže možné zvolit gesta izolovaná (levé dlaně, pravé dlaně, hlavy) i gesta kombinovaná (levá a pravá dlaň, levá dlaň a hlava, levá i pravá dlaň současně s hlavou atd.). Tyto gesta jsou asociovány s určitými akcemi, které si opět uživatel může buďto sám nastavit, nebo zvolit základní nastavení.

Zhodnocení výsledků je obsáhlejšího charakteru, proto jsem pro něj vyčlenil vlastní kapitolu, která bezprostředně předchází tuto kapitolu. Provedeným testováním jsem prokázal, že ovládání počítače pomocí gest v mé aplikaci funguje.

Cíl, který jsem si nastavil v úvodu práce, a sice vytvořit takový program, který bude schopen na základě gest hlavy a obou dlaní asociovat zvolené akce, považuji tímto za splněný.

Myslím si, že možností pro další rozvoj této aplikace je několik. Jak jsem již zmínil v kapitole „Realizace“, je možné aplikaci po kratší úpravě využít i pro jiné operační systémy. Aplikaci je dále možné zkvalitnit v oblasti detekce kůže, přičemž pokud by k tomuto došlo, bylo by možné snadněji vyhledávat tvary objektů, které mají barvu kůže (dlaň, lepší rozpoznání jednotlivých prstů atd.). Další alternativou pro zkvalitnění programu je oblast detekce hlavy, kde dosahované výsledky nejsou ideální. Zjednodušení detekce pohybu hlavy na zkoumání pouze jejího středu a změny jeho polohy nebo zkvalitnění detekce částí obličeje jsou jednou z cest k dosažení úspěchu. Zrychlení aplikace by mohlo být účinné v případě upravení predikce polohy hlavy v souvislosti s trendem změny předchozí pozice.

Pro zvýšení uživatelského komfortu by mohlo posloužit kvalitní zpracování grafického uživatelského rozhraní.

Seznam obrázků

Seznam tabulek

Seznam vzorců

Reference

[1] SONKA, M.; HLAVAC, V.; BOYLE, R. *Image processing, analysis, and machine vision.* 3rd edition: Thomson, 2008. 829 p. ISBN 978-0-495-08252-1.

[2] ZEMČÍK, P.; ŠPANĚL, M. *Zpracování obrazu – Studijní opora.* FIT VUT v Brně. 2006.

[3] STRACHOTA, P. *Lidský zrak, vnímání a reprezentace barev*. [online]. FJFI ČVUT v Praze. [cit. 27.12.2010]. Dostupné na URL: <http://saint-paul.fjfi.cvut.cz/base/public-filesystem/admin-upload/POGR/POGR1/03.barvy.pdf>

[4] REICHL, J.; VŠETIČKA, M. *Encyklopedie fyziky.* Vznik a druhy vlnění. [online]. [cit. 27.12.2010]. Dostupné na URL: <http://fyzika.jreichl.com/index.php?sekce=browse&page=165>

[5] PHYSICS.MFF.CUNI.CZ. *Rovinné vlny.* [online] [cit. 28.12.2010]. Dostupné na URL: <http://physics.mff.cuni.cz/kfpp/skripta/kurz_fyziky_pro_DS/display.php/optika/1_3>

[6] WIKIPEDIA. *Electomagnetic radiation.* [online]. [cit. 27.12.2010]. Dostupné na URL: <http://en.wikipedia.org/wiki/Electromagnetic_radiation>

[7] PLING.ORG. *Computer Graphics and Visualisation.* [online]. [cit. 28.12.2010]. Dostupné na URL: <http://www.pling.org.uk/cs/cgv.html>

[8] WIKIPEDIE. *Fotografický film.* [online]. [cit. 27.12.2010]. Dostupné na URL: <http://cs.wikipedia.org/wiki/Fotografick%C3%BD_film>

[9] PEDERSEN, D. *The RGB Color Wheel*. [online]. [cit. 28.12.2010]. Dostupné na URL: <http://blulob.com/2009/03/08/the-rgb-color-wheel/>

[10] SAS INSTITUTE. *Defining Colors using Hex Values.* [online]. [cit. 29.12.2010]. Dostupné na URL: <http://support.sas.com/techsup/technote/ts688/ts688.html>

[11] ACA SYSTEMS. *What is HSB/HSV Color Spaces?* [online]. [cit. 29.12.2010]. Dostupné na URL: <http://www.acasystems.com/en/color-picker/faq-hsb-hsv-color.htm>

[12] E-LEARNING. *Digitalizace obrazu.* [online]. [cit. 29.12.2010]. Dostupné na URL: <http://e-learning.tul.cz/cgi-bin/elearning/elearning.fcgi?ID_tema=67&ID_obsah=1176&stranka=publ_tema&akce=polozka_vstup>

[13] KOMRSKA. *Vzorkovací teorém.* [online]. [cit. 2.1.2011]. Dostupné na URL: <http://physics.fme.vutbr.cz/~komrska/Fourier/KapF14.pdf>

[14] SENSAGENT. *Shannonův teorém*. [online]. [cit. 2.1.2011]. Dostupné na URL: <http://others.sensagent.com/shannonuv+teorem/cs-cs/>

[15] AUER, K.; NORRIS, T. *„Arrieros Alife“ a Multi-Agent Approach Simulating the Evolution of a Social System: Modeling the Emergence of Social Networks with „Ascape“*. [online]. Journal of Artificial Societies and Social Simulation. Vol. 4. No. 1. [cit. 2.1.2011]. Dostupné na URL: <http://jasss.soc.surrey.ac.uk/4/1/6.html>

[16] THOMPSON, K. *Taxicab Geometry. The basics.* [online]. [cit. 4.1.2011]. Dostupné na URL: <http://taxicabgeometry.net/general/basics.html>

[17] ASK.COM. *Taxicab Geometry.* [online]. [cit. 4.1.2011]. Dostupné na URL: <http://www.ask.com/wiki/Taxicab_geometry>

[18] SOCIETY OF ROBOTS. *Programming – Computer Vision Tutorial.* [online]. [cit. 3.1.2011]. Dostupné na URL: <http://www.societyofrobots.com/programming_computer_vision_tutorial_pt2.shtml>

[19] HLAVAC, V. *TeachPres.Cz.* [online]. ČVUT. [cit. 3.1.2011]. Dostupné na URL: <http://cmp.felk.cvut.cz/~hlavac/TeachPresCz/11DigZprObr/>

[20] DOUBEK, P. *Mean-Shift segmentace.* [online]. CMP FEL ČVUT Praha. [cit. 5.1.2011]. Dostupné na URL: <http://cmp.felk.cvut.cz/cmp/courses/ZS1/Cviceni/cv4/meanshift.pdf>

[21] TUZEL1. *Mean Shift Clustering.* [online]. [cit. 5.1.2011]. Dostupné na URL: <http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/TUZEL1/MeanShift.pdf>

[22] ŠPANĚL, M. *Klasifikace a rozpoznávání.* [online]. Prezentace přednášek 2009. Ústav počítačové grafiky a multimédií. FIT VUT v Brně. [cit. 6.1.2011]. Dostupné na URL: <http://www.fit.vutbr.cz/study/courses/IKR/public/prednasky/08_clustering/IKR_Clustering.pdf>

[23] HRADIŠ, M. *Detekce v obraze a AdaBoost.* [online]. Počítačové vidění. Prezentace přednášek. UPGM FIT BRNO University of Technology. [cit. 6.1.2011]. Dostupné na URL: <http://www.google.cz/url?sa=t&source=web&cd=8&ved=0CEUQFjAH&url=http%3A%2F%2Fweb.lwg.cz%2Fzakazky%2Fmanocentrum%2F%3F_site%3Dmanocentrum_web%26_file%3D%2Fwebsite%2Fmainmenu%2Fproducts%2Ftlakomery%2Fmanomersk%2Fstandard-kov%2Ftest%2Fpov_03_detekce_objektu_-_adaboost.pdf%26_op%3DdownloadFile&rct=j&q=adaboost%20vut&ei=8o4zTbjdMtHoOaGnuLYC&usg=AFQjCNEMdpIauRcI5ingBgZilj5uJcMzWw&cad=rja>

[24] PAALANEN, P.; KÄMÄRÄINEN, J.; KÄLVIÄINEN, H. *Gaussian Mixture Model Methods.* [online]. LAPPEENRANTA UNIVERSITY OF TECHNOLOGY. [cit. 7.1.2011]. Dostupné na URL: <http://www2.it.lut.fi/project/gmmbayes/>

[25] ZEMČÍK, P. *HDR obrazy (Přednáška)* FIT VUT 2010 [cit. 8.1.2011].

[26] SYNAPTIC. *The talented Dr Hebb, Part 2, PCA.* [online]. [cit. 14.4.2011]. Dostupné na URL: <http://blog.peltarion.com/2006/06/20/the-talented-drhebb-part-2-pca/>

[27] GARCIA, CH.; TZIRITAS, G. *Face Detection Using Quantized Skin Color Regions Merging and Wavelet Packet Analysis.* IEEE TRANSACTIONS ON MULTIMEDIA, Vol. 1, No. 3, September 1999. [online]. [cit. 5.4.2011]. Dostupné na URL: <http://www.google.com/url?sa=t&source=web&cd=3&ved=0CCgQFjAC&url=http%3A%2F%2Fciteseer.ist.psu.edu%2Fviewdoc%2Fdownload%3Bjsessionid%3DFF395C81A6A990DC94F30B95A2DDA521%3Fdoi%3D10.1.1.25.9044%26rep%3Drep1%26type%3Dpdf&rct=j&q=C.%20Garcia%20and%20G.%20Tziritas.%20Face%20Detection%20Using%20Quantized%20Skin%20Color%20Region%20Merging&ei=YjO3Tae1DcSBOq_s8YwP&usg=AFQjCNFCF21XPFqcLjUWEqSU_tGAr2qgJQ&cad=rja>

[28] PATWARDHAN, K.S.; ROY, S.D. *Dynamic Hand Gesture Recognition using Predictive Eigen Tracker*. [online]. [cit. 20.4.2011]. Dostupné na URL: <http://www.cse.iitd.ernet.in/~sumantra/publications/icvgip04_gesture.pdf>

[29] BILLINGHURST, M. *Gesture based interaction*. [online]. [cit. 23.4.2011]. Dostupné na URL: <http://www.billbuxton.com/input14.Gesture.pdf>

[30] HLAVÁČ, V. *Rozpoznávání s Markovskými modely*. [online]. [cit. 23.4.2011]. Dostupné na URL: <http://cmp.felk.cvut.cz/~hlavac/TeachPresCz/31Rozp/61MarkovianPR.pdf>

[31] O'DONOVAN, P. *Static Gesture Recognition with Restricted Boltzmann Machines*. [online]. [cit. 30.4.2011]. Dostupné na URL: <http://www.dgp.toronto.edu/~donovan/gesture/report.pdf>

[32] ŠMÍD, R. *Úvod do vlnkové transformace*. 2001. [online]. [cit. 2.5.2011]. Dostupné na URL: <http://measure.feld.cvut.cz/usr/staff/smid/wavelets/Wavelet-intro8859.pdf>

[33] HRADIŠ, M. *Boosting*. Klasifikace a rozpoznávání. [online]. [cit. 2.5.2011]. Dostupné na URL: <http://www.fit.vutbr.cz/study/courses/IKR/public/prednasky/07_adaboost/IKR-Boosting.pdf>

[34] VERMA, B.; BLUMENSTEIN, M. *Pattern Recognition Technologies and Applications*. IGI Global. 2008. 435p. ISBN-13:978-1-59904-807-9.

[35] FREUND, Y.; SCHAPIRE, R.E. *A Short Introduction to Boosting*. Journal of Japanese Society for Artificial Intelligence, 14(5):771-780, 1999. [online]. [cit. 3.5.2011]. Dostupné na URL: <http://www.google.cz/url?sa=t&source=web&cd=8&sqi=2&ved=0CGYQFjAH&url=http%3A%2F%2Fciteseerx.ist.psu.edu%2Fviewdoc%2Fdownload%3Fdoi%3D10.1.1.93.5148%26rep%3Drep1%26type%3Dpdf&rct=j&q=boosting&ei=urTWTYagBIOWOrXj3bUH&usg=AFQjCNGWN_mj1JWxsqGEFXcKAL-VQvap7Q&cad=rja>

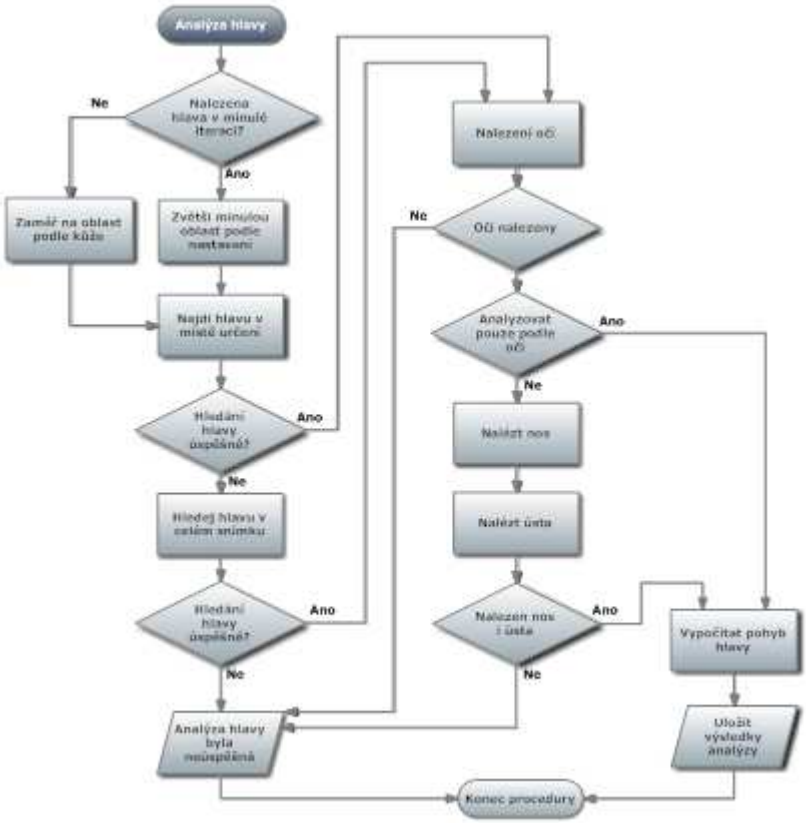
[36] SMITH, L.I. *A tutorial on Principal Components Analysis*.2002. [online]. [cit. 15.5.2011]. Dostupné na URL: <http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf>

[37] OpenCV REFERENCE MANUAL. December 2010.

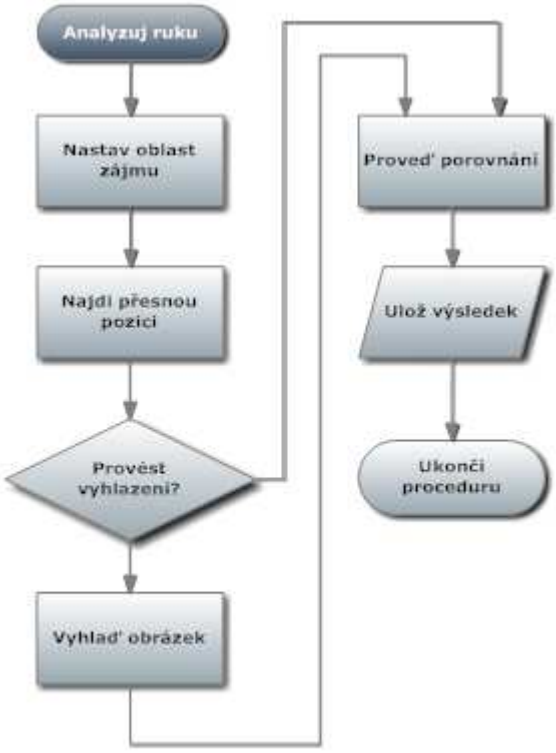
[38] BRADSKI, G.; KAEBLER, A. *Learning OpenCV*. O'Reilly, 2008. 576 p. ISBN: 978-0-596-51613-0

Seznam příloh

Příloha 1: Analýza hlavy



Příloha 2: Analýza ruky



Příloha 3: Ukázka konfiguračního souboru settings.xml

```
<?xml version="1.0" ?>
- <opencv_storage>
    <Maska_hledanych_gest>7</Maska_hledanych_gest>
    <Metoda_hledani_kuze>1</Metoda_hledani_kuze>
    <Rychlost_pohybu_mysi>30</Rychlost_pohybu_mysi>
    <Uroven_podvzorkovani>1</Uroven_podvzorkovani>
    <Zvetseni_okna_predikce>20</Zvetseni_okna_predikce>
    <Sirka_ruky>192</Sirka_ruky>
    <Vyska_ruky>192</Vyska_ruky>
    <Oblast_hlavy_sirka>300</Oblast_hlavy_sirka>
    <Oblast_hlavy_vyska>260</Oblast_hlavy_vyska>
    <Minimalni_sirka_hlavy>48</Minimalni_sirka_hlavy>
    <Minimalni_vyska_hlavy>48</Minimalni_vyska_hlavy>
    <Maximalni_sirka_hlavy>160</Maximalni_sirka_hlavy>
    <Maximalni_vyska_hlavy>160</Maximalni_vyska_hlavy>
    <Velikost_neplatneho_pohybu>40</Velikost_neplatneho_pohybu>
    <Minimalni_pohyb_hlavy>3</Minimalni_pohyb_hlavy>
    <Uroven_vyhlazeni>15</Uroven_vyhlazeni>
  </opencv_storage>
```

Příloha 4: Ukázka konfiguračního souboru priority.xml

```
<?xml version="1.0" ?>
<opencv_storage>
  <intFirst2Dcombination>6</intFirst2Dcombination>
  <intSecond2Dcombination>5</intSecond2Dcombination>
  <intThird2Dcombination>3</intThird2Dcombination>
  <intFirst1Dcombination>4</intFirst1Dcombination>
  <intSecond1Dcombination>2</intSecond1Dcombination>
  <intThird1Dcombination>1</intThird1Dcombination>
</opencv_storage>
```

Pozn.: 6 znamená binárně $4 + 2 = 1*2^2 + 1*2^1 + 0*2^0$ což představuje levou a pravou ruku.

Příloha 5: Ukázka konfiguračního souboru gesta.xml

```
<?xml version="1.0" ?>
<opencv_storage>
  <Gesto1D_Left_1>772</Gesto1D_Left_1>
  <Gesto1D_Left_2>773</Gesto1D_Left_2>
  <Gesto1D_Left_3>770</Gesto1D_Left_3>
  <Gesto1D_Left_4>771</Gesto1D_Left_4>
</opencv_storage>
```

Pozn.: Popis hodnot i tagů je možné nalézt v souboru gesta.xml