VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

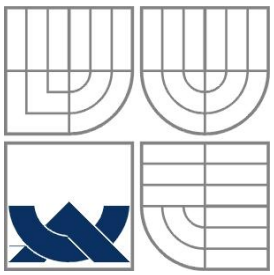
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

3D MONITOR POMOCÍ DETEKCE POZICE HLAVY

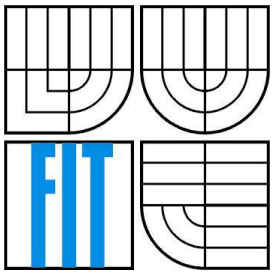
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. Jan Zivčák
AUTHOR

BRNO 2011



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

3D MONITOR POMOCÍ DETEKCE POZICE HLAVY

3D MONITOR BASED ON HEAD POSE DETECTION

DIPLOMOVÁ PRÁCE
MASTER‘S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Jan Zivčák

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Vítězslav Beran , Ph.D.

BRNO 2011

Abstrakt

S vývojem možností zpracování obrazu, stereoskopického zobrazení, cen webových kamer a výkonu počítačů vyvstala možnost jak znásobit zážitek uživatele během práce s 3D programy. Z obrazu z webové kamery lze odhadnout polohu uživatelovy hlavy a podle této polohy natočit trojrozměrnou scénu zobrazovanou na monitoru počítače. Uživateli se potom při pohybu hlavy bude zdát, jako by byl monitor okno, skrze které může nahlížet na scénu za ním. Pomocí systému, který je výsledkem této práce, bude možné jednoduše a levně dodat uvedené chování libovolnému 3D programu.

Abstract

With the development of posibilities of image processing, stereoscopy, prices of web cameras and power of computers an opportunity to multiply an experience with working with 3D programs showed. From the picture from webcamera an estimation of a pose of user's head can be made. According to this pose a view on 3D scene can be changed. Then, when user moves his head, he will have a feeling as if monitor was a window through which one can see the scene behind. With the system which is the result of this project it will be possible to easily and cheaply add this kind of behaviour to any 3D application.

Klíčová slova

3D monitor, detekce pozice hlavy, detekce rohových bodů, tracking, Viola-Jones, KLT

Keywords

3D monitor, head pose detection, corner detection, tracking, Viola-Jones, KLT

Citace

Zivčák Jan: 3D monitor pomocí detekce pozice hlavy, diplomová práce, Brno, FIT VUT v Brně, 2011

3D monitor pomocí detekce pozice hlavy

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Vítězslava Berana, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jan Zivčák
25. května 2011

Poděkování

Chtěl bych poděkovat svému vedoucímu Ing. Vítězslavu Beranovi, Ph.D.
za odborné vedení této práce.

Obsah

1 Úvod

Ovládání počítače je pro současného člověka každodenní činností. Počítače provází většinu lidí při širokém okruhu činností – při zábavě, hledání informací, práci, komunikaci atd. Pro usnadnění práce na počítači a tedy i zpříjemnění spousty času, který lidé s počítačem tráví, je čím dál tím víc kladen důraz na uživatelské rozhraní – na způsob jakým uživatel s počítačem komunikuje. Vypadá to, že klasické prostředky jako myš a klávesnice jsou už dávno překonány. Počítače lze ovládat více způsoby, než tomu bylo doposud. Dotykové displeje už jsou v současné době téměř standardem. Ovládání hlasem je podrobeno intenzivnímu výzkumu, je součástí mnoha projektů a navzdory tomu, že ještě neexistují bezproblémově použitelná řešení, je to oblast velice slibná.

Avšak obrovská pozornost je v současné době věnována ovládání gesty. Téměř každý nový mobilní telefon obsahuje akcelerometry, jež jsou schopny zjistit přesnou polohu daného přístroje. Existuje spousta programů, jež těchto informací dokáže využívat pro ovládání telefonu gesty. Pro ovládání osobních počítačů pomocí gest existuje také několik řešení, ovšem ta nejrozšířenější a nejúspěšnější z nich vyžadují speciální přídavný hardware.

Další oblast uživatelského rozhraní počítačových systémů, které se v současné době dostává velké pozornosti, je stereoskopické zobrazení. Toho využívá velké množství nově vznikajících filmů a už tak přestává být pouze dominantou filmových pláten, ale postupně proniká i do počítačových monitorů a dokonce i do mobilních telefonů.

Cílem této práce bylo vytvořit systém, který za pomoci obyčejné webové kamery sleduje pohyb uživatele a podle polohy jeho hlavy umožňuje natočit trojrozměrnou (3D) scénu libovolného programu. Uživateli se potom při pohybu hlavy zdá, jako by byl monitor okno, skrze které může nahlížet na scénu za ním. V kombinaci se stereoskopickým zobrazením by takový systém mohl nabídnout vskutku pravý 3D zážitek.

Výsledkem práce je programová knihovna, která umožňuje takové rozšíření přidat do libovolné 3D aplikace. Byl kladen důraz na to, aby knihovna byla snadno použitelná, s velmi jednoduchým nastavením a s co nejmenší nutností uživatelovy interakce. Ve výsledku uživatel může k počítači s běžící aplikací přistoupit a bez nutnosti kalibrace či inicializace se začít rozhlížet. To umožňuje použít systém i na místech, kde uživatel nemá přístup k ovládacím prvkům – např. výstavní síně, výlohy obchodů či jiné prezentace.

Jelikož se očekává, že velká část aplikací, které by takovou knihovnu mohly využívat, budou počítačové hry, simulace či jiné aplikace náročné na výkon počítače, byl kladen velký důraz na optimalizaci knihovny.

Kvůli testování funkčnosti systému a k jeho prezentaci vznikla testovací aplikace, která využívá systém k zobrazení jednoduché trojrozměrné scény.

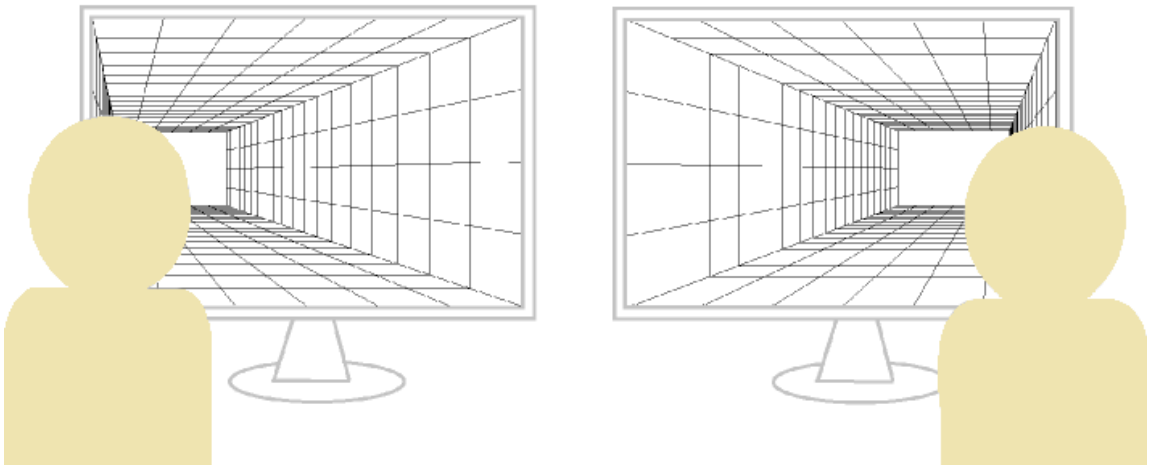
Samotné testování systému mělo za úkol zjistit přesnost a omezení systému. Dále byly provedeny testy zaměřující se na výkonnost systému a v neposlední řadě byl systém představen několika uživatelům k zjištění jejich odezvy.

V tomto dokumentu budou představeny teoretické podklady důležité pro tvorbu takového systému, návrh celého systému a také popis implementace a testování. Dokument navazuje na semestrální projekt, z něhož byla převzata převážně celá druhá kapitola.

2 Specifikace problému a jeho praktická a teoretická východiska

Aby bylo možné zobrazit 3D scénu na monitoru počítače tak, aby měl uživatel při pohybu pocit, že se dívá skutečně na 3D scénu a ne pouze na její dvojrozměrnou projekci, je potřeba znát co možná nejpřesněji polohu uživatelovy hlavy a očí vzhledem k monitoru počítače. Podle této polohy je pak natočena zobrazovaná 3D scéna – viz obrázek 1.

Zjistit polohu uživatelovy hlavy je tedy hlavním problémem, který je třeba vyřešit při realizaci 3D monitoru. V následující kapitole bude tento problém upřesněn, budou stručně popsána již existující průmyslová řešení a bude uveden seznam a popis metod, které je možné použít pro řešení tohoto problému. Dále budou detailněji uvedena teoretická východiska dílčích problémů týkajících se jak detekce polohy hlavy, tak samotného principu 3D zobrazovaní. Důvod uvedení řešení těchto dílčích problémů, jejich využití a způsob kombinace naleznete v kapitole 3.

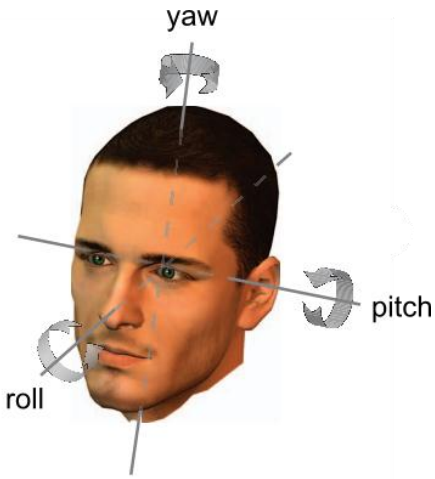


Obrázek 1 – Problém 3D monitoru nastíněný graficky – při změně pohybu hlavy se mění i obraz na monitoru.

2.1 Odhad polohy hlavy

Pro kompletní popis polohy hlavy je potřeba znát jednotlivé stupně volnosti (degrees of freedom). Pro trojrozměrný prostor rozlišujeme šest stupňů volnosti:

- posunutí podél osy X
- posunutí podél osy Y
- posunutí podél osy Z
- rotaci kolem osy X (pitch)
- rotaci kolem osy Y (yaw)
- rotaci kolem osy Z (roll)



Obrázek 2 - Stupně volnosti hlavy (zdroj: [16])

Odhad těchto veličin lze rozdělit na odhad posunutí a na odhad rotací. Pro fungovaní 3D monitoru je potřeba znát minimálně hodnoty posunutí. Znalost orientace hlavy není rozhodující, ale lze jí využít pro přesnější odhad pozice nebo pro různá rozšíření.

Hodnoty posunutí a orientace se vztahují k určitému bodu v reálném světě. Zpravidla tímto bodem bývá samotná kamera či jiný systém, který hlavu snímá.

2.2 Průmyslová řešení detekce polohy hlavy

Jak již bylo nastíněno v úvodu, řešení problému detekce polohy hlavy má v současné době několik úspěšných průmyslových řešení. Většina z nich se týká oblastí počítačových her a především herních konzolí.

Po úspěchu herní konzole Wii [11], která těží právě z ovládání gesty, se i další výrobci konzolí zaměřily právě na tento fenomén. Wii používá k detekci gest dvě zařízení (Sensor Bar a Wii Remote) a využívá kombinaci akcelerometrů a infračerveného signálu. I když jsou tato zařízení určena k detekci pohybu rukou, objevilo se spoustu amatérských projektů využívajících Wii k mnohem rozmanitějším účelům. Jeden z nich využil Wii právě k detekci hlavy a vytvoření iluze 3D monitoru. Snadno tak ukázal, jak působivý tento efekt může být a stal se tím inspirací pro ostatní [14].

Firma Microsoft zareagovala na úspěch Wii systémem Kinect pro herní konzoli Xbox 360 [12]. Tento vskutku revoluční systém umožňuje za pomoci dvou hloubkových senzorů rozpoznávat jakýkoliv pohyb lidského těla. I když je Kinect oficiálně podporován pouze konzolí Xbox 360, již se objevily neoficiální způsoby, jak tento systém napojit na PC a tak jej naplno začít využívat.

Systém FaceAPI od firmy Seeing Machines na rozdíl od obou předchozích využívá k detekci polohy obličeje pouze standardní webové kamery [15]. FaceAPI navíc dokáže detekovat nejen přesnou polohu a orientaci hlavy v rozmezí $\pm 90^{\circ}$, ale dokonce i tři body na hraně každého z obočí a osm bodů po obvodu rtů. Takto lze tedy detekovat i obličejovou mimiku. Systém je navíc velice

robustní – je odolný vůči špatnému a nekonstatnímu osvětlení, variabilnímu vzhledu uživatele, nošení brýlí či plnovousu a také rychlému pohybu uživatele. Ačkoliv je systém pro nekomerční užití dostupný zdarma, jeho implementace není otevřená, a tudíž zůstává tajemstvím. FaceAPI tak může sloužit alespoň jako ideální cíl, ke kterému bych se rád se svým systémem přiblížil.

2.3 Metody pro detekci polohy hlavy

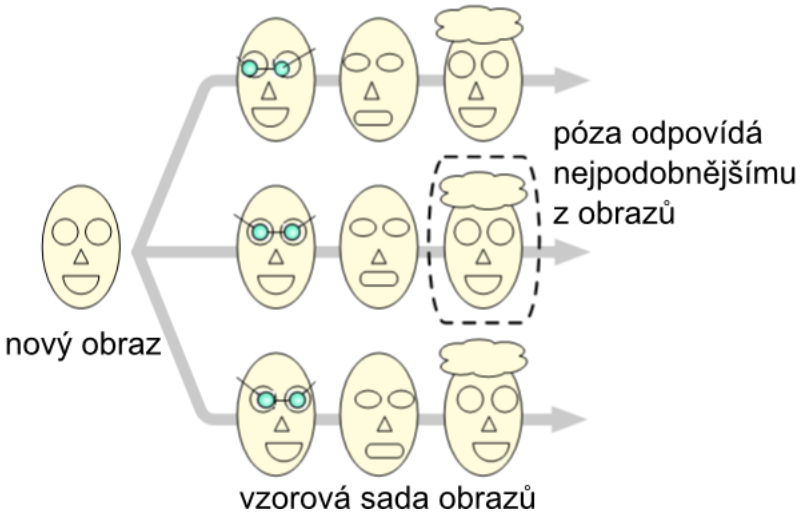
V této podkapitole bude uveden seznam metod používaných pro detekci polohy hlavy. U každé z nich bude nastíněn princip jejich činnosti a budou zmíněny jejich výhody či nevýhody.

2.3.1 Metody založené na vzorových obrazech

Tyto metody porovnávají vstupní obraz se sadou vzorových obrazů, u kterých je předem označeno v jaké poloze se hlava nachází. Jako výslednou polohu a orientaci metody dosadí hodnoty, jež se nachází u vzorového obrazu, který je nejvíce podobný obrazu vstupnímu. Činnost takových metod je znázorněna na obrázku 3.

Výhodou těchto metod je, že sadu vzorových obrazů, lze kdykoliv rozšířit či upravit. Pro činnost těchto metod není potřeba mít negativní příklady (obrazy s pozadím, kde se hlava nenachází). Metody lze použít pro vysoká i nízká rozlišení.

Mezi nevýhody patří diskrétní výstup polohy a orientace hlavy, který ale lze interpolovat. Pro správnou činnost je také třeba předpokládat, že sada vzorových obrazů obsahuje všechny možné polohy a orientace. Avšak i po splnění tohoto předpokladu může docházet k chybným výstupům, kdy je jako výstup označena hodnota u vzorového obrazu, který se sice nejvíce podobá vstupnímu obrazu, ale jeho orientace a poloha je značně odlišná. Tyto metody mohou být také velmi neefektivní, a to pokud je sada vstupních obrazů příliš rozsáhlá.



Obrázek 3 - Činnost metod založených na vzorových obrazech (zdroj: [16])

2.3.2 Pole detektorů

Přístup těchto metod je částečně podobný přístupu metod založených na vzorových obrazech. Opět máme anotovanou sadu vzorových obrazů, které se ale tentokrát nepoužívají přímo pro porovnávání se vstupním obrazem, ale jsou použity pro natrénování sady detektorů. Každý detektor je schopen rozpoznávat právě jednu polohu nebo orientaci hlavy. Výsledek je daný detektorem, který měl na vstupních datech nejlepší odezvu.

Výhodou těchto metod je, že je možné vynechat detekci hlavy v obraze, protože detektory jsou natrénovány i na negativních příkladech. Další výhodou oproti metodám založených na vzorových obrazech je, že nepodléhají chybě, která označí výsledek pouze na základě podobnosti vzhledu obličejů na obrazech, navzdory jejich rozdílné orientaci.

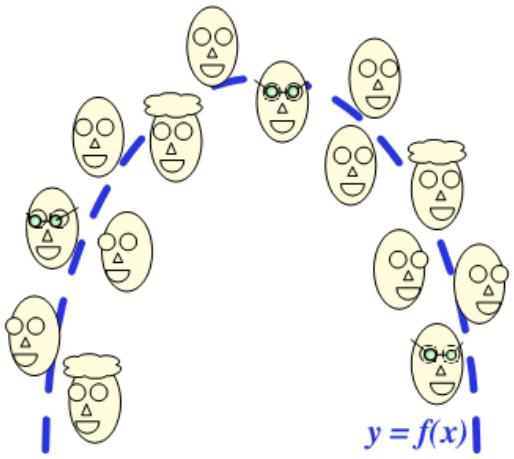
Mezi nevýhody těchto metod rozhodně patří náročnost na trénování sady detektorů, která stoupá s každým přidaným detektorem. Při vysokém počtu detektorů může být náročné i samotné klasifikování. V praxi se ukázalo, že pro real-time zpracování lze použít maximálně 12 detektorů, které odhadují jen jeden stupeň volnosti [16]. Vzhledem k tomu, že výstup jednotlivých detektorů bývá často pouze diskrétní, bude výstup sady jen těžko převeditelný na spojitý a navíc mohou vznikat nejednoznačnosti.

2.3.3 Metody založené na regresní analýze

Tyto metody odhadují polohu hlavy pomocí nelineárního mapování z obrazového prostoru přímo do jednotlivých stupňů volnosti. Princip činnosti je znázorněn na obrázku 4. Ze sady označných trénovacích dat se vytvoří model, který bude buď diskrétně, nebo spojitě dodávat výstupní hodnoty i pro neznámá vstupní data. Problémem těchto metod je vysoký počet rozměrů vstupního obrazu, který je možný snížit například metodou analýzy hlavních komponent – PCA (Principal Component Analysis) nebo pokud budeme předem znát polohu jednotlivých obličejových rysů. Používanými metodami jsou SVRs (Support Vector Regressors) [18] nebo vícevrstevné neuronové sítě MLP (Multi-Layer Perceptron).

Tyto metody mohou mít spoustu výhod – jsou relativně přesné a obzvláště výpočet výstupních hodnot může být velmi rychlý [16].Také nejsou kladeny vysoké požadavky na trénovací vstupní data.

Hlavní nevýhodou je, že tyto metody jsou velmi závislé na detekci hlavy v obraze, která musí výpočtu předcházet. Pokud je detekce hlavy nepřesná, jsou nepřesné i odhady orientace.



Obrázek 4 - Regresní metody (zdroj: [16])

2.3.4 Manifold metody

Jelikož počet rozměrů vstupního obrazu mnohonásobně převyšuje počet rozměrů polohy a orientace hlavy (šest stupňů volnosti), je vhodné převést vstupní obraz do jiného méně rozměrného prostoru (manifold, varieta). Po transformaci (vnoření) vstupních dat do méně rozměrného prostoru lze provést odhad polohy buď porovnáváním se vzorovými obrazy, nebo metodami regresní analýzy. Pro vnoření vstupních dat do méně rozměrného prostoru lze použít jakékoliv metody pro snižování dimenze. Používanými metodami jsou opět metody analýzy hlavních komponent - PCA (Principal Component Analysis) nebo KCPA (Kernel Principal Component Analysis). Avšak při použití těchto metod není zaručené, že hlavní komponenty budou odpovídat pouze změnám v orientaci hlavy. Mohou totiž opět odpovídat změnám ve vzhledu, což není vhodné. Pro odstranění tohoto problému je vhodné rozdělit trénovací data do skupin sdílejících shodnou orientaci hlavy a analýzu provádět na těchto skupinách odděleně. Takto ale zase nebudou výstupní data spojitá. Pro ta hrubý odhad je vhodnější použít metody LDA (Linear Discriminant Analysis) nebo KLDA (Kernel Linear Discriminant Analysis). Jako nejvhodnější se ukázaly metody Isomapping (Isometric feature mapping), LLE (Locally Linear Embedding) nebo LE (Laplacian Eigenmaps) [16].

Hlavním problémem všech těchto metod stále zůstává vliv podobnosti obličejů ve vstupních datech s obličeji v trénovacích datech – důležitá není podobnost ve vzhledu ale podobnost v orientaci.

2.3.5 Metody založené na flexibilních modelech

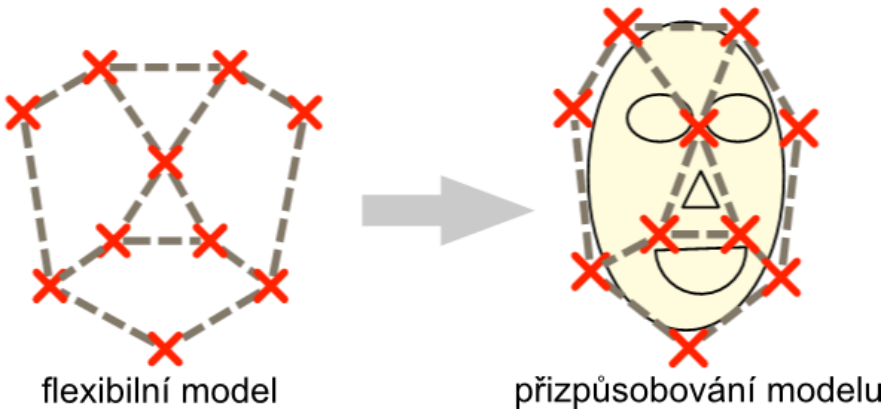
Všechny předchozí metody se na vstupní data dívají spíše jako na signál, který se snaží namapovat na konkrétní hodnotu polohy a orientace. Metody z této skupiny používají flexibilní modely (nazývané též grafy), které se snaží napasovat model na vstupní obraz tak, aby kopíroval strukturu obličejových

rysů - viz obrázek 5. Obličejové rysy mohou být například koutky očí a úst, špičky uší, nosní dírky atd. Na trénovacích datech těchto metod musí být označena nejen konkrétní orientace obličeje, ale i všechny používané obličejové rysy. Během trénovaní se pak na místech, kde se nachází obličejový rys, vypočítají lokálních deskriptory – používají se např. gabor-jets. Je vhodné získat těchto deskriptorů pro každý obličejový rys více, a to z obrazů různých lidí, čímž se dosáhne větší invariantnosti. Při vyhodnocování je graf umístěn na vstupní obraz a upravuje se tak, aby byl rozdíl rysů minimální – proces je znám pod zkratkou EGM (Elastic Graph Matching). Je nutné, aby byl pro každou orientaci hlavy vytvořen jeden graf. Tím se stane výstup diskrétní a jeho zjemnění lze dosáhnout jen přidáním více grafů. Výpočet se ale stane časově náročnější.

Active Appearance Model (AAM) a Active Shape Model (ASM) jsou další modely, které jsou podobným postupem využívané pro odhad polohy a orientace hlavy. Jsou to obecné modely používané pro popis tvaru libovolného objektu. ASM pracují pouze se strukturou objektu, která ale nemusí být pouze 2D - za využití metody Structure From Motion lze z několika snímků objektu z různých úhlů získat i 2.5D či 3D model [19]. AAM jsou navíc schopné pracovat i s texturou a dalšími vlastnostmi.

Celkově jsou metody založené na flexibilních modelech a obzvláště AAM schopné vykazovat velmi přesné výsledky.

Mezi nevýhody patří složitost a také to, že je vždy nutné znát polohu všech potřebných obličejových rysů, což nemusí být pokaždé možné.



Obrázek 5 – Přizpůsobování flexibilního modelu (zdroj: [16])

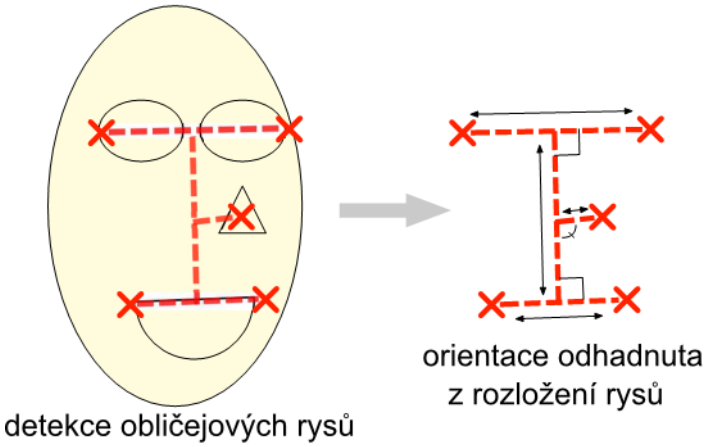
2.3.6 Geometrické metody

Tyto metody nepracují přímo s obrazem, ale vychází z předpokladu, že známe polohu obličejových rysů. Polohu obličejových rysů lze zjistit různými způsoby, viz předchozí kapitola nebo kapitola 2.4. U většiny metod je důležitá znalost antropometrie. Využít lze různé obličejové rysy a jejich různou kombinaci. Minimálně jsou však potřeba tři. Dále je důležité znát relativní vzájemnou polohu těchto rysů. Při výpočtech je důležitá znalost perspektivního zkreslení. Například je možné použít vnější a

vnitřní koutky očí a úst a špičku nosu [20]. Natočení okolo osy y lze vyčíst z rozdílu velikostí očí. Natočení okolo osy z lze zjistit jednoduše změřením úhlu, který svírá spojince koutků se spodní či horní hranou obrazu. Natočení okolo osy x lze zjistit porovnáním vzdálenosti špičky nosu od spojnice očí s antropometrickým modelem – viz obrázek 6.

Hlavní nevýhodou těchto metod je, že vyžadují znalost přesné polohy obličejových rysů, a že předem předpokládají určitou vzájemnou polohu obličejových rysů, která se však může obličej od obličeje mírně lišit.

Výhodu je, že nejsou příliš výpočetně náročné, nevyžadují žádné trénování (pokud nepočítáme trénování nutné k detekci obličejových rysů), a že jsou velmi intuitivní. Proto existuje spousta různých postupů, mezi které patří i ten můj – detailněji rozebraný v kapitole 3.



Obrázek 6 – Odhad orientace z geometrie obličejových rysů (zdroj: [16])

2.3.7 Trackovací metody

Tyto metody sledují (trackují) pohyb jednotlivých rysů mezi jednotlivými snímky a odhad provádí ze změn poloh těchto rysů. Důležité ale je, aby na začátku sledování proběhla inicializace, během které musí být hlava v předem předepsané poloze po určitou dobu - obvykle se jedná o přímý, vycentrovaný pohled do kamery. Pokud se během sledování ztratí stopa, je nutné provést inicializaci znovu. Dřívější postupy využívaly pro sledování pouze metodu nejmenších čtverců a pohyb celé hlavy odhadovaly ze slabé perspektivy. V současné době se pro sledování rysů používá velmi robustní metoda SIFT (Scale-Invariant Feature Transform). Pro odhad orientace je opět možné využít 3D modelu hlavy, který je možný získat v průběhu sledování, a poté najít takovou orientaci tohoto modelu, která nejlépe odpovídá současnému obrazu. Toho lze dosáhnout namapování textury získané během inicializace na model hlavy a po té porovnáváním vstupního obrazu a obrazů tohoto otexturovaného modelu v různých orientacích.

Výhodou těchto metod je, že také nevyžadují žádné trénování. Hlavně ale dokážou být velmi přesné, avšak pouze za předpokladu, že proběhla řádná inicializace. Další výhodou je, že metody pracují s modelem, který byl vytvořený za běhu a odpovídá přímo dané osobě.

Mezi nevýhody patří již zmíněná nutnost inicializace a také to, že metody jsou schopny pracovat jen se sekvencí snímků.

2.3.8 Hybridní metody

Již z předchozího textu je patrné, že se metody velmi prolínají. Hybridní metody používají kombinaci více různých metod k eliminování nedostatků dílčích metod. Často dochází ke kombinaci odhadu statické metody s trackovacím systémem, kdy statická metoda může sloužit k inicializaci, následné kontrole a případné reinicializaci trackování. Mezi další oblíbenou kombinaci patří kombinace trackovaní bodů s geometrickými metodami [16]. Do této kategorie spadá i mnou navrhovaný system.

2.4 Detekce obličeje a obličejových částí

Pro většinu z metod pro detekci polohy hlavy je nutné, nebo alespoň z hlediska optimalizace vhodné, znát oblast, kde se ve vstupním obraze nachází uživatelův obličej a jiné obličejové rysy. Z tohoto důvodu následuje výčet možností, jak lze detekovat obecně libovolné objekty v obraze:

- **Znalostní metody** používají pro detekci soupis znalostí a přesně definovaných pravidel, které detekovaný objekt má mít – např. rozmístění a velikost obličejových částí při detekci obličeje. Vzhledem k tomu, že i u jednoduchých objektů může být počet těchto pravidel a počet jejich kombinací vysoký, vede použití těchto metod k velice složitým algoritmům, které navíc budou použitelné pouze pro objekt, pro který byly sestrojeny.

- **Metody založené na neměnných rysech** využívají znalost základních rysů objektů, které jsou i pro jiné objekty z dané třídy neměnné. U detekce obličeje může jít například o barvu kůže, polohu nosu, očí, atd. Výhodou těchto metod je snadná implementace a univerzálnost. Do této skupiny patří i metoda Viola-Jones, která je vysvětlena dále.

- **Metody vzorových objektů** porovnávají detekovaný objekt s databází vzorových obličejů. Výhodou je snadná implementace, nevýhodou je závislost na velikosti a kvalitě databáze.

- **Metody založené na vzhledu** jsou podobné předcházející skupině. Rozdílem je, že z databáze vzorových objektů se vytvoří vhodným způsobem obecnější modely a až s těmi se detekovaný objekt porovnává.

2.4.1 Viola-Jones

Systém Viola-Jones je určen pro detekci objektů v obraze. Byl navržen a poprvé implementován Paulem Violou a Michaelem Jonesem v roce 2001 [2]. Systém je schopen podávat kvalitní výsledky v reálném čase [1]. Lze jej použít pro detekci různých objektů, ale osvědčil se především pro detekci obličeje a obličejových částí.

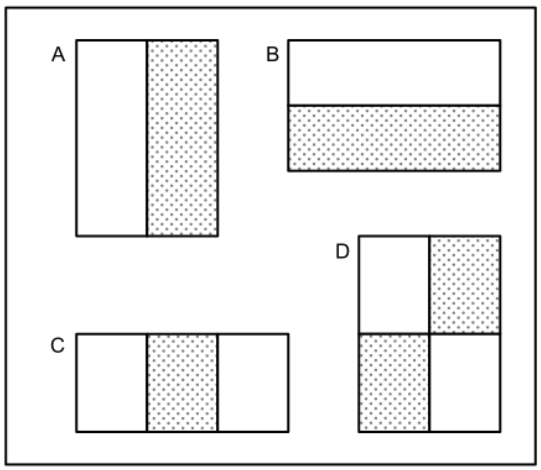
Systém je založen na kombinaci tří postupů. Prvním z nich je metoda používaná pro rychlý výpočet příznaků obrazu. Používají se 'Haar-like' příznaky, které mají svůj název odvozen od podobnosti s Haarovou vlnkou. Pro výpočet těchto příznaků je vhodná metoda integrálního obrazu, která dokáže příznaky spočítat v krátkém, konstantním čase nezávisle na velikosti či umístění. Příznaky jsou potřebné pro tvorbu klasifikátorů určujících, zda daný výřez původního obrazu obsahuje detekovaný objekt či nikoliv. Pro výběr nejdůležitějších příznaků (je jich mnohem více než pixelů v obraze) a pro vytvoření klasifikátorů a jejich natrénovaní je použita další podstatná část systému Viola-Jones – metoda AdaBoost. Třetím důležitým rysem je metoda pro seřazení složitějších klasifikátorů do kaskády, díky které se detekce zaměřuje pouze na slibné části obrazu a části, které určitě neobsahují hledaný objekt, jsou co nejdříve vyloučeny.

2.4.2 Příznaky vhodné pro klasifikaci

Pro konstrukci klasifikátorů je výhodnější použít místo hodnot pixelů příznaky. V systému Viola-Jones jsou použity příznaky, jež vycházejí z Haarovy vlnky. Příznaky se počítají ze součtů pixelů v obdélníkových oblastech obrazu. Tyto oblasti musí mít stejnou velikost i tvar a musí na sebe navazovat svislou nebo vodorovnou hranou.

Existují tři typy těchto příznaků. Nejjednodušší jsou dvouobdélníkové příznaky. Jejich hodnota je spočítána rozdílem mezi sumami pixelů pod dvěma obdélníky. Hodnota trojobdélníkového příznaku se spočítá ze sumy pixelů ve dvou krajních obdélnících odečtené od sumy pixelů ve středním obdélníku. Hodnota čtyřobdélníkového příznaku je dána rozdílem mezi sumami obdélníků ležících na diagonálách.

Počet těchto příznaků může být velmi vysoký. V obrázku o rozměru $24x24$, který má 576 pixelů se nachází více než 180 000 obdelníkových příznaků [1]. I přes to, že nakonec nejsou použity všechny tyto příznaky, by výpočet samotných příznaků byl za použití klasického postupu náročný. Pro daleko rychlejší výpočet příznaků se používá integrální obraz.



Obrázek 7 - Příklady příznaků

Suma pixelů ležících ve vyšrafovaných oblastech se odečte od sumy pixelů ležících v bílých oblastech. *A)* a *B)* jsou dvouobdelnékové příznaky, *C)* je trojobdelníkový a *D)* je čtyřobdelníkový příznak.

2.4.3 Integrální obraz

Integrální obraz umožňuje výpočet obdélníkových příznaků v konstantním čase [2]. Především při řešení problému mapování textur se používá integrálního obrazu pod pojmem tabulka součtů oblastí (summed area table).

Integrální obraz v bodě $[x, y]$ se rovná součtu všech pixelů ležících nalevo a nahoře od pixelu se souřadnicemi $[x, y]$ včetně tohoto pixelu:

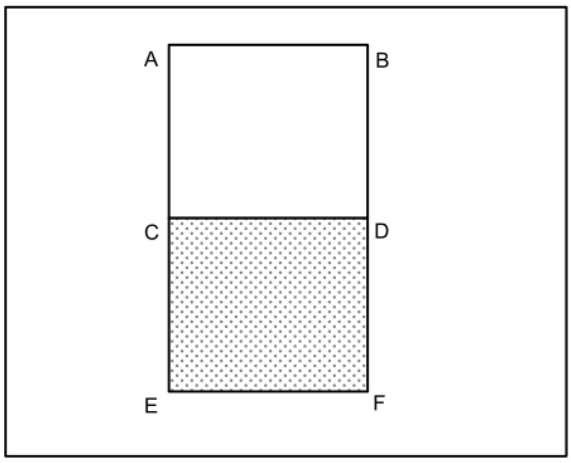
$$ii(x, y) = \sum_{x' \leq x,\ y' \leq y} i(x', y') \tag{1}$$

kde *ii(x,y)* značí integrální obraz v bodě a $i(x, y)$ originální obraz (hodnotu pixelu).

Pomocí následujících rekurentních vztahů lze integrální obraz spočítat v jednom průchodu obrazem:

$$\begin{aligned} s(x, y) &= s(x, y-1) + i(x, y) \\ s(x, -1) &= 0 \\ ii(x, y) &= ii(x-1, y) + s(x, y) \\ ii(-1, y) &= 0 \end{aligned} \tag{2}$$

Pokud máme integrální obraz jednou spočítán a uložen, lze dvouobdélníkový příznak spočítat pomocí šesti hodnot integrálního obrazu, trojobdélníkový pomocí osmi a čtyřobdélníkový pomocí devíti.



Obrázek 8 - Výpočet příznaku pomocí integrálního obrazu

Tento dvouobdélníkový příznak lze spočítat pomocí šesti hodnot integrálního obrazu, hodnota příznaku je

$(A + D) - (B + C) - ((C + F) - (D + E)) = A - B - 2C + 2D + E - F$

2.4.4 Učení klasifikátorů pomocí metody AdaBoost

Jelikož je obdélníkových příznaků i v malém obraze velice velké množství, které několikanásobně přesahuje počet pixelů v obraze, není možné využít pro klasifikaci všechny tyto příznaky. Je potřeba zvolit několik málo důležitých příznaků. Metoda AdaBoost se v systému Viola-Jones používá jak pro výběr těchto několika příznaků, tak pro samotné učení klasifikátorů.

Obecně metoda AdaBoost umožňuje sestrojit z několika slabých klasifikátorů jeden silný klasifikátor. Na slabé klasifikátory je kladena pouze jediná podmínka – jejich chyba nesmí překročit hodnotu 0.5. V případě systému Viola-Jones každému slabému klasifikátoru odpovídá jeden obdélníkový příznak.

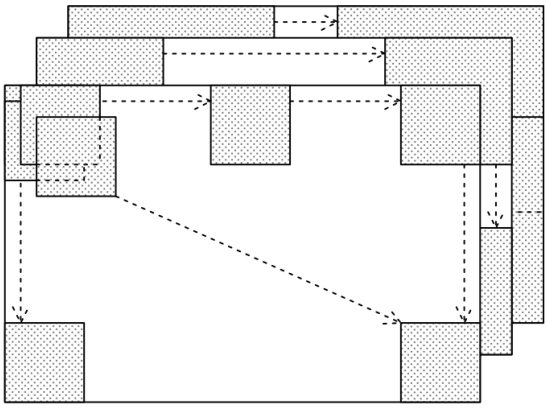
Celý průběh metody AdaBoost potom probíhá na trénovacích vzorcích cyklicky po kolech, dokud není splněna podmínka ukončení (pokud proběhl určitý počet kroků nebo chyba klesla pod určitou mez). Při inicializaci jsou každému trénovacímu vzorku nastaveny váhy. Tyto váhy postupně stoupají těm vzorkům, které se dlouhodobě nedaří správně klasifikovat. V každém kroku pak proběhne učení klasifikátorů a do finálního klasifikátoru je vybrán ten dílčí klasifikátor, který měl na váhovaných trénovacích vzorcích nejmenší chybu.

V praxi neexistují jednoduché klasifikátory, které by byly schopny pracovat s nízkou chybou. Obvyklé hodnoty chyb příznaků vybíraných v prvních fázích jsou mezi 0.1 a 0.3, v pozdějších fázích chyby vybíraných příznaků ještě více stoupají a pohybují se mezi 0.4 a 0.5 [1].

Obrázek 9 – První a druhý příznak vybraný metodou AdaBoost pro detekci obličeje
První příznak porovnává jas v oblasti očí s jasem v oblasti lící, jelikož oblast očí bývá v obličeji tmavší. Podobně druhý příznak porovnává jas v oblasti očí s jasem v oblasti kořene nosu.

2.4.5 Řazení klasifikátorů do kaskády

Podle dosavadních poznatků lze sestrojit klasifikátor, který je schopen rozhodnout, zda daný výřez obrazu obsahuje detekovaný objekt či nikoliv. Avšak takový klasifikátor není schopen najít objekt v celém původním obraze. Banálním řešení by bylo vzít všechny možné výřezy původního obrazu a přiložit je na vstup klasifikátoru.



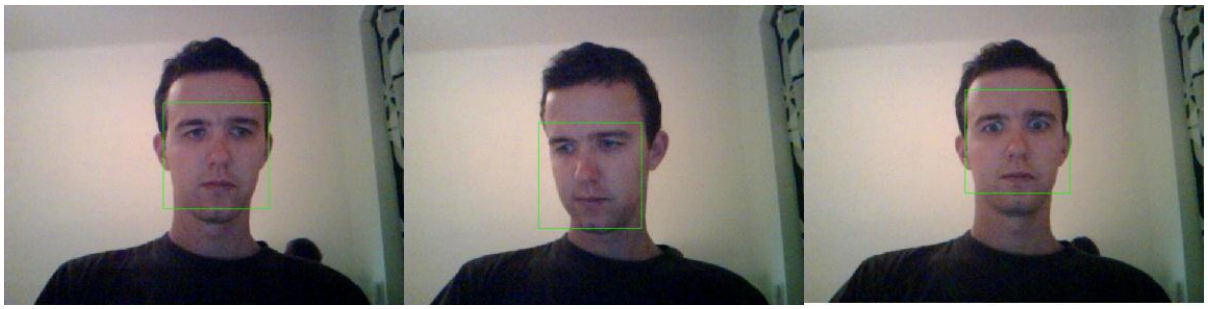
Obrázek 10 – Všechny možné výřezy jednoho obrazu

Jelikož je ale počet všech výřezů i v malém obraze vysoký, bylo by takové řešení výpočetně značně náročné. Systém Viola-Jones přichází s řešením v podobě kaskády klasifikátorů. Klasifikátory na začátku kaskády jsou jednodušší a jejich úkolem je vyloučit z dalšího zpracování co nejvíce výřezů, které neobsahují obličej (klasifikátory by měly mít co nejmenší *false positive rate*). Avšak neměly by vyloučit žádný výřez, který obličej obsahuje (měly by mít pokud možno nulovou *false negative rate*). Takové klasifikátory vzniknou natrénovaním pomocí metody AdaBoost a dodatečným upravením prahu. Podle [2] lze pro tvorbu prvního klasifikátoru kaskády použít pouze dva příznaky shodné s příznaky na obrázku 9, který na validační části trénovacích dat správně detekuje 100% obličejů s false positive rate 40%.

2.5 Sledování polohy objektů

Detektor objektů používající metodu Viola-Jones podává značně nestabilní výsledky – udávané pozice a velikosti objektů se liší i při minimální změně vstupního obrazu. Proto lze po detekci objektů zjišťovat jejich polohu v dalších krocích jinými prostředky. Pokud už polohu objektu jednou známe, není potřeba ji v dalším kroku hledat celou od znova. Jeho polohu v dalších snímcích lze zjistit sledováním. Sledování objektů v obraze je jeden z důležitých problémů počítačového vidění pro který existuje dostatek řešení.

Existují dvě skupiny metod ke sledování pohybujících se objektů. První z nich jsou založeny na optickém toku, který reprezentuje směr a rychlost pohybu každého bodu v obraze. Do druhé skupiny patří metody založené na sledování významných bodu v obraze (features).



Obrázek 11 – Nestabilita metody Viola-Jones
I při malých změnách vstupního obrazu jsou pozice a velikosti detekovaných obličejů různé.

Pro náš systém se hodí metody z druhé skupiny. Pro tyto metody je potřeba nejprve najít v obraze významné body. To jsou body, které se významně liší od svého okolí. Jejich hledání bude probíhat pro každý obličejový rys zvlášť a to v oblasti, kde byl rys detekován. Pro hledání lze použít nějakého hranového detektoru. Mezi nejpoužívanější patří Harris-Stephensův detektor nebo metoda Good Features to Track, jejímiž autoři jsou Jianbo Shi a Carlo Tomasi.

2.5.1 Harris-Stephensův detektor

Tento detektor vychází z Moravcova hranového detektoru. Hlavním nedostatkem Moravcova detektoru je, že není izotropní – detekce významného bodu závisí na orientaci jeho okolí. Tuto chybu Harris-Stephensův eliminuje.

Detektor nejprve vypočítá autokorelační matici pro každý bod snímku. Jedná se o Hessovu matici druhých parciálních derivací:

$$A = \begin{bmatrix} \langle I_x^2 \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I_y^2 \rangle \end{bmatrix} \tag{3}$$

kde lomené závorky značí konvoluci oknem:

$$\langle X \rangle = \sum_u \sum_v w(u,v) X(u,v) \tag{4}$$

Pokud je použito okno s kruhovými vahami - např. Gaussovo:

$$w(u,v) = e^{\frac{-(u^2+v^2)}{2\sigma^2}} \tag{5}$$

bude odezva izotropní [5].

Parciální derivaci lze aproximovat konvolucí s nějakým hranovým operátorem – např. Sobelovým:

$$I_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * I \tag{6}$$

$$I_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * I$$

který je použit v knihovně OpenCV [3].

Zkoumáním vlastních čísel autokorelační matice lze pro každý bod rozhodnout, zda je či není významný. Vlastní čísla lze získat ze vztahu:

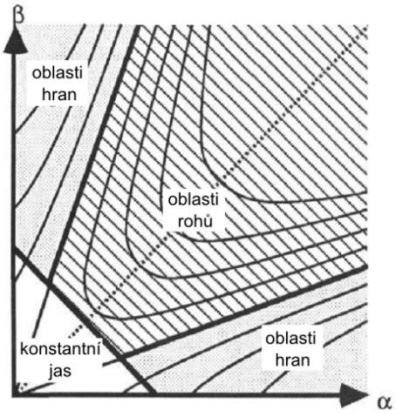
$$\lambda_{1,2} = \frac{\langle I_x^2 \rangle + \langle I_y^2 \rangle \pm \sqrt{(\langle I_x^2 \rangle - \langle I_y^2 \rangle)^2 + 4\langle I_x I_y \rangle \langle I_x I_y \rangle}}{2} \tag{7}$$

Pokud jsou obě vlastní čísla kladná a vysoká, jedná se o významný, rohový bod. Bod ležící na hraně dvou oblastí s různým jasem má jedno vlastní číslo kladné a vysoké, druhé se přibližně rovná nule. Pokud jsou obě vlastní čísla téměř nulová, jedná se o bod ležící v oblasti s konstantním jasem [5].

Jelikož je výpočet vlastní čísel relativně náročný – obsahuje odmocninu – rozhodování o významnosti bodu se provádí podle následujícího vztahu pro míru rohovitosti:

$$R = \lambda_1 \lambda_2 - \kappa(\lambda_1 + \lambda_2)^2 = \det(A) - \kappa\, trace^2(A) \tag{8}$$

kde κ je empiricky zjištěná hodnota, vhodné jsou hodnoty z intervalu $0.04 - 0.15$. Vlastní čísla se tedy nemusí počítat, stačí pouze determinant a stopa autokorelační matice. Bod se označí jako významný, pokud míra rohovitosti přesáhne zvolený práh.



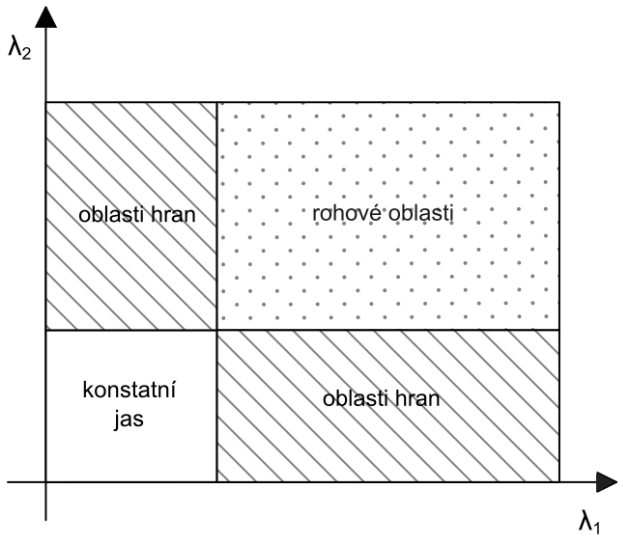
Obrázek 12 – Funkce závislosti míry rohovitosti na vlastních číslech autokorelační matice (α,β) (zdroj: [5])

2.5.2 Shi-Tomasi detektor

Shi-Tomasi hranový detektor vychází z Harris-Stephensova detektoru. Jedinou změnou je výpočet míry rohovitosti, jež přímo používá hodnoty vlastních čísel autokorelační matice. Ty lze získat ze vztahu (7). V [6] autoři ukázali, že mnohem lépe rohovitost bodu popisuje jednoduchý vztah:

$$R = \min(\lambda_1, \lambda_2) \tag{9}$$

Pro označení bodu za významný tudíž stačí, aby obě vlastní čísla jeho autokorelační matice byla vyšší než prahová hodnota.



Obrázek 13 – Závislost míry rohovitosti na vlastních číslech autokorelační matice

2.5.3 Sledování pomocí KLT

KLT tracker patří mezi nejpoužívanější metody pro sledování objektů v obraze. Název je odvozen od počátečních písmen jmen autorů – Kanade, Lucas a Tomasi. Z těchto jmen lze poznat, že tato metoda je založena na používání významných bodů z detektoru Shi-Tomasi.

Problém sledování významných bodů lze ukázat a vyřešit v jednorozměrném prostoru podle [7]. Máme-li jednorozměrnou funkci $F(x)$ a $G(x)$, kde funkce $G(x)$ byla nějakým neznámým způsobem posunuta, cílem je najít posunutí d tak, aby platilo:

$$F(x + d) = G(x) \tag{10}$$

Za využití derivace funkce $F(x)$, lze hodnotu $F(x + d)$ aproximovat jako:

$$F(x + d) \approx F(x) + dF'(x) \tag{11}$$

Chyba daná odhadem d pomocí rozdílu čtverců je dána vztahem:

$$E = \sum_x (F(x + d) - G(x))^2 \tag{12}$$

Cílem je tuto chybu minimalizovat, proto položíme:

$$0 = \frac{\partial E}{\partial d} \tag{13}$$

To lze s využitím vztahu (11) přepsat na:

$$0 \approx \frac{\partial E}{\partial d} \sum_x (F(x) + dF'(x) - G(x))^2 \tag{14}$$

A z toho lze odvodit vztah pro výslednou hodnotu posunutí:

$$d \approx \frac{\sum_x F'(x)(G(x) - F(x))}{\sum_x F'(x)^2} \tag{15}$$

Ve vícerozměrném prostoru se výsledný vztah liší jen minimálně:

$$d \approx \frac{\sum_x \left(\frac{\partial F}{\partial x}\right)^T \left(G(x) - F(x)\right)}{\sum_x \left(\frac{\partial F}{\partial x}\right)^T \left(\frac{\partial F}{\partial x}\right)} \tag{16}$$

$$\left(\frac{\partial}{\partial x}\right) = \left(\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, \dots, \frac{\partial}{\partial x_n}\right)$$

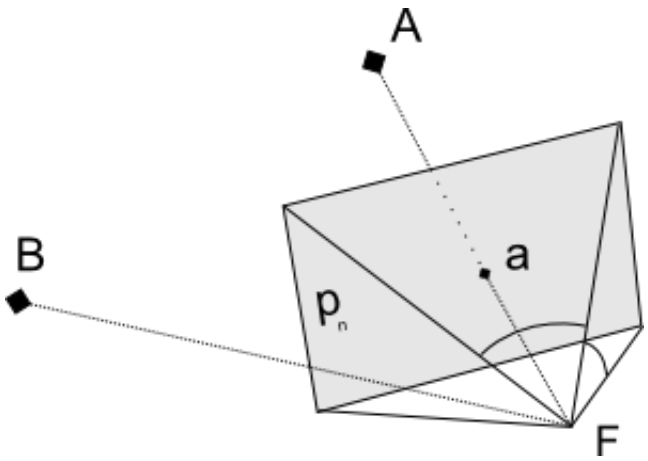
Výpočet posunutí jednotlivých významných bodů se provádí pouze v malých oknech okolo významných bodů. Výsledný odhad posunutí se zpřesňuje iterativním výpočtem pomocí Newton-Raphsonovy metody.

Metoda dokáže spolehlivě odhadnout jen velmi malá posunutí. Proto byla v [8] navržena pyramidová implementace KLT. Ta pracuje s různými zmenšeními původního obrazu. Začíná se od nejmenších velikostí obrazu, ve kterých je hledán hrubý odhad velikosti posunutí, a postupně se zvětšováním rozlišení vstupního obrazu se odhad posunutí zpřesňuje.

2.6 Perspektivní projekce

Perspektivní projekce popisuje vznik obrazu tak, jak jej vytváří lidské oko či jiné reálné optické systémy (např. objektiv kamery). Je rovněž využívána v počítačové 3D grafice a to právě v případech, kdy je třeba napodobit reálné zobrazování (pro nezkreslené zobrazování 3D objektů se používá např. orthogonální projekce).

Princip perspektivní projekce je zobrazen na následujícím obrázku:



Obrázek 14 – Perspektivní projekce
Bod ***A*** se promítl do bodu ***a***. Bod ***B*** leží mimo zobrazovací jehlan, takže se nepromítl.

Projekce je určena polohou ohniska F a polohou a rozměry promítací plochy p_n. Zobrazení každého bodu je pak dáno průsečíkem spojnice bodu s ohniskem a promítací roviny. Ohniskem a promítací plochou je dán zobrazovací jehlan. Body, které leží mimo tento jehlan, se nezobrazí vůbec. Vzdálenost d_f, což je vzdálenost ohniska od promítací roviny, se nazývá ohnisková vzdálenost. Úhly, které svírají spojnice ohniska se středem a oběma okraji promítací plochy, se nazývají zorné úhly (field of view). Vztahy mezi ohniskovou vzdáleností, šířkou w a výškou h promítací roviny a zornými úhly jsou následující:

$$\tan\left(\frac{fov_w}{2}\right) = \frac{\frac{w}{2}}{d_f} \tag{17}$$

$$\tan\left(\frac{fov_h}{2}\right) = \frac{\frac{h}{2}}{d_f}$$

$$\frac{\tan\left(\frac{fov_w}{2}\right)}{\tan\left(\frac{fov_h}{2}\right)} = \frac{w}{h} \tag{18}$$

Pokud předpokládáme, že ohnisko leží v počátku souřadného systému $[0,0,0]$ a zobrazovací rovina leží od tohoto bodu směrem po jedné ose (obvykle to bývá osa z, tak jak je tomu na obrázku), jsou pro popis perspektivní projekce potřebné pouze ohnisková vzdálenost, jeden zorný úhel (obvykle vertikální zorný úhel – fov_h) a poměr stran promítací roviny [17].

Při perspektivním zobrazovaní v počítačové grafice nebo při vzniku digitálního snímku se reálné souřadnice promítnutého bodu v promítací rovině převádí na souřadnice v pixelech. Pokud bychom chtěli provést opačný postup – převést souřadnice v pixelech na reálné souřadnice – lze za stejných předpokladů jako v předchozím odstavci a za využití vztahu 18 použít následující vztah:

$$x = \frac{x_p - \frac{w_p}{2}}{r_x} \tag{19}$$

$$y = \frac{y_p - \frac{h_p}{2}}{r_y}$$

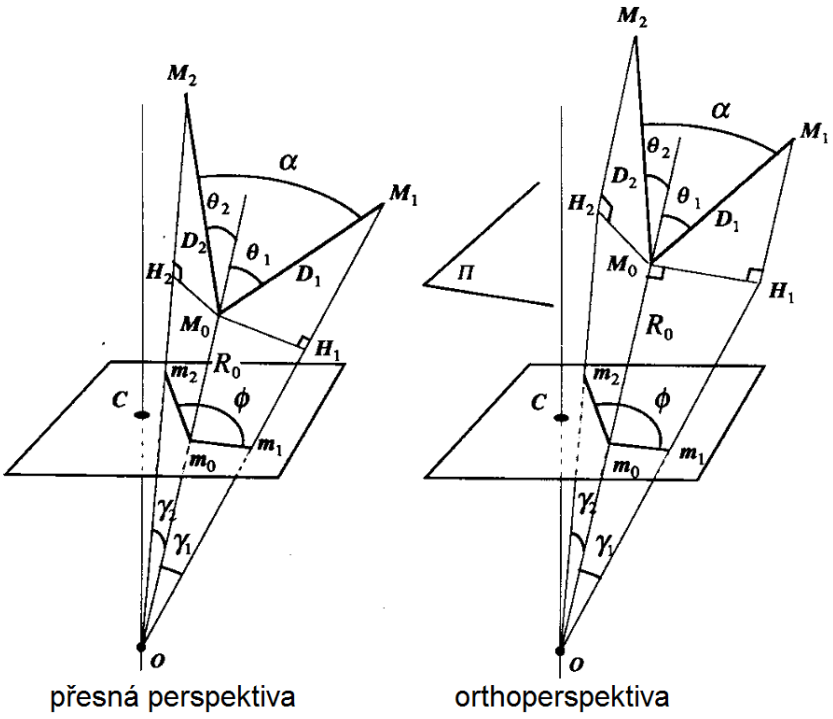
$$z = d_f$$

$$r_x = r_y = \frac{\frac{-w_p}{2}}{tan\left(\frac{fov_w}{2}\right) d_f} = \frac{\frac{-h_p}{2}}{tan\left(\frac{fov_h}{2}\right) d_f}$$

kde $[x_p, y_p]$ značí souřadnice bodu v pixelech, $[x, y, z]$ reálné souřadnice bodu, w_p šířku obrazu v pixelech, h_p výšku obrazu v pixelech, d_f ohniskovou vzdálenost, fov_w horizontální zorný úhel a fov_h vertikální zorný úhel.

2.7 Perspektivní tříbodový problém

Perspektivní tříbodový problém (v angličtině *perspective three point problem* nebo také *triangle pose problem*) je charakterizován tím, že známe promítnuté polohy třech bodů a jejich relativní reálné pozice a potřebujeme dopočítat jejich reálnou polohu. V [9] je popsáno přesné řešení a několik přibližných.



Obrázek 15 – Přesná a pozměněná perspektiva použitá při řešení tří-bodového problému (zdroj: [9])

Na obrázku 15 je problém nastíněn graficky. Potřebujeme spočítat reálné souřadnice bodů M_0, M_1 a M_2. Známe m_0, m_1 a m_2., což jsou promítnuté souřadnice hledaných bodů. Pro převod souřadnic v pixelech na reálné souřadnice lze použít vztah 19. Dále známe úhel ϕ, který svírají přímky m_0m_1 a m_0m_2, D_1 a D_2, což jsou vzdálenosti $|M_0M_1|$ a $|M_0M_2|$, a úhel α, který svírají úsečky M_0M_1 a M_0M_2

Pro přesný výsledek je třeba postupovat podle levé části obrázku, kde jsou body m_0, m_1 a m_2 získány obyčejným promítnutím bodů M_0, M_1 a M_2 do promítací roviny. U přibližného řešení se využívá takzvané orthoperspektivy, která předpokládá, že body M_0, M_1 a M_2 se nejprve promítnou do roviny Π, která obsahuje bod M_0 a která je kolmá na přímku OM_0. Tak vzniknou body M_0, H_1 a H_2 a teprve jejich promítnutím získáme body m_0, m_1 a m_2. Úhel ϕ je při použití orthoperspektivy dán úhlem, který svírají roviny Om_0m_1 a Om_0m_2. Použití orthoperspektivy výrazně zjednodušší výpočet – viz dále.

Pro získání souřadnic bodů M_0, M_1 a M_2 stačí vypočíst velikost R_0, což je vzdálenost bodů O a M_0, a velikosti úhlů θ_1 a θ_2. θ_1 je úhel, který svírá přímka OM_0 s přímkou M_0M_1. θ_2 je úhel, který svírá přímka OM_0 s přímkou M_0M_2. Pokud známe R_0, θ_1 a θ_2, souřadnice bodů M_0, M_1 a M_2 lze vypočíst z následujících vztahů [10]:

$$\overrightarrow{OM_0} = R_0\overrightarrow{e_0} \tag{20}$$

$$\overrightarrow{OM_1} = \left(R_0 + D_1\left(cos\theta_1 - \frac{sin\,\theta_1}{tan\,\gamma_1}\right)\right)\overrightarrow{e_0} + D_1\frac{sin\,\theta_1}{sin\,\gamma_1}\overrightarrow{e_1}$$

$$\overrightarrow{OM_2} = \left(R_0 + D_2\left(cos\theta_2 - \frac{sin\,\theta_2}{tan\,\gamma_2}\right)\right)\overrightarrow{e_0} + D_2\frac{sin\,\theta_2}{sin\,\gamma_2}\overrightarrow{e_2}$$

kde $\overrightarrow{e_0}$, $\overrightarrow{e_1}$ a $\overrightarrow{e_2}$ jsou jednotkové vektory, které mají stejný směr jako přímky Om_0, Om_1 a Om_2. A kde γ_1 je úhel, který svírá $\overrightarrow{e_0}$ a $\overrightarrow{e_1}$, a γ_2 je úhel, který svírá $\overrightarrow{e_0}$ a $\overrightarrow{e_2}$.

Podle [9] lze odvodit následující vztahy nejprve pro přesnou perspektivu:

$$\frac{sin(\theta_1 - \gamma_1)}{R_0/D_1} = sin\,\gamma_1 \tag{21}$$

$$\frac{sin(\theta_2 - \gamma_2)}{R_0/D_2} = sin\,\gamma_2$$

$$\frac{sin(\theta_1 - \gamma_1)}{sin(\theta_2 - \gamma_2)} = K \tag{22}$$

$$K = \frac{sin\,\gamma_1/D_1}{sin\,\gamma_2/D_2} \tag{23}$$

a pro orthoperspektivu:

$$\frac{sin\,\theta_1}{R_0/D_1} = sin\,\gamma_1 \tag{24}$$

$$\frac{sin\,\theta_2}{R_0/D_2} = sin\,\gamma_2$$

$$\frac{sin\ \theta_1}{sin\ \theta_2} = K \tag{25}$$

$$K = \frac{tan\ \gamma_1/D_1}{tan\ \gamma_2/D_2} \tag{26}$$

Poslední vztah nutný pro výpočet platný pro obě řešení:

$$cos\ \alpha = sin\ \theta_1\ sin\ \theta_2 \cos\phi + cos\ \theta_1\ cos\ \theta_2 \tag{27}$$

Při použití přesné perspektivy, bychom neznámé θ_1, θ_2 a R_0 museli počítat ze vztahů (21), (22) a (23). Díky rozdílu úhlů ve funkci sinus ve vztazích (21) a (22) by byl výpočet velice složitý s až 16 možnými řešeními, z nichž by bylo třeba vybrat pouze dvě, která jsou správná.

Zato při použití orthoperspektivy, vede soustava rovnic (24), (25) a (26) ke kvadratické rovnici s řešením:

$$sin\ \theta_1 = \sqrt{\frac{-B - \sqrt{\Delta}}{2}} \tag{28}$$

$$B = -(K^2 - 2K cos\ \alpha \cos\phi\ + 1)$$

$$\Delta = B^2 - 4K^2 sin^2\alpha\ sin^2\ \phi$$

Hodnotu R_0 lze získat ze vztahu (24). Z řešení je patrné, že získáme dvě hodnoty θ_1 (kvůli funkci sinus) a k nim podle vztahu (25) dvě hodnoty θ_2. Tyto čtyři hodnoty nelze kombinovat libovolně, ale musí platit:

$$cos\ \theta_1 \cos\ \theta_2\ (\cos\alpha - \sin\ \theta_1 \sin\ \theta_2 \cos\phi)\ > 0 \tag{29}$$

Ve výsledku získáme za použití vztahu (20) vždy jednu možnou polohu bodu M_0 a dvoje možné polohy M_1 a M_2, které jsou zrcadlově symetrické. Otázku, které z těchto řešení je správné už nelze bez dalších informací rozhodnout.

Chyby, jakých se dopustíme při použití orthoperspektivy jsou rozebrány v [9].

3 Návrh systému

Jak bylo naznačeno v úvodu, na systém je kladeno několik požadavků, jež se budou muset zohlednit při výběru metody, která bude použita pro odhad orientace hlavy. Požadavky, které již byly zmíněny:

- možnost pracovat bez kalibrace či inicializace
- co nejmenší náročnost na prostředky
- použití jedné standardní webové kamery
- robustnost
- plynulost

Další požadavky vznikly hlavně kvůli tomu, aby byl systém realizovatelný:

- vyloučit sběr trénovacích dat a samotné trénovaní
- využít volně dostupná řešení dílčích algoritmů
- vyloučit příliš komplikované postupy

Vzhledem k těmto požadavkům jsem z metod pro odhad polohy a orientace hlavy (kapitola 2.3) navrhnul systém, který by se dal popsat jako kombinace geometrické metody s trackovaním. V následující kapitole je výsledný systém stručně popsán. Jednotlivé části systému jsou pak detailněji popsány v dalších kapitolách.

3.1 Schéma systému

Systém funguje kombinací detekce obličeje a tří obličejových částí, sledování těchto částí a odhadu polohy hlavy z 2D polohy těchto částí. Podle odhadu polohy hlavy je provedeno vykreslení zobrazované 3D scény. Celý proces je popsán data-flow diagramem na obrázku.

Hlavním a prvním blokem celého systému je **kamera**. Kamera na začátku běhu systému poskytuje obraz pro detekci obličeje, ale později i pro další bloky.

Detekce obličeje proběhne vždy na začátku běhu systému – v průběhu inicializace. Detekce obličeje probíhá v celém obraze. Pokud se nepodaří v prvním snímku obličej nalézt, musí se provádět i v dalších krocích dokud se obličej nenalezne. Jinak je v dalších krocích detekce obličeje prováděna pouze, pokud je to vyžadováno jinými bloky – viz dále.

Po úspěšné detekci obličeje je možné provést **detekci obličejových částí.** Ty nejsou detekovány v celém obraze, ale pouze v části, kde se detekoval obličej. Pro každou obličejovou část je navíc možné v oblasti detekovaného obličeje zvolit podoblast, kde se daná obličejová část musí nacházet. Detekce obličejových částí vždy navazuje na úspěšnou detekci obličeje – je tedy prováděna ve všech snímcích, kdy byl nalezen obličej.

Každé obličejové části je přidělen určitý minimální a maximální počet významných bodů. Významné jsou ty body obrazu, které se výrazně liší od svého okolí. **Doplňování významných bodů** je prováděno vždy, když počet významných bodů klesl pod minimální úroveň, což zahrnuje i hledaní nových bodů během inicializace. Hledání významných bodů samozřejmě probíhá jen v oblasti, která přísluší obličejové části. Oblast je dána při inicializaci detekcí, později je tato oblast upravována výsledkem sledování. Při hledání je vhodné nalézt více bodů než je potřeba a podle dalších kriterií vybrat ze skupiny nově nalezených ty nejvhodnějších – viz níže. U každého nově nalezeného bodu je navíc uložena jeho relativní pozice v rámci oblasti příslušející dané obličejové části

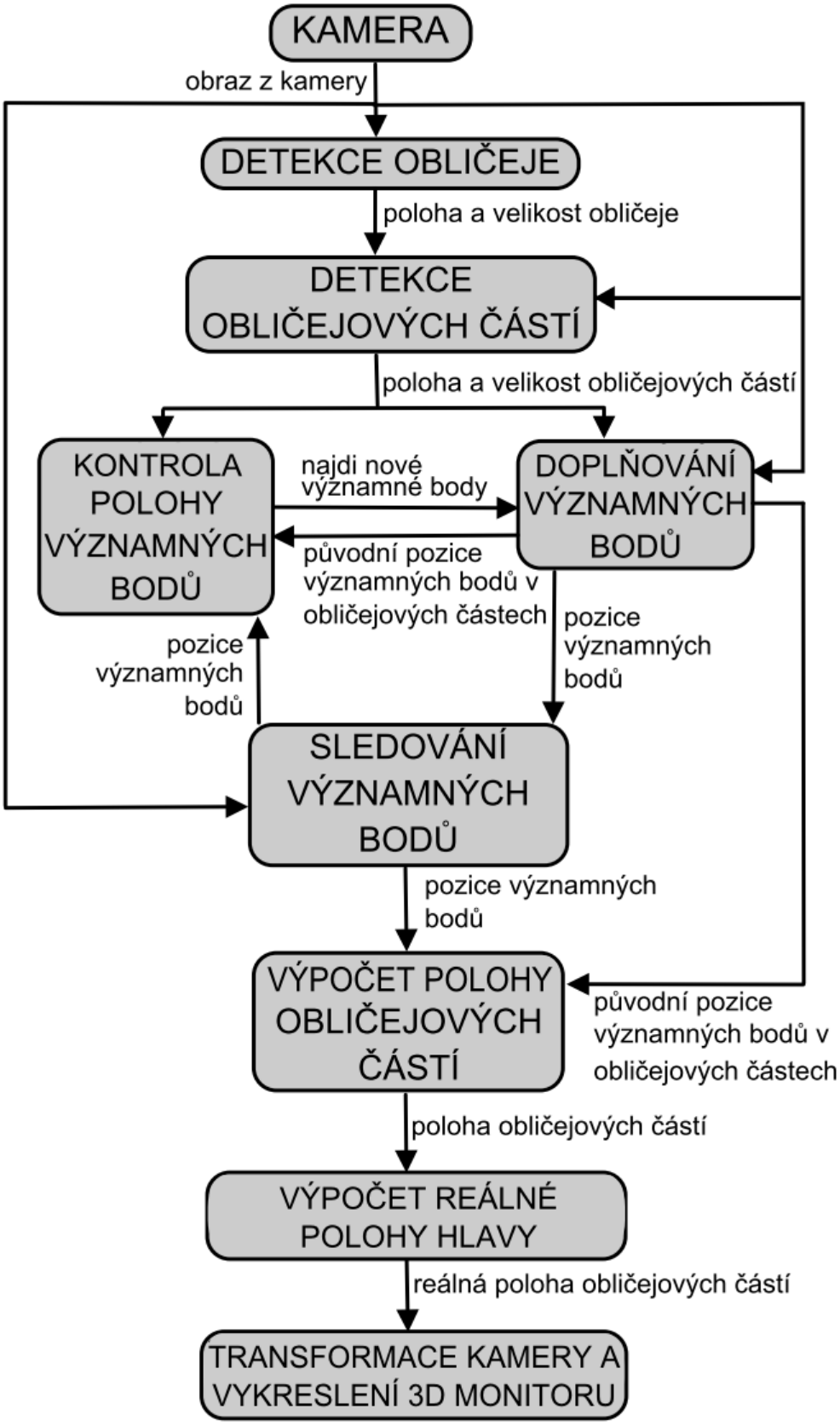
Sledování významných bodů probíhá v každém novém snímku z kamery. Úkolem sledování je pro každý významný bod nalézt při znalosti jeho pozice v minulém snímku jeho novou pozici v aktuálním snímku. Pokud se nějaké body nepodaří nalézt, provede se hledání nové polohy podle polohy ze snímku staršího, než je ten minulý. Takto lze dohledat i body, které byly v minulém snímku chybně vyhledané nebo v něm nebyly nalezeny vůbec – například kvůli zakrytí.

Při sledování významných bodů může docházet k chybám, kdy je jako nová poloha sledovaného bodu označeno jiné místo. K tomu muže dojít například vlivem stínů či odlesků nebo vlivem problému apertury (aperture problem) [3]. Proto se provádí **kontrola významných bodů**. U bodů se kontroluje jejich rychlost vzhledem k průměrné rychlosti všech bodů a jejich pozici v rámci oblasti příslušející dané obličejové části. Také se jednou za čas provede kontrolní detekce obličejových částí a u bodů se pak ověřuje, jestli leží v oblasti detekce. Pokud bod neprojde kontrolními testy, jeho nově nalezená poloha je smazána a bod je označen jako nenalezený.

Výpočet polohy obličejových částí je prováděn s využitím všech bodů nalezených v novém snímku a jejich relativních poloh v obličejovém rysu. Po výpočtu nové polohy obličejové části, je každému sledovanému bodu upravena jeho relativní poloha v obličejové části.

Během **výpočtu reálné polohy hlavy** se ze tří 2D souřadnic obličejových částí vypočtou jejich 3D souřadnice. Z nich se odvodí 3D poloha hlavy a případně její orientace.

Poloha hlavy je využita k **transformaci kamery** zobrazující 3D scénu. Kamera je posunuta a natočena tak, aby to odpovídalo pohledu uživatele na monitor. Scéna se ovšem nemůže pouze vykreslit z této kamery přímo na monitor. Obraz je nejprve vykreslen do paměti a až z ní je vhodným způsobem namapován na monitor – viz níže.



Obrázek 16 – Schéma systému 3D monitoru

3.2 Výběr obličejových částí

Systém v obličeji detekuje obě oči a ústa. Tento výběr má několik opodstatnění:

- všechny tyto objekty jsou vhodné k detekci, protože patří k nejvýraznějším částem v obličeji
- je dostupná řada řešení zajišťujících tuto detekci
- prvky neleží na okrajích obličeje, takže k jejich zakrytí dochází až při velkých natočení hlavy
- žádná z částí vyloženě nevyčnívá z obličeje

Poslední bod je důležitý, protože kdyby části vyčnívaly, docházelo by při rotaci hlavy k velké změně vzhledu těchto částí, což by znesnadňovalo detekci i sledování. Například obraz nosu se při rotaci hlavy výrazně mění, protože jej vidíme z různých stran. Na druhou stranu ústa se při mluvení uživatele také mění. Jelikož nelze odhadnout, která obličejová část je z dvojice ústa – nos výhodnější, je nutno podotknout, že systém je navržen tak, aby případná změna vyžadovala minimální úsilí.

3.3 Detekce obličeje a obličejových částí

První krok odhadu polohy a orientace hlavy je detekce obličeje. Tato detekce je stejně jako detekce obličejových částí prováděna pomocí algoritmu Viola-Jones popsaném v kapitole 2.4.1. Tento krok není nezbytný, ale je zde z následujících důvodů:

- obličejové části se nemusí hledat v celém obraze, ale jen v malé části příslušející obličeji – tím dojde k optimalizaci
- z velikosti obličeje lze pro každou z obličejových částí odvodit minimální velikost, tu lze pak zadat algoritmu Viola-Jones jako startovací velikost detekční kaskády (kapitola 2.4.5) - tím dojde opět k optimalizaci
- pokud dojde k omezení prostoru pro algorimtus Viola-Jones, dojde také k omezení false positive výsledků – tímto se sníží chybovost

Po detekci obličeje je možné přistoupit k detekci jednotlivých obličejových částí. Ty se nemusí hledat v celé oblasti, která přísluší obličeji, ale lze vybrat jen určitou podoblast, o které budeme jistě vědět, že se tam hledaná obličejová část jistě nachází. Levé oko se určitě bude nacházet v levé horní části obličeje, pravé oko v pravé horní části a ústa ve spodní části.

Algoritmus Viola-Jones vrací při úspěšné detekci obličeje nenatočený obdélník, který by měl obsahovat obličej. Pro obdélník určíme body A_F a C_F, kde A_F je levý dolní roh se souřadnicemi $[x_{AF}, y_{AF}]$ a C_F je pravý horní roh obdelníka se souřadnicemi $[x_{CF}, y_{CF}]$. Pro popis oblastí, kde se budou hledat další obličejové části, použijeme normalizované souřadnice:

$$x_n = \frac{x - x_{AF}}{x_{CF} - x_{AF}} \tag{30}$$
$$y_n = \frac{y - y_{AF}}{y_{CF} - y_{AF}}$$

Jako příklad jsou uvedeny hodnoty, které byly určeny experimentálně na konkrétních detekčních kaskádách.

Levé oko je hledáno v oblasti určené body A_{LE} a C_{LE}:

$$A_{LE} = [0,0.4] \tag{31}$$
$$C_{LE} = [0.5,1]$$

Oblast pro hledání pravého oka, je určena body A_{RE} a C_{RE}:

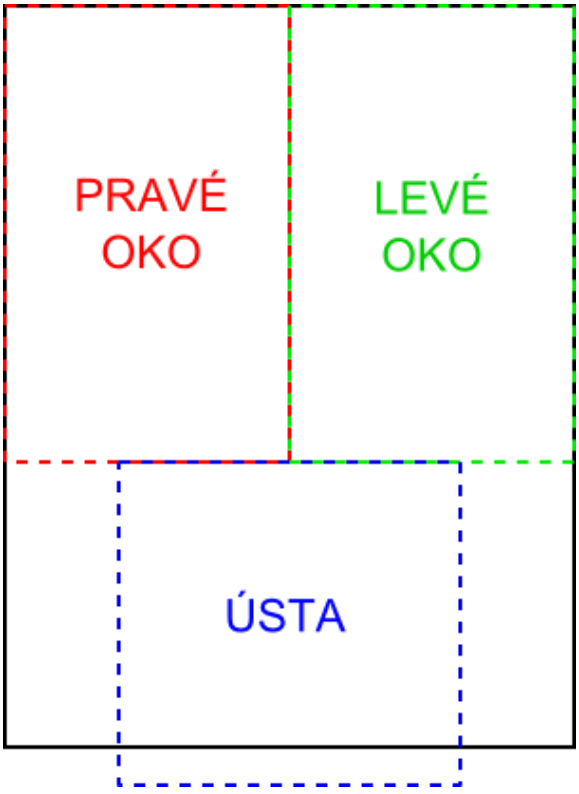
$$A_{RE} = [0.5,0.4] \tag{32}$$
$$C_{RE} = [1,1]$$

Oblast, kde budou hledána ústa, je určena body:

$$A_M = [0.2, -0.05] \tag{33}$$
$$C_M = [0.8,0.4]$$

Oblast úst leží částečně i pod oblastí obličeje, protože se často stává, že oblast obličeje neobsahuje ústa celá. K tomu dochází obzvláště v případech, kdy je hlava natočená po ose X (pitch).

Všechny oblasti jsou znázorněny v příkladu na následujícím obrázku:



Obrázek 17 – Detekovaný obličej a podoblasti, kde se budou detekovat obličejové části

3.4 Sledování obličejových částí

Sledování obličejových částí opět není nezbytný krok vedoucí k detekci polohy hlavy. Avšak užitím sledování lze systém obohatit z následujících důvodů:

- detekce obličejových částí poskytuje velmi nestabilní výsledky, díky sledování lze systém stabilizovat
- detekce obličejových částí je více časově náročnější, sledováním dojde k optimalizaci systému
- detekce obličejových částí podává výsledky jen v určitých orientacích hlavy – např. při rotaci hlavy kolem osy Z (roll) detekce pomocí algoritmu Viola-Jones selhává úplně [21], sledování tedy zvýší robustnost systému

V této kapitole je průběh sledování rozebrán na několik částí, které jsou vysvětleny pomocí algoritmů zapsaných v pseudokódu.

3.4.1 Inicializace sledování

Sledování obličejové části je zahájeno její první úspěšnou detekcí. Při ní se uloží poloha středu obdélníka označující detekci a jeho rozměry (výška a šířka). Dokud neproběhne první úspěšná detekce sledované obličejové části, není možné pokračovat dál. Proto se systém při svém startu snaží detekovat obličejové části v každém snímku, dokud nejsou všechny nalezeny.

Po první úspěšné detekci dojde k inicializaci sledování. Ačkoliv se jedná pouze o nastavování hodnot, je inicializace zapsána v pseudokódu kvůli návaznosti na další části:

$$stred = stredDetekce;\ velikost = velikostDetekce;\ priblizovatDetekci = 0;$$
$$historiePoloh[0][1..N] = null;\ historieChyb[0][1 \dots N] = null;$$
$$rozdilyOdStredu[1 \dots N] = 0;\ rozdilDetekce = (0,0);$$

Algoritmus 1 – Inicializace sledování obličejové části její detekcí

Význam konstant a proměnných je následující:

- N je počet významných bodů, pomocí kterých se obličejová část sleduje
- $stred$ vždy obsahuje aktuální polohu středu obličejové části
- $velikost$ popisuje aktuální rozměry obličejové části – výšku a šířku (mění se jen po úspěšné detekci)
- $historiePoloh$ obsahuje všechny polohy sledovaných významných bodů v historii (tohle v praxi samozřejmě není možné implementovat, takže hloubka historie je omezená)
- $historieChyb$ obsahuje všechny chyby sledování významných bodů v historii
- $rozdilyOdStredu$ obsahuje aktuální rozdíl (vektor) všech významných bodů od středu obličejové části
- $priblizovatDetekci$ a $rozdilDetekce$ jsou použity ke srovnání sledování a detekce

3.4.2 Sledování významných bodů

Po inicializaci již začne v každém snímku probíhat sledování významných bodů příslušejících k obličejové části. Průběh sledování je popsán následujícím algoritmem:

1. pokud neplatí $priblizovatDetekci == 0$
   $stred = stred + rozdilDetekce;$
   $UpravRozdily(rozdilyOdStredu, historiePoloh[t], stred);$
   $priblizovatDetekci - -;$
2. $novePolohy[1 \dots N] = null;\ noveChyby[1 \dots N] = null; historiePokusy = 0;$
3. $h = VyberHistorii();\ historiePokusy + +;$
4. pro $i = 1 \dots N$
   $(novaPoloha, novaChyba)$
   $= NajdiNovouPolohu(historiePoloh[h][i], historieObrazu[h]);$
   pokud $(novaChyba < chybaLimit)$
   $novePolohy[i] = novaPoloha;\ noveChyby[i] = novaChyba;$
5. $novyStred = VypocitejNovyStred(novePolohy, rozdilyOdStredu);$
6. ZkontrolujRychlosti(historiePoloh[h], novePolohy);
7. ZkontrolujPoziciVObdelníku(novyStred, velikost, novePolohy);
8. pokud $historiePokusy < historiePokusyLimit$ a současně
   $PrumernaChyba(noveChyby) > prumernaChybaLimit$
   jdi na krok 3
9. $historiePoloh[t] = novePolohy;\ historieChyb[t] = noveChyby;$
10. $stred = novyStred;$
11. $UpravRozdily(rozdilyOdStredu, historiePoloh[t], stred);$
12. $spatneBody[] = NajdiSpatneBody(historiePoloh, historieChyb)$
13. pokud $\big(N - Pocet(spatneBody)\big) < dobreBodyLimit$
    $NahradSpatneBodyNovymi(historiePoloh, historieChyb, spatneBody);$
    pro všechna i z $spatneBody$
    $rozdilyOdStredu[i] = stred - historiePoloh[t][i];$

Algoritmus 2 – Průběh sledování obličejové části, který se provádí ve všech snímcích

Krok 1 slouží pro srovnávní sledování obličejového rysu s kontrolní detekcí, což je popsáno níže.

Při prvním volání tohoto algoritmu ještě nejsou vytvořeny žádné významné body, takže kroky 2 až 11 nemají žádný účinek. Až kroky 12 a 13 vytvoří první významné body, jejichž polohy se hledají v následující snímcích a voláních tohoto algoritmu.

Hledání nových poloh významných bodů probíhá pomocí trackeru KLT, který je popsaný v kapitole 2.5.3. KLT je schopný najít nové polohy sledovaných bodu v aktualním obraze za využití jiného obrazu se známými polohami sledovaných bodů. Každý nález nového bodu je navíc označen chybou.

Nejprve je tedy potřeba vybrat snímek a polohy významných bodů z historie, pomocí nichž budou nalezeny nové polohy sledovaných bodů. To je znázorněno krokem 3. Tento výběr je popsán níže.

Po výběru historie je již možné se pokusit najít nové polohy sledovaných bodů (krok 4). Nová poloha je uznána, jen pokud chyba nálezu nepřesáhla hodnotu $chybaLimit$. Jakmile známe nové polohy sledovaných bodů, je možné vypočíst nový střed obličejové části (krok 5). Jak probíhá tento výpočet je také vysvětleno níže.

Dále je provedena kontrola nových poloh významných bodů. Kontroluje se jejich rychlost vzhledem k minulému snímku (krok 6, viz níže). Také se kontroluje, zda body leží v dané obličejové části (krok 7).

Může se stát, že některým sledovaným bodům nebyly nalezeny nové polohy a to znásledujících důvodů:

- v historii používané k hledání nových poloh nebyl daný bod nalezen (mohl být například zakryt)
- KLT novou polohu nenalezl, nebo byla chyba nálezu příliš vysoká
- poloha bodu byla smazána na základě neúspěšných kontrol rychlosti či polohy

Aby se zvýšila úspěšnost hledání nových poloh, je možné ji provést vícekrát za využití dat z jiné historie (krok 8). K opakovému hledání nových poloh bodů dochází jen pokud počet pokusů nepřekročil hodnotu $historiePokusyLimit$ a současně pokud průměrná chyba překročila $prumernaChybaLimit$. Pokud dojde k opakovanému hledání nových poloh bodů (průměrná chyba byla vyšší než limit), provede se také v příštím kroku kontrolní detekce.

V kroce 9 jsou uloženy nové hodnoty poloh sledovaných bodů a chyby těchto nálezů. V kroce 10 je akutalizována i hodnota středu obličejového rysu. V kroce 11 jsou upraveny hodnoty vektorů rozdílů poloh sledovaných bodů od středu obličejového rysu. Kroky 12 a 13 slouží k vylučování některých významných bodů a jejich nahrazení, což je vysvětleno níže.

3.4.3 Kontrolní detekce

Jelikož se stává, že sledované body postupně „sjíždí“ ze svých původních pozic a tím negativně ovlivňují pozici obličejové části, je vhodné provádět jednou za čas kontrolní detekci. Kontrolní detekce se také provádí tehdy, pokud byla průměrná chyba sledování větší než limit.

Samotná detekce probíhá nezávisle na sledování. Výsledek detekce, který je platný pro snímek s časem t, musí být zpracován po hledání nových poloh sledovaných bodů v čase t a dřív než proběhne hledání v čase $t + 1$. Výsledek detekce je zpracován následujícím způsobem:

1. $velikost = velikostDetekce$;
2. $vzdalenostDetekce = Vzdalenost(stred, stredDetekce)$;
3. pokud $vzdalenostDetekce > stredBezZmenyLimit$ a současně
   $vzdalenostDetekce < stredPriblizovatLimit$
   $rozdilDetekce = (stred - stredDetekce)/pomerPriblizovaniDetekce$;
   $priblizovatDetekci = pomerPriblizovaniDetekce$;
   jinak pokud $vzdalenostDetekce > stredPriblizovatLimit$
   $stred = stredDetekce$;
   pro $i = 1 \dots N$
   $rozdilyOdStredu[i] = stred - historiePoloh[t][i]$;

Algoritmus 3 – Reakce sledování obličejové části na její detekci

Velikost sledované obličejové části je automaticky aktualizována (krok 1). Poté je spočítána vzdálenost středu detekce od středu sledování (krok 2). To jak se změní střed sledování, závisí právě na této vzdálenosti.

Jak již bylo řečeno, detekce podává nestabilní výsledky. Pokud by se střed sledování automaticky opravil podle středu detekce, ona nestabilita detekce by se přesunula do sledování a ve výsledku by při každé kontrolní detekci poloha sledovaných obličejových částí poskočila. Proto jsou jemné rozdíly sledování od detekce tolerovány. Jestliže je vzdálenost středu detekce od sledování menší než $stredBezZmenyLimit$, vůbec se neupraví střed sledování.

Pokud je vzdálenost větší než $stredBezZmenyLimit$ avšak stále menší než $stredPriblizovatLimit$, upravuje se během následujících několika kroků střed sledování tak, aby se přiblížil středu detekce – viz krok 3 algoritmu 3 a krok 1 algoritmu 2. Rychlost přibližování středu sledování ke středu detekce určuje hodnota $pomerPriblizovaniDetekce$.

Pokud je vzdálenost větší než $stredPriblizovatLimit$, rozdíl detekce od sledování již nemůže být tolerován ani postupně upravován a dojde ke skokové změně středu sledování (krok 3).

Všechny tyto uváděné vzdálenosti jsou normalizovány šířkou obličejové části. Tím se zajistí podobné chování při různých vzdálenostech uživatele od kamery.

3.4.4 Výběr historie pro hledání nových poloh bodů

Při hledání nových poloh sledovaných bodů je potřeba nejprve vybrat pomocí jakého obrazu a jakých poloh z historie se bude vyhledávat (krok 3 algoritmu 2). Jelikož KLT tracker není schopen najít

odpovídající polohy sledovaných bodů při velkých posunutích [3], je při prvním pokusu vždy využito poloh a obrazu z předchozího sledování.

Hledání je však možné provést vícekrát (krok 8 algoritmu 2), což je vhodné například v situacích, kdy bylo předchozí sledování bodů neúspěšné. To mohlo vzniknout například zakrytím sledované obličejové části. Proto je potřeba nějakým způsobem vybrat jinou vhodnou historii. Výběr je proveden s ohledem na to, kolik dosud nenalezených bodů má v dané historii známou polohu. Díky tomu je větší šance, že nenalezené body budou dohledány.

Pokud byla v historii s časem $t - h$ průměrná chyba sledování $\bar{e}$, je tato historie ohodnocena výrazem:

$$h_{eval} = \bar{e} \cdot vyberHistoriePomer^{h} \tag{34}$$

Kde $vyberHistoriePomer$ je hodnota menší 1. Podle této hodnoty jsou zvýhodněny hodnoty novější před hodnotami staršími. Pro hledání nových poloh sledovaných bodů je pak použita historie s největší hodnotou h_{eval} .

3.4.5 Výpočet středu obličejové části

Jelikož je rozmístění významných bodů v obličejové části náhodné, nelze střed obličejové části ztotožnit s těžištěm významných bodů. Střed obličejové části je tedy z významných bodů počítán jiným způsobem.

Při vytváření významného bodu (krok 13 algoritmu 2), při přibližování středu sledování detekci (krok 1 algoritmu 2) nebo při skokové úpravě středu sledování (krok 3 algoritmu 3) je potřeba si uložit vektor rozdílu polohy významného bodu od aktuální polohy středu:

$$rozdilyOdStredu[i] = stred - historiePoloh[t][i] \tag{35}$$

Potom po vyhledání nových poloh významých bodů je nová poloha středu obličejové části dána vztahem (viz krok 5 algoritmu 2):

$$stred = \frac{1}{N}\sum_{i=1}^{N}(novePolohy[i] + rozdilyOdStredu[i]) \tag{36}$$

3.4.6 Kontrola rychlostí sledovaných bodů

Kontrola rychlostí sledovaných bodů má za úkol vyloučit ze zpracování ty body, jejichž pohyb je výrazně odlišný od pohybu ostatních bodů. Takové body pravděpodobně nebyly vyhledány správně a mohly by negativně ovlivnit výslednou polohu obličejové části. Body s odlišným pohybem vznikají především pohybem stínů, odlesků či vlivem problému apertury (aperture problem) [3].

Může se ale stát, že se všechny body pohybují jiným směrem (například při přibližovaní či oddalování uživatele od kamery). V tomto případě by bylo vhodné, aby byla kontrola rychlostí benevolentnější. Toho lze docílit použitím směrodatné odchylky velikostí rychlostí.

Označme aktuální polohu sledovaného bodu jako p_new_i a minulou polohu jako p_old_i. Potom je rychlost bodu dána vztahem:

$$v_i = p\_new_i - p\_old_i \tag{37}$$

Průměrná rychlost všech bodů je daná vztahem:

$$\bar{v} = \frac{1}{N}\sum_{i=0}^{N} v_i \tag{38}$$

A směrodatná odchylka velikostí rychlostí je dána vztahem:

$$\sigma_v = \sqrt{\frac{1}{N}\sum_{i=0}^{N} |v_i - \bar{v}|^2} \tag{39}$$

K vyřazení nové polohy bodu dojde tehdy, platí-li:

$$\frac{|v_i - \bar{v}|}{\sigma_v} > rychlostLimit \tag{40}$$

Kde $rychlostLimit$ je hodnota větší než 1. Čím menší je tato hodnota, tím je kontrola rychlostí přísnější.

3.4.7 Hledání nových významných bodů

Během sledování se stává, že se některé body nedaří dlouhodobě úspěšně sledovat. Potom je vhodné tyto body nahradit nějakými stabilnějšími.

Označme velikost historie každého bodu jako h. Je to počet snímků, který uplynul od jeho vytvoření (vyhledání hranovým detektorem). Kontrola, zda bude bod označen jako špatný, proběhne pouze pokud

$$h > historieTestyLimit \tag{41}$$

Jinými slovy – bod bude ověřován a případně označen jako špatný pouze pokud je starší než $historieTestyLimit$, což je celé kladné číslo. Tato podmínka je zde kvůli tomu, aby se nestávalo, že body budou nahrazeny záhy po svém vytvoření.

Jestliže má bod dostatečně velkou historii pro další testování, je označen jako špatný pokud:

$$\frac{n_p}{h} < historiePokrytiLimit \tag{42}$$

kde n_p udává, kolikrát ve své historii měl daný bod známou polohu. Parametr $historiePokrytiLimit$ je číslo z intervalu $(0,1)$.

Dále může být sledovaný bod označen za špatný, pokud je jeho průměrná chyba historie menší než $historieChybaLimit$:

$$\bar{e} < historieChybaLimit \tag{43}$$

Samotné hledání nových významných bodů probíhá v oblasti příslušející dané obličejové části pomocí hranového detektoru Shi-Tomasi. Hledání nových bodů probíhá jednou po inicializaci sledování a poté později, když je potřeba nahradit špatné body novými. Jelikož detektor Shi-Tomasi při hledání významných bodů počítá pro každý bod obrazu autokorelační matici (viz kapitola 2.5.2), což je operace relativně náročná, je vhodné provádět hledání nových významných bodů po dávkách. Proto je hledání nových bodů zahájeno pouze tehdy, klesne-li počet dobrých bodů pod $dobreBodyLimit$ (krok 13 algoritmu 2).

Také je vhodné najít více významných bodů, než je třeba, a mezi těmito body vybrat ty nejvhodnější. Výběr lze určit na základě míry hranovosti, kterou lze získat z detektoru Shi-Tomasi, nebo podle rozmístění významných bodů v obličejové oblasti. Výběr může například zajistit rovnoměrné rozmístění bodů v obličejové části. Také je možné se u určitých obličejových částí vyhnout některým oblastem, které nejsou vhodné pro sledování. V oblasti očí rozhodně není vhodné vybírat body pro sledování umístěné v samotném oku. Nejenže se v této oblasti objevují často sledování nepříznivé odlesky, ale i pohyb duhovky by mohl negativně ovlivnit sledování.

3.5 Odhad reálné polohy hlavy

Nyní předpokládáme, že v obraze z kamery již známe polohy všech sledovaných obličejových částí (oči a ústa). Tato kapitola obsahuje popis, jak lze z těchto poloh získat reálnou polohu hlavy.

Odhad polohy hlavy je poveden pomocí vyřešení tříbodového perspektivního problému, který je popsán v kapitole 2.7. Jako bod M_0 (viz obrázek 15) jsou zvolena ústa. Body M_1 a M_2 potom odpovídají očím. Při řešení problému je třeba znát relativní rozložení těchto bodů – jsou to vzdálenosti $|M_0M_1|$ a $|M_0M_2|$, a úhel α, který svírají úsečky M_0M_1 a M_0M_2. Ty jsou vypočteny z následujících vztahů:

$$|M_0M_1| = |M_0M_2| = \sqrt{\left(\frac{D_e}{2}\right)^2 + (D_m)^2} \tag{44}$$

$$\alpha = 2\ tan\left(\frac{D_e}{2D_m}\right) \tag{45}$$

kde D_e je vzdálenost očí a D_m je vzdálenost úst od spojnice obou očí. Odhadem byly tyto vzdálenosti určeny následovně :

$$D_e = 6.5cm$$

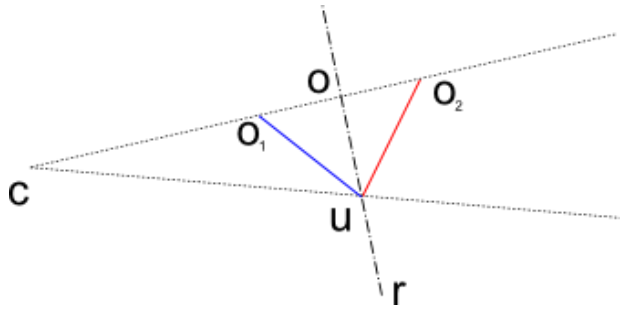
$$D_m = 7cm$$

Další parametry důležité pro výpočet se týkají kamery, která obličej snímá. Je potřeba znát ohniskovou vzdálenost a zorné úhly kamery (viz kapitola 2.6).

Jednotky, ve kterých vyjde řešení, jsou dány jednotkami, v jakých byly zadány vzdálenosti $|M_0M_1|$ a $|M_0M_2|$.

Již zmíněným nedostatkem tříbodového perspektivního problému je, že existují vždy dvě možná řešení a nelze jednoduše určit, které z nich je to správné.

Pro ústa mají obě řešení vždy stejnou polohu. Pro každé oko existují vždy dvě zrcadlově převrácená řešení, což je zachyceno na následujícím obrázku:



Obrázek 18 – Příklad dvou možných řešení polohy oka u perspektivního tříbodového problému
$\boldsymbol{c}$ je kamera, $\boldsymbol{u}$ jsou ústa, $\boldsymbol{o_1}$ a $\boldsymbol{o_2}$ jsou dvě možné polohy jednoho oka, které jsou symetrické podle osy $\boldsymbol{r}$, $\boldsymbol{o}$ je průměr obou řešení

Problém je vyřešen nejjednodušším způsobem – jako poloha hlavy je označen průměr obou možných řešení obou očí. Tím získáme přibližnou polohu středu očí. Jelikož nám nejde o přesné řešení a jelikož jsou již tato dvě řešení zatížena chybou díky použití orthoperspektivy (viz obrázek 17), je chyba, které se tímto průměrováním dopustíme, zanedbatelná.

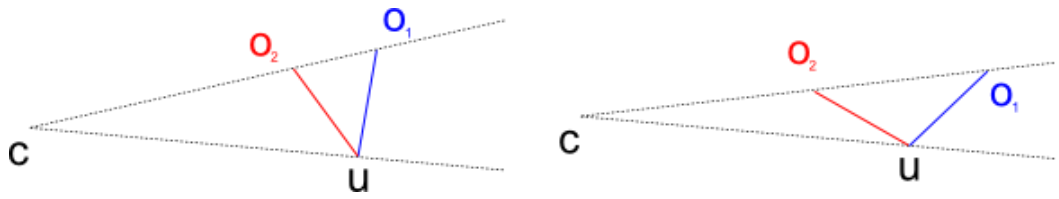
Způsob, jakým by bylo možné vybrat ze dvou řešení to správné, je nastíněn v následující kapitole.

3.6 Odhad orientace hlavy

Kdybychom znali reálnou polohu všech tří obličejových částí, mohli bychom snadno pomocí vektorového součinu dopočítat orientaci hlavy. Stále je tu ale problém nastíněný v předchozí kapitole – problém dvojího řešení perspektivního tříbodového problému. Ačkoliv to není pro potřeby 3D monitoru nutné, budou v této kapitole uvedeny dva možné způsoby, jak vybrat z těchto dvou řešení to správné a umožnit tak odhadnout orientaci hlavy. Oba způsoby jsou pouze teoretické a jejich použitelnost nebyla ověřena.

První způsob předpokládá, že by bylo možné mít u každé obličejové části informaci o tom, zda se v porovnání s minulým snímkem zvětšila či zmenšila. Zvětšení či zmenšení je totiž důsledkem přiblížení respektive oddálení obličejové části od kamery. Změna velikosti by šla například odhadnout porovnáním relativních vzdáleností významných bodů obličejové části mezi dvěma snímky.

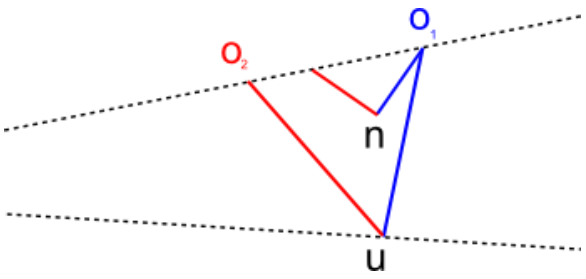
Způsob, jakým je možné vybrat správné řešení, je ukázán pomocí následujícího příkladu a obrázku:



Obrázek 19 – Dvě řešení polohy oka při změně polohy hlavy
$\boldsymbol{c}$ je kamera, $\boldsymbol{u}$ jsou ústa, $\boldsymbol{o_1}$ a $\boldsymbol{o_2}$ jsou dvě možné polohy jednoho oka

Na obrázku je znázorněna změna polohy hlavy. Kromě změny promítnuté polohy obličejových částí víme na základě předchozího předpokladu také to, že oko se mezi snímky zmenšilo. Z toho můžeme usuzovat, že se oko od kamery vzdálilo. Správné řešení musí tedy změnit svou polohu stejným způsobem. Z obrázku vidíme, že každé řešení zareagovalo na změnu jiným způsobem. Řešení $\boldsymbol{o_1}$ se od kamery vzdálilo, kdežto řešení $\boldsymbol{o_2}$ se ke kameře přiblížilo. Je tedy zřejmé, že správným řešením je $\boldsymbol{o_1}$ a hlava se pravděpodobně otočila směrem nahoru.

Druhý způsob je sice jednodušší z hlediska implementace, ale je výpočetně náročnější. Lze totiž mimo tři obličejové rysy sledovat ještě čtvrtý, který bude sloužit pouze pro výběr správného řešení z dvojice řešení tříbodového perspektivního problému. Je například možné sledovat nos a využít jej způsobem zobrazeným na následujícím obrázku:



Obrázek 20 – Výběr ze dvou řešení polohy oka u perspektivního tříbodového problému použitím čtvrtého bodu
$\boldsymbol{u}$ je poloha úst, n je poloha nosu, $\boldsymbol{o_1}$ a $\boldsymbol{o_2}$ jsou dvě možné polohy jednoho oka

Jak vidíme, tak při řešení dvojího perspektivního tříbodového problému získáme dvě a dvě řešení, z nichž však jen dvě půjdou zkombinovat.

3.7 Transformace virtuální kamery

V předchozích kapitolách byl představen způsob, jak zjistit polohu uživatelovy hlavy. V této kapitole bude uveden postup, jak využít tuto polohu k transformaci kamery snímající 3D scénu počítačového programu.

Při řešení tříbodového perspektivního problému, z něhož je spočítána poloha hlavy, jsou výsledkem pouze relativní souřadnice vůči kameře, která uživatele snímá. Pro správné zobrazení je tedy důležité znát relativní polohu kamery vůči středu monitoru. Poté, pokud budeme chtít umožnit uživateli rozhlížet se po virtuální 3D scéně se středem c_s, bude poloha virtuální kamery, pomocí které budeme 3D scénu snímat, dána vztahem:

$$c_v = c_s + c_r + h \tag{46}$$

kde c_v je výsledná poloha virtuální kamery, c_r je relativní poloha reálné kamery vůči středu monitoru a h je relativní poloha hlavy vůči kameře.

Předchozí vztah předpokládá, že je reálná kamera natočena ve směru kolmém od roviny monitoru. Pokud by byla reálná kamera natočena jiným způsobem, byl by výpočet složitější.

Virtuální kamera bude vždy natočená ve směru středu virtuální 3D scény.

3.8 Vykreslení 3D monitoru

Pro splnění funkčnosti 3D monitoru, by nyní stačilo vykreslit 3D scénu pomocí virtuální kamery definované v předchozí kapitole. Představme si ale, že bychom chtěli mít ve virtuální scéně okno, které by přesně odpovídalo monitoru počítače, a uživateli bychom chtěli umožnit skrze svůj monitor nahlížet do virtuální scény, která se rozprostírá za tímto oknem.

Pro takové vykreslení je potřeba speciální kamera, jejíž ohnisko je dáno pozicí hlavy a promítací plocha je dána virtuálním oknem. Spojnice ohniska kamery a středu promítací plochy pak nemusí být k této ploše kolmá a zobrazovací jehlan může být zkosený. Nejedná se tedy o typickou perspektivní projekci – viz kapitola 2.6. Navzdory tomu ale počítačové grafické knihovny takové nastavení umožňují (např. funkce $glFrustum$ u OpenGL [22]).

4 Implementace

Tato kapitola obsahuje popis implementace systému navrženého v kapitole 3. Budou zde uvedeny prostředky použité k tvorbě systému a řešení některých implementačních problémů. Také bude popsána knihovna Monitor3D, která je výsledkem implementace.

Systém byl napsán v jazyce C++ s využitím volně dostupných a multiplatformních knihoven OpenCV a OpenGL. Pro získání obrazu z kamery byl použit multimediální framework DirectShow. Jako vývojové prostředí bylo použito Microsoft Visual Studio 10. Systém je určen pro operační systém Windows, ale po vynechání DirectShow a několika málo úpravách, které jsou zmíněny níže, by měl být snadno přeložitelný i na jiných platformách.

4.1 OpenCV

OpenCV je svobodná a otevřená multiplatformní knihovna zaměřená na real-time zpracování obrazu a počítačové vidění. Jak již zaměření knihovny napovídá, byla v systému použita na problémy spojené s detekcí a sledováním obličejových částí.

V systému jsou využity následující funkce z knihovny OpenCV (výčet neobsahuje různé pomocné funkce):

- $cvHaarDetectObjects$ je funkce, která implementuje detekci objektů pomocí systému Viola-Jones. Funkce byla použita pro detekování obličeje a obličejových částí. Pro detekci byly použity následující detekční kaskády, které jsou dodávány spolu s OpenCV:
  $haarcascade\_frontalface\_alt.xml$ – pro detekci obličeje
  $haarcascade\_mcs\_righteye.xml$ – pro detekci pravého oka
  $haarcascade\_mcs\_lefteye.xml$ – pro detekci levého oka
  $haarcascade\_mcs\_mouth.xml$ – pro detekci úst
- $cvGoodFeaturesToTrack$ implementuje vyhledávání bodů vhodných pro sledování pomocí hranového detektoru Shi-Tomasi.
- $cvCalcOpticalFlowPyrLK$ implementuje sledování bodů užitím KLT trackeru.

Ačkoliv OpenCV poskytuje funkce k získávání obrazu z kamer nainstalovaných v operačním systému, není téhle možnosti v systému 3D Monitoru využito. Využití OpenCV v tomhle směru bylo totiž značně pomalé v porovnání s ostatními řešeními. Například z kamery, která má snímkovou frekvenci 30 Hz, bylo možné použitím OpenCV získat jen zhruba 15-16 snímků za vteřinu.

K získávání obrazu z kamery byl tedy použit framework DirectShow, jehož použití bylo sice značně složitější, ale zase bylo dosaženo kýžené rychlosti. Navíc je s využitím DirectShow snadné přesměrovat vstup z kamery na video soubor, což bylo vhodné zejména ve fázi testovací.

Použitím DirectShow jako vstupního zdroje obrazových dat ale vznikl malý problém s orientací vstupních obrazových dat. Ačkoliv OpenCV poskytuje možnost označit, jakým způsobem jsou vstupní obrazová data orientována, zdá se, že výše uvedené funkce pro detekci a sledování toto označení nerespektují. Proto je nutné vstupní obrazová data před použitím OpenCV funkcí převrátit. Výstupy těchto funkcí jsou potom ale také převrácené. Při důrazu na optimalizaci to vše značně znesnadňovalo práci a vznikala tak spousta zbytečných chyb, které zpomalovaly vývoj systému.

Problém blokující detekce

Další problém, který vyvstal při použití OpenCV, je spojen s použitím funkce $cvHaarDetectObjects$.

Funkce je volána s parametrem $CV\_HAAR\_FIND\_BIGGEST\_OBJECT$. Tento parametr zajistí, že se vždy detekuje pouze největší objekt nacházející se v obraze. To je velice výhodné při použití v systému 3D monitor, jelikož je tak sledován pouze uživatel, který se nachází nejblíže kameře. Problém ovšem nastane, pokud se funkci $cvHaarDetectObjects$ nedaří nalézt žádný objekt. K tomu dochází nejen v případech, kdy daný objekt v obraze není, ale také v případech, kdy je objekt natočen okolo osy z (roll). Poté je výpočet výsledku několikanásobně delší (v některých případech i 3x), než pokud je detekce úspěšná. Takové výpočty, které navíc ani nepřináší žádné výsledky, systém blokují a díky tomu může docházet k zahazování některých snímků z kamery. To se velmi negativně projevuje na sledování obličejových částí, které díky chybějícím obrazům z kamery selhává.

Proto je v systému vytvořeno jedno speciální vlákno, které zajišťuje pouze výpočet detekce. Tím dojde na počítačích s více procesorovými jádry k optimalizaci a hlavně celý systém pak není detekcí blokován. Za běhu systému se počítá průměrná doba úspěšné detekce $\bar{t_d}$. Hlavní vlákno pak v případě potřeby požádá druhé o výpočet detekce, avšak na výsledek čeká pouze po dobu $1.1\bar{t_d}$.

K vytváření a správě vláken je využito Windows API, což je druhý důvod (první je použití DirectShow), proč je systém v současném stavu přeložitelný pouze pod Windows.

4.2 Knihovna Monitor3D

Knihovna Monitor3D je výsledkem implementace systému 3D monitoru. Knihovna poskytuje jednoduché rozhraní, pomocí kterého lze libovolné 3D aplikaci, která používá pro vykreslení OpenGL a která je určena pro operační systém Windows, dodat výše popsaný 3D efekt. Pro testování této knihovny byla napsána jednoduchá aplikace, jejíž popis naleznete v příloze.

Rozhraní knihovny je rozděleno do tří tříd podle jejich funkčnosti. Programátor tak může rozhodnout, co přenechá knihovně Monitor3D a co si naprogramuje sám.

$DirectVideo$ je třída, pomocí které lze enumerovat kamery napojené v systému, vybrat jednu z nich jako zdroj videa a následně pak voláním jedné metody pravidelně získávat aktuální obrazová data. Třída podporuje také užití video souborů, jako zdrojů videa.

$HeadPoseEstimator$ je nejdůležitější třída, která provádí detekci, sledování a odhad uživatelovy hlavy z obrazových dat. Toto vše lze provést pravidelným voláním jediné metody – $ProcessInputData$. Parametry této metody jsou vstupní obrazová data a jejich rozměry. Po skončení této metody lze získat voláním metody $GetHeadPosition$ odhadnutou relativní pozici uživatelovy hlavy vzhledem ke kameře.

Mimo volání těchto metod je samozřejmě nutné nastavit parametry potřebné pro odhad, jako jsou zorný úhel a ohnisková vzdálenost kamery či rozměry obličeje (viz kapitola 3.5).

$Monitor3D$ je velmi jednoduchá třída, která zajišťuje vykreslování 3D monitoru pomocí knihovny OpenGL. Tato třída poskytuje pouze jednu metodu – *BeginRender*, která očekává jako vstupní parametry relativní polohu hlavy ke středu monitoru a velikosti virtuálního okna, pomocí kterého bude nahlíženo do scény. Poloha tohoto okna bude vždy rovna počátku souřadného systému. Tato metoda pouze nastaví OpenGL pro vykreslování. Poté je možné pomocí standardních prostředků OpenGL vykreslit scénu.

Obvykle programátor musí po získání polohy hlavy pomocí třídy $HeadPoseEstimator$ a před vykreslením scény pomocí třídy $Monitor3D$ upravit polohu tak, aby se vztahovala ke středu monitoru, a ne ke středu kamery. Informaci, jak polohu měnit, je potřeba získat přímo od uživatele, protože záleží na něm, jak je kamera vůči monitoru počítače umístěna.

5 Výsledky a testování

Systém byl prověřen testy na náročnost na systémové prostředky, přesnost, plynulost, robustnost a uživatelskou přívětivost. V této kapitole bude na výsledcích těchto testů ověřeno, zda systém splňuje požadavky, které byly stanoveny v úvodu.

5.1 Náročnost na systémové prostředky

Jelikož byl jeden z požadavků na systém jeho co nejmenší náročnost na systémové prostředky, byl proveden test tuto vlastnost ověřující. Systém byl testován s jednoduchou grafickou scénou, jejíž náročnost na vykreslení je minimální, a tudíž by tím testování nemělo být ovlivněno. Po celou dobu testu se uživatel aktivně pohyboval.

Další detaily testu:

- délka trvání: 2 minuty
- CPU: Intel(R) Core(TM)2 Duo, 2x 2.0 Ghz
- RAM: 2.00 GB
- GPU: NVIDIA GeForce 8600M
- kamera: 640x480, 30 FPS

Výsledky testu:

- celkový procesorový čas: 55.568 s
- využití procesoru = $55.568 / 120 / 2 = 23\ \%$
- využití paměti = 40.064

Jelikož testovací sestava již v této době nepatří k nejvýkonnějším, lze vzhledem k uvedeným výsledkům prohlásit, že požadavek systému na jeho nenáročnost byl splněn. Na další výpočty zbývá ještě spousta výkonnostní kapacity, takže by jistě nebyl problém využít systém i v náročnějších aplikacích – například v počítačových hrách.

Jelikož je systém optimalizován pro běh na procesoru s více jádry, lze očekávat, že na procesoru s jedním jádrem, budou dosažené výsledky horší.

5.2 Přesnost, plynulost a robustnost systému

Pro následující testy byla využita testovací data z projektu, který detekuje obličej a obličejové části (oči a ústa) pomocí integrálních projekcí a tyto objekty sleduje [23]. K tomuto projektu je dostupná sada testovacích videí společně s anotací, která obsahuje přesnou polohu obličejových částí (ground truth). Videa jsou dostupná na webové adrese <http://www.dis.um.es/~ginesgm/fip/videos.html>

Videa byla použita jako vstup systému. V průběhu sledování byla v každém snímku ukládána poloha obličejových částí v pixelech. Pro studium vlastností systému byly použity následující charakteristiky:

- chyba (vzdálenost v pixelech) polohy jednotlivých obličejových částí v každém snímku
- chyba všech obličejových častí (průměr) v každém snímku
- průměr těchto chyb v celém videu
- variační koeficient těchto chyb v celém videu

Variační koeficient charakterizuje variabilitu chyby a lze jej spočítat vztahem:

$$c_e = \frac{\sigma_e}{|\bar{e}|} \tag{47}$$

kde σ_e je směrodatná odchylka chyby (viz např. vztah 40) a $\bar{e}$ je průměrná hodnota chyby.

Test 1

Pro následující test bylo použito video ggm4.avi. Na videu je protagonista s brýlemi, v průběhu si brýle odloží a nasadí zpět. Celý pohyb je relativně rychlý, provázen různorodou obličejovou mimikou a dochází i k zakrytí obličeje. Video je tedy vhodné pro ověření přesnosti, plynulosti i robustnosti zároveň.

Parametry videa:

- rozlišení: 640x480
- snímková frekvence: 15 Hz
- délka: 489 snímků

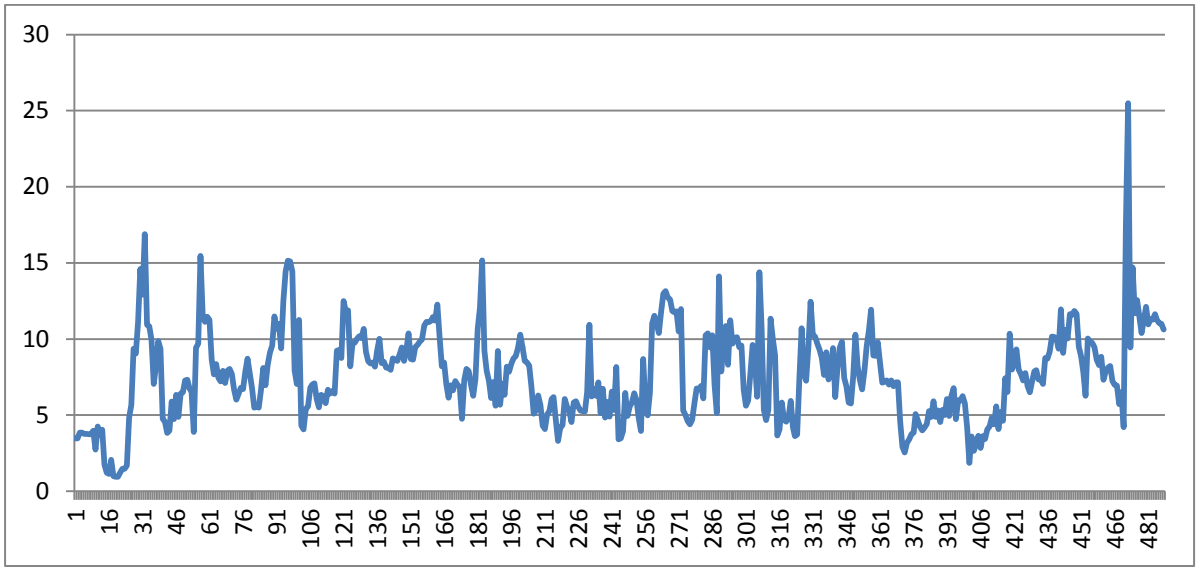
Výsledky:

	levé oko	pravé oko	ústa	vše
$\bar{e}$	5.48	6.09	11.3	7.62
c_e	0.74	0.64	0.55	0.38

Tabulka 1 – Výsledky testu 1

Z výsledků vidíme, že průměrná chyba všech obličejových částí není velká – pouze 7.62 pixelů. Zatímco levé a pravé oko mají zhruba stejnou průměrnou chybu, u úst je chyba téměř dvojnásobná.

To lze vysvětlit tím, že ústa nabývají ve videu různých tvarů a jen těžko lze určit jejich střed. Hodnoty variačních koeficientů budou porovnávány s dalšími testy.



Graf 1 – Graf celkové chyby v testu 1 (horizontální osa – číslo snímku videa, vertikální osa – velikost chyby)

Test 2

Účelem tohoto testu bylo zjistit, zda kontrola rychlostí významných bodů navrhovaná v kapitole 3.4.6 má očekávaný vliv na výsledek. Test byl proveden se stejným videem jako test první, avšak během sledování neprobíhala kontrola rychlostí sledovaných bodů.

Výsledky:

	levé oko	pravé oko	ústa	vše
$\bar{e}$	6.49	6.28	14.49	9.09
c_e	0.85	0.80	0.57	0.50

Tabulka 2 – Výsledky testu 2

Po porovnání výsledků z tabulky 1 a tabulky 2 vidíme, že stouply nejen průměrné hodnoty chyb, ale také hodnoty variačních koeficientů. Kontrola rychlostí sledovaných bodů má tedy pozitivní vliv na přesnost systému, ale také na jeho stabilitu (plynulost).

Test 3

Další rozšíření, které bylo navrženo ke zlepšení sledování, je použití obrazů a poloh starších, než je poslední snímek – viz kapitola 3.4.4. Test 3 měl za cíl potvrdit užitečnost tohoto rozšíření. Test opět proběhl se stejným videem jako test 1, avšak podpora historie byla v tomto testu vypnuta.

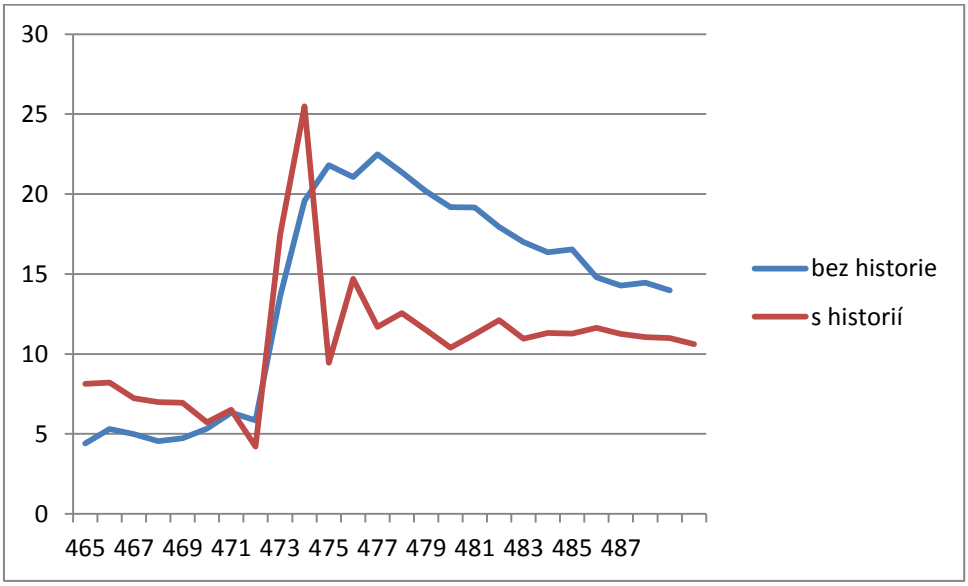
Výsledky:

	levé oko	pravé oko	ústa	vše
$\bar{e}$	5.27	6.32	13.28	8.29
c_e	0.73	0.78	0.59	0.50

Tabulka 3 – Výsledky testu 3

Při porovnání výsledků z tabulky 1 a tabulky 3 lze vidět, že sice chyba a variačních koeficient levého oka klesly, avšak ostatní hodnoty včetně celkové chyby jsou vyšší.

Zajímavé je také porovnat průběh chyby u konce videa. Ve videu totiž zhruba ve snímku 471 nastane překrytí obličeje. To se projevuje na grafu skokovou změnou chyby – viz graf 1. Při použití historie je však tato chyba téměř okamžitě srovnána, kdežto bez použití historie trvá srovnání chyby déle. Rozdíly jsou porovnané na grafu 2. To, aby se systém dobře vyrovnal s překrytím obličeje, byl hlavní důvod, proč bylo použití historie zavedeno. Použití historie tedy zlepšuje přesnost a stabilitu systému očekávaným způsobem.



Graf 2 – Porovnání odezvy systému na zakrytí obličeje bez a s použitím historie (horizontální osa – číslo snímku videa, vertikální osa – velikost chyby)

Test 4

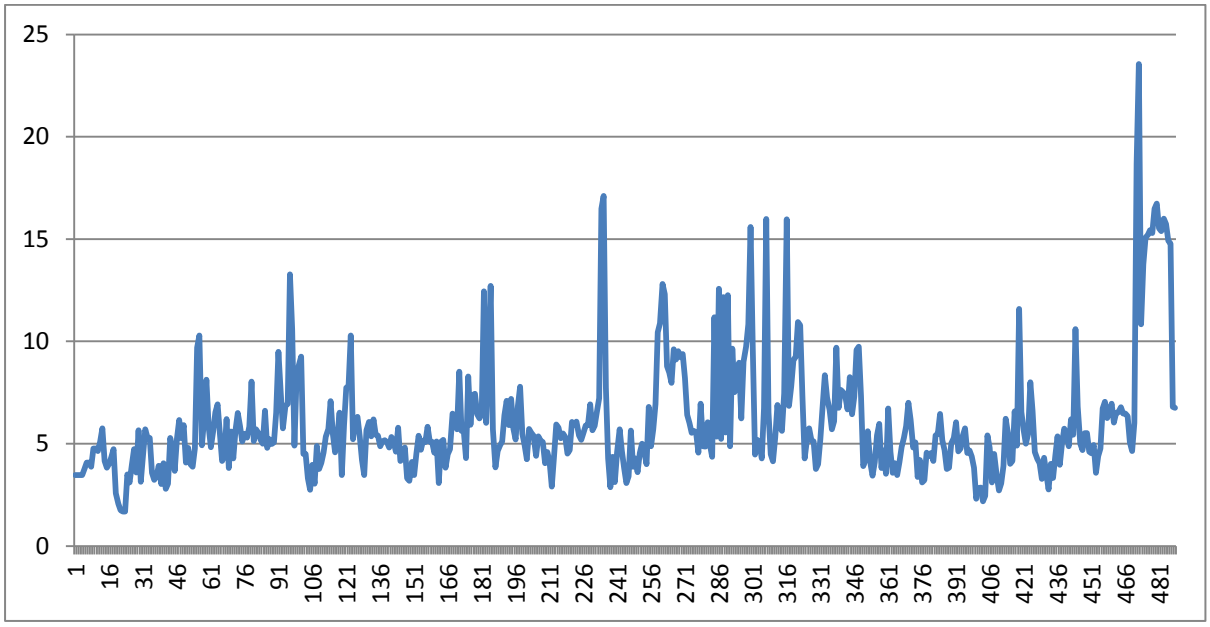
Jelikož je podle požadavků na systém preferována stabilita systému před jeho přesností, je podle kapitoly 3.4.3 po případech kontrolní detekce prováděno postupné dorovnávání změny rozdílu detekce od sledování. To je také předmětem testu 4. Opět bylo využito stejného videa jako v testu 1, avšak tentokrát po kontrolní detekci neprobíhá postupné vyrovnávání rozdílu detekce a sledování, ale skokové.

Výsledky:

	levé oko	pravé oko	ústa	vše
$\bar{e}$	3.21	4.11	10.80	6.04
c_e	1.09	0.77	0.59	0.48

Tabulka 4 – Výsledky testu 4

Jak je vidět z porovnání tabulky 1 a tabulky 4, bez postupného vyrovnávání detekce lze dosáhnout menších průměrných chyb. Na druhé straně ale všechny variační koeficienty vzrostly. Systém je tak méně stabilnější, což lze jasně vidět i porovnáním grafu 1 a grafu 3. Jelikož by taková nestabilita způsobovala nepříjemné kmitání obrazu, je dána přednost stabilitě před přesností a ve výsledném systému je postupné vyvažování rozdílu detekce a sledování použito.



Graf 3 – Průběh celkové chyby bez použití postupného vyvažování rozdílu detekce a sledování v testu 4 (horizontální osa – číslo snímku videa, vertikální osa – velikost chyby)

Další testy

Také proběhly testy s více uživateli, aby se zjistilo, zda systém rozpoznává různé tváře. Systém úspěšně detekoval každého ze sedmi dobrovolníků ve věku 19-50 let, z nichž byly tři ženy, dva nosí brýle a jeden delší vousy. Pouze uživatelé s brýlemi měli problémy, pokud se na ploše brýlí objevovaly odlesky. Dojmy těchto dobrovolníků jsou uvedeny v následující kapitole.

Systém nemá problém, pokud je před kamerou více uživatelů, vždy je pro sledování vybrán ten nejbližší. Rovněž pohyb v pozadí nečiní systému obtíže. Obecně je ale vhodnější, aby pozadí nebylo příliš členité.

Další testy zkoumaly vliv osvětlení na výsledky detekce. Systém je schopný pracovat v přirozeném i umělém světle, s nižší i vyšší intenzitou. Ovšem důležité je, aby osvětlení bylo pokud možno rovnoměrné. Při velkých rozdílech jasů na obličeji selhává detekce a tudíž celý systém.

5.3 Uživatelské testy

V předchozí kapitole byla krátce představena skupina uživatelů, na kterých byla vyzkoušena schopnost systému rozpoznávat a sledovat různé obličeje. Tito uživatelé měli možnost si systém v klidu vyzkoušet. Po té bylo uživatelům položeno několik otázek, na které mohli odpovědět známkou od 1 do 5, kdy 1 znamená „Vynikající“ a 5 „Nedostačující“ .

Otázka: „Mohli jste se po příchodu k počítači začít bez problémů rozhlížet?“
Průměrná odpověď: 1.8
Uživatelé většinou hodnotili tuto vlastnost systému pozitivně. Většina z nich se mohla začít rozhlížet bez problémů. Jeden uživatel měl však brýle a mohl systém bez problému používat až po jejich odložení. Jeden uživatel musel přistoupit blíže ke kameře.

Otázka: „Připadal vám běh programu plynulý, docházelo k nějakým problémům?“
Průměrná odpověď: 2
U této otázky se odpovědi velice shodovaly. Většina uživatelů tvrdila, že když drželi hlavu v klidu, tak zobrazovaná scéna byla také téměř v klidu. Všem uživatelům se ale stalo, že scéna alespoň jednou poskočila, většinou při prudkých pohybech. Naopak nikomu se nestalo, že by program poté nešlo dál používat.

Otázka: „Jaký máte dojem z 3D efektu“
Průměrná odpověď: 1
Uživatelům se výsledný efekt líbil. Většinou nečekali, že je něco takového možné. Typickou reakcí bylo: „Úplně to vyčnívá.“

Z výsledku těchto testů lze usoudit, že uživatelům se se systémem pracuje dobře a bez velkých problému. Většina z nich potřebovala minimální instrukce. Dojem z 3D efektu byl též dobrý. Jedním z uváděných problémů byla potřeba přistoupit ke kameře blíže při inicializaci. Dalším problémem jsou občasné chyby ve sledování, které jsou častější při rychlých pohybech spojených s natočením hlavy.

6 Závěr

V této práci byl navržen systém 3D monitoru, který pomocí webové kamery sleduje uživatele a podle polohy jeho hlavy mění pohled na scénu zobrazovanou na monitoru počítače. Díky tomu má uživatel pocit, že scéna není připoutaná pouze k povrchu obrazovky, ale prostupuje celým prostorem před obrazovkou i za ní.

Na systém bylo vznešeno několik požadavků, které měly zajistit jeho co největší použitelnost. Mezi tyto požadavky například patřily:

- možnost pracovat bez kalibrace či inicializace
- co nejmenší náročnost na systémové prostředky
- použití jedné standardní webové kamery
- robustnost
- plynulost

Výsledkem návrhu systému je celek, jehož hlavní částí je detektor obličejových částí založený na metodě Viola-Jones. Detekované obličejové části jsou pro zajištění stability sledovány pomocí KLT trackeru. Takto jsou sledovány tři obličejové části – obě oči a ústa. Z poloh těchto obličejových částí je řešením perspektivního tříbodového problému odhadnuta reálná poloha uživatelovy hlavy. Po zjištění polohy hlavy je jednoduchým nastavením 3D zobrazování a vykreslením scény dosaženo výsledku.

V průběhu návrhu bylo předloženo několik vylepšení, která měla za úkol splnění výše uvedených požadavků. Mezi tato vylepšení patří například zavedení historie sledování, statistické zkoumání rychlostí sledovaných bodů či pokročilé řízení kontrolní detekce. Ačkoliv to nebylo pro potřeby systému nutné, byl také navržen způsob, jak malým rozšířením systému umožnit odhadování orientace hlavy.

Systém byl implementován přesně podle návrhu. Během implementace byla použita již hotová řešení dílčích problémů. Obzvláště knihovna OpenCV se ukázala jako výkonný pomocník programátora aplikací počítačového vidění. Výsledkem implementace je snadno použitelná a relativně nenáročná programová knihovna Monitor3D.

Tato knihovna byla podrobena testům, které měly za úkol zjistit, zda byly splněny požadavky na systém. Také byly provedeny testy orientované přímo na potvrzení užitečnosti výše uvedených vylepšení. Ukázalo se, že vylepšení byla vhodně navržená a že systém víceméně splňuje všechny požadavky. Je tedy schopen pracovat s různými uživateli, kteří se pohybují různým způsobem a při různých světelných podmínkách. Uživatelé, kteří se se systémem setkali, měli vždy pozitivní reakce a výsledný 3D efekt na ně zanechal dojem.

Avšak je zde stále prostor pro zlepšení vlastností systému. Ačkoliv je systém celkem stabilní, vždy při delším používaní se objeví chyby, které způsobují kmitání obrazu a tak negativně ovlivňují pocity uživatele. Zamezení výskytu těchto chyb by bylo určitě vhodným vylepšením. Také navrhovaná schopnost odhadu orientace hlavy uživatele by v systému jistě našla uplatnění.

7 Literatura

[1] VIOLA, Paul; JONES, Michael. Rapid Object Detection using a Boosted Cascade of Simple Features. *IEEE Conference on Computer Vision and Pattern Recognition*. 2001

[2] VIOLA, Paul; JONES, Michael. Robust Real-time Object Detection. *Second International Workshop on Statistical and Computational Theories of Vision - Modeling, Learning, Computing, and Sampling*. 2001.

[3] BRADSKI, Gary; KAEHLER, Adrian. *Learning OpenCV : Computer Vision with the OpenCV Library*. Sebastopol : O'Reilly Media, 2008. 576 s.

[4] FREUND, Yoav; SCHAPIRE, Robert E. A decision-theoretic generalization of on-line learning and an application to boosting. *AT&T Bell Laboratories*. 1995.

[5] HARRIS, Chris; STEPHENS, Mike. A Combined Corner and Edge Detector. *Plessey Research Roke Manor*. 1988.

[6] SHI, Jianbo; TOMASI, Carlo. Good Features to Track. *Computer Vision and Pattern Recognition*. 1994.

[7] LUCAS, Bruce D. ; KANADE, Takeo. An Iterative Image Registration Technique with an Application to Stereo Vision. *International Joint Conference on Artificial Intelligence*. 1981.

[8] BOUGUET, Jean-Yves. Pyramidal Implementation of the Lucas Kanade Feature Tracker : Description of the algorithm. *Intel Corporation*. 2002.

[9] DEMENTHON, Daniel; DAVIS, Larry S. Exact and Approximate Solutions of the Perspective-Three-Point Problem. *IEEE Transactions on Pattern Aanalysis and Machine Inteligence*. 1992.

[10] DEMENTHON, Daniel; DAVIS, Larry S. New Exact and Approximate Solutions of the Perspective-Three-Point Problem. *IEEE International Conference on Robotics and Automation*. 1990.

[11] *Wii.com* [online]. 2009 [cit. 2011-01-11]. Dostupné z WWW: <http://wii.com>.

[12] *Microsoft Kinect Xbox 360* [online]. 2010 [cit. 2011-01-11]. Dostupné z WWW: <http://www.kinect.cz/>.

[13] *NVIDIA 3D Vision* [online]. 2009 [cit. 2011-01-11]. Dostupné z WWW: <http://www.nvidia.com/object/3d-vision-main.html>.

[14] *Johnny Chung Lee Wii Projects* [online]. 2007 [cit. 2011-02-22]. Dostupné z WWW: <http://johnnylee.net/projects/wii/>.

[15] *faceAPI* [online]. 2008 [cit 2011-02-22]. Dostupné z WWW: <http://www.seeingmachines.com/product/faceapi/>.

[16] MURPHY-CHUTORIAN, Erik; TRIVEDI, Mohan Manubhai. Head Pose Estimation in Computer Vision: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2008

[17] SONG HO, Ahn. *Songho.ca* [online]. 2011 [cit. 2011-05-15]. OpenGL Transformation. Dostupné z WWW: <http://www.songho.ca/opengl/gl_transform.html>.

[18] LI, Yongmin; GONG, Shaogang; LIDDELL, Heather. Support Vector Regression and Classification Based Multi-view Face Detection and Recognition. *IEEE International Conference on Automatic Face & Gesture Recognition.* 2000

[19] MARTINS, Pedro; CASEIRO, Rui; BATISTA, Jorge. Face Alignment Through 2.5D Active Appearance Models. *British Machine Vision Conference.* 2010

[20] HORPRASERT, Thanara; YACOOB, Yaser ; DAVIS , Larry S. Computing 3-D Head Orientation from a Monocular Image Sequence. *IEEE International Conference on Automatic Face & Gesture Recognition.* 1996

[21] BARCZAK, Andre L. C.; Toward an Efficient Implementation of a Rotation Invariant Detector Using Haar-like Features. *Image and Vision Computing New Zealand*. 2005

[22] *OpenGL API Documentation* [online]. 2011 [cit. 2011-05-18]. Dostupné z WWW: <http://www.opengl.org/documentation/>.

[23] *Human Face Processing Using Integral Projections* [online]. 2007 [cit. 2011-05-20]. Dostupné z WWW: <http://www.dis.um.es/~ginesgm/fip/index.html>

8 Seznam příloh

Příloha 1 – Specifikace parametrů pro sledování

Příloha 2 – Popis testovací aplikace Monitor3DTest

Příloha 3 – CD se zdrojovými texty a spustitelnou testovací aplikací

Příloha 1 – Specifikace parametrů pro sledování

Hodnoty parametrů z kapitoly 3, které byly použity pro implementaci:

$N = 10$ (počet sledovaných bodů)

$chybaLimit = 200$

$historiePokusyLimit = 2$

$prumernaChybaLimit = 130$

$dobreBodyLimit = 5$

$stredBezZmenyLimit = 0.15$

$stredPriblizovatLimit = 0.5$

$pomerPriblizovaniDetekce = 15$

$vyberHistoriePomer = 0.95$

$rychlostLimit = 2.5$

$historieTestyLimit = 1$

$historiePokrytiLimit = 0.5$

$historieChybaLimit = 100$

Příloha 2 – Popis testovací aplikace Monitor3DTest

Aplikace při svém spouštění čte nastavení kamery ze souboru $camera.conf$

Formát souboru je následovný:

<číslo kamery, která se má použít>\n

<zorný úhel kamery ve stupních>\n

<ohnisková vzdálenost kamery v centimetrech>

Při spouštění aplikace se vypíše seznam kamer nainstalovaných v systému.

Aplikace podporuje tři pohledy:

- data z kamery (čárkovaně je znázorněno sledování, plnou čárou detekce)
- pohled na scénu se zobrazenou polohou detekované hlavy (lze ovládat myší)
- výstup 3D Monitor

Klávesové zkratky:

1 – přepínaní pohledu v hlavním okně

2 – přepínaní pohledu ve vedlejším okně

3 – zobrazení/skrytí vedlejšího okna

4 – zapnutí/vypnutí 3D výstupu

f – zapnutí/vypnutí fullscreen režimu

<mezerník> – zapnutí/vypnutí čtení obrazu z kamery