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

HRA S OVLÁDÁNÍM POMOCÍ OČÍ A VÝRAZŮ OBLIČEJE

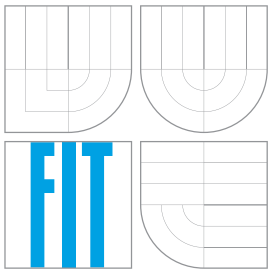
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAROMÍR VAŇHARA

BRNO 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

HRA S OVLÁDÁNÍM POMOCÍ OČÍ A VÝRAZŮ OBLIČEJE

GAME WITH EYE AND FACE EXPRESSION CONTROL

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JAROMÍR VAŇHARA
AUTHOR

VEDOUCÍ PRÁCE — Ing. MICHAL HRADIŠ
SUPERVISOR

BRNO 2009

Abstrakt

Tato bakalářká práce ukazuje jeden z možných způsobů ovládání počítače pomocí očí. Je zde představen způsob detekce oblasti očí pomocí mrknutí, dále detekce duhovky pomocí Houghovy transformace a také jednoduchá metoda k určení směru pohledu. K demonstraci principu ovládání také vznikla jednoduchá hra.

Abstract

This bachelor's thesis shows a posible way, how to control computer with human eyes. It describes method of detection eyes using blinking, also pupil detection using Hough transform and simple way, how to estimate gaze. Also the simple game was created to demonstrate the principles of control.

Klíčová slova

Detekce očí, detekce duhovky, určení směru pohledu, segmentace obrazu, Houghova transformace, OpenCV

Keywords

Eye detection, pupil detection, gaze estimation, image segmentation, Hough transform, OpenCV

Citace

Jaromír Vaňhara: Hra s ovládáním pomocí očí a výrazů obličeje, bakalářská práce, Brno, FIT VUT v Brně, 2009

Hra s ovládáním pomocí očí a výrazů obličeje

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením ing. Michala Hradiše. Uvedl jsem všechny literární prameny, ze kterých jsem při tvorbě práce čerpal.

.....
Jaromír Vaňhara
31. května 2009

Poděkování

Děkuji ing. Michalu Hradišovi za odborné vedení a poskytnutí rad a pomoci při tvorbě této bakalářské práce. Také bych chtěl poděkovat mé rodině a přítelkyni za podporu.

Obsah

Kapitola 1

Úvod

Na rozdíl od jiných oblastí informatiky jsou způsoby interakce člověka s počítačem značně konzervativní. Mezi klasické přístupy k ovládání počítače tak patří klávesnice a myš, tedy způsoby, které se bez větší obměny používají již dlouhou dobu. Samozřejmě existují další, specializovanější způsoby, například joystick nebo různé druhy volantů. Jiné techniky ovládání počítače jsou sice vyvíjeny, ale většina z nich je stále v experimentálním stádiu. Výjimkou je například konzole Nintendo Wii, která dokáže snímat pohyb ovladače ve všech směrech.

Různé jiné způsoby ovládání počítače lze využít v mnoha směrech. Je to třeba snadnější přístup hendikepovaných osob k počítači. V tomto případě rozpoznání směru pohledu, mrknutí nebo například rozpoznání hlasových příkazů může být pro tyto osoby jediný způsob, jak počítač ovládat.

Využití speciálnějších ovládacích technik ale nemusí být omezeno jen na hendikepované osoby. I pro zdravého člověka může alternativní přístup k ovládání počítače usnadnit každodenní práci. Také počítačové hry mohou doznat zásadních změn a být mnohem atraktivnější.

Tato práce si dává za cíl nalézt způsob, jak ovládat počítač pomocí očí. Bude využívat pouze běžnou web kameru a nebude vyžadovat žádný další hardware. Z těchto důvodů se od tohoto systému nebude vyžadovat příliš velká přesnost. Důležitou podmínkou ale bude, aby systém pracoval v reálném čase a mohl být využit k ovládání počítačové hry.

V této práci jsou popsány principy, které vedly ke vzniku knihovny implementující funkce pro nalezení očí ve video sekvenci získané z běžné web kamery, dále pak funkce pro detekci duhovky a určení směru uživatelova pohledu. K demonstraci ovládání byla vytvořena velmi jednoduchá počítačová hra.

Práce má 8 kapitol. V následující 2. kapitole stručně popisuji publikace, které se věnují jednotlivým problémům, které jsou součástí mého řešení. Kapitola 3 se věnuje teoretickému podkladu této práce. Jsou zde popsané základní pojmy počítačové grafiky. Vysvětlen je princip detekce očí. Dále je popsána detekce duhovky jako kruhového objektu a princip Houghovi transformace. Hlavním cílem kapitoly 4 je přesné definování požadavků na řešení a detailní popis použitých algoritmů. 5. kapitola popisuje implementační detaily. Hlavně popis vytvořené knihovny a její rozhraní pro její použití. Stručně jsou také zmíněny funkce a struktury z knihovny OpenCV použité při řešení. Poslední část této kapitoly se věnuje demonstrační hře, zejména způsobu ovládání. Další kapitola 6 se zabývá návrhem testů. Popisuje na co by se testy měly zaměřit. Výsledky testu popisuje následující, 7. kapitola. Závěrečná 8. kapitola shrnuje dosažené výsledky a diskutuje možnosti dalšího vývoje.

Kapitola 2

Dosavadní způsoby řešení

V této kapitole jsou stručně komentovány publikace na téma detekce očí v obraze a detekce duhovky a zorničky. Je tak získán obecný přehled o možném řešení jednotlivých problémů souvisejících s tématem bakalářské práce.

K detekci oblasti, ve které se nacházejí oči, lze využít jejich pohybu. Předpokládá se, že je snímán obličej uživatele. Když se provádí rozdíl po sobě následujících snímku, zjistí se pohyb na obraze. Pokud je tento pohyb mrknutí, jsou tímto způsobem detekovány oblasti očí. Tento způsob je využit například v [4].

Jiný možný způsob vychází ze specifických vlastností oblasti očí v barevném modelu YCbCr. Y v tomto modelu reprezentuje jas, Cb a Cr reprezentují modrou, respektive červenou barevnou složku obrazu. Tento postup využívá specifických vlastností oblasti očí při použití chromizačních komponent na jedné straně a jasu na straně druhé. Takto je provedena detekce oblasti očí například v [9].

Nalezení zorničky se dá provést pomocí osvícení scény infračervenými LED diodami. Jsou použity dva zdroje, první v blízkosti osy kamery a druhý vzdálenější od této osy. Je-li zornička osvícena prvním zdrojem, blízkým k ose kamery, světlo se odráží od sítnice a je zachyceno na kameru v podobě světlé zorničky. Pokud je zornička osvícena druhým zdrojem, vzdálenějším od osy kamery, světlo se od sítnice odrazí jiným směrem a kamerou odchycen není. Tímto způsobem detekují zorničku v [8].

Pokud se používá pouze normální osvětlení, pak k detekci zorničky resp. duhovky lze využít jejich tvaru. Detekují se pak kruhové objekty, například jednou z variant Houghovy transformace. Takto je detekce duhovky popsána například v [11].

K určení směru pohledu se často využívá odrazů různých světelných zdrojů od duhovky nebo zorničky. Po zjištění přesné polohy středu zorničky se dopočítá vzdálenost k těmto odrazům. Počet a poloha světelných zdrojů se může lišit. Například v [6] jsou využity čtyři světelné zdroje a dvě kamery.

Kapitola 3

Teoretická východiska

Kapitola detailněji popisuje teoretická východiska, ze kterých jsem čerpal při tvorbě vlastní práce. Na začátku kapitoly je stručný popis technik matematické morfologie a segmentace obrazu využité při detekci očí a duhovky. Vysvětlen je postup detekce hran pomocí Cannyho hranového detektoru. Popsán je také princip Houghovy transformace a jeho využití při detekci kruhových oblastí. Tyto všechny zmíněné postupy se využijí k detekci oblasti, ve které jsou oči a k detekci duhovky v oblasti očí.

3.1 Prahování

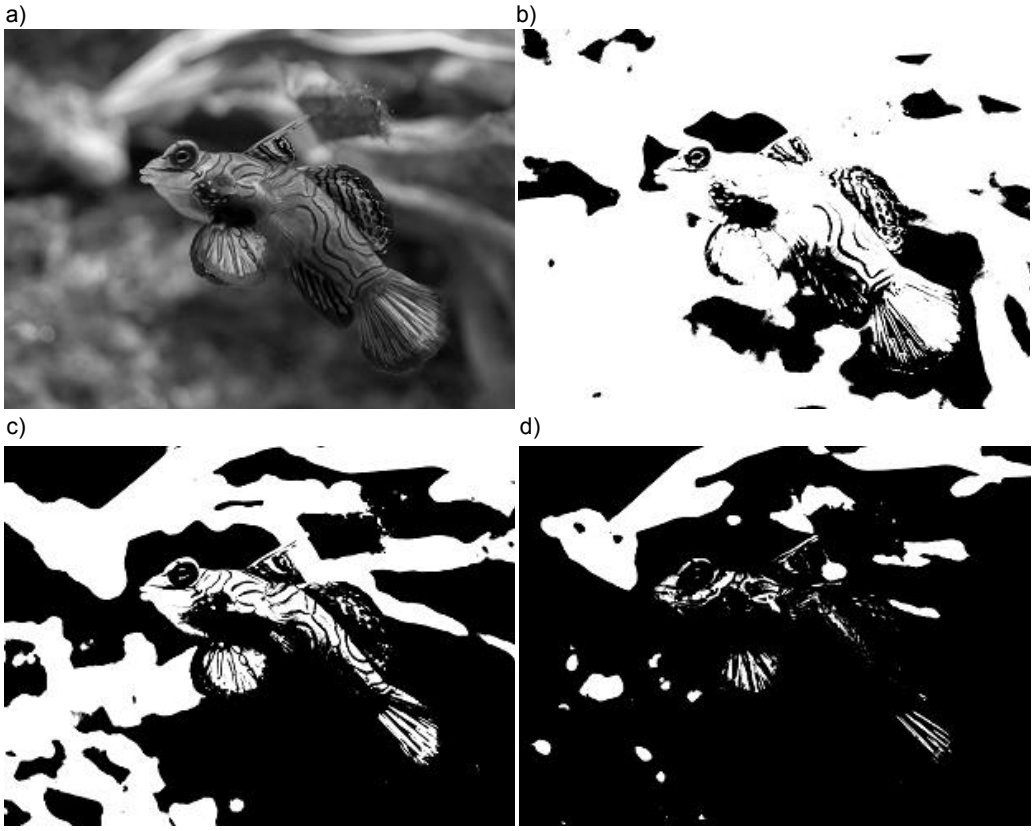
Prahování (thresholding) je nejjednodušší formou segmentace obrazu. Jedná se o operaci, při níž se upravují hodnoty barvy nebo jasu podle následujícího vztahu:

$$f(c) = \begin{cases} A, & \text{pokud} \quad c < \mathit{práh} \\ B, & \text{pokud} \quad c \geq \mathit{práh} \end{cases}, \tag{3.1}$$

kde c je hodnota jasu nebo barvy daného pixelu, *práh* je hodnota, se kterou se porovnává hodnota pixelu a A, resp. B jsou nové hodnoty, které se nastaví aktuálnímu pixelu. Jeden z konkrétních případu takto obecně definované operace je pokud $A = 1$ a $B = 0$. V tomto případě z prahování vznikne binární obraz. Body, které na zdrojovém obrazu mají nižší hodnotu než je práh, budou mít na cílovém obrazu hodnotu 0 a zbylé budou mít hodnotu 1.

Prahování se používá k redukci barevného prostoru z obrazu v odstínech šedi na binární obraz. V binárním obraze body nabývají hodnot 0 nebo 1, jsou tedy černé nebo bílé. Je důležité určit vhodně hodnotu prahu. Vhodný práh se dá například zvolit z jasového histogramu obrazu. Práh může být určen náhodně (toho se využívá při náhodném rozptýlení), nebo hodnoty prahu mohou být určeny z vhodné matice. Na obr. 3.1 příklad prahování na obraze s různými hodnotami prahu.

Jiný případ užití prahování je při určení rozdílu obrazů. Pokud se odpovídající pixely obrazů od sebe odečtou, výsledkem je rozdílová mapa. Při následném vyprahování se získá binární obraz, který ukazuje rozdílová místa v obraze. Změnou prahu v tomto případě se nastaví citlivost, neboli jak velký musí být rozdíl jasu, aby byl zaznamenán do výsledné binární mapy.



Obrázek 3.1: Příklad prahování. Na obrázku a) je zdrojový obraz a zbylé obrazy vznikly prahováním. Obr. b) má práh 50 c) 100 a d) 150. Zdrojový obraz byl převzat z www.wikipedia.org.

3.2 Morfologické operace

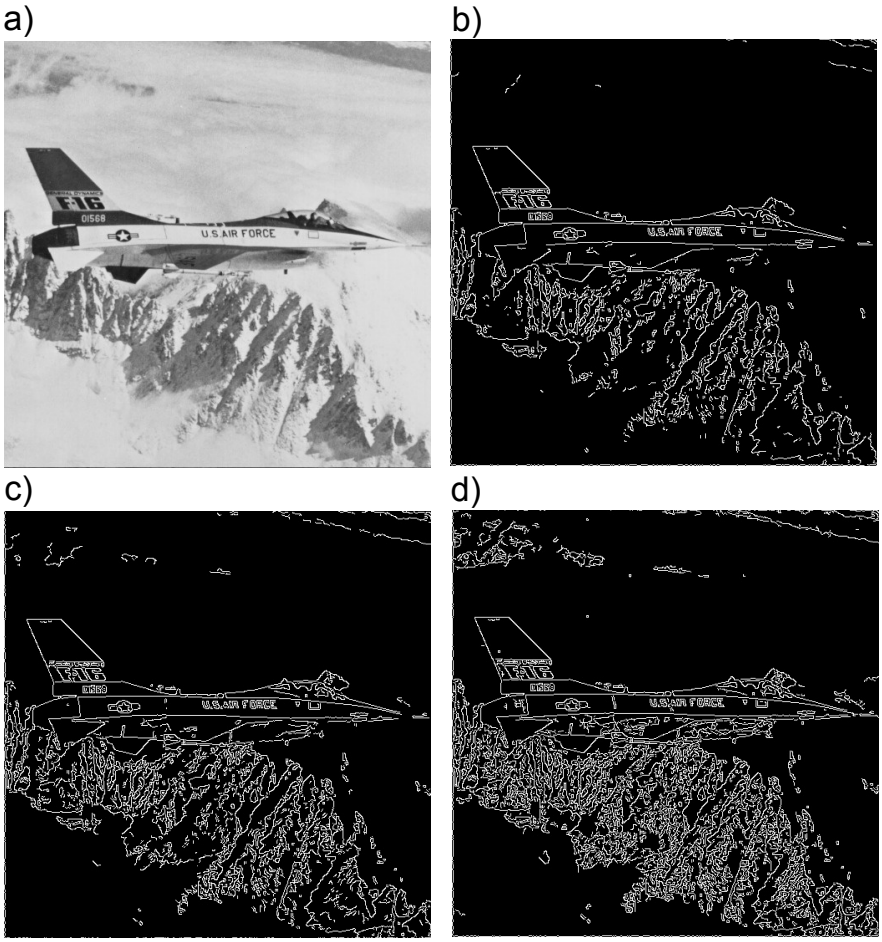
Morfologie je součást mnoha různých vědeckých disciplín obecně zabývající se tvarem. Matematická morfologie v sobě zahrnuje funkce na zpracování obrazu, jeho předzpracování a segmentaci. Operace binární matematické morfologie mají dva operandy. Jedná se o binární obraz, tedy takový, ve kterém jednotlivé pixely tvoří množinu a nabývají pouze dvou hodnot 0 a 1. Druhý operand, tzv. strukturní element je také binární množina, typicky menší než obraz. Ve strukturním elementu má také definovaný lokální počátek. Typický je strukturní element čtverec o velikosti 3x3 s počátkem ve středu. Mezi základní operace binární matematické morfologie patří mimo jiné dilatace a eroze.

Dilatace [10] je operace matematické morfologie, kterou lze vyjádřit vztahem:

$$B \oplus S = \bigcup_{b \in B} S_b, \tag{3.2}$$

kde B je vstupní binární obraz a S je strukturní element, který provádí dilataci. Obraz B se prochází pomocí strukturního elementu. V případě, že počátek strukturního elementu překryje v binárním obraze hodnotu 1, přenese se strukturní element do výstupního obrazu S_b inicializovaného na samé 0.

Dilatace se používá k zaplnění malých děr a úzkých oblastí v obraze. Výsledkem dilatace



Obrázek 3.2: Příklad Cannyho hranového detektoru s různě nastavenými hodnotami prahu pro prahování s hysterezí. Hodnoty pro b) jsou 200 a 150 pro c) 150 a 100 a pro d) 100 a 50.

je zvětšení původního objektu. Kompenzace zvětšení objektů po dilataci pomocí použití eroze se nazývá otevření.

Eroze [10] je operace opačná k dilataci. Lze ji vyjádřit vztahem:

$$B \ominus S = \bigcap_{b \in B} S_{-b}. \tag{3.3}$$

V tomto případě se binární obraz B také prochází pomocí strukturního elementu S. Pokud se překrývá pixel strukturního elementu s hodnotou 1 s odpovídajícím pixelem binárního obrazu s hodnotou 1, přenese se strukturní element do výstupního obrazu.

Eroze se používá ke zjednodušení struktury obrazu a k rozložení objektů na jednodušší oblasti. Výsledkem eroze je objekt menší než původní. Kompenzace tohoto zmenšení se dá provést pomocí dilatace. Tento postup se nazývá otevření.

3.3 Cannyho hranový detektor

Cannyho hranový detektor[5] je, jak název napovídá, algoritmus, který slouží k vyhledání hran v obraze. Tento postup byl vytvořen, aby splňoval zejména tři základní požadavky:

- Minimální počet chyb – musí být nalezeny všechny hrany v obraze, ale nesmí být detekována místa, která hranami nejsou.
- Přesnost – poloha hrany musí být co nejpřesněji určena.
- Jednoznačnost – každá hrana by měla být detekována právě jednou, nesmí docházet ke zdvojení.

Nejprve se obraz zbaví šumu, například Gaussovým filtrem. Následuje standardní detekce hran, například pomocí Sobelova operátoru pro nalezení velikostí gradientů a jejich směrů. Třetím krokem je ztenčení (thinning), při kterém se hledají lokální maxima z hodnot získaných v předešlém kroku. Cílem tohoto kroku je odebrat body, které nejsou maximem. Posledním krokem je prahování s hysterezí, neboli se dvěma prahy. Účel této fáze je odstranění nevýznamných hran a docílení jednoznačnosti detekce. Příklad výsledku Cannyho detektoru hran je na obr. 3.2. Na tomto obrázku je vidět, jak různé hodnoty prahu ovlivňují výslednou detekci.

3.4 Houghova transformace

Houghova transformace [7] je metoda pro nalezení parametrického vyjádření objektu v obraze. Používá se hlavně pro nalezení jednoduchých objektů jako jsou přímky a kružnice. Hlavní výhoda této metody je její odolnost vůči nepravidelnostem, různým přerušením objektu atp.

Vezmeme si nejjednodušší příklad Houghovy transformace, tedy přímku. Jedno z jejich parametrických vyjádření je:

$$r = x \cos\theta + y \sin\theta, \tag{3.4}$$

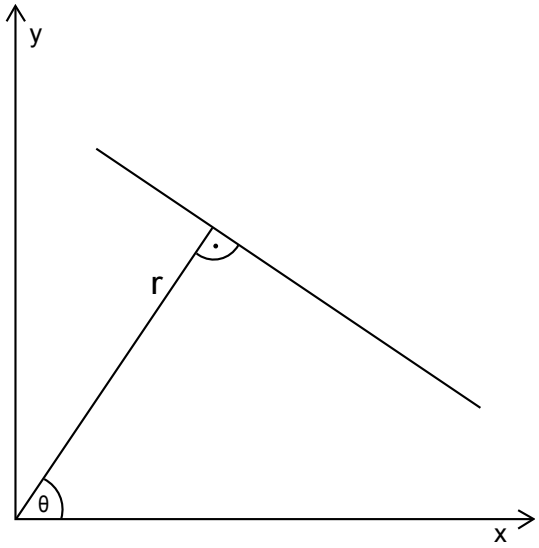
kde r je vzdálenost přímky od počátku souřadnicového prostoru a θ je úhel, který svírá normála přímky s osou x (viz obrázek 3.3.)

Jednotlivé body vstupního obrazu mají parametry x a y, které určují jejich pozici v obraze. Vlastní transformace probíhá tak, že pro každý bod se jeho parametry dosadí do rovnice 3.4. Za proměnnou θ se dosadí hodnoty z intervalu $\langle 0, 2\pi)$. Tento interval je nekonečně velká množina hodnot, v případě reálné implementace se dosazují hodnoty podle nastavení citlivosti Houghovy transformace. Parametr r se následně dopočítá. V parametrickém prostoru takto vznikne pro každý bod jedna křivka. Parametrický prostor je obecně n-rozměrný prostor, kde n je počet parametrů. V případě přímky $n = 2$. Leží-li body v obraze v přímce, potom křivky spočítané pro tyto body se nejčastěji protínají v jednom bodě. Souřadnice tohoto bodu představují parametry θ a r dané přímky.

Obdobný je princip nalezení kružnice. Tady lze využít analytické rovnice kružnice:

$$r^2 = (x - a)^2 + (y - b)^2, \tag{3.5}$$

kde r je poloměr kružnice a bod $[a, b]$ je střed kružnice. Houghovou transformací se v tomto případě hledají tyto tři parametry. Tudíž má parametrický prostor 3 dimenze.



Obrázek 3.3: Parametrický popis přímky

3.5 Normalizovaná korelace

Korelací[12] se chápe vzájemná podobnost mezi procesy, veličinami nebo třeba i obrazy. Korelace dvou obrazů tedy znamená podobnost mezi nimi. Normalizovaná varianta potlačuje různé světelné podmínky na obou obrazech. Normalizovaný korelační koeficient dvou stejně velkých obrazů se dá spočítat podle rovnice 3.6. Výsledkem je číslo v intervalu $\langle 0, 1 \rangle$ udávající podobnost mezi obrazy A a B. Čím je toto číslo větší, tím jsou si oba obrazy podobnější.

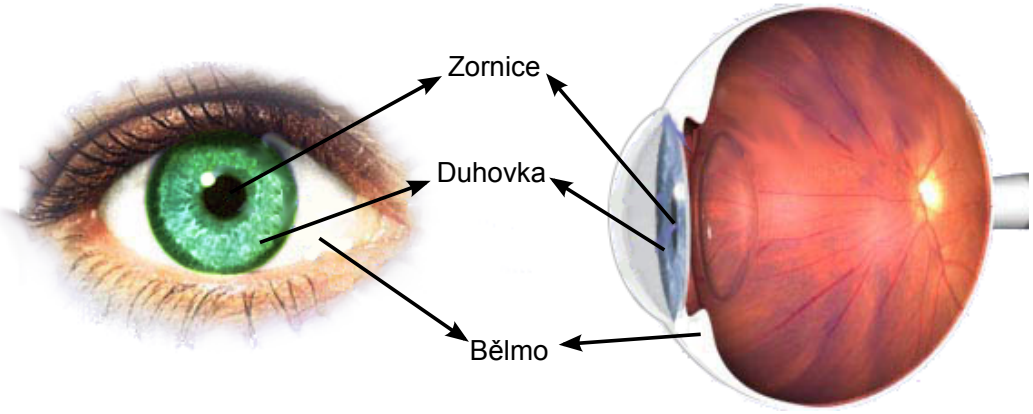
$$NCC = \frac{\sum_{x,y}[A(x,y) \cdot B(x,y)]}{\sqrt{\sum_{x,y} A(x,y)^2 \cdot \sum_{x,y} B(x,y)^2}} \tag{3.6}$$

Zjištění hodnoty korelace obrazů se využívá při technice vyhledávání vzorů (template matching). Účelem této techniky je nalezení šablony v obraze. Pro každou část ve obraze o velikosti šablony se spočítá korelační faktor. V místech, kde korelační faktor vysoký, byla pravděpodobně šablona nalezena. Pokud jsou odlišné světelné podmínky vyhledávané šablony a obrazu, je třeba využít normalizovanou variantu korelace.

3.6 Optický tok

Pohyb objektů v obraze lze sledovat metodou optického toku (optical flow). Každému bodu obraze lze najít dvourozměrný vektor, který vypovídá o směru a rychlosti pohybu mezi snímky. Metoda je výpočetně náročná, ale představuje možnost, jak sledovat různé oblasti například v sekvenci snímků z kamery.

Metoda optického toku vychází z předpokladu, že světelné podmínky se ve snímcích nemění a hodnota jasu pixelu v čase t_1 a t_2 je stejná, ovšem daný pixel je posunut ve směru vektoru optického toku [14].



Obrázek 3.4: Lidské oko.

3.7 Nalezení oblasti očí

Nalezení očí v obraze lze provést pomocí metody autorů Chau a Betke [4]. Při tomto postupu se detekuje mrknutí ve video sekvenci. Každé dva po sobě následujících snímky z kamery se od sebe odečtou. Takto se získá rozdílová mapa. Tato rozdílová mapa se poté vyprahuje. Pro odstranění šumu se provede morfologická operace otevření, tedy eroze obrazu a jeho následná dilatace. Tímto způsobem se získá binární rozdílová mapa, na které se naleznou spojité komponenty. V ideálním případě při mrknutí jsou tyto spojité komponenty dvě – levé a pravé oko.

Z nalezené oblasti očí se následně vytvoří šablona pro vyhledávání vzorů v obraze (template mathing). Toto umožní uživateli určitou možnost se hýbat. Šablona je dále využita pro detekci mrknutí. Při každém nalezení očí s využitím template matching se spočítá podobnost nalezené oblasti s původním vzorem. V případě zavření očí se podobnost sníží. Když naopak uživatel oči otevře, podobnost se zvýší. Detekcí taktových výkyvů se určí, zda uživatel mrknul.

Metoda představuje jednoduchý způsob detekce očí a využívá výhod použití sekvence snímků z kamery. Funguje dobře i na obrazech s nízkým rozlišením.

3.8 Detekce duhovky

Pro určení směru pohledu je důležité přesně nalézt duhovku nebo zorničku. Duhovka je orgán oka, kruhového tvaru, tvořený z vaziva obsahujícího hladké svalstvo, cévy a pigmentové buňky. Pigmentové buňky dávají duhovce různá zabarvení. Zornička je kruhový otvor ve středu duhovky, kterým procházejí paprsky světla dovnitř oka, kde na sítnici tvoří obraz. Svaly duhovky dokáží svými pohyby měnit velikost zorničky. Tím plní funkci světelné clony, tedy regulace množství světla, které pronikne do oka. Schéma a reálný obraz lidského oka je znázorněn na obr. 3.4.

Z hlediska rozpoznání duhovka a zornička jsou dva soustředné kruhy. Při normálním osvětlení je vnitřní černý a vnější barevný. K určení směru pohledu se využije pouze střed těchto kruhů, takže nezáleží na tom, který kruh se detekuje. Takže dále budu pro zjednodušení psát o detekci duhovky, ale pokud takový způsob nalezne zorničku, na řešení to nic nemění.

Pro nalezení duhovky, tedy kruhu, v oblasti, ve které se nachází celé oko, lze použít například metodu popsanou v [2]. Metoda využívá nalezení vektorů gradientů a následné hledání párů těchto vektorů, které jsou opačně orientovány. Jiný způsob detekce kruhových oblastí je nalezení hran, například pomocí Cannyho hranového detektoru a následné použití Houghovy transformace.

Kapitola 4

Návrh řešení

4.1 Požadavky na řešení

Mým cílem je vytvořit knihovnu, která dokáže v sekvenci snímků z kamery nalézt oblast, ve které se nacházejí oči. V oblasti očí nalezne pozici duhovky. Z této pozice se dále dá zjistit, na který bod na obrazovce se uživatel dívá. Hlavní využití pro tuto knihovna bude při vytvoření hry, která se ovládá očima. Z toho vyplývá potřeba určovat směru pohledu v reálném čase.

Hlavním požadavkem tedy bude, aby kamera snímala obličej uživatele nebo jeho část obsahující oči. Snímek dále musí být dobře osvětlený a kontrastní, hlavně v okolí očí. V průběhu ovládání se uživatel nebude moci hýbat. Pouze velmi malé pohyby se budu snažit kompenzovat. K demonstraci funkčnosti rozhraní bude využita jednoduchá hra.

4.2 Nalezení oblasti očí

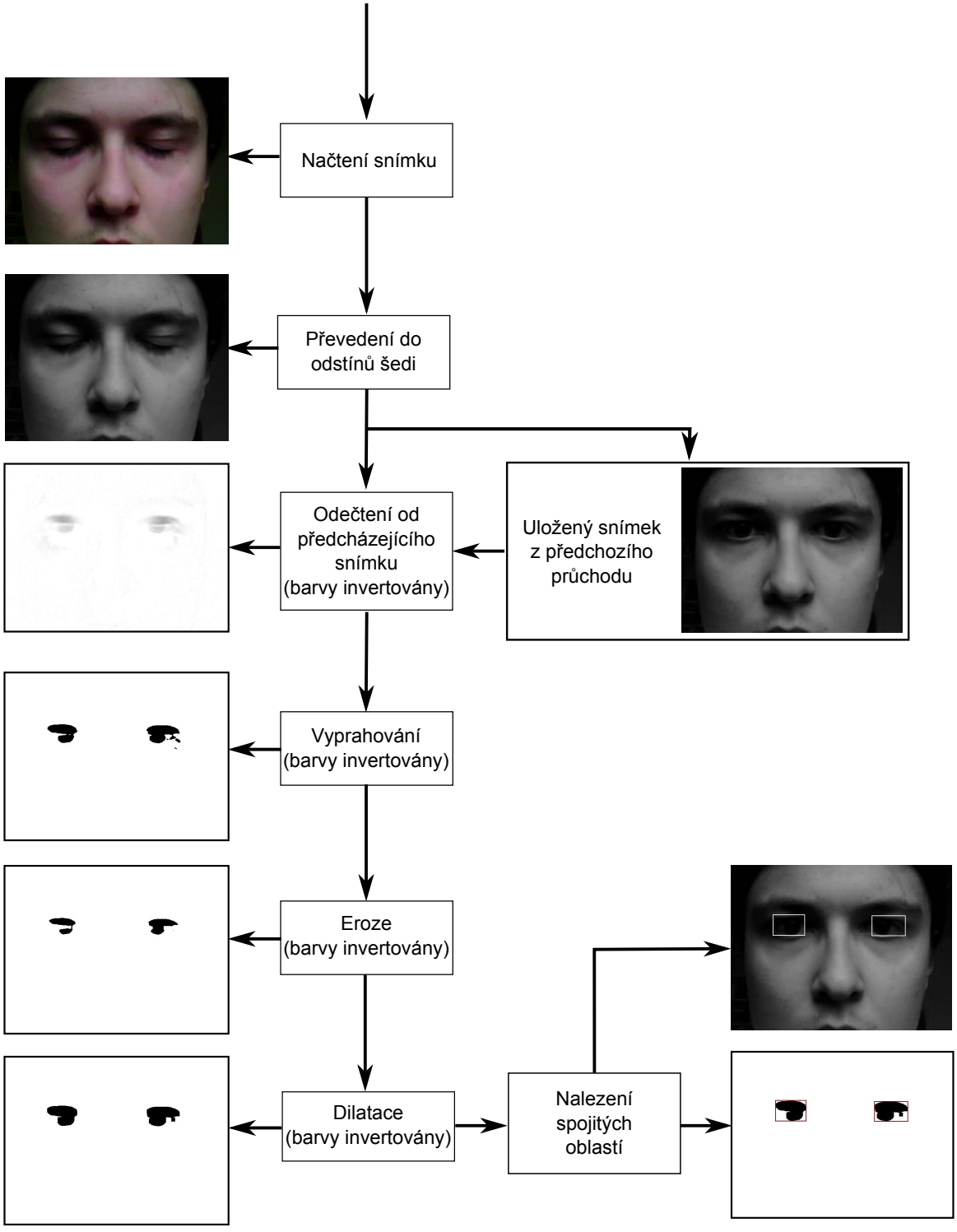
Postup nalezení očí se v podstatných částech shoduje s metodou [4] detailně popsanou v kapitole 3. Tato metoda je použita pouze k detekci očí. Použití k detekci mrknutí a případné použití mrknutí jako ovládacího prvku hry není plně doimplementováno.

Princip tedy spočívá v zjištění rozdílu po sobě následujících snímků. Následuje prahování, které vyústí v binární rozdílovou mapu. Práh je implicitně nastaven na hodnotu 15. Tato hodnota zajistí, že na rozdílové mapě se projeví změny, jako například mrknutí, které se tímto snažím detekovat, ale zároveň jsou potlačeny drobné rozdíly, šum apod. Pokud tato hodnota z nějakého důvodu nevyhovuje, je možné ji změnit.

Na binární rozdílové mapě je následně provedeno otevření a nalezeny spojité oblasti. Detekce spojitých oblastí probíhá rekurzivní metodou podobnou semínkovému vybarvování. Obraz se prochází do nalezení prvního bílého pixelu. Bod, pro který se funkce volá se označí jako zpracovaný. Funkce se pak rekurzivně volá pro pixely sousedící s tímto bílým pixelem. Pokud pixel, pro který je funkce volána černý nebo již zpracovaný funkce končí. Pozice krajních pixelů se zaznamená. Hloubka zanoření rekurze bude omezena. Pokud jsou v oblasti velké plochy značící rozdíl v obrazech, pravděpodobně se nejedná o oblasti očí.

Nalezené oblasti se prohlásí za oblasti očí v případě, že splní jednoduchá pravidla, které vycházejí z lidské anatomie.

- Oblasti jsou dvě
- Oblasti leží v přibližně stejné výšce



Obrázek 4.1: Diagram znázorňující postup nalezení očí. Každá fáze je doprovázena aktuální podobou snímku.

- Oblasti leží v minimální vzdálenosti od sebe
- Oblasti mají určitou minimální velikost

Algoritmus je popsaný na diagramu na obr. 4.1. Pro větší názornost je v každé části ukázka aktuální podoby snímku. Tyto ukázkové snímky byly zmenšeny z původního rozlišení 640x480 bodů.

4.3 Nalezení duhovky

K nalezení duhovky v oblasti oka je využita Houghova transformace, detailněji popsaná v části 3.4. Lepší podmínky pro detekci se dají docílit normalizací oblasti, ve které duhovku hledáme. Vznikne tak mnohem kontrastnější obraz a detekce se zjednoduší.

Na normalizovaném obraze oblasti očí se následně vyhledají hrany pomocí Cannyho hranového detektoru (část 3.3). Prahy pro prahování s hysterezí použitém v Cannyho detektoru se nastaví na pevnou hodnotu. V případě, že toto nastavení nevyhovuje pro detekci hran, je ho možné změnit.

Po nalezení hran se na tomto obraze naleznou kruhové oblasti pomocí Houghovy transformace. Výsledkem je soubor kruhů, nalezených v oblasti ve tvaru souřadnic středu a poloměru.

Pro zlepšení detekce je možné použít uložení šablony duhovky z nalezené oblasti pomocí Houghovy transformace. Je-li nalezeno pomocí Houghovy transformace více kruhových oblastí, spočítá se korelace (část 3.5) mezi šablonou a všemi nalezenými oblastmi. Oblast nejpodobnější se šablonou se prohlásí za duhovku. Jelikož duhovka je odlišena od okolí barvou, je vhodné použít pro toto porovnání barevné obrazy.

4.4 Určení směru pohledu

Pro určení místa, na které se uživatel na obrazovce dívá, se použije lineární interpolace.

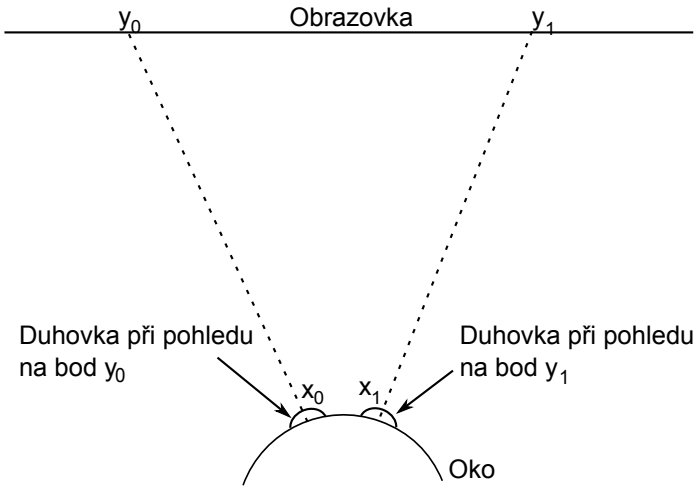
Celý systém zkalibruje tím, že se zaznamená pozice očí uživatele, který se dívá na významné body na obrazovce, například do rohů. Pro zpřesnění a odfiltrování případných chyb se pozice zaznamená po dobu několika snímků a následně se spočítá průměr těchto hodnot. Po zaznamenání pozice duhovky ve všech významných bodech se pozice duhovky pro ostatní body dopočítá podle rovnice 4.1:

$$y = y_0 + (x - x_0)\frac{y_1 - y_0}{x_1 - x_0}, \tag{4.1}$$

kde x značí pozici středu duhovky na obraze a y místo na obrazovce, na které směřuje pohled. Hodnoty x_0 a x_1 značí pozici duhovky při pohledu na významné body a y_0 a y_1 jsou odpovídající souřadnice těchto významných bodů. Tato situace je znázorněna na obr. 4.2.

Hlavní nevýhoda tohoto postupu tkví v tom, že se uživatel nesmí v průběhu kalibrace hýbat. Při vlastním ovládání se ale velmi malé pohyby budou částečně kompenzovat použitím metody optického toku (optical flow). Pokud se tato využije, spočítá se posun oblasti očí od původní pozice, ve které probíhala kalibrace. Tento posun se odečte od aktuálně zjištěné pozice duhovky a teprve poté se zjistí směr pohledu.

Jiný možný přístup k určení směru pohledu je nalezení odrazů na duhovce ze světelných zdrojů. Pak ke kalibraci je možné využít pouze jeden kalibrační bod. Metoda je méně citlivá



Obrázek 4.2: Kalibrace systému. Zaznamenává se pozice duhovky (x_0 resp. x_1) při pohledu na kalibrační body y_0, y_1.

vůči pohybům, speciálně vůči rotaci hlavy. Tento způsob, popsaný například v [6], ovšem vyžaduje umístění zdrojů světla, které se na duhovce budou odrážet a také přesnou detekci těchto jejich odrazů.

Kapitola 5

Implementace

Postupy popsané výše jsem implementoval v jazyce C++ za pomocí knihovny OpenCV [1]. Vlastní demonstrační hru jsem implementoval také v jazyce C++ s použitím knihovny opengl.

5.1 OpenCV

Při implementaci jsem použil knihovnu OpenCV (Open Source Computer Vision Library) společnosti Intel. Tato knihovna implementuje velké množství funkcí, z oblasti zpracování obrazu a počítačového vidění. Knihovna OpenCV je napsána v jazyce C/C++ a funguje pod operačními systémy Windows, Linux a Mac OS X.

Použití této knihovny usnadnilo práci s obrazem a umožnilo zaměřit se na vlastní algoritmus detekce. Nebylo třeba znovu implementovat základní operace s obrazem. V programu byly využity mimo jiné tyto funkce:

`cvThreshold` – funkce, která daný obraz vyprahuje. Podle nastavení posledního parametru se určí, jaký typ prahování se použije.

`cvHoughCircles` – tato funkce implementuje Houghovu transformaci k nalezení kruhových oblastí. V rámci této funkce je také volán Cannyho detektor hran.

`cvDilate` a `cvErode` – provádí dilataci, respektive erozi obrazu.

Důležité jsou také tyto datové typy:

`IplImage` – struktura zapouzdřuje obraz v OpenCV. Kromě vlastních dat daného obrazu obsahuje také informace o velikosti obrazu, jeho typu, počtu kanálů apod.

`CvRect` – struktura vyjadřující obdélník. Parametry `x` a `y` určují pozici levého horního rohu. Parametr `width` značí šířku a `height` výšku obdélníku.

Další informace k jednotlivým funkcím OpenCV lze nalézt buď v dokumentaci na webových stránkách projektu [1], nebo v knize Learning OpenCV [3].

5.2 Popis rozhraní

Implementovaná knihovna s funkcemi pro ovládání očima se dohromady skládá z pěti objektů a hlavičkového souboru. Při použití knihovny se pouze tento hlavičkový přidá do projektu. Objekty knihovny jsou následující:

- `Resolver`
- `Eyes`

- `Eye`
- `Pupil`
- `Blobs`

`Resolver` – objekt zajišťující nalezení očí v obraze. Jako vstup vyžaduje snímek z kamery v podobě datového typu ukazatele na `IplImage` z OpenCV. Snímek se nahrává pomocí metody `loadImg()`, která má právě jeden parametr a tím je daný snímek. Další důležitou metodou objektu Resolver je `findEyes()`. Tato metoda zajišťuje nalezení očí. Jako parametr má ukazatel na objekt `Eyes`. Poslední metoda, o které se zmíním u objektu Resolver, je metoda `OptFlow`, jejíž funkcí je zjištění posunu uložených snímků s pomocí Optical Flow. Parametry typu `double` metody `OptFlow` se po zavolání sčítají s nalezeným posunem. Proto voláním této metody v průběhu snímání kamery se zjistí posun oproti poloze při prvním zavolání.

`Eyes` – objekt zapouzdřující hlavně dva objekty typu `Eye` (neboli levé a pravé oko). Objekt Eye obsahuje metody zajišťující práci s oběma objekty Eye zaráz. Je to například metoda `draw()` na vykreslení očí do obrazu předaném metodě jako parametr. Dále pak `setOff()`, `setXOff()`, `setYOff()`, které nastavují posun objektu vůči první nalezené poloze. Jako parametry mohou tyto metody mít posun zjištěný objektem Resolver pomocí metody `OptFlow()`.

`Eye` – objekt vyjadřující oko. Obsahuje polohu určenou pozicí `x` a `y` na obraze a výškou (parametr `height`) a šířkou (parametr `width`). Nejdůležitější metodou tohoto objektu je `DetectPupil()`. Nalezne duhovku v oblasti tohoto objektu na obraze předaném jako první parametr. Druhým parametrem metody `DetectPupil()` je objekt typu `Pupil`.

`Pupil` – tento objekt vyjadřuje duhovku. Je určen parametry `x` a `y`, které značí polohu středu duhovky v obraze a parametrem `radius`, který vyjadřuje její poloměr.

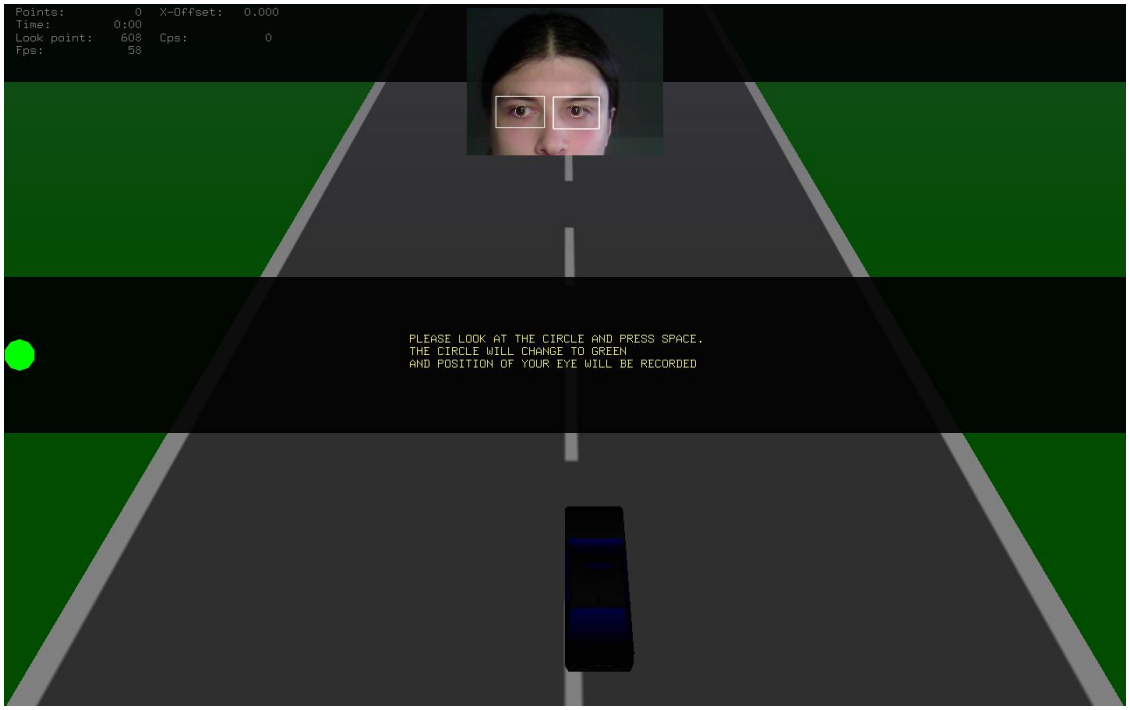
`Blobs` – zde je implementována práce se spojitými komponentami. Zajišťuje funkce pro označení spojitých komponent v binárním obraze. Dále je možné nalezené spojité komponenty spojit, zjistit jestli leží v zadané vzdálenosti od se apod. Tento objekt je využit při detekci očí. Při implementování hry nebo jiného programu s použitím knihovny na detekci očí, nebude tento objekt přímo využit.

Další popis jednotlivých objektů a jejich metod je v dokumentaci vytvořené programem Doxygen.

5.3 Demonstrační hra

V demonstrační hře hráč ovládá pohyb auta, které se vyhýbá překážkám. Cílem hry je dosáhnout co největšího počtu bodů. Body hráč získá jednak za dobu, po kterou jede a také za sbírání bonusových kostek. Naopak, v případě kolize s překážkou hráč část bodů ztrácí. Hra se dá ovládat buď očima, nebo myší. Silnice, po které se auto pohybuje, je rozdělena do tří pomyslných pruhů (levý, pravý a středový). Kurzorem myši nebo pohledem hráč určuje, do které části má auto zamířit.

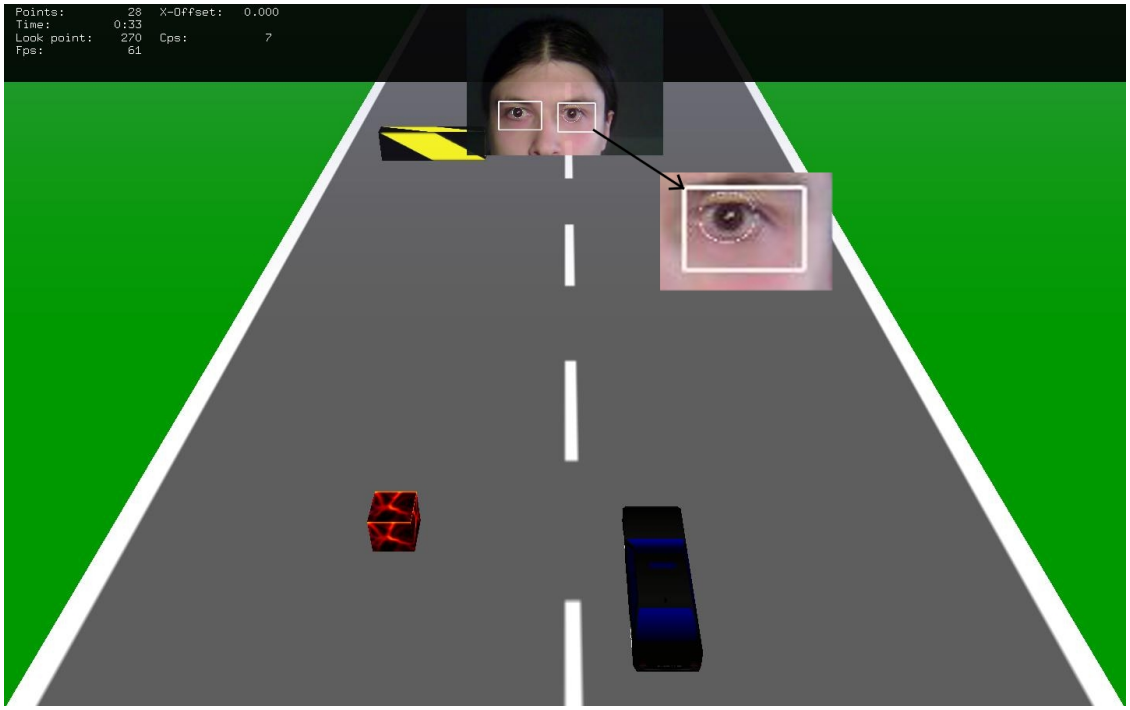
Na začátku hry si uživatel zvolí, jakým způsobem chce hru ovládat. Pokud si zvolí myš, spustí se odpočítávání a za tři sekundy hra začne. Při ovládání očima se na začátku musí nalézt oči a zkalibrovat celý systém. Hráč je tedy požádán, aby mrkal do té doby, než jsou oči nalezeny. Poté je nutné, aby se uživatel podíval do levé části obrazovky na červený kruh. Jakmile se na tento kruh zadívá, musí uživatel zmáčknout mezeru a začne se zaznamenávat pozice očí. Opětovným zmáčknutím mezery se začne kalibrace na opačné straně. Po skončení



Obrázek 5.1: Obrázek ze hry - kalibrační fáze. Uživatel se dívá na zelený kruh.

kalibrační fáze se spouští vlastní hra. V případě špatné kalibrace je možné vynutit novou kalibraci zmáčknutím mezery v průběhu hry. Zmáčknutí klávesy backspace způsobí zrušení nalezené oblasti očí a smazání nastavených šablon pro detekci duhovky. Všechny klávesy, které mají ve hře nějakou funkci, jsou popsány v tabulce 5.1.

Hra byla implementována s pomocí knihovny opengl. Některé funkce, například funkci pro načtení textury z bmp souboru, jsem převzal z ukázkových příkladů z předmětu Počítačová grafika (PGR). Takto převzaté funkce jsou v kódu odlišeny komentáři.



Obrázek 5.2: Obrázek ze hry - vlastní hra. Na obrázku je vidět překážka a bonus, který hráč má sbírat. Je také vidět, že u levého oka nebyl správně určen poloměr duhovky. Střed byl ale určen správně, proto přesnost určení směru pohledu byla zachována. Pravé oko bylo zaměřeno správně.

Klávesa	**Funkce ve hře**
mezera	Při kalibraci posun do další fáze kalibrace, v herní fázi vynucení nové kalibrace
o	Zapnutí / vypnutí optical flow
1, 2 , 3	Změna polohy kamery
m	Na začátku výběr ovládání pomocí myši
e	Na začátku výběr ovládání pomocí očí
escape, x, q	Vypnutí aplikace
backspace	Smaže oblast nalezených očí.
b	Oblast nalezené levé duhovky se vezme jako šablona pro další hledání
n	Oblast nalezené pravé duhovky se vezme jako šablona pro další hledání

Tabulka 5.1: Klávesové ovládání demonstrační hry

Kapitola 6

Návrh testů

Je zapotřebí provést testy ke zjištění funkčnosti a přesnosti určení směru pohledu. Vytvořil jsem tedy jednoduchou testovací aplikaci. Tato aplikace se na začátku zkalibruje na určení směru pohledu. Dále jsou na obrazovce generovány kruhy s náhodnou polohou. Uživatel je požádán, aby se na kruh podíval. Po zmáčknutí mezery se zaznamená směr pohledu a dopočítá se chyba vůči místu, kde je vykreslen kruh. Zjišťován bude pouze směr pohledu v horizontální ose, který je použit v demonstrační hře. Na tomto testu budu sledovat jednak počáteční inicializační fázi, kde si zaznamenám počet mrknutí testovací osoby, do nalezení oči a případný neúspěch této detekce. Dále se při tomto testu zjistí přesnost směru pohledu.

Stejný postup testu bude zopakován celkem třikrát. Poprvé se použije základní způsob detekce. Při druhém opakování se použije optický tok pro kompenzaci pohybů. V poslední části tohoto testu se použije k verifikaci správného nalezení duhovky porovnání se šablonou. Tímto testem se bude zjišťovat, jakým způsobem ovlivní tyto metody přesnost detekce a přesnost určení směru pohledu.

Další test bude věnován vlastní hře. Budu sledovat objektivní i subjektivní porovnání ovládání hry očima a za pomocí myši. Budu zaznamenávat čas, po který uživatel hrál a počet bodů, které získal. Dále nechám uživatele vyplnit dotazník, ve kterém se budu ptát na subjektivní názor na přesnost ovládání očima a který způsob ovládání je zajímavější.

Předpokládám, že ovládání pomocí očí bude v porovnání s ovládáním pomocí myši méně přesné. Hráči budou nejspíš dosahovat méně bodů. Ovládání očima však může být označeno jako zábavnější.

Kapitola 7

Výsledky testů

Testování bylo provedeno na notebooku Dell Inspiron 1420 s konfigurací uvedenou v tabulce 7.1. Testy byly prováděny většinou na půdě Fakulty informačních technologií VUT a převážná většina testujících osob byla studenty této fakulty. Většina také byla mužského pohlaví. Tyto faktory mohly ovlivnit výsledky testu.

Testy byly provedeny na dobře osvětlených místech a parametry kamery (jas, kontrast, sytost barev) byly upraveny podle světelné situace tak, aby na obraze byly dobře viditelné oči. Použitá byla kamera integrovaná do testovacího notebooku a umístěná v rámu nad obrazovkou.

Celkově bylo testování provedeno na vzorku 19 studentů. V barvě očí převažovala hnědá (10 osob) následovaly modré oči (6 osob), 2 studenti měli zelené a jeden šedé oči.

7.1 Přesnost detekce

Před započetím určování směru pohledu je zapotřebí nalézt oblasti očí. Toto se děje mrknutím. Průměrný počet mrknutí, než jsou oči správně detekovány, je 6,2. Stačí relativně krátká doba k tomu, než jsou oči správně nalezeny. Postupně se také počet mrknutí snižuje. Uživatelé postupně zjistili, jakým způsobem musí mrknout, aby systém toto mrknutí detekoval. Zejména bylo důležité, aby se uživatel nehýbal a také, aby mrknutí nebylo doprovázeno pohybem obočí.

Na tabulce 7.2 jsou výsledky testování přesnosti detekce určení směru pohledu. Chyba v této tabulce značí průměrnou vzdálenost v pixelech mezi bodem, který byl vyhodnocen, jako bod, na který se testovací osoba dívá a bodem, na které se ve skutečnosti má dívat. Dále je zde uvedena směrodatná odchylka [13] (σ) spočítaná podle vzorce 7.1:

Procesor	Intel Core2Duo T7250 2,0GHz
Velikost RAM	2GB
Grafická karta	nVidia GeForce 8400M GS
Velikost obrazovky	14,1”
Rozlišení obrazovky	1440 x 900px
Rozlišení kamery	640 x 480px
Operační systém	Windows XP SP3

Tabulka 7.1: Konfigurace sestavy použité na testování

	Chyba [px]	Směrodatná odchylka
Základní detekce	321	103
s použitím TM	342	100
s použitím OF	380	96

Tabulka 7.2: Výsledky testu přesnosti detekce. Byla použita buď základní detekce, dále srovnání duhovky se šablonou (TM) a korekce pomocí optického toku (OF).

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2}, \tag{7.1}$$

kde N značí počet hodnot, x_i je aktuální hodnota a $\bar{x}$ je průměrná hodnota.

Směrodatná odchylka určuje míru rozptylu množiny dat. Při normálním rozložení dat, které je v tomto případě očekáváno, obecně platí, že asi 68% hodnot je v rozmezí do velikosti jedné směrodatné odchylky a přibližně 95% hodnot je od průměrné hodnoty ve vzdálenosti menší než je dvojnásobek směrodatné odchylky.

V tabulce 7.2 je porovnána přesnost základní detekce s detekcí za použití korekce pomocí optického toku (v tabulce značeného OF) a s použitím verifikace detekce duhovky pomocí srovnání se šablonou (v tabulce TM). Obě metody, které byly implementovány s cílem zlepšit a zpřesnit detekci, ovšem dosahují neuspokojivých výsledků. Použití těchto metod většinou detekci naopak zhoršilo. U některých uživatelů ale došlo ke zlepšení, pokud se použilo srovnání se šablonou.

Průměrná chyba je velmi vysoká. Hlavním cílem práce ovšem nebylo zajistit úplnou přesnost určení směru pohledu. I takto vysoká odchylka však může být přijatelná, pokud bude možno ovládat demonstrační hru.

U některých osob ovšem duhovka nebyla správně nalezena. Může to mít spojitost s barvou duhovky. Výsledky nejsou kvůli malému vzorku testovacích osob dostatečně průkazné, ale dá se říct, že popsaná metoda špatně nachází duhovku, pokud její barva příliš splývá s okolními částmi oka nebo obličeje. Toto se dělo hlavně u světle modrých a šedých očí. Detekce duhovky se také z pochopitelných důvodů nepovedla u uživatelů, kteří měli během testu nasazeny brýle, které komplikují detekci duhovky, vznikajícími odrazy obrazovky popřípadě jiných světelných zdrojů.

Celkově lze říct, že správnost a přesnost detekce směru pohledu byla velmi individuální. Záleželo nejspíše na barvě duhovky a dále pak na kvalitě kalibrace. Pokud byl systém špatně zkalibrován, byly výsledky mnohem horší. Důležité také bylo, aby se testovací osoba v průběhu testu nehýbala, což také nebylo často zajištěno.

7.2 Demonstrační hra

V rámci této části testu byly testovací osoby požádány, aby hrály demonstrační hru za pomoci myši a následně pouze očima. Zjišťována byla doba hraní a počet dosažených bodů. Dále uživatelé zaznamenávali své pocity, týkající se přesnosti a zábavnosti obou způsobů ovládání. Hodnocení bylo vyjádřeno známkami 1 - 5, kde hodnota 1 znamenala nejpřesnější resp. nejzábavnější, naopak hodnota 5 značila nejméně přesnou nebo nejméně zábavnou hru.

Přesnost ovládání myší	1,5
Přesnost ovládání očima	2,5
Zábavnost ovládání myší	3,7
Zábavnost ovládání očima	1,4

Tabulka 7.3: Tabulka ukazuje průměrnou známku, kterou uživatelé hodnotili přesnost a zábavnost ovládání očima a myší. 1 znamená nejpřesnější resp. nejzábavnější.

Průměrná délka hraní při ovládáním myší	1 min 21 sek
Průměrná délka hraní při ovládáním očima	3 min 51 sek
Průměrný počet bodů při ovládáním myší	583
Průměrný počet bodů při ovládáním očima	742
Průměrný počet bodů při ovládáním myší za 1 min	193
Průměrný počet bodů při ovládáním očima za 1 min	431

Tabulka 7.4: Tabulka ukazuje průměrnou dobu hraní a průměrný počet bodů při ovládání očima i myší. Dále je uveden počet bodů přepočítaný na 1 min.

Jestli uživatel měl v předchozím testu lepší výsledky přesnosti nalezení duhovky, pokud se použilo srovnání se šablonou, byla tato funkce zapnuta při hraní. Korekce pohybů pomocí optického toku využita nebyla, protože nepodávala přesvědčivé výkony.

Tabulka 7.3 ukazuje výsledky subjektivního hodnocení demonstrační hry uživateli. Je zde dobře vidět, že hra není příliš zábavná, pokud je ovládána myší. Mnohem zábavnější byla pro uživatele hra, pokud ji ovládali očima.

Překvapivě dobrý výsledek má hra při hodnocení přesnosti. Průměrná známka ukazující přesnost ovládání očima je 2,5. Toto je v rozporu s předchozím testem, kdy přesnost určení směru pohledu byla nízká. Nejspíše se tedy projevilo to, že při hraní uživatelé měli zpětnou vazbu v podobě pohybujícího se auta. Uživatelé si patrně dokázali najít způsob, jakým docílit takové akce, kterou zamýšleli udělat.

Větší zábavnost hry dokazuje i tabulka 7.4. Hráči vydrželi hru hrát mnohem delší dobu, pokud hru ovládali očima. Vyšší průměrný bodový zisk při ovládání očima je způsoben právě jenom délkou hraní. Po přepočtení na 1 min hraní se jasně ukáže, že ovládání myší dosahuje vyšších hodnot. I z této tabulky je vidět, že ovládání myší je mnohem přesnější.

Systémové nároky testovací aplikace jsou v tabulce 7.5. Ovládání pomocí očí je sice výpočetně náročnější, ale počet obrázků za sekundu (frames per second, fps) je jenom o málo menší, než při ovládání myší. Tento způsob ovládání tedy bez problémů funguje v reálném čase. Pokud se k vyhledávání duhovky použije šablona (v tabulce značeno TM), náročnost aplikace zvýší jen o málo a plynulost hry se v podstatě nezmění. Proto pokud u

	Ovládání očima				Ovládání myší
	Bez TM i OF	TM	OF	TM i OF	
Zatížení CPU [%]	20 – 22	35 – 37	43 – 46	58 – 61	8 – 12
Využití RAM [MB]	23	26	25	27	14
FPS	53 – 56	45 – 48	32 – 35	30 – 32	60

Tabulka 7.5: Systémové nároky demonstrační hry. Je zde porovnáno ovládání pomocí myši s ovládáním očima, při kterém bylo nebo nebylo využito srovnání se šablonou (TM) a korekce pomocí optického toku (OF).

některého uživatele dojde při použití této metody ke zlepšení, není důvod ji nevyužít. Ovšem při použití optického toku (v tabulce OF) dojde k výraznějšímu snížení fps a aplikace není plynulá. Vezmeme-li v úvahu to, že použití optického toku nepřináší zlepšení, je zbytečné ho použít.

Kapitola 8

Závěr

V této práci byl popsán možný způsob detekce směru pohledu a jeho využití při hraní počítačových her. Vznikla knihovna implementující funkce pro detekci očí v obraze, detekci zorničky a určení směru pohledu. Obsahuje i funkci na korekci pohybů pomocí optického toku a funkci na verifikaci nalezené duhovky pomocí šablony. Vše bylo zkoušeno na obyčejné kameře s nízkým rozlišením.

V průběhu testu bylo zjištěno, že detekce směru pohledu neprobíhá dostatečně přesně. Naproti tomu demonstrační hru byli uživatelé schopni ovládat. Subjektivně byla přesnost ovládání očima hodnocena relativně dobře.

Použité metody detekce duhovky a určení směru pohledu jsou ovlivňovány dvěma zásadními faktory. Je to zejména přesnost detekce duhovky. Zejména světlejší duhovky nejsou správně detekovány. Z důvodu této v některých případech špatné detekce byla přidána možnost vzít šablonu duhovky a použít ji k ověření detekce. Toto ovšem nepřineslo žádné průkazné výsledky. Druhým faktorem je omezení pohybu osoby, která ovládá tímto způsobem počítač. Častý jev je přivírání očí, potom není dobře na obraze vidět duhovka. Dále, v podstatě všechny testující osoby, zvlášť při hraní hry, se při ovládání pohybovaly, otáčely hlavou a nakláněly ji.

Pokud je duhovka detekována správně a pokud je věnována dostatečná pozornost přesné kalibraci, detekce směru pohledu může být celkem přesná.

Detekce oblasti očí funguje dobře, rychle a s dostatečnou přesností nalezne oblasti očí. Této detekce lze využít i jinak, než bylo popsáno. Oblast očí je dobře rozpoznatelná v obraze a lze ji celkem přesně sledovat pomocí optického toku. Toto může být využito k dalším způsobům ovládání hry, třeba pohybem hlavy, jejím nakláněním apod. Možné je také dopracování detekce mrknutí a použití mrknutí jako dalšího ovládacího prvku.

Hlavní možností zlepšení je nalezení lepšího způsobu detekce duhovky nebo zorničky v oblasti očí. Použití Houghovy transformace k detekci kruhových objektů nepřineslo dostatečně přesné výsledky.

Literatura

[1] Dokumentace ke knihovně OpenCV. [online], Naposledy navštíveno 18. 4. 2009.
URL http://opencv.willowgarage.com/wiki/

[2] Ajdari Rad, A.; Faez, K.; Qaragozlou, N.: Fast Circle Detection Using Gradient Pair Vectors. In *DICTA*, CSIRO Publishing, 2003, s. 879–888.

[3] Bradski, G.; Kaehler, A.: *Learning OpenCV: Computer Vision with the OpenCV Library*. Cambridge, MA: O'Reilly, 2008.

[4] Chau, M.; Betke, M.: Real Time Eye Tracking and Blink Detection with USB Cameras. Technická zpráva, Massachusetts Institute of Technology, December 2005.
URL http://www.cs.bu.edu/techreports/pdf/2005-012-blink-detection.pdf

[5] Forsyth, D. A.; Ponce, J.: *Computer Vision: A Modern Approach*. Prentice Hall, August 2002, ISBN 0130851981, 224–234 s.

[6] Guestrin, E. D.; Eizenman, M.: Remote point-of-gaze estimation requiring a single-point calibration for applications with infants. In *ETRA '08: Proceedings of the 2008 symposium on Eye tracking research applications*, New York, NY, USA: ACM, 2008, ISBN 978-1-59593-982-1, s. 267–274,
doi:http://doi.acm.org/10.1145/1344471.1344531.

[7] Johnson, M.: The Hough Transform. [online], Naposledy navštíveno 14. 5. 2009.
URL http://cs-alb-pc3.massey.ac.nz/notes/59318/l11.html

[8] Morimoto, C. H.; Koons, D.; Amir, A.; aj.: Pupil detection and tracking using multiple light sources. *Image and Vision Computing*, ročník 18, č. 4, 2000: s. 331 – 335, ISSN 0262-8856.
URL http://www.sciencedirect.com/science/article/B6V09-3YKKDPY-8/2/ab357cc181c8eff57c29bb537226de14

[9] Nasiri, J.; Khanchi, S.; Pourreza, H.: Eye Detection Algorithm on Facial Color Images. In *Asia International Conference on Modelling and Simulation*, May 2008, s. 344–349.

[10] Shapiro, L. G.; Stockman, G. C.: *Computer Vision*. Prentice Hall, January 2001, ISBN 0130307963.

[11] Toennies, K.; Behrens, F.; Aurnhammer, M.: Feasibility of Hough-transform-based iris localisation for real-time-application. 2002, ISSN 1051-4651, s. 1053–1056 vol.2, doi:10.1109/ICPR.2002.1048486.

[12] Wikipedia: Cross-correlation. [online], Naposledy navštíveno 13. 5. 2009.
URL http://en.wikipedia.org/wiki/Cross-correlation

[13] Wikipedia: Standard deviation. [online], Naposledy navštíveno 14. 5. 2009.
URL http://en.wikipedia.org/wiki/Standard_deviation

[14] Španěl, M.: *Rozpoznávání gest ve video sekencích*. Diplomová práce, Fakulta informačních technologií, Vysoké učení technické, 2003.

Příloha A

Obsah CD

K práci je přiloženo cd. Obsahuje zdrojové kódy knihovny, demonstrační hry a aplikace zjišťující přesnost detekce. Dále obsahuje plakát, kterým je práce doplněna a tuto technickou zprávu jako pfd i ve formě zdrojových kódů.

Adresářová struktura cd je následující:

/Doc/	Dokumentace k projektu vytvořená v programu Doxygen.
/Latex/	Tato technická zpráva ve formě pdf a zdrojových kódů v Latexu.
/Plakat/	Složka obsahující stručný plakát prezentující práci.
/Src/	Složka obsahuje zdrojové kódy.
/Src/EyesLib/	Vlastní knihovna.
/Src/Test/	Testovací aplikace k určení přesnosti detekce směru pohledu.
/Src/Game/	Demonstrační hra.

Příloha B

Plakát

K prezentaci práce byl vytvořen stručný plakát. Je obsažen na přiloženém cd a jeho náhled je na obr. B.1

Obrázek B.1: Náhled plakátu prezentujícího tuto práci.