VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Detekce míry otevření očí a frekvence mrkání ve videosekvencích

Detecting the Measure of Eye Opening and the Frequency of Blinking in Videosequences

2016 Bc. Miroslav Sedlák

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. Miroslav Sedlák**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Detekce míry otevření očí a frekvence mrkání ve videosekvencích
Detecting the Measure of Eye Opening and the Frequency of Blinking in Videosequences

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem diplomové práce je vytvořit software pro detekci míry otevření očí a stanovení frekvence mrkání. Software je vytvářen s představou jeho využití v automobilu pro stanovení míry únavy řidiče. Obrázky, s nimiž bude software pracovat, jsou ve stupních šedi. Jsou pořízené za běžného nebo případně infračerveného osvětlení.

V diplomové práci proveďte následující:
1. Navrhněte/zvolte vhodnou metodu nalezení oblasti očí.
2. Navrhněte/zvolte vhodnou metodu detekce očních víček.
3. Navrhněte vhodnou metodu sledování polohy víček v čase a metodu pro stanovení mrkání.
4. Navržené řešení realizujte v C/C++.
5. Řešení řádně experimentálně prověřte.

Způsob a výsledky řešení popište v textové části práce.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Dr. Ing. Eduard Sojka**

Datum zadání: 01.09.2014

Datum odevzdání: 29.04.2016

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prehlasujem, že som túto diplomovú prácu vypracoval samostatne. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

V Ostrave, 29.4. 2016 .....Sedlák.....

Týmto sa chcem poďakovať predovšetkým vedúcemu mojej práce, doc. Dr. Ing. Eduardovi Sojkovi, za odborné rady, trpezlivosť a ochotu pri vedení práce. Tiež chcem poďakovať svojej rodine a blízkym za neustálu podporu.

Abstrakt

Táto diplomová práca je zameraná na návrh metód detekcie miery otvorenia očí a frekvencie žmurkania. K detekcii stavu oka využívam detekciu dúhovky, očných viečok a klasifikáciu pomocou natrénovaného SVM. Ich kombináciou som zlepšil detekciu žmurkania. Z detekovanej vzdialenosti očných viečok je určená miera otvorenia očí. Navrhnuté metódy sú experimentálne overené na štyroch videách s tromi subjektami pri rôznych svetelných podmienkach. Výsledkom práce je implementácia navrhnutých metód v jazyku C++.

Klíčová slova: detekcia tváre, detekcia očí, SVM, HOG, očné viečka, frekvencia žmurkania, miera otvorenia očí

Abstract

This master thesis is focused on designing the methods for detection of the measure of eye opening and frequency of blinking. Detection of iris, eyelids and classification using SVM are used to detect eye states. By combining these detections, eye blinking detection is improved. The measure of eye opening is determined from detected distance of eyelids. Proposed methods were experimentally evaluated in four videosequences of three different subjects in varying lighting conditions. Implementation is done in C++.

Key Words: face detection, eyes detection, SVM, HOG, eyelids, frequency of blinking, measure of eye opening

Obsah

Zoznam použitých skratiek a symbolov

SVM	–	Support Vector Machine
HOG	–	Histogram of Oriented Gradients
HSI	–	Hue Saturation Intensity
SIFT	–	Scale Invariant Feature Transform
GPU	–	Graphics Processing Unit
FMM	–	Fast Marching Method

Zoznam obrázkov

Zoznam tabuliek

1 Úvod

Cieľom tejto diplomovej práce je návrh metódy a softwaru pre detekciu očných viečok, miery otvorenia očí a frekvencie žmurkania vo videosekvenciách. Takýto software sa dá využiť v programoch pre monitorovanie vodiča automobilu na stanovenie miery únavy pri riadení vozidla.

Každým rokom na vozovkách pribúda veľké množstvo motorových vozidiel. Je prirodzené, že s tým súvisí aj nárast dopravných nehôd. V dnešnom svete plnom stresu a časovej tiesni, medzi hlavné príčiny dopravných nehôd, okrem samotnej nepozornosti a zlom odhade vlastných schopností, patrí aj samotná únava za volantom a teda dobre známy mikrospánok najmä profesionálnych vodičov. Veda a pokrok idú stále dopredu a teda aj na túto situáciu môžeme dnes nájsť čiastočné riešenie.

Do automobilov sa postupne dostávajú nové technológie, ktoré nás dokážu upozorniť nie len na dianie okolo nášho vozidla, ale aj na naše správanie. V moderných autách existujú systémy, ktoré sledujú vozovku a oznámia nám, ak vybočíme z jazdného pruhu, alebo ideme príliš rýchlo a pred nami je prudká zákruta. V takom prípade vodiča upozornia zvukovovizuálnym znamením, prípadne vibráciami v sedadle. Častokrát však samotný impulz prichádza neskoro alebo nevhodne. Kombinácia sledovania vozovky a správania vodiča by mohla viesť k presnejšej detekcii miery únavy a tým predchádzať zbytočným nehodám. Navrhnutý software by mohol nájsť uplatnenie najmä v medzinárodných dopravných prepravách, kde vodiči trávia prevažnú časť dňa aktívne za volantom a častokrát nerešpektujú zákonom stanovené pauzy kvôli vyššie spomenutej časovej tiesni.

Celkovo je diplomová práca členená do 6 kapitol. Najdôležitejšou úlohou v predkladanej práci sú samotné metódy detekcie očných viečok a určenie miery otvorenia očí. Aj preto popis a ozrejmenie týchto metód tvorí väčšiu časť tejto diplomovej práce.

V úvode, a teda v prvej kapitole v krátkosti popisujem cieľ práce, využitie asistenčných systémov v automobiloch a rozpisujem štruktúru práce. Ďalšie tri kapitoly sú tvorené literárnym prehľadom zaoberajúcim sa danou oblasťou, a tiež popisom navrhovaných metód detekcie. V druhej kapitole uvádzam prehľad súčasných metód venujúcich sa detekcii polohy očných viečok, detekcii žmurkania a určovaniu miery otvorenia očí. Kapitola tri obsahuje návrh metódy detekcie tváre a očí, kde bližšie popisujem Haarové kaskádové klasifikátory a ich využitie v detekcii. V štvrtej kapitole sa venujem samotnej detekcii miery otvorenia očí, kde popisujem trénovanie detektora za pomoci *Support Vector Machine* (SVM) a *Histogram of Oriented Gradients* (HOG), ktorý slúži na rozoznanie otvorených a zatvorených očí. Ďalej prezentujem metódy pre odstránenie nežiadúcich odrazov svetla z oblasti očí. Potom sa venujem detekcii dúhovky modifikovanou Houghovou transformáciou s využitím smeru gradientu a detekcii polohy očných viečok, kde využívam polohu detekovanej dúhovky. Spojením týchto krokov sa pokúsim zlepšiť detekciu stavu očí a samotného určenia miery otvorenia očí. V poslednej časti štvrtej kapitoly píšem o detekcii frekvencie žmurkania. Piata kapitola tvorí experimentálnu časť, kde sa venujem testovaniu algoritmov navrhnutých v predchádzajúcich kapitolách diplomovej práce. V poslednej, šiestej

kapitole, zhrniem, ako sa mi podaril splniť cieľ diplomovej práce a taktiež navrhnem oblasti, kde a ako by sa program mohol v ďalšej práci vylepšiť.

2 Súčasný stav v danej oblasti

Existuje viacero techník, ktorými je možné zistiť stav oka (otvorené alebo zatvorené). Jednou z techník je využitie *Hue Saturation Intensity* (HSI) farebného modelu pri rozlišovaní pixelov patriacich oku, a tým rozlíšiť stav očí [1].

Niektoré metódy využívajú porovnávanie šablón. Táto technika spočíva v porovnávaní oblasti záujmu so šablónou otvoreného oka. Chau a Betke [2] použili k detekcii žmurknutia normalizovanú koreláciu medzi šablónou otvoreného oka a oblasťou oka v aktuálnom snímku videa. V reálnom čase dosiahli celkovú presnosť detekcie žmurnkutí 95% na videách s rozlíšením 320×240.

Awais a kol. [3] túto metódu upravili a používajú šablónu získanú v prvom snímku videa. Dosiahli presnosť detekcie 99,6%. Výhodou týchto techník je nízka výpočtová náročnosť. Avšak tieto metódy dokážu detekovať iba zatvorené a otvorené oko, nie stavy medzi tým.

V roku 2000 Tian a kol. [4] predstavili techniku, ktorá najskôr detekuje stav oka hľadaním dúhovky. Dúhovku detekujú pomocou polkruhovej masky, pretože horná časť dúhovky je väčšinou prekrytá očným viečkom. Ak je dúhovka detekovaná, oko je považované za otvorené a hranice oka sú určené stredovými bodmi na očných viečkach. Ak je oko zatvorené, hranicu oka tvorí čiara medzi vnútorným a vonkajším kútikom oka. Dobré výsledky dosiahli v 493 sekvenciách obrázkov z 500.

Yang a kol. [5] sa zaoberajú vytvorením šablóny oka pomocou dvoch parabolických kriviek. Úlohou tohto algoritmu je deformovanie šablóny, aby čo najlepšie zodpovedala oku. Presnosť testovali na dvoch videách. Na videu, kde bol subjekt ospalý dosiahli senzitivitu detektora $87,3\%$ a špecifickosť $92,7\%$. Na druhom videu bol subjekt v bdelom stave a autori uvádzajú senzitivitu 94% a špecifickosť $89,4\%$.

Lalonde a kol. [6] navrhli metódu detekcie žmurknutia, ktorá využíva detekciu pohybu v oblasti záujmu. Táto oblasť je na začiatku zarovnaná tak, aby obsahovala oblasť očí. Táto oblasť je udržiavaná sledovaním bodov *Scale Invariant Feature Transform* (SIFT) príznakov, ktoré sú použité pri odhadovaní afinnej transformácie medzi počiatočnou polohou tváre a polohou tváre v ďalších snímkoch. Na urýchlenie sledovania SIFT príznakov je použitá *Graphics Processing Unit* (GPU).

Pan a kol. [7] použili silný klasifikátor na detekciu žmurknutia a stupňa zatvorenia oka. Prechod oka z otvoreného stavu do zatvoreného stavu modelujú skrytým Markovovým modelom. Metóda beží v reálnom čase na obrázkoch s rozlíšením 320×240, ktoré sú získavané pomocou webkamery. Uvádzajú priemernú presnosť detekcie žmurknutí $95,7\%$ na dvoch očiach a $88,8\%$ na jednom oku. Pri dvoch očiach je presnosť vyššia vďaka predpokladu, že pri prirodzenom žmurknutí sú zatvorené obe oči.

Liu a kol. [8] najskôr v oblasti oka hľadajú očné kútiky. Definujú filtre, pomocou ktorých nájdu kandidátov. Z nich následne vyberú len silné rohy s použitím postupu navrhnutého Harrisom a Stephensom [9]. Potom opäť pomocou filtrov detekujú očné viečka v oblasti medzi detekovanými kútikmi oka. Táto oblasť je najskôr rozdelená na 5 zvislých častí. V každej časti

sa aplikuje navrhnutý filter, pričom maximálna odozva filtra zodpovedá bodu na hornom viečku a minimálna odozva bodu na dolnom viečku. Z detekovaných bodov určia mieru otvorenia očí. Porovnaním s prahovou hodnotou ďalej určujú stav oka a únavu vodiča.

3 Detekcia tváre a očí

Detekcia tváre v obraze je veľmi dôležitá. Je to prvý krok napríklad pred rozpoznávaním tváre, alebo v monitorovacích systémoch. Dnes je detekcia tváre stále viac rozšírená. Môžeme ju vidieť na sociálnych sieťach, v mobilných telefónoch, ktoré sa dajú odomknúť rozpoznaním tváre, alebo tiež vo fotoaparátoch a kamerách. Správna detekcia tváre môže byť obtiažna kvôli rôznym faktorom ako sú pozícia hlavy, výraz tváre, čiastočné prekrytie tváre iným objektom alebo prvky tváre (brada, fúzy, okuliare).

V mojej práci najskôr detekujem oblasť tváre a v nej následne oči. Tak sa vyhnem nesprávnym detekciám v oblasti mimo tváre. K detekcii tváre a očí využívam kaskádové klasifikátory Haarových príznakov, ktoré predstavili v roku 2001 P. Viola a M. Jones [10].

3.1 Kaskádové klasifikátory Haarových príznakov

Každá tvár má typické črty, ktoré sa prejavia v obraze ako zmena v jase. Toto využili aj P. Viola a M. Jones pri detekcii tváre pomocou Haarových príznakov. K detekcii je použité posuvné okienko, ktorému sa postupne mení veľkosť a pozícia. Toto okienko sa skladá z čiernych a bielych obdĺžnikov. Príklad Haarových príznakov môžeme vidieť na obrázku 4. Kaskádové klasifikátory mávajú typicky 20 – 30 fáz. Každá fáza je silný klasifikátor, vytvorený z viacerých slabých klasifikátorov. Slabý klasifikátor označí podoblasť obrazu, x, ako objekt porovnaním f s prahovou hodnotou θ.

$$h(x) = \begin{cases} 1 & f \cdot p \geq \theta \cdot p \\ -1 & inak \end{cases} \tag{1}$$

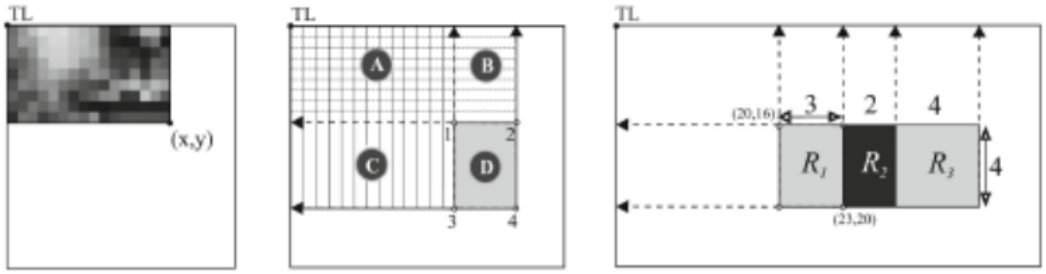
Sú nazývané slabými klasifikátormi, pretože očakávaná kvalita ich detekcie je len o niečo vyššia ako náhodné hádanie.

Každá z fáz kaskádového klasifikátora obsahuje postupne viac príznakov a má presnejšie prahové hodnoty, na základe ktorých sa rozhoduje, či je daný príznak prítomný. Ak v prvej fáze klasifikátor nedetekuje danú oblasť obrazu ako hľadaný objekt, algoritmus môže túto oblasť preskočiť. Na obrázku 1 môžeme vidieť aplikáciu rôznych Haarových príznakov. Ak všetky fázy úspešne prejdú, daná oblasť je klasifikovaná ako hľadaný objekt.

Pri každom príznaku sa musí dokola počítať súčet pixelov v kontrolovanej oblasti. Tu prichádza na radu caching. Na zistenie prítomnosti Haarových príznakov sa využívajú integrálne obrazy. Jedná sa o vytvorenie tabuľky súčtu plôch pôvodného obrazu [15]. Integrálna hodnota $I(p)$ pre pixel $p = (x, y)$ je súčet všetkých hodnôt $P(q)$ pixelov $q = (i, j)$, pričom q je naľavo a hore od p.

$$I(p) = \sum_{\substack{1 \leq i \leq x \\ 1 \leq j \leq y}} P(i, j) \tag{2}$$

Obr. 1: Príklad aplikácie Haarových príznakov



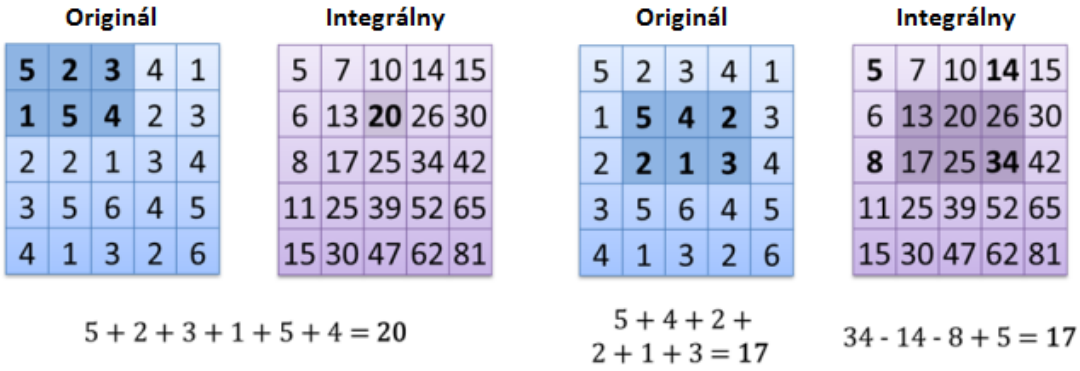
Obr. 2: Príklad výpočtu súčtu pixelov z integrálneho obrazu (upravené podľa [16])

Majme obdĺžnik D, ktorého rohy sú v bodoch p_1, p_2, p_3, p_4 (viď obrázok 2). Súčet hodnôt pixelov v D zistíme z integrálneho obrazu.

$$I(D) = I(p_4) - I(p_3) - I(p_2) + I(p_1) \tag{3}$$

Na obrázku 3 je príklad vytvorenia integrálneho obrazu. Takisto v ňom môžeme vidieť postupy pre získanie súčtu pixelov časti obrazu z vytvoreného integrálneho obrazu použitím vzorca 3. Použitím integrálneho obrazu znížime zložitosť výpočtu Haaroveho príznaku z $O(n^2)$ na $O(1)$.

Túto techniku môžeme použiť pri efektívnom výpočte Haaroveho príznaku oblasti $R_1R_2R_3$. Hodnotu zistíme vypočítaním $I(R_i)$ a prenásobením váhami $\omega_i > 0$. Pre regióny R_1 a R_3 sú ω_1 a ω_3 kladné, naopak pre R_2 je ω_2 záporná. Tieto váhy bývajú bežne priradené obdĺžnikom v



Obr. 3: Príklad integrálneho obrazu

Haarových príznakoch tak, aby spĺňali:

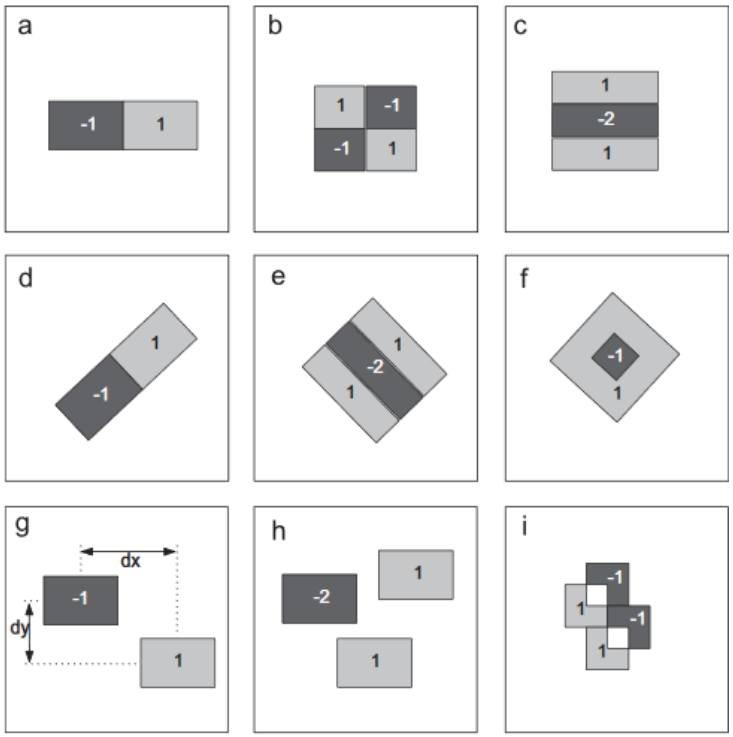
$$\sum_{i=1}^{k} \omega_i = 0 \tag{4}$$

Napríklad obdĺžniky Haaroveho príznaku z obrázku 4(a) majú priradené váhy 1 a -1. Podobne obdĺžniky Haaroveho príznaku z obrázku 4(c) alebo 4(e) majú priradené váhy $1, -2$ a 1. Na obrázku ďalej môžeme vidieť, že hlavným rozdielom medzi rôznymi Haarovými príznakmi je počet obdĺžnikov a orientácia obdĺžnikov vzhľadom k šablóne[11].

3.2 Detekcia pomocou kaskádových klasifikátorov

Pri implementácii som použil knižnicu OpenCV [12], ktorá už obsahuje niekoľko predtrénovaných klasifikátorov. Tie dokážu detekovať tvár, pokiaľ nie je veľmi naklonená.

Najskôr pomocou kaskádového klasifikátora Haarových príznakov detekujem tvár. Použil som predtrénovaný klasifikátor zo súboru *haarcascade_frontalface_alt2.xml*, ktorý je natrénovaný z tvárí za rôznych podmienok (rôzne osvetlenie, okuliare, brada, fúzy, výraz tváre). Tento krok je veľmi dôležitý. Zabránim tak falošným detekciám očí mimo oblasti tváre. Pretože oči sa nachádzajú v hornej časti, oblasť tváre následne orežem o 15% zhora a výšku zmenším na 40% pôvodnej veľkosti. Týmto zmenšením detekovanej oblasti urýchlim ďalšiu detekciu očí. Ľavé a pravé oko detekujem zvlášť. Používam k tomu opäť predtrénované klasifikátory. Pre ľavé oko používam klasifikátor zo súboru *haarcascade_lefteye_2splits.xml* a pre pravé oko *haarcascade_righteye_2splits.xml*. Tieto klasifikátory boli natrénované z viac ako 6000 vzoriek. Pozitívne vzorky pochádzajú z databáz FERET, BioID a VALID [13]. Pred samotnou detekciou ešte raz orežem detekovanú oblasť tváre. Tentokrát však na 70% šírky. Pravé oko tak detekujem iba v ľavej časti detekovanej oblasti. To nie len urýchli detekciu, ale takisto presnosť detekcie. Nemôže sa stať, že by nám klasifikátor detekoval ľavé oko, keď zrovna detekujeme pravé.



Obr. 4: Haarove príznaky so štandardnými váhami (upravené podľa [11])

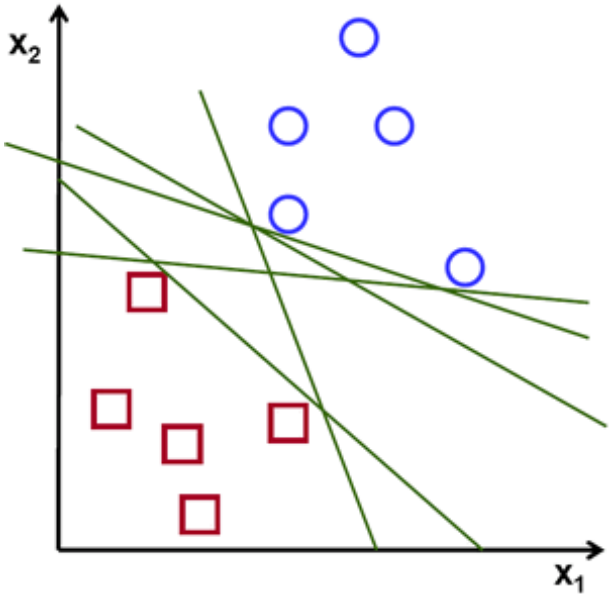
Pri detekcii pomocou kaskádových klasifikátorov v OpenCV môžeme z výstupných parametrov funkcie *detectMultiScale()* získať váhy detekcií. Ak by teda klasifikátor detekoval viacero objektov, za oko považujem ten, ktorý má najväčšiu váhu.

4 Detekcia otvorenia očí

Detekcia otvorenia očí sa skladá z troch krokov. Najskôr s použitím svojpomocne natrénovaného SVM detektora zistím stav očí. Následne sa snažím v oblasti očí nájsť dúhovku bez ohľadu na to, či je oko otvorené, alebo zatvorené. Ďalej detekujem pozíciu očných viečok. Na základe vzdialenosti očných viečok, detekovanej dúhovky a stavu očí zistenom SVM detektorom, určím výsledný stav očí. Oko považujem za otvorené, ak aspoň dve z detekcií určia oko ako otvorené. Ak by teda SVM detektor nesprávne klasifikoval oko ako zatvorené, je stále možné nájsť dúhovku a na základe vzdialenosti očných viečok a detekovanej dúhovky určiť správny stav oka.

4.1 Support Vector Machine

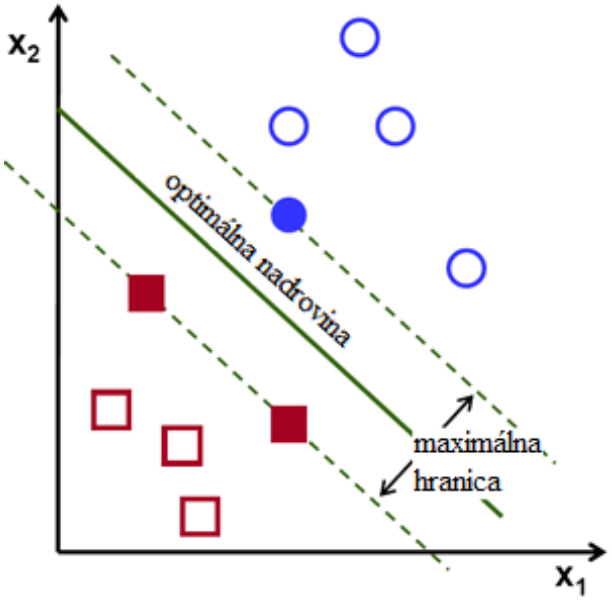
Jedná sa o klasifikátor, ktorý je definovaný deliacou nadrovinou. Pri označených trénovacích dátach bude výstupom SVM nadrovina, ktorá ich rozdeľuje a dokážeme ňou klasifikovať ďalšie dáta. Binárna klasifikácia je základnou úlohou riešenou pomocou SVM.



Obr. 5: Príklad oddeliteľného problému v 2 rozmernom priestore

Na obrázku 5 sú dve množiny bodov, ktoré chceme rozdeliť priamkou. Priamok, ktoré by jednoznačne rozdelili tieto body, môže existovať viac. Priamky, ktoré prechádzajú blízko bodov, nie sú vhodné, pretože klasifikátor by potom nemusel správne predvídať ďalšie dáta. Pri určovaní optimálnej deliacej nadroviny (viď obrázok 6) hľadá SVM najväčšiu minimálnu vzdialenosť medzi bodmi dvoch tried.

Majme trénovacie dáta (x_i, y_i) pre $i = 1...N$, kde x_i je i-ta trénovacia vzorka a $y_i \in \{-1, 1\}$ je trieda, ku ktorej daná vzorka patrí. Natrénovaním SVM získame klasifikátor $f(x)$, pre ktorý



Obr. 6: Optimálna nadrovina maximalizujúca hranicu medzi trénovacími dátami

platí

$$f(w^T x_i + b) \begin{cases} \geq 1 & \text{ak } y_i = +1 \\ \leq -1 & \text{ak } y_i = -1 \end{cases} \tag{5}$$

Potom pre správnu klasifikáciu platí $y_i f(w^T x_i + b) \geq 1$. Pri definovaní nadroviny nemusíme brať do úvahy celú trénovaciu množinu, ale iba časť dát, nazývanú podporné vektory (support vectors), ktoré túto nadrovinu určujú. Pre podporné vektory platí $y_i f(w^T x_i + b) = 1$. Optimálna nadrovina

$$w^T x + b = 0 \tag{6}$$

rozdeľuje trénovacie dáta tak, že medzi podpornými vektormi dvoch rôznych tried je maximálna vzdialenosť. Nadrovina pretína úsečku medzi najbližšími bodmi konvexných obalov dvoch trénovacích množín. Vzdialenosť medzi týmito bodmi a nadrovinou je určená ako $d = \frac{|w^T x + b|}{\|w\|} = \frac{1}{\|w\|}$. Hľadáme maximálnu hranicu $M = \frac{2}{\|w\|}$. Problém maximalizácie M je teda ekvivalentný problému minimalizácie $\|w\|$. Jedná sa teda o kvadratický optimalizačný problém s lineárnymi obmedzeniami. Riešenie je určené sedlovým bodom Lagrangiánskej funkcie

$$L_p = \frac{1}{2} w^T w - \sum_{i=1}^{n} a_i (y_i (w^T x_i + b) - 1) \tag{7}$$

kde $a_i \geq 0$) [14].

Predošlý problém bol lineárne oddeliteľný. Boser, Guyon a Vapnik v roku 1992 prišli so spôsobom ako klasifikovať nelineárne dáta. Trénovacie dáta sa namapujú do viac rozmerného priestoru, v ktoromu sú dáta lineárne oddeliteľné. Táto metóda je nazývaná kernelový trik. Bežne

používané kernely sú:

1. polynomiálny – $K(x, y) = (x \cdot y + c)^d$, homogénny ak $c = 0$
2. Gaussova radiálna funkcia – $K(x, y) = \exp(-\gamma\|x - y\|^2)$ kde $\gamma > 0$
3. Exponenciálna radiálna funkcia – $K(x, y) = \exp(-\gamma\|x - y\|)$ kde $\gamma > 0$
4. Hyperbolický tangens – $K(x, y) = tanh(\rho x \cdot y + c)$ kde $\rho > 0$ a $c < 0$

4.2 Histogram orientovaných gradientov

HOG bol pôvodne navrhnutý pre detekciu chodcov N. Dalalom a B. Triggsom [17]. Dnes je využívaný napr. aj pri detekcii vozidiel [18], alebo dopravných značiek [19]. Základnou myšlienkou je, že tvar objektu môže byť dobre definovaný orientáciou, alebo intenzitou gradientu. Gradient $\nabla f = [\frac{\partial f}{dx}, \frac{\partial f}{dy}]$ je definovaný smerom a veľkosťou. Pomocou konvolúcie s jadrom $[1, 0, -1]$ získame deriváciu obrazu v horizontálnom smere (dx) a s jadrom $[1, 0, -1]^T$ deriváciu vo vertikálnom smere (dy). Smer gradientu je definovaný ako:

$$\theta = tan^{-1}(\frac{\partial f}{dx}/\frac{\partial f}{dy}). \tag{8}$$

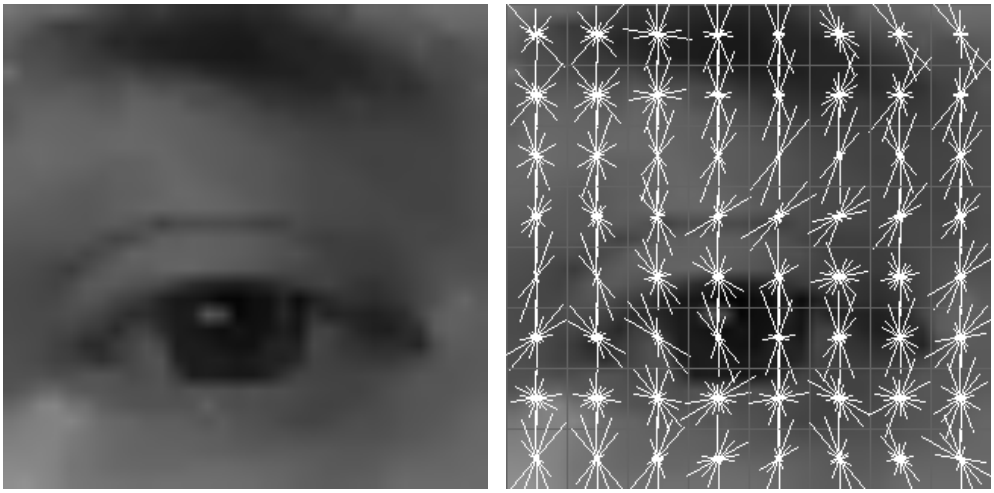
Veľkosť gradientu je

$$\|\nabla f\| = \sqrt{((\frac{\partial f}{dx}^2) + (\frac{\partial f}{dy}^2))}. \tag{9}$$

Potom rozdelíme obraz na bunky. Tieto bunky môžu byť obdĺžnikové, alebo kruhové (C-HOG). Gradient má uhol $0° - 360°$ (gradient so znamienkom), alebo $0° - 180°$ (znamienko gradientu je ignorované). Podľa [17] je vhodné použiť rozdelenie orientácií do 9 smerov v rozmedzí $0° - 180°$. Každý pixel v bunke prispieva k jednej orientácii. Kvôli lepšej invariancii voči tieňom, osvetleniu a kontrastu hrán je dôležitá normalizácia histogramov. Bunky sú rozdelené do blokov, ktoré sa prekrývajú. Každý z týchto blokov je normalizovaný samostatne. Na obrázku 7 je znázornená vizualizácia HOG príznakov. Veľkosť pôvodného obrazu je 32×32, veľkosť bunky použitej pri výpočte je 4×4 a gradienty boli normalizované v blokoch veľkosti 8×8.

4.3 Použitie SVM a HOG

V mojej práci používam knižnicu SVMlight [20] na klasifikáciu otvorených a zatvorených očí. Najskôr pomocou OpenCV načítam obrázok z trénovacej množiny. Zmením veľkosť obrázku, ak nesedí s veľkosťou posuvného okna použitého v HOG detektore. Pre každý obrázok vypočítam HOG príznak a uložím ho do súboru spolu s triedou, ku ktorej patrí (-1 pre negatívnu, +1 pre pozitívnu). Následne sa všetky vektory príznakov načítajú, vykoná sa tréning pomocou SVM a výsledný model sa uloží do súboru pre ďalšie použitie.



(a) pôvodný obraz (b) vizualizácia

Obr. 7: Vizualizácia HOG príznakov

Pri trénovaní SVM som použil 2728 obrázkov zatvorených očí a 1340 obrázkov otvorených očí. Zo začiatku som trénoval SVM s množinou 300 pozitívnych a 300 negatívnych obrázkov. Po natrénovaní som detektor otestoval a zle klasifikované obrázky som pridal do trénovacej množiny. Tento proces som niekoľkokrát zopakoval. Keďže výsledné trénovacie množiny nie sú vyvážené, je potrebné pri trénovaní SVM nastaviť váhový faktor. Ten nastavím ako pomer medzi negatívnymi a pozitívnymi vzorkami, teda 2,03.

Po úspešnej detekcii oblasti oka použijem natrénovaný HOG detektor na detekciu stavu oka. Pri žmurknutí je prirodzené, že sú zatvorené obe oči. Ak teda obe detekované oči klasifikuje detektor ako otvorené, považujem oko za otvorené. O tom, či sa jedná skutočne o žmurknutie, alebo sú oči otvorené, rozhodne až ďalšia detekcia dúhovky a vzdialenosti medzi očnými viečkami.

4.4 Úprava obrazu a detekcia dúhovky

Ďalším krokom je detekcia dúhovky. Najskôr však detekovanú oblasť oka orežem zhora o $\frac{1}{2,5}$. Tým sa zbavím časti oka s obočím. Priemer detekovanej dúhovky je približne trikrát menší ako šírka oka. Keďže detekovaná oblasť presne neohraničuje oko, tak hľadaný priemer musí byť väčší. Pokusmi som zistil, že najvhodnejšie je hľadať dúhovku s priemerom 3,5 krát menším ako je šírka detekovanej oblasti oka. Pred detekciou dúhovky je vhodné obraz ďalej upraviť.

4.4.1 Odstránenie šumu

Pred aplikáciou hranového detektora(Sobelov filter) je potrebné odstrániť z obrazu šum, pretože hranové detektory sú na šum citlivé. Toto je možné docieliť pomocou konvolúcie Gaussovho filtra s obrazom. Gaussov filter je nízkofrekvenčný filter. Nízkofrekvenčné filtre odstraňujú prudké zmeny jasu a tým obraz rozostrujú.

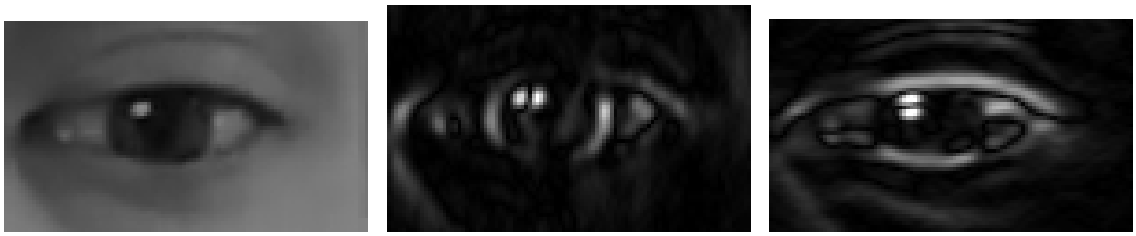
4.4.2 Sobelov filter

Sobelov filter, alebo tiež Sobel-Feldmanov filter je používaný v analýze obrazu na detekciu hrán. Konvolúciou obrazu so Sobelovým filtrom vo vertikálnom a horizontálnom smere dostaneme aproximáciu derivácií G_x (10) a G_y (11) v diskrétnom obraze I. Často používané sú konvolučné jadrá s veľkosťou 3×3. Použitím väčšieho konvolučného jadra by detekované hrany boli viac rozostrené.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * I \tag{10}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * I \tag{11}$$

Na obrázku 8 môžeme vidieť obraz oka pred a po aplikovaní Sobelovho operátora. Smer a veľkosť gradientu vypočítame podľa rovníc 9 a 8.



(a) pôvodný obraz (b) derivácia v horizontálnom smere (c) derivácia vo vertikálnom smere

Obr. 8: Obraz pred a po aplikácii Sobelovho filtra

4.4.3 Binarizácia obrazu

Binárny obraz je obraz, v ktorom každý pixel môže mať jednu z dvoch hodnôt. Zvyčajne je to buď hodnota bielej alebo čiernej farby. Binárny obraz I_B z obrazu I dostaneme pomocou prahovania. Prahovanie obrazu môžeme definovať ako:

$$I(x,y) = \begin{cases} 0 & \text{ak } I(x,y) < T \\ 255 & inak \end{cases} \tag{12}$$

kde T je prahová hodnota. Hodnoty jednotlivých pixelov obrazu teda nahradíme hodnotou 0, ak hodnota jasu v danom pixely je menšia ako prahová hodnota T. Na obrázku 9 môžeme vidieť obraz veľkosti gradientu detekovaných hrán pred prahovaním a výsledný obraz po binarizácii s $T = 180$.

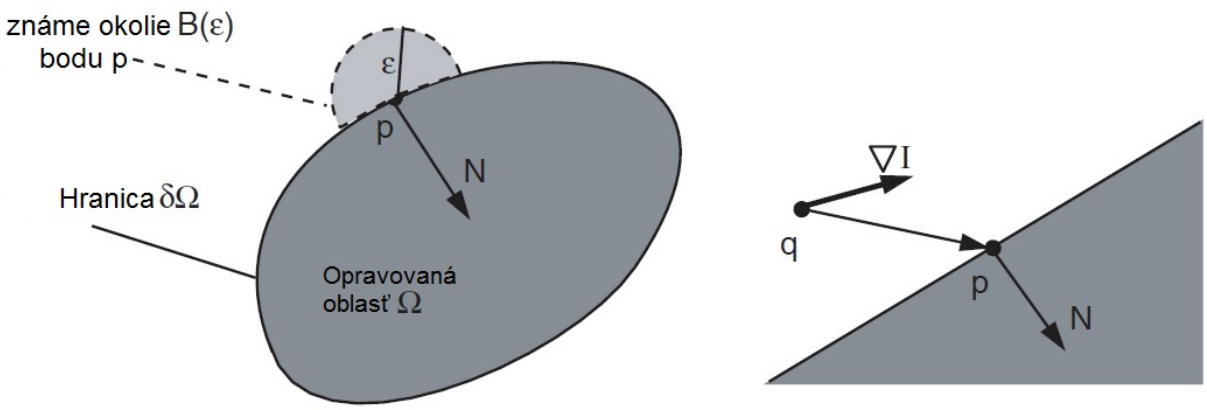
(a) pôvodný obraz (b) binárny obraz

Obr. 9: Príklad binarizácie obrazu

4.4.4 Odstránenie odleskov z oblasti oka

K odstráneniu odleskov svetelných zdrojov v obraze používam metódu *image inpainting*. Je to technika nedetekovateľnej úpravy obrazu, ktorá sa používa na rôzne účely ako napríklad na obnovovanie zničených malieb a fotografií, alebo na nahradzovanie a odstraňovanie objektov v obraze. Bertalmio a kolektív predstavili formu inpaintingu pomocou aproximácie riešenia Navier–Stokesovej rovnice [21]. Telea [22] popísal ďalšiu metódu inpaintingu s použitím Fast Marching Method (FMM), ktorá je rovnako efektívna a nevyžaduje komplexné výpočty ako Bertalmiove riešenie.

Na obrázku 10 je bod p na hranici $\delta\Omega$ opravovanej oblasti Ω, na ktorú chceme aplikovať metódu image inpainting. Výsledná hodnota bodu p bude rozhodnutá na základe jeho okolia



Obr. 10: Princíp metódy image inpainting (upravené podľa [22])

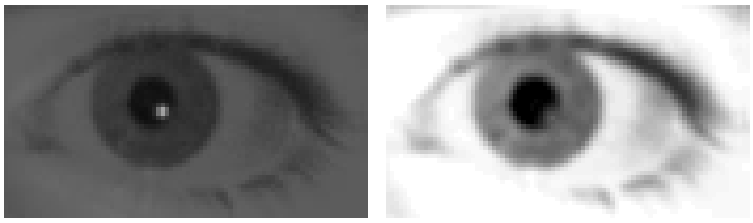
$B_\epsilon(p)$ o veľkosti ϵ, ktoré je v obraze známe. Majme obraz $I(q)$ a gradient $\nabla I(q)$. Dostatočne malé ϵ získame aproximáciou prvého rádu $I_q(p)$ obrazu v bode p.

$$I_q(p) = I(q) + \nabla I(q)(p - q) \tag{13}$$

Bod p nahradíme funkciou všetkých bodov q v okolí $B_\epsilon(q)$ sčítaním všetkých odhadov bodov q. Táto funkcia je ďalej vážená normalizovanou váhovou funkciou $w(p,q)$. Tá je potrebná, aby inpainting bodu p propagoval šedé hodnoty, ale aj hrany a ostré detaily obrazu v okolí $B_\epsilon(p)$. Niektoré metódy inpaintingu počítajú mapu vzdialeností od hranice $\delta\Omega$. Telea využil FMM, vďaka čomu nie je potrebné počítať mapu vzdialeností. FMM zároveň udržiava hranicu oddeľujúcu známu a neznámu oblasť obrazu a určuje nasledujúci pixel, ktorý bude nahradený.

$$I(p) = \frac{\sum_{q \in B_\epsilon(p)} w(p,q)[I(q) + \nabla I(q)(p-q)]}{\sum_{q \in B_\epsilon(p)} w(p,q)} \tag{14}$$

V mojej práci použijem Teleov algoritmus založený na FMM. Vstupom metódy je pôvodný obraz a maska, podľa ktorej budú pixely obrazu nahradzované. V mojom prípade chcem odstrániť nežiaduce odlesky z detekovanej oblasti oka. Predpokladám, že odlesky na oku majú vysokú intenzitu jasu. Najskôr aplikovaním Sobelovho operátora na pôvodný obraz získam jeho derivácie v horizontálnom a vertikálnom smere. Potom vypočítam celkovú veľkosť gradientu podľa vzorca 9. Veľkosť gradientu bude najväčšia v oblasti výrazných odleskov a to hlavne v oblasti dúhovky a zreničky, pretože sú výrazne tmavšie. Následne prahovaním veľkosti gradientu získam binárny obraz. Na získaný obraz aplikujem morfologickú transformáciu uzatvorenia (dilatácia nasledovaná eróziou), aby sa prípadné diery v obraze zacelili. Výsledný binárny obraz použijem ako masku pri inpaintingu. Na obrázku 11 môžeme vidieť príklad obrazu s výrazným odrazom svetla v oblasti zreničky a obraz po úprave a aplikovaní metódy image inpainting.



(a) pôvodný obraz

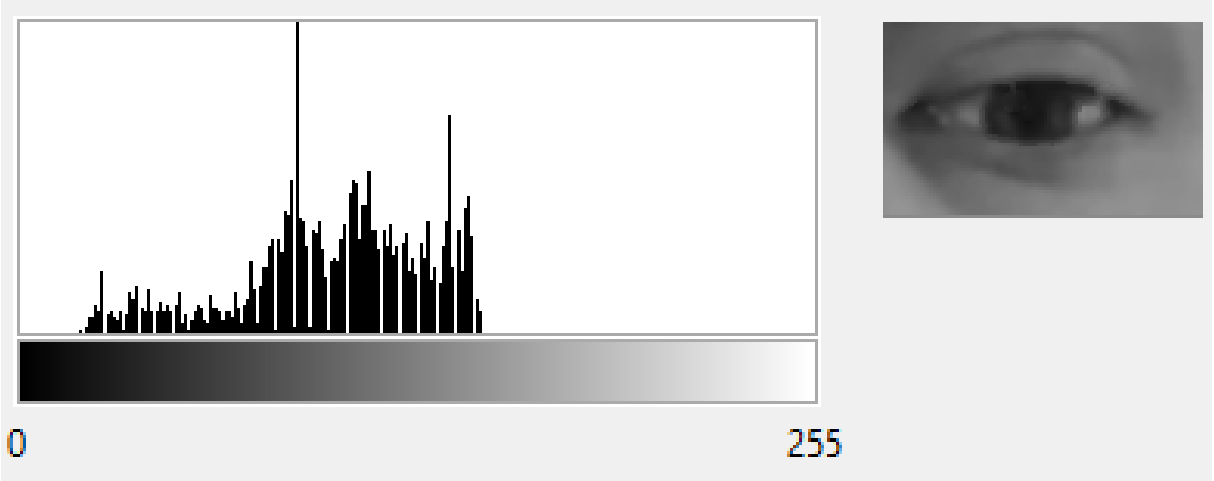
(b) po vynásobení intenzity a aplikácii metódy inpainting

Obr. 11: Image inpainting

4.4.5 Úprava kontrastu

Kontrast obrazu je rozsah intenzít, ktoré sú v ňom obsiahnuté. Môžeme tiež povedať, že je to rozdiel medzi maximálnou a minimálnou intenzitou v obraze. Pred detekovaním jednotlivých častí oka je vhodné kontrast upraviť. K tomu používam roztiahnutie histogramu obrazu.

Na obrázku 12 je oblasť oka pred roztiahnutím kontrastu. Minimálna intenzita pixelu v tomto obraze je $f_{min} = 17$ a maximálna intenzita $f_{max} = 148$. Povedzme, že chceme intenzitu jasu v obraze upraviť tak, že minimálna intenzita bude 0 a maximálna 255. Chceme teda z obrazu



Obr. 12: Oblasť oka s histogramom

$f(x, y)$ s kontrastom 131 dostať obraz $g(x, y)$ s kontrastom 255.

$$g(x, y) = \frac{f(x, y) - f_{min}}{f_{max} - f_{min}} \cdot 255 \tag{15}$$

V mojom prípade nechcem kontrast roztiahnuť podľa minimálnej a maximálnej hodnoty jasu v obraze. Najskôr si vypočítam histogram obrazu $f(x, y)$ a z neho následne vypočítam pravdepodobnosti P_i pre jednotlivé intenzity v obraze ako:

$$P_i = \frac{h(i)}{n} \tag{16}$$

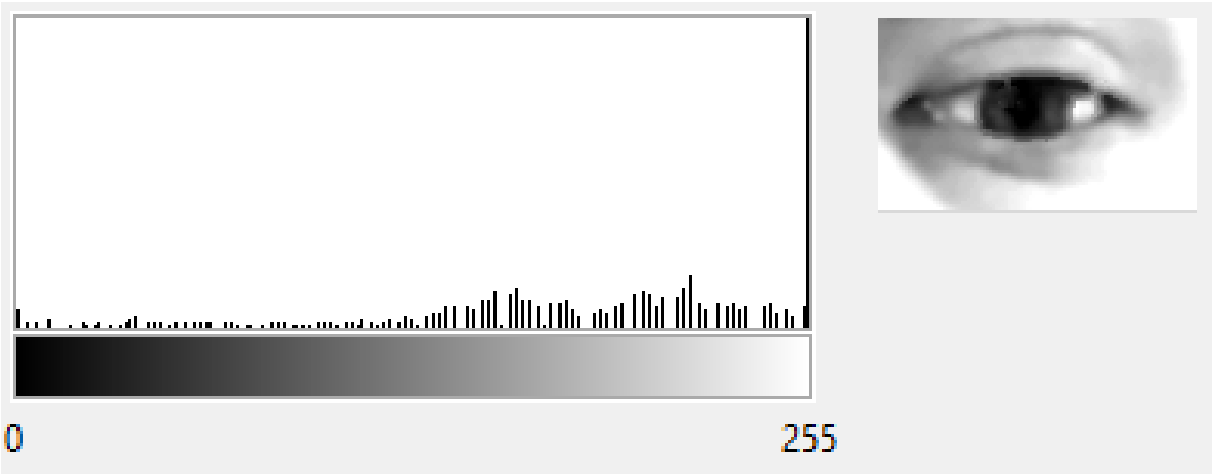
kde $h(i)$ je hodnota v histograme pre intenzitu $i \in \{0, 1, \ldots 255\}$. f_{min} a f_{max} určím zo zistených pravdepodobností ako:

$$\sum_{i=0}^{f_{min}} P_i \geq P_{low} \tag{17}$$

$$\sum_{i=f_{max}}^{255} P_i \geq P_{high} \tag{18}$$

P_{low} a P_{high} som určil experimentálne. Pri detekcii dúhovky je to 0,01 a 0,2. Pri detekcii očných viečok je možné tieto hodnoty v aplikácii meniť, ale východzie hodnoty sú nastavené na 0,1.

Na obrázku 13 je oblasť oka s histogramom po zmene kontrastu s použitím $P_{low} = 0,01$ a $P_{high} = 0,2$. Tým roztiahneme kontrast obrazu tak, že nízke intenzity jasu zostanú v obraze zachované a niektoré vyššie intenzity budú mať hodnotu 255. Obraz sa tým zosvetlí, ale oblasť dúhovky zostane tmavá.



Obr. 13: Oblasť oka po roztiahnutí histogramu

4.4.6 Detekcia dúhovky

K samotnej detekcii dúhovky využívam kruhovú Houghovu transformáciu. Vďaka využitiu veľkosti gradientu a jeho smeru je tento algoritmus efektívnejší. Pred výpočtom gradientu zvýšim kontrast oblasti oka roztiahnutím histogramu. Následne pomocou Sobelovho operátora spočítam deriváciu v smere osi x a osi y. Po získaní derivácií spočítam veľkosť gradientu podľa rovnice 9. Hľadanie kružnice v obraze zjednoduším tým, že nepočítam smer gradientu a stred kružnice pre každý pixel. Využijem vypočítanú veľkosť gradientu a prahovaním získam binárny obraz veľkostí gradientu (viď obr. 9). Stred detekovanej dúhovky potom hľadám podľa pixelov, ktorých hodnota v binárnom obraze je 255. Každý takýto pixel leží na nejakej kružnici. Keď zoberieme do úvahy aj smer gradientu vypočítaný podľa rovnice 8, potom hľadáme dva potencionálne stredy, ktoré ležia na opačných stranách dotyčnice kružníc. Obrázok 14 napovedá, ako je možné tieto stredy vypočítať. Majme polomer r a súradnice pixela $[x, y]$. Stredy $[c_x^1, c_y^1]$ a $[c_x^2, c_y^2]$ vypočítame ako:

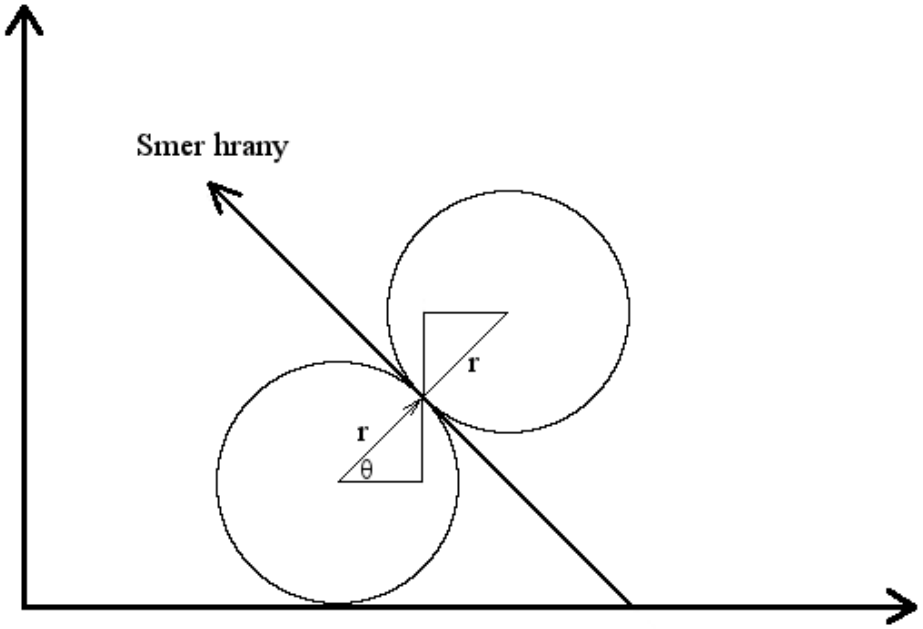
$$c_x^1 = x + rsin(\theta(x,y) - \frac{\pi}{2}) \tag{19}$$

$$c_y^1 = y - rcos(\theta(x,y) - \frac{\pi}{2}) \tag{20}$$

$$c_x^2 = x - rsin(\theta(x,y) - \frac{\pi}{2}) \tag{21}$$

$$c_y^2 = y + rcos(\theta(x,y) - \frac{\pi}{2}) \tag{22}$$

Keďže predpokladám, že priemer hľadanej dúhovky je 3,5 krát menší ako šírka detekovanej oblasti oka w_e, hľadám kružnice s polomerom od $[\frac{w_e}{7}] - 1$ do $[\frac{w_e}{7}] + 1$. Každý pixel binárneho obrazu s hodnotou 255 prispieva k dvom trojiciam $[c_x, c_y, r]$ v Houghovom akumulačnom priestore.



Obr. 14: Potencionálne stredy kružníc s ohľadom na smer gradientu

Potencionálny stred dúhovky je ignorovaný, ak zasahuje mimo detekovanú oblasť oka. Trojica, ktorej prispieva najviac pixelov definuje hľadanú dúhovku s polomerom r a stredom $[c_x, c_y]$, pokiaľ je toto maximum väčšie ako prahová hodnota.

Prahovú hodnotu určujem počas prvých snímkov, kedy by mal používateľ pozerať do kamery a prirodzene žmurknúť. Prahová hodnota je určená ako:

$$T_I = V_I[70\% \cdot S] - 2 \tag{23}$$

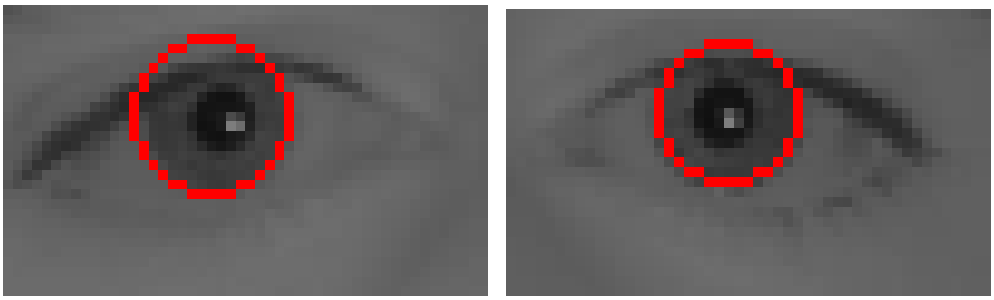
kde V_I je vektor zoradených maxím získaných z akumulátorov počas prvých snímkov a S je veľkosť vektora V_I.

Na obrázku 15 vidíme úspešnú detekciu dúhovky, ktorú môžeme ďalej využiť pri detekcii očných viečok

4.5 Detekcia očných viečok

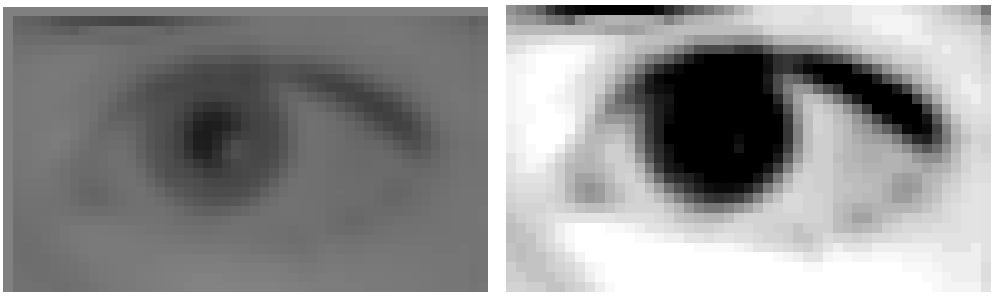
Pri detekcii očných viečok vychádzam podobne ako v [8] z týchto predpokladov:

1. Okolie oka je svetlejšie ako oko.
2. Horné viečko je nad spodným viečkom.
3. Horné viečko je nad detekovaným stredom dúhovky.



Obr. 15: Detekcia dúhovky

4. Spodné viečko je pod detekovaným stredom dúhovky.



Obr. 16: Ukážka obrazu pred a po roztiahnutí histogramu

Ak sa v predchádzajúcom kroku podarilo detekovať dúhovku, využijem informáciu o jej polohe pri detekcii očných viečok. Podobne ako pri detekcii dúhovky roztiahnutím histogramu zmením kontrast obrazu (viď obrázok 16). Očné viečka sa tak zvýraznia. Samotná detekcia spočíva v aplikovaní filtrov navrhnutých podľa vyššie uvedených podmienok.
Filter pre detekciu horného viečka:

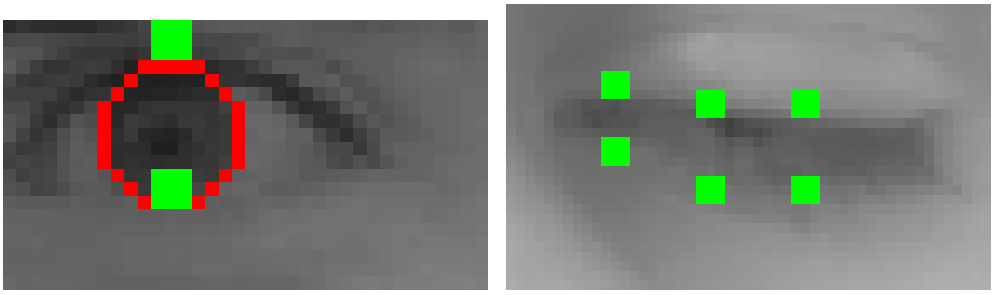
$$F_U = \begin{bmatrix} 1 & \cdots & 1 \\ -1 & \cdots & -1 \end{bmatrix} \tag{24}$$

Filter pre detekciu spodného viečka:

$$F_L = \begin{bmatrix} -1 & \cdots & -1 \\ 1 & \cdots & 1 \end{bmatrix} \tag{25}$$

Pri hľadaní horného viečka aplikujem filter F_U na časť oka nad detekovaným stredom dúhovky. Najväčšia odozva filtra bude na pozícii viečka. Spodné viečko hľadám aplikovaním filtra F_L v oblasti pod stredom dúhovky. Výslednú vzdialenosť medzi viečkami určím ako rozdiel polohy spodného viečka a horného viečka

$$D = P_L(s) - P_U(s) \tag{26}$$



(a) S využitím polohy dúhovky (b) Zatvorené oko

Obr. 17: Príklad detekcie polohy očných viečok

kde $P_L(s)$ je poloha spodného a $P_U(s)$ poloha horného viečka v snímku s.

Ak dúhovka nebola detekovaná, rozdelím oblasť oka na 5 rovnako veľkých zvislých častí. Prvú a poslednú časť vynechám, keďže sa jedná o krajné oblasti oka. Krajné oblasti obsahujú očné kútiky, alebo neobsahujú žiadnu časť oka. V stredných troch častiach pomocou vyššie popísaných filtrov detekujem približnú polohu viečok, pričom polohu spodného viečka hľadám v oblasti pod detekovaným horným viečkom. Vzdialenosť medzi horným a spodným viečkom určím váženým priemerom

$$D = 0,3 \cdot D_1(s) + 0,4 \cdot D_2(s) + 0,3 \cdot D_3(s) \tag{27}$$

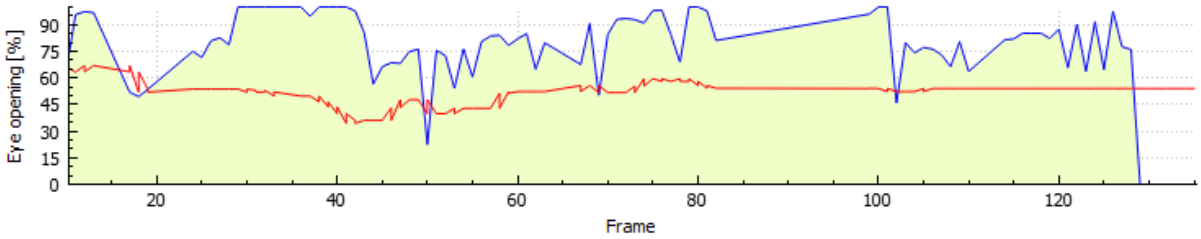
kde $D_i(s)$ je vzdialenosť na osi y medzi detekovanými bodmi horného a spodného viečka v snímku s. Vzdialenosť v strednej časti má pridelenú vyššiu váhu, pretože je väčšinou najpresnejšia.

4.6 Určenie miery otvorenia očí

Ak sú oči detekované ako otvorené, zisťujem mieru otvorenia, inak je miera 0%. Pri určovaní miery otvorenia očí predpokladám, že obe oči mi poskytujú rovnakú informáciu o stave oka. Vzdialenosť očných viečok určím zo zistených vzdialeností viečok ľavého $D_L(s)$ a pravého oka $D_P(s)$ v snímku s ako

$$D(s) = \begin{cases} max(D_L(s), D_P(s)) & \text{ak } \dfrac{D_L(s) + D_P(s)}{2} > T_Z \\ \dfrac{D_L(s) + D_P(s)}{2} & \text{ak } \dfrac{D_L(s) + D_P(s)}{2} \leq T_Z \end{cases} \tag{28}$$

kde T_Z je hraničná hodnota vzdialenosti medzi očnými viečkami pri zatvorenom oku. Túto hranicu môžeme získať dvomi spôsobmi. Jednen spôsob je určiť hranicu počas prvých snímkov videosekvencie a druhý určiť hranicu podľa aktuálne detekovanej oblasti oka.



Obr. 18: Určenie miery otvorenia očí v prvých snímkoch

4.6.1 Určenie prahovej hodnoty z prvých snímkov

Tento spôsob som už využil pri detekcii dúhovky. Prahovú hodnotu určím ako:

$$T_Z = V_D[5\% \cdot S] \tag{29}$$

V_D je vektor zoradených vzdialeností medzi viečkami získaných počas prvých snímkov a S je veľkosť vektora V_D. Maximálnu vzdialenosť očných viečok pri otvorenom oku potom získam takisto z vektora V_D ako:

$$D_{max} = V_D[82\% \cdot S] \tag{30}$$

Aby bol algoritmus invariantný voči zmene veľkosti, normalizujem hodnoty T_Z a D_{max} tak, že ich vydelím šírkou oblasti oka. Mieru otvorenia očí následne určím ako

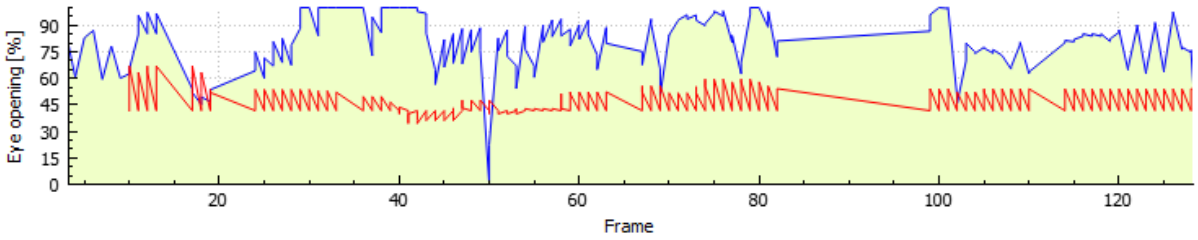
$$M = min(100, \frac{D(s)}{D_{max} \cdot w_e} \cdot 100) \tag{31}$$

kde w_e je šírka detekovanej oblasti oka aktuálneho snímku s. Na obrázku 18 môžeme vidieť určovanie miery otvorenia očí vo videosekvencii, kedy počas prvých 105 snímkov bola upravovaná prahová hodnota T_Z a maximálna vzdialenosť viečok D_{max}.

4.6.2 Určenie prahovej hodnoty podľa šírky detekovanej oblasti

Jedná sa o stanovenie prahovej hodnoty na základe šírky detekovanej oblasti oka. Pokiaľ by sme detekovali vzdialenosť očných viečok na statickom obraze, alebo vo videosekvencii, kde počas prvých snímkov subjekt nežmurkne, alebo naopak, bude mať oči stále zatvorené, prvá metóda určenia prahovej hodnoty sa stáva nepresnou. V takom prípade je vhodné určiť pevnú prahovú hodnotu. Avšak chceme zachovať invarianciu voči zmene veľkosti oblasti oka. Experimentálne som určil prahovú hodnotu zatvoreného oka ako:

$$T_Z = \frac{w_e}{6} \tag{32}$$



Obr. 19: Určenie miery otvorenia očí podľa veľkosti oblasti oka

kde w_e je šírka detekovanej oblasti oka aktuálneho snímku. Rovnako je určená aj maximálna vzdialenosť medzi očnými viečkami pri otvorenom oku.

$$D_{max} = \frac{w_e}{2,5} \tag{33}$$

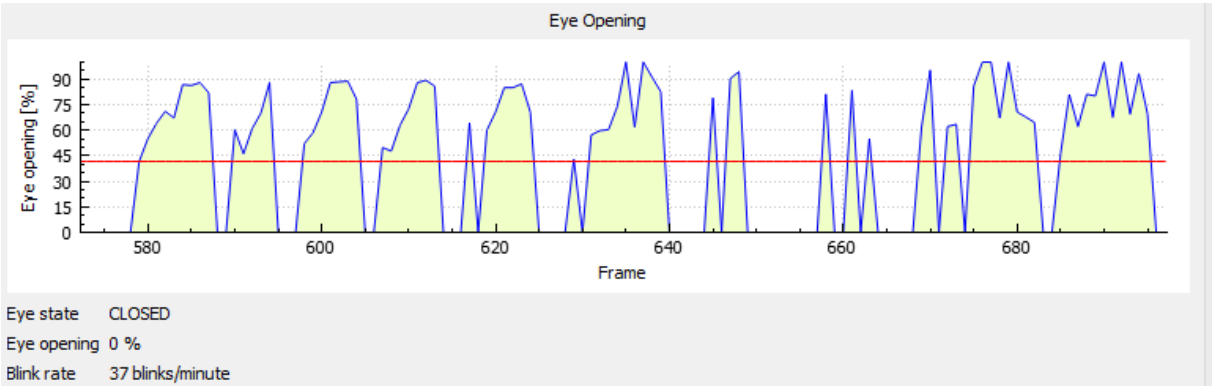
Pri určovaní miery otvorenia očí v tomto prípade už nemusíme T_Z a D_{max} normalizovať. Miera otvorenia sa vypočíta ako:

$$M = min(100, \frac{D(s)}{D_{max}} \cdot 100) \tag{34}$$

Na obrázku 19 je vidieť ako sa pri použití tejto metódy hranica zatvoreného oka mení v každom snímku.

4.7 Detekcia frekvencie žmurkania

Žmurknutie je definované ako rýchle zatvorenie oka. V mojej práci určujem frekvenciu žmurkania ako počet žmurknutí za minútu, pričom za žmurknutie považujem aj dlhšie zatvorené oči. Pre každý snímok videosekvencie v rámci poslednej minúty si uložím stav očí. Môžu nastať tri stavy. Buď neboli oči, alebo tvár detekované vôbec, alebo bolo aspoň jedno oko detekované a je možné jeho stav určiť. Každú sekundu sa frekvencia žmurkania aktualizuje. Prejdem všetky uložené stavy a spočítam počet prechodov z otvoreného stavu do zatvoreného stavu. Na obrázku 20 môžeme vidieť ako program správne detekoval 4 žmurknutia po sebe, no potom začal falošne detekovať viacero žmurknutí. Napríklad od snímku 655 po snímok 665 detekoval program tri žmurknutia namiesto jedného.

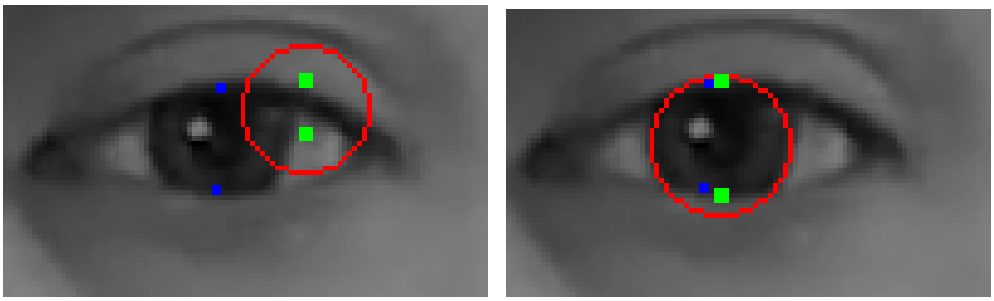


Obr. 20: Určenie frekvencie žmurkania v aplikácii

5 Experimentálna časť

V tejto kapitole popíšem metódy testovania navrhnutých algoritmov, ktoré som popísal v predchádzajúcich kapitolách. K testovaniu detekcie vzdialenosti očných viečok som do aplikácie pridal možnosť manuálneho zadávania bodov na očných viečkach. Používateľ určí bod na hornom a spodnom očnom viečku a stlačí klávesu "o", ak je oko otvorené. Vzdialenosť týchto bodov sa následne uloží do súboru spolu s číslom aktuálneho snímku a so stavom oka. Ak je oko zatvorené, stačí stlačiť klávesu "c" a do súboru sa uloží vzdialenosť medzi očnými viečkami 0.

Na obrázku 21 môžeme vidieť príklad nekorektne detekovaných očných viečok a rozdiel medzi detekovanou vzdialenosťou (body vyznačené zelenou) a manuálne označenou vzdialenosťou očných viečok. Takisto vidíme správne detekované očné viečka, kde je tento rozdiel minimálny. Z obrázku je jasné, že nekorektná detekcia očného viečka bola spôsobená nesprávnou detekciou dúhovky.



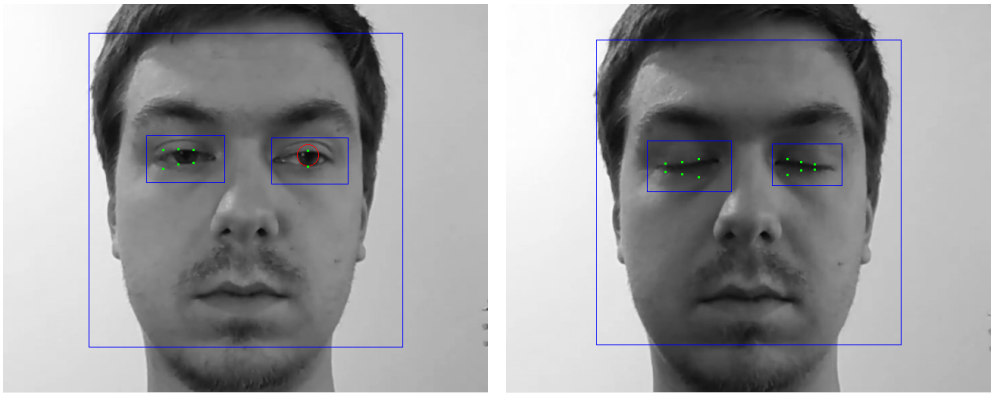
Obr. 21: Príklad rozdielu vypočítaných a reálnych vzdialeností očných viečok

5.1 Test detekcie vzdialenosti očných viečok

Na testovanie detekcie vzdialenosti očných viečok som pomocou aplikácie manuálne označil viečka v dvoch testovacích videách. Prvé video má 23 sekúnd a obsahuje 188 snímkov. V druhom videu som označil 661 snímkov. K určeniu presnosti detekcie porovnávam manuálne vyznačené vzdialenosti s detekovanými vzdialenosťami. Chybu detekcie určím ako:

$$e = \frac{|D_r - D_v|}{w_e} \tag{35}$$

kde D_r je skutočná vzdialenosť medzi očnými viečkami, D_v je vypočítaná vzdialenosť medzi očnými viečkami a w_e je šírka detekovanej oblasti oka. Vydelením výsledného rozdielu šírkou w_e zaistím, že chyba detekcie bude invariantná voči veľkosti oka. Na obrázku 22 sú príklady detekcií očných viečok pri otvorenom a zatvorenom oku. Môžeme vidieť, že detekciou dúhovky môžeme vylepšiť detekciu viečok. Takisto vidíme, že detekované body neležia presne na viečkach. To však pri určovaní vzdialenosti očných viečok a miery otvorenia nevadí, pretože pri zatvorenom oku je priemer vzdialeností detekovaných bodov na viečkach menší ako prahová hodnota. Preto budú



Obr. 22: Ukážka detekcií očných viečok

oči detekované ako zatvorené a miera otvorenia bude 0%. Detekcia vzdialenosti očných viečok je teda závislá na správnej detekcii žmurkania.

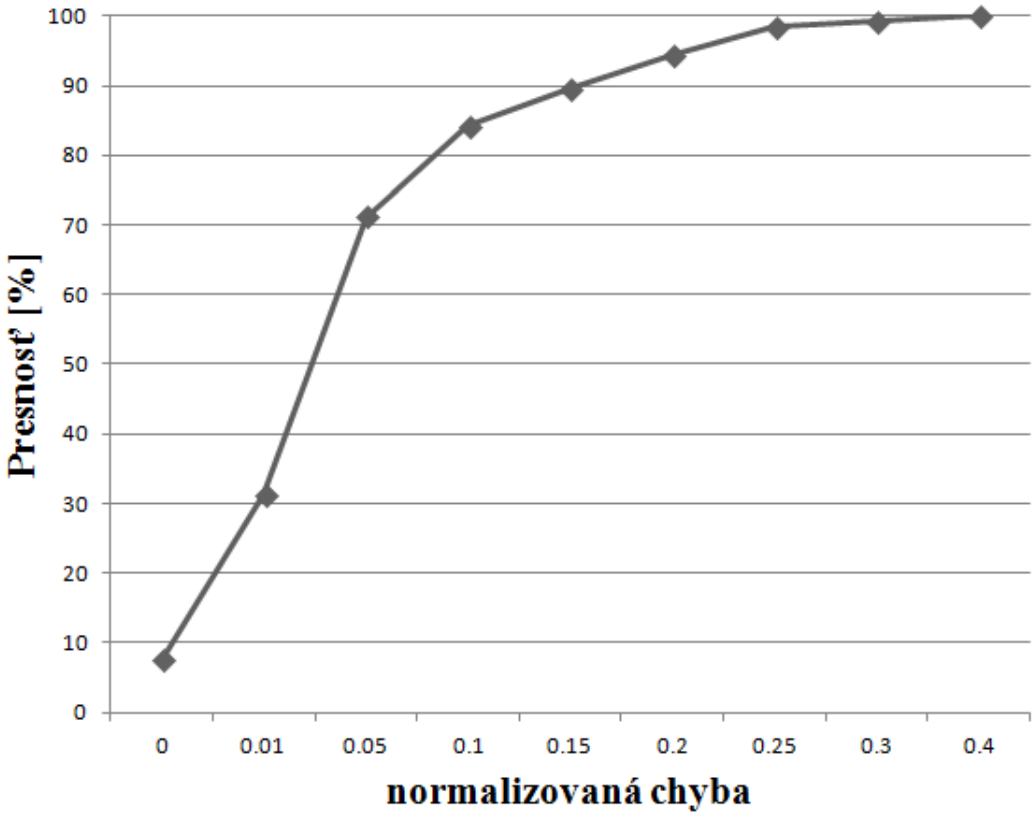
V grafe na obrázku 23 je znázornená závislosť presnosti detekcie vzdialenosti očných viečok na normalizovanej chybe. Môžeme teda povedať, že presnosť detekcie pre $e \leq 0,15$ je $89,6\%$. Normalizovaná chyba $e \leq 0,15$ zodpovedá veľkosti $0,15 \cdot w_e$, čo je približne veľkosť priemeru zreničky. Chyba pri detekcii môže nastať pri nesprávnom určení oblasti oka. To môžeme vidieť aj na obrázku 24, kde detekovaná oblasť oka je príliš malá. Namiesto dúhovky bola detekovaná zrenička a hranica medzi dúhovkou a zreničkou je nesprávne považovaná za očné viečka.

5.2 Porovnanie automaticky získaných prahových hodnôt so skutočnými

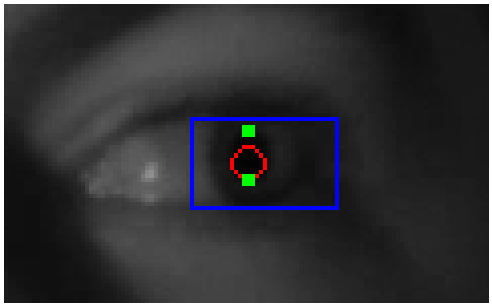
Správne určenie miery otvorenia očí priamo závisí od detekcie vzdialenosti očných viečok, a tiež od určenia správnych prahových hodnôt. Navrhol som dve techniky získavania prahových hodnôt.

Prvou technikou je získanie minimálnej a maximálnej miery otvorenia z prvých snímkov videosekvencie. V jednom testovacom videu bola takto získaná prahová hodnota pre určenie zatvoreného oka 15 pixelov. Bližším skúmaním manuálne vyznačených vzdialeností v tomto videu som zistil, že vzdialenosť očných viečok pred zatvorením očí je 12 pixelov. Prahová hodnota určená z prvých snímkov je o 3 pixely väčšia. Pre lepšiu predstavu môžeme vidieť na obrázku 25 nesprávne detekované viečka označené zelenými bodmi s veľkosťou 3 pixely. V tomto snímku bola manuálne vyznačená vzdialenosť očných viečok najväčšia a to 30 pixelov. Maximálna vzdialenosť očných viečok zistená z prvých snímkov je v tomto prípade 24 pixelov. Tento rozdiel je už podstatný, avšak vzdialenosť očných viečok je 30 pixelov iba ak je oko otvorené viac ako zvyčajne, napr. ak subjekt pozerá smerom hore. Ak toto zistenie vezmeme do úvahy, ideálna maximálna miera otvorenia očí by bola 26 pixelov.

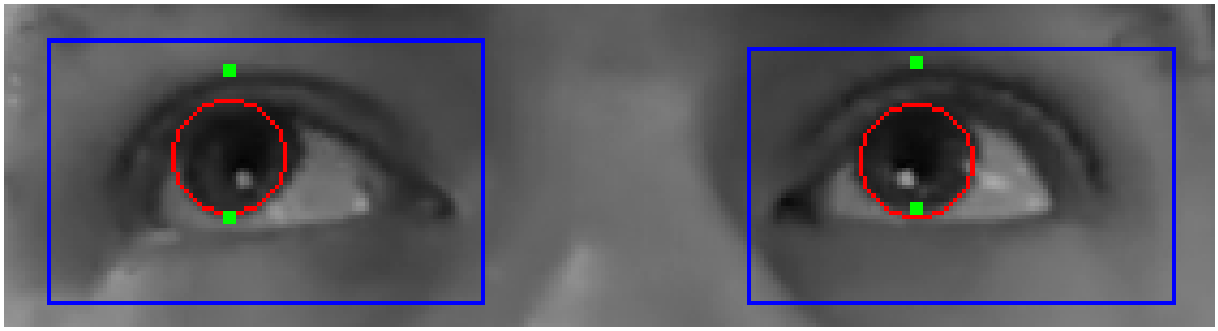
Použitie druhej techniky je závislé na detekovanej oblasti oka. Je vhodná, pokiaľ vo videosekvencii počas prvých snímkov nemôžeme určiť prahové hodnoty prvou technikou. Použitím



Obr. 23: Graf závislosti presnosti na normalizovanej chybe



Obr. 24: Ukážka nesprávnej detekcie oka



Obr. 25: Maximálna miera otvorenia očí

tejto techniky v testovacom videu získame priemernú prahovú hodnotu 17 pixelov a maximálnu mieru otvorenia 40 pixelov. Vďaka vyššej prahovej hodnote budú detekcie žmurknutia presnejšie. Určovanie miery otvorenia je týmito hodnotami značne ovplyvnené. Miera otvorenia 100% bude dosiahnutá len zriedka.

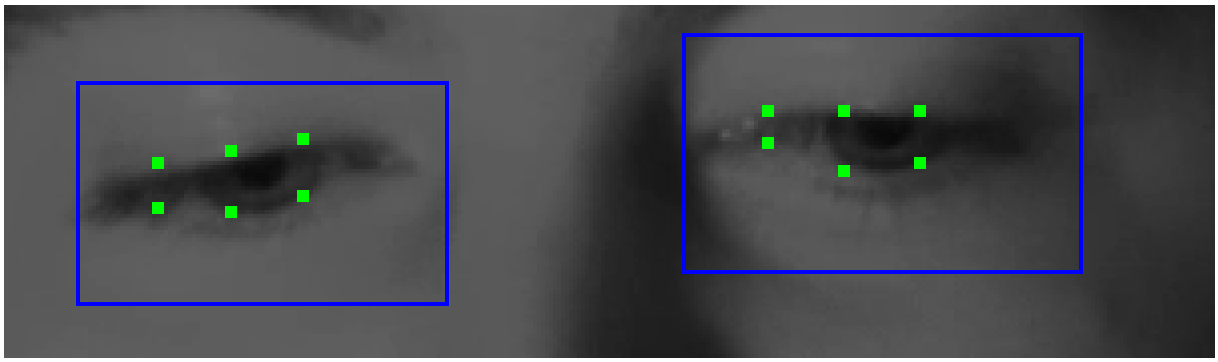
5.3 Test detekcie žmurknutia

K testovaniu detekcie žmurknutia som použil rovnaké videá ako v kapitole 5.1. K nim som pridal ďalšie dve videá, v ktorých som už označil len stav očí (zatvorené/otvorené). Dokopy mám teda k dispozícii 1471 snímkov s označenými stavmi očí, ktoré použijem na overenie presnosti algoritmu.

Celkovú presnosť detekcie spočítam ako:

$$P = \frac{TP + TN}{TP + FP + TN + FN} \cdot 100 \tag{36}$$

kde TP je počet snímkov, v ktorých boli oči správne detekované ako otvorené, TN je počet snímkov, v ktorých boli oči správne detekované ako zatvorené, FP je počet snímkov, v ktorých boli oči nesprávne detekované ako otvorené a FN je počet snímkov, v ktorých boli oči nesprávne detekované ako zatvorené. Obrázok 26 znázorňuje otvorené oči, ktoré boli nesprávne klasifikované ako zatvorené. Detektor v tomto prípade nedokázal nájsť dúhovku, vzdialenosť očných viečok je nižšia ako prahová hodnota a SVM klasifikátor takisto detekoval oči ako zatvorené.



Obr. 26: Nesprávne klasifikované oči ako zatvorené

V ďalších častiach otestujem detekciu žmurknutia s použitím adaptívnej prahovej hodnoty získanej z prvých snímkov videosekvencie a následne s použitím prahovej hodnoty určenej podľa šírky aktuálne detekovanej oblasti oka. Výsledky testov potom porovnám.

5.3.1 Test detekcie žmurknutia s použitím prahovej hodnoty určenej z prvých snímkov

V tabuľke 1 sú uvedené výsledky detekcie zatvorených a otvorených očí s použitím adaptívnej prahovej hodnoty získanej podľa postupu v kapitole 4.6.1.

Tabuľka 1: Počty správne a nesprávne detekovaných stavov očí pri použití adaptívneho prahu

Stav očí	otvorené	zatvorené
detekované otvorené	1126(TP)	42(FP)
detekované zatvorené	59(FN)	244(TN)

Výsledná presnosť detekcie podľa vzorca 36 je $91,1\%$. Ďalej je vhodné určiť senzitivitu, špecifickosť detektora. Špecifickosť hovorí o tom, ako presne detektor klasifikuje skutočne zatvorené oči. Senzitivita zase hovorí o jeho schopnosti správne klasifikovať otvorené oči. Môžeme tiež povedať, že je to pravdepodobnosť, že detektor klasifikuje oči ako otvorené práve vtedy, keď sú skutočne otvorené. Čím vyššia je senzitivita a špecifickosť, tým je detektor lepší. Senzitivitu vypočítame ako:

$$senzitivita = \frac{TP}{TP + FN} \cdot 100 \tag{37}$$

Špecifickosť je daná ako:

$$špecifickosť = \frac{TN}{FP + TN} \cdot 100 \tag{38}$$

Môžeme teda povedať, že navrhnutý detektor má špecifickosť $85,3\%$ a senzitivitu 95%. Mal by byť presnejší pri klasifikácii otvorených očí ako pri klasifikácii zatvorených očí.

5.3.2 Test detekcie žmurknutia s použitím prahovej hodnoty určenej šírkou detekovanej oblasti

Prahová hodnota určená šírkou detekovanej oblasti je vhodná pri určovaní stavu oka v statickom obraze, alebo vo videosekvencii, kde zo začiatku nemôže byť správne nakalibrovaná adaptívna prahová hodnota.

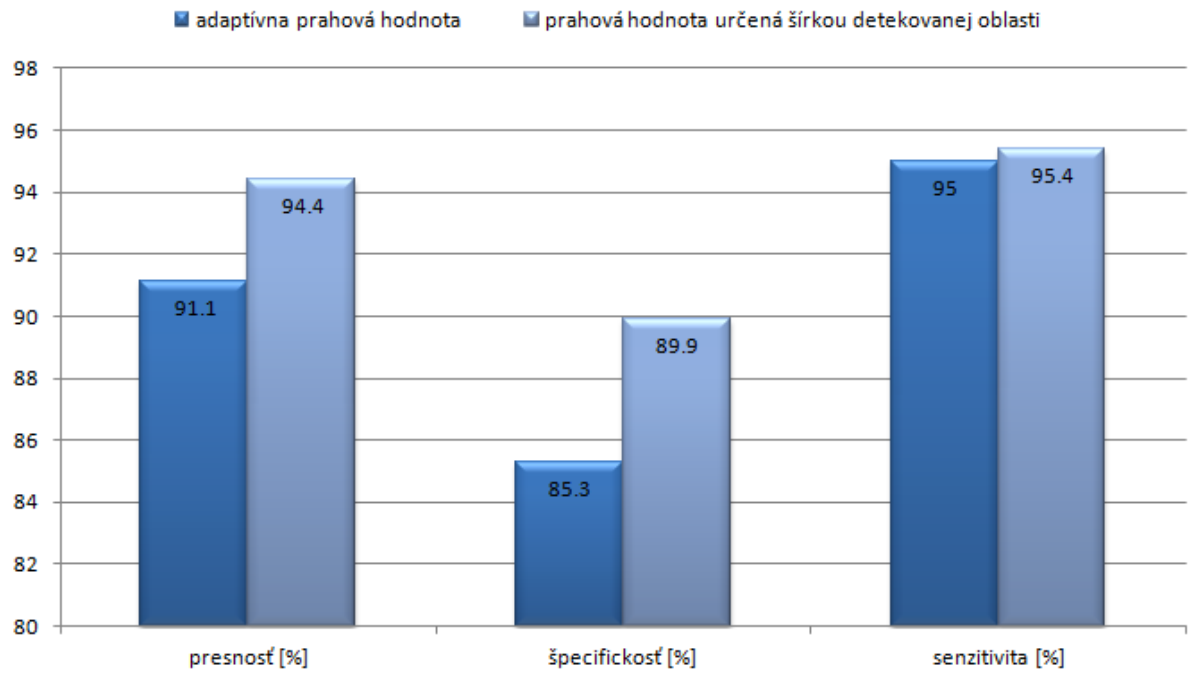
V tabuľke 2 sú výsledky z testovania detekcie na videách použitých v predchádzajúcej podkapitole. Môžeme vidieť, že počet očí správne detekovaných ako otvorené je len o niečo vyšší, ale za to stúpol počet správne detekovaných zatvorených očí. Z toho aj vyplýva vyššia senzitivita detektora, ktorá je $95,4\%$. Špecifickosť detektora stúpla na $89,9\%$. Celková presnosť detekcie je v tomto prípade $94,4\%$.

Tabuľka 2: Počty správne a nesprávne detekovaných stavov očí pri použití prahovej hodnoty určenej šírkou detekovanej oblasti

Stav očí	otvorené	zatvorené
detekované otvorené	1131(TP)	29(FP)
detekované zatvorené	54(FN)	257(TN)

5.3.3 Porovnanie testov detekcie žmurknutia

Podľa výsledkov testov detekcie žmurknutia môžeme vidieť, že adaptívna prahová hodnota v prvom teste nebola určená ideálne. Porovnanie výsledkov týchto testov môžeme vidieť v grafe



Obr. 27: Porovnanie testov detekcie žmurkania

Tabuľka 3: Počty správne a nesprávne detekovaných stavov očí pomocou SVM

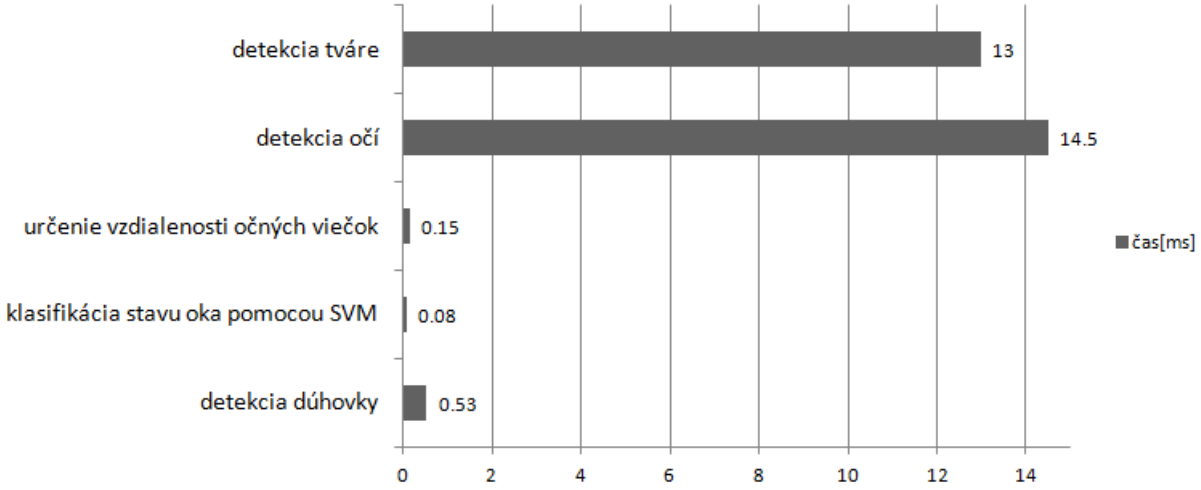
Stav očí	otvorené	zatvorené
detekované otvorené	1163(TP)	83(FP)
detekované zatvorené	21(FN)	203(TN)

na obrázku27. S použitím prahovej hodnoty určenej šírkou oblasti oka bola dosiahnutá lepšia celková presnosť detekcie a to o $3,3\%$. To sa môže zdať nepodstatné, ale keď ďalej vezmeme do úvahy špecifickosť detekcie, je rozdiel nezanedbateľný. Špecifickosť detektora v druhom teste je $89,9\%$, čo je o $4,6\%$ lepšie ako v prvom teste. Detektor teda bude lepšie klasifikovať nielen zatvorené oči, ale vďaka lepšej senzitivite aj otvorené oči. Preto by bol vhodnejší na detekciu žmurkania.

5.4 Test SVM klasifikátora

V tejto kapitole sú zhrnuté výsledky testu detekcie pomocou natrénovaného SVM klasifikátora bez využitia detekcie dúhovky a očných viečok. Z tabuľky 3 môžeme vidieť, že SVM klasifikátor detekuje správne viacej otvorených očí ako s využitím detekcie vzdialenosti očných viečok a detekcie dúhovky. Avšak správnosť týchto detekcií je za cenu viacerých falošne pozitívnych detekcií. Je teda opäť potrebné vypočítať celkovú presnosť klasifikátora, jeho senzitivitu a špecifickosť.

Celková presnosť natrénovaného SVM klasifikátora je $92,9\%$. Je teda presnejší ako v kombinácii s detekciou očných viečok a dúhovky pri použití adaptívnej prahovej hodnoty určenej z



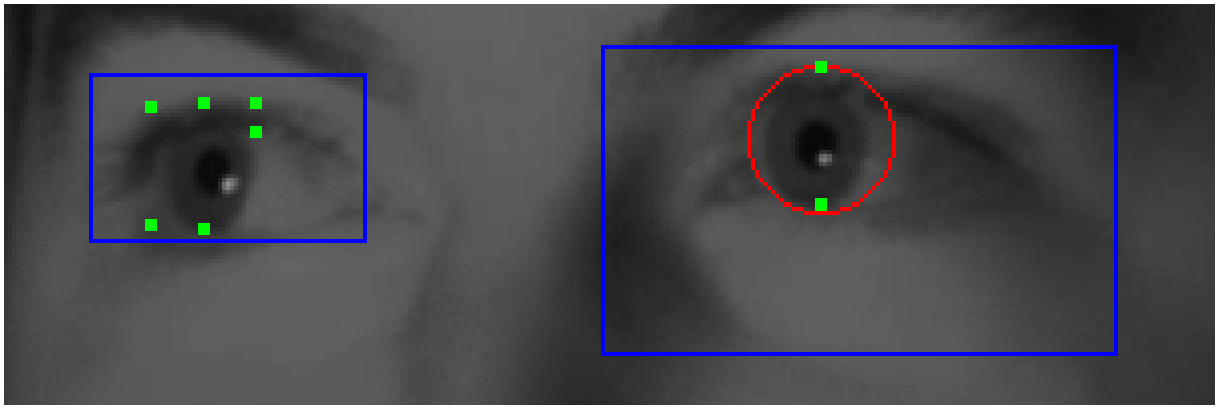
Obr. 28: Časová zložitosť navrhovaných metód

prvých snímkov vo videosekvenciách. Senzitivita SVM klasifikátora je $98,2\%$ a špecifickosť je 71%. Vidíme teda, že aj napriek vyššej presnosti SVM detektora, je stále vhodné ho vylepšiť pomocou detekcie dúhovky a vzdialenosti očných viečok. A to hlavne kvôli nízkej špecifickosti SVM klasifikátora, ktorá je o necelých 19% nižšia ako pri vylepšenej detekcii s použitím prahovej hodnoty určenej šírkou detekovanej oblasti.

5.5 Časová zložitosť

V tejto kapitole sa venujem časovej zložitosti navrhnutých metód. Čas behu jednotlivých metód som meral na pripravenom testovacom videu s dĺžkou 53 s, ktoré obsahuje 1054 snímkov. Video má rozlíšenie 720×576, no pri detekcii je každý snímok zmenšený na veľkosť 640×512. Všetky merania som uskutočnil na osobnom počítači s procesorom Intel Core i5 s taktom jadier 2,67 GHz a s pamäťou RAM 16GB.

Zistený priemerný čas potrebný na detekciu častí oka a následné určenie miery otvorenia a frekvencie žmurkania je 41 ms. Toto je celkový čas pre detekciu na oboch očiach. Môžeme teda povedať, že navrhnuté metódy sú vhodné na detekciu žmurkania a určenie miery otvorenia očí v reálnom čase. Z grafu na obrázku 28 vidíme, že najdlhšie trvá samotná detekcia oblasti tváre a očí pomocou kaskádových klasifikátorov. Detekcia očí trvá 14,5 ms a detekcia tváre trvá 13 ms. Detekcia tváre je rýchlejšia, pretože detektor začína s väčším posuvným okienkom a zväčšuje sa rýchlejšie ako pri detekcii očí. Ďalej môžeme vidieť, že detekcia dúhovky s použitím Houghovej transformácie je náročnejšia ako klasifikácia stavu oka pomocou SVM detektora alebo určovanie vzdialenosti očných viečok. Oproti detekcii tváre a očí je však stále veľmi rýchla. Treba však podotknúť, že tvár je potrebné detekovať v obraze veľkosti 640×512, kdežto oči sú detekované v orezanej časti tváre. Táto orezaná časť má v prípade testovaného videa priemernú šírku 278 pixelov a výšku 158 pixelov.



Obr. 29: Nevhodná detekcia očí bez detekcie tváre

Pri snahe zrýchliť detekciu tváre a očí som otestoval program bez použitia detekcie tváre. Skúsil som teda detekovať oči v pôvodnom obraze. Bez ďalšej zmeny parametrov detektora bola detekcia očí približne päťkrát pomalšia. Po zmene škálovacieho faktora funkcie *detectMultiScale()* z $1,2$ na $1,8$ bola detekcia asi dvakrát pomalšia, celková rýchlosť bola teda už na rovnakej úrovni ako pri použití detekcie tváre a následne očí. Pri škálovacom faktore $1,8$ však neboli oči detekované vždy presne. To môžeme vidieť aj na obrázku 29, kde detekovaná oblasť ľavého oka je neprimerane väčšia ako oblasť pravého oka.

6 Záver

Témou mojej diplomovej práce bolo navrhnúť vhodnú metódu detekcie miery otvorenia očí a frekvencie žmurkania a ich následná implementácia. Tieto metódy som mal tiež experimentálne otestovať.

V práci som popísal princíp detekcie tváre a očí pomocou silných klasifikátorov Haarových príznakov. Ukázal som princíp techniky image inpainting využívajúcu FMM a jej aplikáciu pri odstraňovaní nechcených odrazov svetla v oblasti dúhovky. Navrhol som detekciu dúhovky pomocou kruhovej Houghovej transformácie, pri ktorej je využitá veľkosť gradientu obrazu a tiež jeho smer. Nájdená kružnica je považovaná za hľadanú dúhovku, ak hodnota trojice (x, y, r) v Houghovom priestore je väčšia ako prahová hodnota určená počas prvých snímkov videosekvencie. Opísal som trénovanie SVM klasifikátora s použitím HOG, ktorý som využil pri detekcii stavu oka. Ďalej som navrhol detekciu polohy očných viečok pomocou filtrov, z ktorej je ďalej určená miera otvorenia očí. Prezentoval som dve techniky získania prahovej hodnoty pre určenie zatvoreného oka, teda žmurknutia, a maximálnej vzdialenosti viečok pri otvorenom oku. Jedna technika je určenie miery z prvých snímkov videosekvencie podobne ako pri detekcii dúhovky. Druhá technika určuje prahovú hodnotu na základe detekovanej oblasti oka. Samotná miera otvorenia očí veľmi závisí od správneho určenia týchto prahových hodnôt.

Navrhnuté metódy som implementoval v jazku C++ s použitím knižnice OpenCV [12] a frameworku Qt [23]. Funkčnosť navrhnutých metód som otestoval na niekoľkých videách. Každý spracovávaný snímok bol pred detekciou zmenšený na veľkosť 640×512. Porovnaním presnosti detekcie jednotlivých metód som zistil, že sa pri detekcii žmurkania viacej osvedčila prahová hodnota určená šírkou detekovanej oblasti oka. S použitím tejto prahovej hodnoty bola presnosť detekcie o $3,3\%$ vyššia. Takisto senzitivita a špecifickosť detektora boli vyššie ako pri použití prahovej hodnoty určenej z prvých snímkov videosekvencie. Zistil som tiež, že samotný SVM klasifikátor dokáže lepšie detekovať otvorené oči, ale kvôli nízkej špecifickosti 71% častejšie nesprávne klasifikuje zatvorené oči ako otvorené. Kombinácia navrhnutých metód je vhodnejšia na detekciu žmurkania, pretože ňou dosiahneme vyššiu presnosť. V práci som tiež zhodnotil časovú zložitosť navrhnutých metód a môžem povedať, že sú vhodné na použitie v reálnom čase.

Z vyššie uvedeného môžem konštatovať, že cieľ diplomovej práce som splnil. Navrhujem prácu ďalej vylepšiť pridaním sledovania očí. Aby sa zvýšila presnosť detekcie miery otvorenia očí, bolo by vhodné aby pred detekciou prebehla kalibrácia, ktorá by lepšie určila maximálnu a minimálnu mieru otvorenia očí. Pre lepšiu klasifikáciu stavu očí navrhujem pretrénovanie SVM detektora s väčšou množinou negatívnych vzoriek. V niektorých prípadoch sme mohli vidieť, ako sú detekované viaceré žmurknutia namiesto jedného, alebo v prípade zatvorených očí. Mohli by sme teda na základe stavu očí v posledných pár snímkoch presnejšie určiť aktuálny stav oka a eliminovať falošné detekcie.

Literatura

[1] WEN-BING HORNG, CHIH-YUAN CHEN, YI CHANG a CHUN-HAI FAN. Driver fatigue detection based on eye tracking and dynamk, template matching. IEEE International Conference on Networking, Sensing and Control, 2004. IEEE, 2004, 7-12. DOI: 10.1109/ICNSC.2004.1297400. ISBN 0-7803-8193-9. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1297400

[2] CHAU, Michael a Margrit BETKE. Real Time Eye Tracking and Blink Detection with USB Cameras. Boston University Computer Science Technical Report. 2005, 2005(12). Dostupné z: https://pdfs.semanticscholar.org/76b7/811c5b12717fe2f00ca12a7c26301b5f4d27.pdf

[3] AWAIS, Muhammad, Nasreen BADRUDDIN a Micheal DRIEBERG. Automated eye blink detection and tracking using template matching. 2013 IEEE Student Conference on Research and Developement. IEEE, 2013, 79-83. DOI: 10.1109/SCOReD.2013.7002546. ISBN 978-1-4799-2656-5. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7002546

[4] YING-LI TIAN, T. KANADE a J.F. COHN. Dual-state parametric eye tracking. Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580). IEEE Comput. Soc, 2000, 110-115. DOI: 10.1109/AFGR.2000.840620. ISBN 0-7695-0580-5. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=840620

[5] YANG, Fei, Xiang YU, Junzhou HUANG, Peng YANG a Dimitris METAXAS. Robust eyelid tracking for fatigue detection. 2012 19th IEEE International Conference on Image Processing. IEEE, 2012, 1829-1832. DOI: 10.1109/ICIP.2012.6467238. ISBN 978-1-4673-2533-2. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6467238

[6] LALONDE, Marc, David BYRNS, Langis GAGNON, Normand TEASDALE a Denis LAURENDEAU. Real-time eye blink detection with GPU-based SIFT tracking. Fourth Canadian Conference on Computer and Robot Vision (CRV '07). IEEE, 2007, 481-487. DOI: 10.1109/CRV.2007.54. ISBN 0-7695-2786-8. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4228575

[7] G. Pan, L. Sun, Z. Wu, S. Lao: Eyeblink-based anti-spoofing in face recognition from a generic webcamera. The 11th IEEE International Conference on Computer Vision (ICCV'07), Rio de Janeiro, Brazil, October 2007. Dostupné z https://pdfs.semanticscholar.org/99a5/f5eb492d3a609611268e9d107ce4408fa474.pdf

[8] LIU, Ang, Zhichao LI, Lang WANG a Yong ZHAO. A practical driver fatigue detection algorithm based on eye state. 2010 Asia Pacific Conference on Postgraduate Research in Microelectronics and Electronics (PrimeAsia). IEEE, 2010, 235-238. DOI: 10.1109/PRIMEASIA.2010.5604919. ISBN 978-1-4244-6735-8. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5604919

[9] HARRIS, C. a M. STEPHENS. A Combined Corner and Edge Detection. Proceedings of the 4th Alvey Vision Conference. 1988, 147-151. DOI: 10.1.1.231.1604. Dostupné z: http://www.cis.rit.edu/~cnspci/references/dip/feature_extraction/harris1988.pdf

[10] VIOLA, P. a M. JONES. Rapid object detection using a boosted cascade of simple features. Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001. IEEE Comput. Soc, 2001, I-511-I-518. DOI: 10.1109/CVPR.2001.990517. ISBN 0-7695-1272-0. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=990517

[11] PAVANI, Sri-Kaushik, David DELGADO a Alejandro F. FRANGI. Haar-like features with optimally weighted rectangles for rapid object detection. Pattern Recognition. 2010, 2010(1), 160-172. Dostupné z: http://www.sciencedirect.com/science/article/pii/S0031320309002003

[12] OpenCV [online]. 2016 [cit. 2016-04-11]. Dostupné z: http://opencv.org

[13] Shiqi Yu [online]. [cit. 2016-04-11]. Dostupné z http://yushiqi.cn/research/eyedetection

[14] BOSER, Bernhard E., Isabelle M. GUYON a Vladimir N. VAPNIK. A training algorithm for optimal margin classifiers. Proceedings of the fifth annual workshop on Computational learning theory - COLT '92. New York, New York, USA: ACM Press, 1992, 144-152. DOI: 10.1145/130385.130401. ISBN 089791497X. Dostupné z: http://portal.acm.org/citation.cfm?doid=130385.130401

[15] CROW, Franklin C. Summed-area tables for texture mapping. ACM SIGGRAPH Computer Graphics. 1984, 18(3), 207-212. DOI: 10.1145/964965.808600. ISSN 00978930. Dostupné z: http://portal.acm.org/citation.cfm?doid=964965.808600

[16] REZAEI, Mahdi. Artistic Rendering of Human Portraits Paying Attention to Facial Features. , 90. DOI: 10.1007/978-3-642-33329-3_11. Dostupné z: http://link.springer.com/10.1007/978-3-642-33329-3_11

[17] DALAL, N. a B. TRIGGS. Histograms of Oriented Gradients for Human Detection. 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05). IEEE, 2005, 886-893. DOI: 10.1109/CVPR.2005.177. ISBN 0-7695-2372-

2. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1467360

[18] G. Ballesteros, L. Salgado: Optimized HOG for On-road Video Based Vehicle Verification. http://www.eurasip.org/Proceedings/Eusipco/Eusipco2014/HTML/papers/1569924423.pdf BALLESTEROS, Gonzalo a Luis SALGADO. Optimized HOG for on-road video based vehicle verification. Proceedings of the 22nd European Signal Processing Conference (EUSIPCO). IEEE, 2014, 2014, 805-809. ISSN 2219-5491. Dostupné z: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6952260

[19] GREENHALGH, Jack a Majid MIRMEHDI. Real-Time Detection and Recognition of Road Traffic Signs. IEEE Transactions on Intelligent Transportation Systems. 2012, 13(4), 1498-1506. DOI: 10.1109/TITS.2012.2208909. ISSN 1524-9050. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6287592

[20] Making large-scale support vector machine learning practical. JOACHIMS, Thorsten. Advances in kernel methods: support vector learning. Cambridge, Mass.: MIT Press, 1999, s. 169-184. ISBN 0262194163.

[21] BERTALMIO, Marcelo, Guillermo SAPIRO, Vincent CASELLES a Coloma BALLESTER. Image inpainting. Proceedings of the 27th annual conference on Computer graphics and interactive techniques - SIGGRAPH '00. New York, New York, USA: ACM Press, 2000, , 417-424. DOI: 10.1145/344779.344972. ISBN 1581132085. Dostupné z: http://portal.acm.org/citation.cfm?doid=344779.344972

[22] TELEA, Alexandru. An Image Inpainting Technique Based on the Fast Marching Method. Journal of Graphics Tools. 2004, 9(1), 23-34. DOI: 10.1080/10867651.2004.10487596. ISSN 1086-7651. Dostupné z: http://www.tandfonline.com/doi/abs/10.1080/10867651.2004.10487596

[23] Qt Framework [online]. 2016 [cit. 2016-04-27]. Dostupné z: https://www.qt.io/qt-framework/