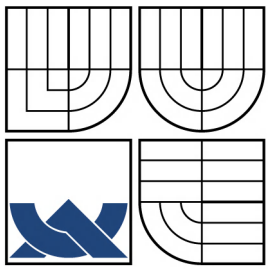
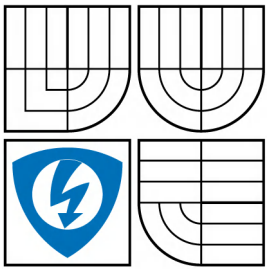


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

DETEKCE MRKÁNÍ A ROZPOZNÁVÁNÍ PODLE MRKÁNÍ OČÍ

EYE BLINK DETECTION AND RECOGNITION

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE *BC. VIKTOR TESÁREK*
AUTHOR

VEDOUCÍ PRÁCE *ING. JAN VLACH*
SUPERVISOR

BRNO 2008

LICENČNÍ SMLOUVA
POSKYTOVANÁ K VÝKONU PRÁVA UŽÍT ŠKOLNÍ DÍLO

uzavřená mezi smluvními stranami:

1. Pan/paní

Jméno a příjmení: Bc. Viktor Tesárek

Bytem: Písečná 5034, 43004, Chomutov

Narozen/a (datum a místo): 28.3.1982, Chomutov

(dále jen „autor“)

a

2. Vysoké učení technické v Brně

Fakulta elektrotechniky a komunikačních technologií

se sídlem Údolní 244/53, 602 00, Brno

jejímž jménem jedná na základě písemného pověření děkanem fakulty:

prof. Ing. Kamil Vrba, Csc.

(dále jen „nabyvatel“)

Čl. 1
Specifikace školního díla

1. Předmětem této smlouvy je vysokoškolská kvalifikační práce (VŠKP):
   - □ disertační práce
   - ⊠ diplomová práce
   - □ bakalářská práce
   - □ jiná práce, jejíž druh je specifikován jako .....

   (dále jen VŠKP nebo dílo)

Název VŠKP:	Detekce mrkání a rozpoznávání podle mrkání očí
Vedoucí/ školitel VŠKP:	Ing. Jan Vlach
Ústav:	Ústav telekomunikací
Datum obhajoby VŠKP:	

VŠKP odevzdal autor nabyvateli v*:

- ⊠ tištěné formě – počet exemplářů 2
- ⊠ elektronické formě – počet exemplářů 1

* hodící se zaškrtněte

2. Autor prohlašuje, že vytvořil samostatnou vlastní tvůrčí činností dílo shora popsané a specifikované. Autor dále prohlašuje, že při zpracovávání díla se sám nedostal do rozporu s autorským zákonem a předpisy souvisejícími a že je dílo dílem původním.
3. Dílo je chráněno jako dílo dle autorského zákona v platném znění.
4. Autor potvrzuje, že listinná a elektronická verze díla je identická.

Článek 2
Udělení licenčního oprávnění

1. Autor touto smlouvou poskytuje nabyvateli oprávnění (licenci) k výkonu práva uvedené dílo nevýdělečně užít, archivovat a zpřístupnit ke studijním, výukovým a výzkumným účelům včetně pořizovaní výpisů, opisů a rozmnoženin.
2. Licence je poskytována celosvětově, pro celou dobu trvání autorských a majetkových práv k dílu.
3. Autor souhlasí se zveřejněním díla v databázi přístupné v mezinárodní síti
   - ⊠ ihned po uzavření této smlouvy
   - □ 1 rok po uzavření této smlouvy
   - □ 3 roky po uzavření této smlouvy
   - □ 5 let po uzavření této smlouvy
   - □ 10 let po uzavření této smlouvy

   (z důvodu utajení v něm obsažených informací)
4. Nevýdělečné zveřejňování díla nabyvatelem v souladu s ustanovením § 47b zákona č. 111/ 1998 Sb., v platném znění, nevyžaduje licenci a nabyvatel je k němu povinen a oprávněn ze zákona.

Článek 3
Závěrečná ustanovení

1. Smlouva je sepsána ve třech vyhotoveních s platností originálu, přičemž po jednom vyhotovení obdrží autor a nabyvatel, další vyhotovení je vloženo do VŠKP.
2. Vztahy mezi smluvními stranami vzniklé a neupravené touto smlouvou se řídí autorským zákonem, občanským zákoníkem, vysokoškolským zákonem, zákonem o archivnictví, v platném znění a popř. dalšími právními předpisy.
3. Licenční smlouva byla uzavřena na základě svobodné a pravé vůle smluvních stran, s plným porozuměním jejímu textu i důsledkům, nikoliv v tísni a za nápadně nevýhodných podmínek.
4. Licenční smlouva nabývá platnosti a účinnosti dnem jejího podpisu oběma smluvními stranami.

V Brně dne: …….

…….. ……

Nabyvatel Autor

ABSTRAKT:

Tato diplomová práce se zabývá rozpoznáváním mrknutí oka a jejím hlavním úkolem je rozbor algoritmů zabývajících se detekcí lidí s jejich následným využitím ve vlastním programu na detekci mrknutí. Rozborem algoritmů, které se používají pro aplikace tohoto typu se zabývá první část této práce. V druhé části je popsán návrh samotného programu. Návrh programu je založen na metodě detekce pohybu pomocí akumulovaných diferencí snímků, které pomáhají určit pozice očí. Po zjištění pozic očí se provádí detekce mrknutí, která spočívá na porovnávání prahovaných šablon obou očí vyjmutých z předešlého snímku a šablonami očí vyjmutých z aktuálního snímku. Na základě shody předešlých a aktuálních šablon je vyhodnoceno, zda došlo k mrknutí. Program je přizpůsoben k sledování sedícího člověka, který se mírně pohybuje před pozadím, které také může být do jisté míry pohyblivé. Jako testovací vzorek bylo zvoleno průměrně kvalitní video s moderátorem a nestálým pozadím. Vlastnosti programu lze přizpůsobit na určitou scénu nebo na rozsah pohybu sledovaného člověka nastavením parametrů. Například zmenšením délky segmentu videa docílíme toho, že akumulační rámec nebude přesycený pohybem a bude jednodušší určit pozici očí.

ABSTRACT:

This master thesis deals with the issues of the eye blink recognition from video. The main task is to analyse algorithms dealing with a detection of persons and make a program that could recognize the eye blink. Analysis of these algorithms and their problems are in the first part of this thesis. In the second part design and properties of my program are described. The realization of the program is based on the method of move detection using the accumulated difference frame, which helps to identify the eye areas. The eye blink detection algorithm tests a match between a tresholded pattern of the eye area taken from the actual frame and the frame before. The resolution whether the eye blink happened or not, is based on the level of the match. The algorithm is designed for watching a sitting man, which is slightly moving. The background can be a little dynamic as well. An average quality video with a moderator and dynamic background was used as a tested subject. The function of my program can be adjusted to a certain scene or a range of the watched person motion by setting parameters, for example if we shorten the video segment length, we will achieve that the accumulated frame won't be overfilled by motion.

OBSAH

1 Úvod

Rozpoznávání mrknutí oka je poměrně obtížným úkolem. Abychom byly schopni jej realizovat, potřebujeme využívat poznatky mnoha vědních disciplín jako jsou pravděpodobnost, geometrie, anatomie, grafika a v neposlední řadě programování, protože díky počítačovému softwaru jsme schopni vytvořit umělé systémy, které mohou nahradit lidské myšlení.

Praktických použití systému na detekci mrkání je několik, ale v této práci není program nijak směrován ke konkrétnímu použití, spíš se snaží ukázat možnosti metod založených na detekci pohybu a metod používaných na rozpoznání lidí z obrazu.

Konkrétně lze detekci mrkání použít jako část ovládacího systému pro pohybově omezené lidi, kde mrknutí jednoho nebo druhého oka anebo jejich kombinace může sloužit jako spouštěcí povel podobně jako při ovládání počítačové myši. Často se stává, že ochrnutý člověk nemůže pohybovat ničím než ústy a očima, a tak jsou tyto programy pro ochrnuté jednou z mála možností jak něco ovládat. Dalším praktickým použitím systému s detekcí mrkání je napodobení lidského chování pro počítačovou animaci nebo jako sběr statistických dat pro animaci postav představující umělou inteligenci, které se snaží simulovat řeč těla. Pomocí detekce mrkání, lze také částečně určit psychické rozpoložení lidí například míru jejich stresu nebo únavu. Například použití v automobilech jako systém včasného varování, které má řidiče upozornit na jeho klesající bdělost v závislosti na klesající rychlosti mrkání.

Většina těchto realizací potřebuje ke svému použití v praxi určitou robustnost, které se zatím pomocí softwaru nepodařilo plně docílit a tak se určité problémy eliminují použitím přídavného hardwaru. Jedním z příkladů může sloužit systém na určování řidičovy ospalosti, kde můžou být v autě velice rozdílné podmínky v osvětlení zanedbány použitím kamery, která snímá infračervené spektrum, které je vyzařováno infračervenými zdroji světla. Systém určený k ovládání počítače pohybem očí a mrkáním zas eliminuje potíže s hledáním oka umístěním kamery na rámeček brýlí. Ve své práci se snažím přiblížit k návrhu robustního systému použitím standardních snímacích prostředků.

Cílem této práce je prostudovat možnosti vypracování algoritmu na detekci mrkání, sepsat jejich vlastnosti, navrhnout vlastní realizaci a tu naprogramovat v prostředí MATLAB.

Dokument je rozdělen na dvě tématické poloviny, kde první je teoretická a snaží se nám ukázat jak rozpoznávací systémy fungují. Ta je ještě rozdělena na dvě podčásti podle základního přístupu algoritmů k detekci lidí v obraze. První typ algoritmů pracuje na bázi předzpracování obrazu tak, aby po zpracování byly výrazné prvky, které jsou oblastí zájmu detekce. Tyto metody se nazývají metodami s oddělením pozadí a většinou slouží k oddělení člověka od scény. Druhou část teoretického úvodu tvoří metody přímé detekce, které nejsou tak závislé na předzpracování obrazu, avšak často obsahují uložená předzpracovaná data v databázích, se kterými se ta testovaná snaží srovnávat a hledat mezi nimi podobnost. Pokud máme již určenu pozici člověka můžeme přejít ke konečné fázi a to k detekci mrknutí, které se dělí na tři typy, jež jsou všechny popsány v závěru teoretického úvodu. Dělí se podle sledované části oka, která signalizuje detekci mrknutí sledováním duhovky, očního víčka anebo porovnáváním šablony celého oka.

Po teoretické části je popsána praktická část tak, že jsou přímo popisovány jednotlivé části programu podle jejich funkce. Celý program je rozdělen do šesti hierarchicky strukturovaných podprogramů. Po popisu funkce jednotlivých podprogramů následuje charakteristika vlastního podprogramu v jeho časové náročnosti a účinnosti.

1.1 Funkce mrkání

Mrkání hlavně slouží k ochraně oka před vyschnutím a také stírá všechny menší částice, které mohou podráždit plochu oka. Mrkání probíhá tak, že se sval ovládající víčka (lat. orbicularis oculi) roztáhne a nutí klouzat hlavně horní víčko směrem dolů po bulvě. Víčko klouže po bulvě díky tomu, že je k ní přisáté a stále je tvořen mezi nimi podtlak. Na vnitřní straně dolního očního víčka je slzný kanálek, kterým je přiváděn sekret k lubrikaci oka. Sekret ze slzného kanálku je roztírán pomocí víček po celém povrchu oka, takže potom tvoří ochranou a průhlednou vrstvu po celém oku. Mrknutí je prováděno převážně horním víčkem, které se pohybuje jeden centimetr dolu a potom zpět nahoru. Celý pohyb přibližně trvá 300 – 400 milisekund a statisticky člověk mrkne jednou během dvou až deseti vteřin[14]. Během jedné minuty dospělý člověk průměrně mrkne desetkrát tato hodnota se, ale může měnit v závislosti na psychickém a fyzickém stavu, třeba když člověk čte mrkne jen třikrát až čtyřikrát do minuty. Také několik nemocí ovlivňuje periodu mrknutí například Tourettův syndrom může být indikován pomocí častého mrkání, mrtvice a poruchy nervového systému také zvyšují frekvenci mrkání. Parkinsonova nemoc naopak snižuje frekvenci mrkání.

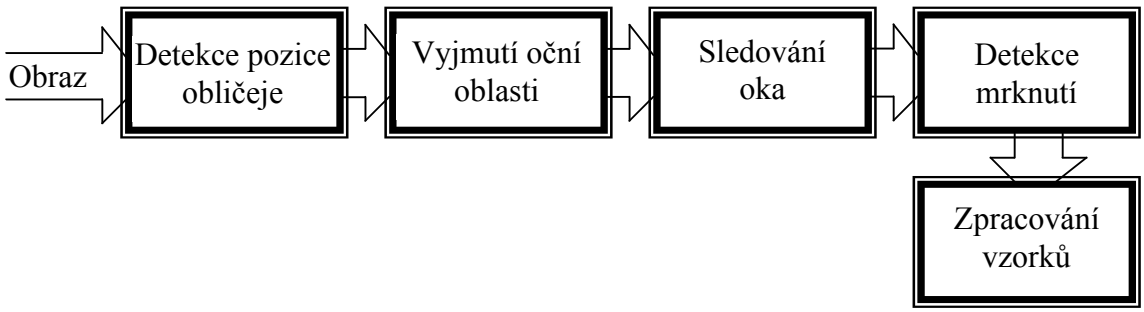
2 Přehled zpracování obrazu a metod detekce

Problematikou detekce mrknutí oka se již zabývalo mnoho lidí a já se v této kapitole budu věnovat jejich metodám řešení, u kterých objasním jejich funkčnost a vlastnosti fungování. Také se pokusím popsat metody, které přímo nebyly navrženy pro detekci mrknutí oka, ale souvisí s touto problematikou anebo jsou předstupněm k detekci mrknutí.

2.1 Obecné schéma pro postup detekce mrknutí

Při prvotním zamyšlení nad úkolem jsem sestavil schéma, které by mělo ukázat postupné zpracování vstupního obrazu a určit úrovně, které jsou potřebné k dosažení výsledku. Schéma také určuje rozdělení celého úkolu na jednotlivé bloky, kde každý pracuje s odlišnými vstupy podle svého algoritmu a pomocí výstupu posouvá zpracování k dalšímu kroku.

Ovšem toto blokové schéma není pravidlem, jak se ukáže v dalších kapitolách. Některé algoritmy pracují zcela jinými způsoby nebo přeskakují jeden či několik bloků v závislosti na jejich funkci.



Obr. 2.1: Blokové schéma zpracování obrazu

Vstupní tok našeho modelu může být tvořen videem nebo obrazem, ale také sekvencí obrázků. Vstup může, ale nemusí být limitován pravidly v závislosti na schopnosti algoritmu odpovědného za detekci pozice obličeje. Blok pro detekci pozice obličeje má za úkol určit, kde se obličej nachází a v závislosti na algoritmu, zda je natočen v nějaké ose. Výstup z tohoto bloku by měl být tvořen souřadnicemi nebo přímo částí obrazu, kterou algoritmus vyhodnotil jako obličej. Další blok má za úkol vyrovnat chybu předchozího bloku a přesně vymezit oblast očí. Sledování oka v průběhu času slouží jako předstupeň k detekci mrknutí, může být založeno na detekci očního víčka, detekci duhovky nebo panenky anebo také používat rozdílů z minulého snímku.

2.2 Metody detekce lidí v obraze

Podle Neeti O.[1] je možné techniky detekce lidí z obrazu rozdělit na metody, které obraz rozdělí na oblasti, kde se můžou nacházet lidé. A na metody, které používají detekci založenou na rysech člověka a nepotřebují předzpracování obrazu v takové míře. První typ metod se označuje jako metody s oddělením pozadí a druhý typ se označuje jako metody založené na přímé detekce.

2.2.1 Metody s oddělením pozadí

Metody s oddělením pozadí pracují ve dvou fázích. První fází je filtrování obrazu, kterým se odliší objekty zájmu v popředí. A na ně se v druhé fázi uplatňuje detekční algoritmus založený na lidských rysech. Algoritmy rozdělující obraz na pozadí mohou používat několik technik k docílení. První možností je využití barevných modelů k detekci lidské kůže. Další možnost detekce je založena na rozdílu v jednotlivých snímcích způsobeným pohybem. Za První takový algoritmus se dá považovat jednoduchý detektor pohybu, založený na odstranění pixelů, které se nepohybují nebo alespoň zůstávají v určité části videa stejné, tak že se po určité době vrátí na stejnou hodnotu.

Detekce kůže v obraze

Algoritmus vychází z Fleck-Forsythova algoritmu [2] [3] na vyhledávání fotografií s pornografií podle poměru oblastí s kůží a pozadí. Jako vstup do algoritmu je potřeba RGB obraz s 256 stupni na každé složce. Dále je RGB obraz konvertován na logaritmický barevný model IRgBy a z něj je spočítány mapa amplitudových textur, odstín a saturace(2)[3]

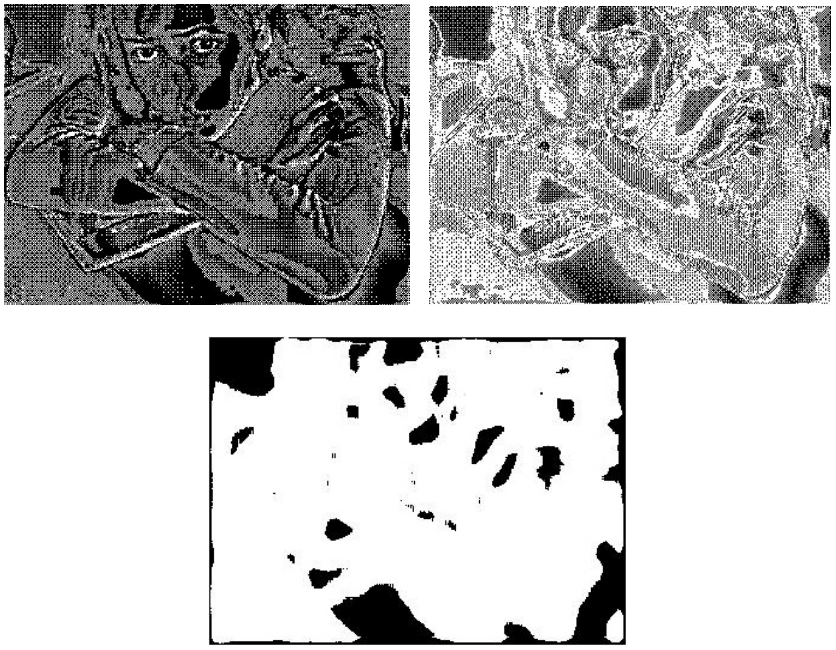
$$\begin{aligned} L(x) &= 105 \cdot \log(x+1), \\ I &= \frac{L(R)+L(B)+L(G)}{3}, \\ Rg &= L(R)-L(G), \\ By &= L(B)-\frac{L(G)+L(R)}{2}, \end{aligned} \tag{1}$$

kde *x* je proměnná reprezentující jednotlivé barevné složky, *R* znamená červenou, *G* zelenou, *B* modrou barevnou složku s rozsahem 256 hodnot a *I*, *Rg*, *By* představují logaritmické hodnoty barevného modelu.

Před konverzí na logaritmický barevný prostor je z 10-ti pixelového okraje obrázku vybrána za každou barevnou složku nejmenší hodnota, která je potom odečtena ze stejné barevné složky každého pixelu. Tím dojde k posunutí barevných

složek k nulové hranici. Poté se z obrazu podle velikosti spočítá hodnota SCALE velká jako součet obou os celého obrázku dělené 320 (320 určil autor algoritmu [2] podle obrázků, které používal 198x122). Hodnota SCALE potom určuje velikost okolí, s nímž bude počítáno. Tím vznikne několik oken, na které je aplikován mediánový filtr, a dojde k vyhlazení a zániku hodně odlišných a málo se vyskytujících pixelů.

Dalším krokem je vypočítání texturových map, které se velice často používají k detekci kůže. Kůže je většinou hladká s malou změnou barvy, takže oblasti s velkým množstvím textur nejsou klasifikovány jako kůže a vyřazují se z dalšího zpracovávání. Texturová mapa je počítána pomocí složky *I*, na kterou aplikujeme mediánový filtr s velikostí osminásobku hodnoty SCALE viz. obr. 2.2. Dále se odečte filtrovaná hodnota I od I a na absolutní hodnotu rozdílu viz. obr. 2.2 se znovu aplikuje mediánový filtr s velikostí matice dvanáctinásobku SCALE.



Obr. 2.2: a) Filtrovaná složka I, b) Rozdíl složky I a I filtr, c) Texturová mapa

Poté co jsme pomocí texturové mapy viz. obr. 2.2 vybrali oblasti, jenž neobsahují kůži z důvodů větší texturové informace, vybereme další oblasti pomocí prahování.

Porovnávání barev s daným rozmezím probíhá pomocí hodnot *hue* (odstín) a *saturation* (nasycení), které získáme výpočtem [2](2) ze složek *Rg* a *By*

$$hue = \arctan\left(\frac{Rg}{By}\right), saturation = \sqrt{Rg^2 + By^2}. \qquad (2)$$

Hranice prahování byly nastaveny podle [2], tak aby zahrnovaly celou barevnou oblast kůže. Rozsahy jsou dva. První je vymezen tak, že texturová mapa nesmí přesahovat hodnotu 4.5, hodnota *hue* musí být od 120 do 160 a hodnota *saturation* se musí pohybovat od 10 do 60. Druhý rozsah je větší a je vymezen hodnotou texturové mapy do 4.5, hodnotou *hue* 150 až 180 a *saturation* 20 až 80. Obrázek projdeme po jednotlivých pixelech a každý pixel budeme testovat, zda nespadá do daných mezí a tam, kde pixely spadají do mezí, uložíme do samostatné matice logickou hodnotu 1. Zobrazením matice, která obsahuje logické hodnoty toho zda příslušné pixely vyhovují, získáme mapu kůže (skin map, viz. obr. 2.3).

Další krok spočívá v tom, že oblasti, které byly definovány jako kůže rozšíříme o pixel v každém směru a tím zvětšíme oblast označenou jako kůže ve skin mapě. Po zvětšení oblastí je obraz ještě jednou zkontrolován prahováním, ale tentokrát se spojeným rozsahem hodnot.

Vynásobením logických hodnot, uložených ve skin mapě a barevných hodnot vstupního obrázku dostaneme výstupní obrázek, na kterém jsou jen oblasti detekované jako kůže viz. obr. 2.3.

Obr. 2.3: a) Vstupní obraz, b) Skin map, c) Oblasti s detekovanou kůží

Převod obrázku na stupně šedi

Některé algoritmy nepoužívají barevných informací, ale vystačí si jen se stupni šedi, které zároveň sníží množství zpracovávané informace. K převodu barevného modelu na 256 stupňů šedi můžeme použít algoritmus, který podle vzorce [5](3) počítá ze všech barevných složek RGB jas I jednotlivých pixelů, který je pak stejně uložen do všech barevných RGB složek, čímž vznikne stupeň šedi.

$$I = 0{,}3 \cdot R + 0{,}59 \cdot G + 0{,}11 \cdot B \qquad (3)$$

Obr. 2.4: a) Vstupní obraz, b) Obraz převedený na stupně šedi pomocí vzorce

Detekce člověka pomocí detekce pohybu počítáním ADF

Detekce člověka na základě pohybu spočívá v tom, že člověk, který je před kamerou alespoň nepatrně pohybuje a my této změny můžeme využít k jeho detekci. Algoritmus se zakládá na změnách pixelů ve snímku (pohybu v čase), takže je nutné, aby pozadí za člověkem bylo statické a na pozadí by nemělo být velké množství výrazně pohybujících se částí, což by ztížilo pozdější detekci obličeje. Na počátku algoritmu [5] se určí z videa obrázek s indexem i, který bude sloužit jako referenční, většinou ten kde by měla začít sekvence, na kterou budeme používat rozpoznávací algoritmus. Pokud bude rozpoznávání aplikováno na celé video nebo v reálném čase bude sekvencí celé video (*i* = 0, *N* = konec videa), nebo můžeme tento interval posouvat v čase a tím zvýšit odolnost algoritmu na změny na pozadí. Dále určíme počet snímků, které budou zpracovány *N*. Pak se celá sekvence [5](4) odečítá od obrázku s indexem *i*. Tam kde zůstanou absolutní hodnoty větší než nula, je detekována změna v jasu, tedy pohyb oproti referenčnímu obrázku.

Po odečtení vznikne *N* rozdílných rozdílových matic, které v intervalu

$$[(i+1),(i+N)] \tag{4}$$

sečteme, vznikne ADF (accumulated diference frame, akumulační rozdílový rámec). Pak se na ADF matici aplikuje prahování tak, aby zanikly hodnoty, jenž vznikly šumem nebo nepodstatným pohybem. Následně se určuje, zda nastal na snímcích pohyb, který se bude zpracovávat. Hranice je dána poměrem pixelů, které překročily prahovací hranici a počtem pixelů v celém snímku, tak že tento poměr

$$\frac{pixely\_nad\_hranicí}{pixelů\_celkem} \succ \Theta \tag{5}$$

musí přesáhnout hodnotu θ = 0.015 [5].

V dalším kroku se odstraňují oblasti s detekovaným pohybem podle velikosti a to pokud změna jasu nenastala ve čtverci větším, něž 16 pixelů. Po vynechání menších oblastí je možné zprůměrovat pozice zbylých pixelů, abychom mohli určit střed výřezu, kde se nachází hlava.

Detekce pohybu člověka pomocí detekce průměrného posunu

Tento algoritmus [4] byl navržen, aby určoval trasu pohybu lidí nebo skupin na základě změn hustoty dat jednotlivých snímcích. Rozdílový binární obraz se počítá každý druhý snímek potom je z něj odfiltrován šum. Dále jsou pomocí funkce lokálních maxim hustot odlišeny jednotlivé objekty

$$\nabla f(x) = \frac{2c_{k,d}}{nh^{d+2}} \sum_{i=1}^{n} (x_i - x) g\left(\left\| \frac{x - x_i}{h} \right\|^2 \right), \qquad (6)$$

kde x je n počet bodů, *h* je poloměr, pro který je hustota počítána, *d* je počet dimenzí prostoru, $c_{k,d}$ jsou normalizační konstanty. Funkce nám pomůže určit trasu pohybu objektů a to i v případě několika se pohybujících objektů, pro každý objekt se vektor pohybu bude lišit. Algoritmus je velice rychlý a může pracovat v aplikacích s reálným časem.

Rozlišení pozice člověka pomocí pohybu, barvy kůže a barevných skvrn

Další možnost, jak detekovat člověka je vidět na postupu [10], který využívá detekci pohybu i barev, aby oddělil člověka od pozadí. Poté co je kandidátní oblast vybrána pomocí detekce pohybu je rozlišena na oblast, která obsahuje kůži a oblast, kde je oblečení. Data získaná pomocí diferenčních matic jsou upravena podle jednotlivých oblastí, tak aby tyto oblasti byly kompaktní v horizontální rovině a nepřekrývali se navzájem. Takto zpracovaný obraz podrobíme výpočtu Mahalanobisovým vzdálenostem pro každý shluk, podle čehož můžeme jednotlivé oblasti separovat. Algoritmus je navržen pro detekci člověka od pasu nahoru a předpokládá, že člověk má na sobě nevýrazné oblečení, které lze pomocí barev i textur rozlišit a tím pomoci k určení pozice člověka.

Mahalanobisovy vzdálenosti

Mahalanobisovy vzdálenosti slouží k určení vzdálenosti bodu od gravitačního středu oblasti jinak, něž jako u euklidovské vzdálenosti dvou bodů:

$$d(x_1, x_2) = \sum_i (x_{1i} - x_{2i}), \tag{7}$$

ale za pomoci korelace vstupních proměnných a za pomoci pravděpodobnosti jsme schopni určit, zda nějaký testovaný bod spadá do určité množiny dat. Korelací vstupních proměnných se docílí toho, že vzdálenosti bodů nebudou závislé na určeném měřítku. Pokud máme skupinu vstupních dat

$$x = (x_1, x_2 \ldots x_p) \tag{8}$$

se středníma hodnotami

$$\mu = (\mu_1, \mu_2 \ldots \mu_p) \tag{9}$$

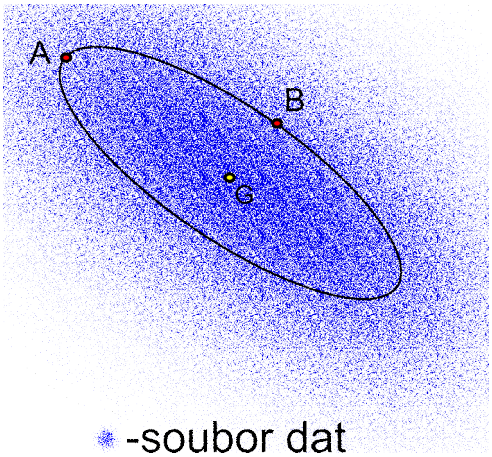
tak Mahalabisovy vzdálenosti spočítáme pomocí vztahu

$$D_M(x) = \sqrt{(x-\mu)^T \sum^{-1} (x-\mu)}. \tag{10}$$

Vztah (10) upravený pro vzdálenosti dvou vektorů, kde *x* a *y* mají shodné rozložení s kovarianční maticí *P*

$$d(\vec{x}, \vec{y}) = \sqrt{(\vec{x}-\vec{y})^T P^{-1} (\vec{x}-\vec{y})}. \tag{11}$$

Pokud víme, kde se nachází střed *G* celé skupiny dat a potřebujeme zjistit, zda nějaký bod spadá do této skupiny, odhadneme odchylku vzdálenosti tohoto bodu od středu a určíme, zda vzdálenost bodu pro tento směr je menší nebo rovná jedné odchylce v tomto směru a tím spadá do množiny, viz. obr. 2.5, kde body *A* a *B* mají stejnou Mahalanobisovu vzdálenost od středu *G*.



Obr. 2.5: Elipsa Mahalanobisovy vzdálenosti

2.2.2 Metody přímé detekce

Tyto metody fungují na principu detekce rysů vlastních pro lidské tělo. Nejedná se jen o geometrické vlastnosti, ale také pohybové, jako je třeba pohyb úst nebo mrkání. Většinou se v metodách s přímou detekcí setkáváme s detekcí tvarů, hran, barevných skvrn, vlasů a celé nebo částečné rozpoznání siluety člověka. Nezřídka kdy bývá u těchto typů detekce použita databáze, se kterou se zpracovaná vstupní data porovnávají.

Metoda detekce založena na vzdálenosti prvků těla

Metoda používá relativní vzdálenosti a umístění prvků lidského těla. Na počátku zpracování se obraz rozdělí na *M* sloupců a *N* řádků. Inicializuje se matice vzdáleností velikostí *MN* sloupců a *MN* řádků a ty jsou naplněny rozdílovými hodnotami jednotlivých bloků. Tato diferenční matice je porovnávána s podobně zpracovanou databází obrázků pomocí Mahalanobisovy vzdálenosti, která rozhodne zda se jedná o statisticky podobný obraz jako v databázi. Pro metody na hledání podobnosti dvou signálů nebo obrazů pomocí matic vzdáleností používáme pojem dynamické programování.

Detekce obličeje s použitím charakterizujících předloh

Metoda k rozeznávání obličeje [9] používá množinu předloh, která je připravena z fotografií lidí. Tyto fotografie jsou vyrovnány na sebe, tak aby se překrývala ústa a oči. Dále je z těchto fotografií vypočítána průměrná hodnota a vypočítána kovarianční matice. Kovarianční matice je tvořena průměry rozdílů hodnot obrázku a středních hodnot, ze kterých potom extrahujeme charakterizující vektory. Vektory jsou rozlišeny dle toho zda jsou výrazné a porovnány s vektory z databáze podle jejich množství. Z vektorů mohou vyplývat i symetrické znaky očí a obočí, protože v těchto oblastech se bude jednat o velkou změnu v okolí.

2.3 Metody detekce mrknutí

Detekovat mrknutí oka, lze využitím několika v základu rozdílných algoritmů, kde každý má své výhody a nevýhody. S úspěšností algoritmu rostou i požadavky na kvalitu obrazu. Například u přesného zaměření duhovky je potřeba vysokého rozlišení, které nám poskytuje úplný popis pohybu očních víček.

2.3.1 Detekce mrknutí pomocí rozpoznání duhovky

Detekce mrknutí pomocí rozpoznání duhovky využitím výpočtu kruhové intenzity

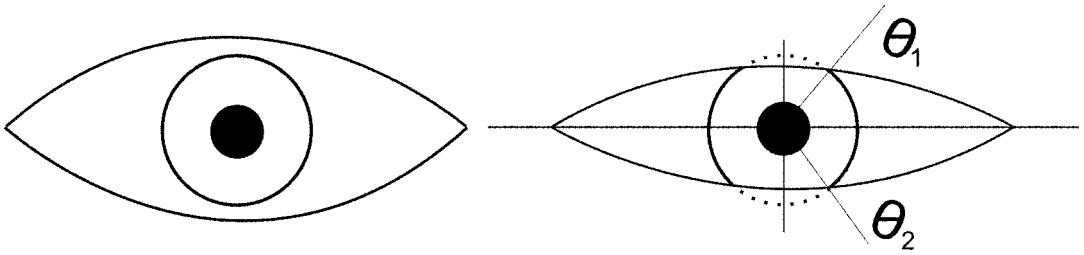
V první řadě se pro algoritmus [7] musí určit střed oka pomocí detekce panenky, která je kruhová a tmavá, nebo pomocí detekce duhovky, která může být podobně tmavá jako panenka. Na výřez oka se aplikuje jeden z vyhlazovacích filtrů, buď mediánový nebo rozostřovací, a to v takové míře, aby nedošlo k velikému posunu hrany duhovky. Dále se pomocí horizontální osy, která prochází vertikálním středem nejtmavší oblastí panenky a změny jasu na ní určí začátek a konec duhovky.

V dalším kroku algoritmus určí přesnou pozici duhovky z již získaného středu oka a prvního obrázku, jenž není filtrován a je součástí předchozích algoritmů na detekci očí. Algoritmus je založen na vlastnostech oka. Duhovka je tmavší než okolní bělmo, je kulatá a nachází se v oblasti výřezu oka.

Kulatosti duhovky využívá vztah, který určuje funkci intenzity kruhové oblasti kolem bodů x_0 a y_0 s poloměrem r, v rozmezí $\Theta = 0$ až 2π:

$$f_{\Theta}(x_0, y_0, r) = \int_{\theta \in \Theta} I(x_0 + r\cos\theta, y_0 + r\sin\theta) d\theta \,. \qquad (7)$$

Rozmezí $\Theta 0$ až 2π se většinou nenalézá v plném rozsahu, protože duhovka může být částečně až plně překryta spodním či horním víčkem, takže výpočet probíhá od horizontální osy oka nahoru i dolu, vlevo i vpravo viz. obr. 2.6 a testuje se jaká intenzita je za hranicí poloměru a jaká před hranicí, aby se určilo, zda se ještě jedná o duhovku nebo o jedno z víček.



Obr. 2.6: Otevřené oko a částečně přivřené s rozmezím

Pokud jsme již našli přesnou pozici duhovky můžeme pomocí vztahu (7) určovat jak moc je oko otevřené a tak ve videu pomocí Θ určovat, zda se oko zavírá nebo otvírá a tím detekovat mrknutí.

Určení pozice duhovky pomocí detekce hran

Tato metoda [8] slouží k detekci a vyjmutí duhovky, která je pak roztažena po obvodu panenky. Tato data jsou dále zpracována a použita k identifikaci majitele. Na rozdíl od předchozího algoritmu se neuvažuje se soustřednými a kruhovými vlastnostmi panenky a duhovky. Prvním krokem této metody je hledání středu využitím aproximace obvodu duhovky podle horizontální osy.

Obraz je také předzpracován vyhlazovacími filtry a poté je navýšen jeho kontrast k výraznějšímu přechodu přes hrany. Detekce hran je provedena filtrováním Haarovou vlnkou po horizontální přímce, která zdůrazní vzrůst či pokles světlosti a tím detekuje hranu. Následně co známe filtrovaný průběh ze středu panenky, jsou v průběhu vyhodnoceny zlomy a tím určeny poloměr duhovky i panenky.

2.3.2 Detekce mrknutí pomocí diferenční matice

Pokud pro detekci mrkání máme již připraven výřez oka, zpracovávat můžeme jen tyto data pomocí algoritmu, který vyvinul M.Turk [9]. Algoritmus má za úkol určit pozici obou očí a to i za okolností kdy je hlava nakloněna do strany. Výpočet algoritmu používá diferenčních map pro detekci.

Prvním krokem je inicializace matice na nulové hodnoty diferenční mapy σ^2 o velikosti *M*x*N*, velikosti výřezu. Dalším krokem je inicializace průměrovaného obrázku μ o velikosti *M*x*N* s rozsahem 24 bitů na pixel, kterému jsou přiřazeny hodnoty prvního obrázku.

Dalším krokem je upravení diferenční matice a průměrové matice podle druhého obrázku podle vzorců (8) a (9). Vzorec (8) má za úkol zpracovat hodnoty, které jsou obsaženy v posloupnosti matic vyjadřujících videosekvenci

$$\mu_{j+1}(x,y) = \frac{j\mu_j(x,y) + I_{j+1}(x,y)}{j+1}, \tag{8}$$

$$\sigma^2_{j+1}(x,y) = \left(1 - \frac{1}{j}\right)\sigma^2_j(x,y) + (j+1)\left(\mu_{j+1}(x,y) - \mu_j(x,y)\right), \tag{9}$$

kde μ označuje matici průměrovacího obrazu, σ je matice varianční mapy a *I* je vstupní obraz. V dalším kroku se diferenční mapa nechá projít prahováním s prahem 255, aby matici nebyly přesycené hodnoty. Poté se diferenční mapa podrobí druhému prahování dle velikostí oblastí, kde došlo k zanechání informací ve varianční mapě, tak aby malé a velké oblasti byly vynechány. Počet pixelů, jenž prošel přes prahovací filtr určuje v poměru s celkovým počtem pixelů v okně výřezu,

že došlo k mrknutí oka. Diferenční mapa je teď matice s binárními pixely, které jsou buď šum anebo jsou spojeny do větších celků. Na tyto pixely jednotlivých oblastí můžeme aplikovat algoritmus, který pixel po pixelu nesoucím stejnou informaci jako jeho soused v diferenční mapě začne procházet oblast ve všech osmi směrech tak, aby odlišil samostatné pixely nebo menší oblasti od oblastí určité velikosti a polohy. Tím, že algoritmus obkrouží celou oblast vznikne kontura této oblasti. V obrázku by po zpracování měly zůstat dvě tyto výrazné kontury v horní třetině našeho výřezu z detekce na základě pohybu. Tyto dvě kontury by měly být přibližně horizontálně elipsovité se špičkami na stranách. Tyto špičky využijeme, abychom zjistili jak jsou oči nakloněny a kde je jejich střed. Po určení bočních mezí oka určíme dolní a horní maximum oka.

2.3.3 Detekce očního víčka

Metoda dle Uzunové [15] pracuje na základě detekce hran, která je uskutečněna horizontálním Sobelovým jádrem. Filtrace Sobelovým operátorem zvýrazní hrany dané oběma víčky a jejich řasami, ale také hranu způsobenou přehybem měkké tkáně mezi obočím a víčkem. Ostatní hrany vrásek nejsou tak masivní, takže je nutné jen správně odlišit hrany přehybu od víčka. Identifikace hran probíhá určením v jaké změně kontrastu se každá hrana nachází. Vrásky a přehyb jsou obklopeny kůží, ale víčko je z jedné strany obklopeno kůží přechodem přes řasy nakonec k oku. Druhou možností jak odlišit vzniklé hrany je přes barevný model, tak že z barevného modelu odstraníme zelenou a modrou složku, tak nám zůstane červená, která je hodně v oblasti kůže zastoupena. Z použití barevného kanálu plyne podmínka, že touto metodou lze zpracovávat jen barevný obraz.

Z již získaných souřadnic hran spodního i horního víčka vzniknou Beziérovy křivky, z kterých můžeme určit do jaké míry je oko otevřeno.

3 Vlastní návrh programu na detekci mrkání

Mé vlastní řešení jsem založil na základu dvou algoritmů. Prvním z nich je algoritmus dle Morris, T., Blenkhorn P., Zaidi F. [5], kde je použit takzvaný ADF (acumulated diference frame). Druhým je algoritmus [13] vytvořeným Grauman K., Betke M., Gips J., Bradski R. G., který má pomáhat lidem s omezeným pohybem ovládat počítač, který následně využívá diference v jednotlivých snímcích. Program jsem navrhnul a rozdělil do několika částí, tak aby se na jednotlivých částech dalo samostatně pracovat s výstupy předchozích podprogramů. Rozdělení programu do několika podprogramů bylo nutné i z časových důvodů, protože prostředí MATLAB nemůže načíst větší část videa, jelikož je video hned ukládáno do nekomprimovaných matic, což je výrazně paměťově náročné. Proto tak jsem video nechal zpracovávat postupně po částech o 10-20 snímcích. Z časových důvodu také, protože načtení videa v MATLABU a následné celkové zpracování dat je velice náročné na čas. V následujících kapitolách popisuji funkci těchto podprogramů a jejich vlastnosti.

3.1 Hlavní program

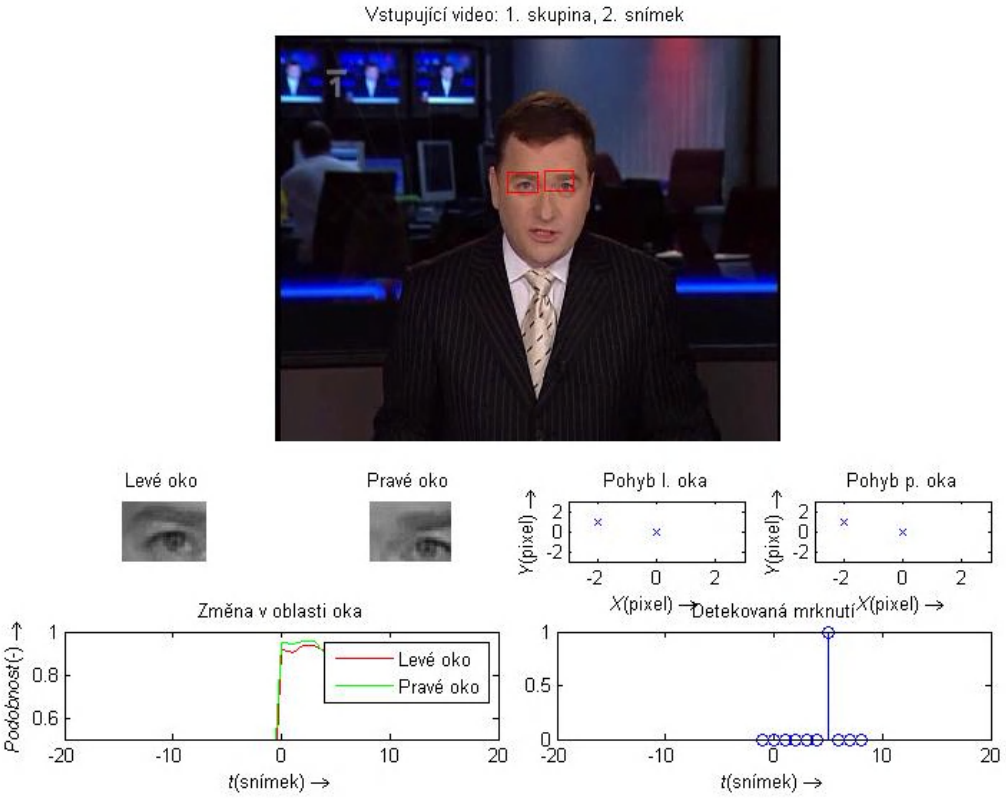
Funkcí hlavního programu je, aby načítal vstupní video, inicializoval proměnné, které se budou v hlavním programu zobrazovat. Načítání videa probíhá postupně (podle délky videa dělené velikostí segmentu). Takže pokud délku segmentu zvolíme 10 snímků z videa se vyjme 10 snímku, které se uloží do matic, které jsou zpracovávány dalšími volanými podprogramy.

Délka segmentu je prvním parametrem ovlivňujícím funkčnost celého programu, jelikož se v dalším stupni počítá akumulační diferenční matice, délka segmentu zásadně ovlivní to, jak bude vypadat diferenční akumulační rámec. Délka segmentu ovlivňuje akumulační rámec v závislosti na rozmezí pohybu sledovaného člověka. Například, pokud zvolíme malý segment a pohyb člověka nebude dostatečně výrazný, akumulační rámec bude mít velice malé hodnoty, takže další algoritmus na detekci oblasti očí nebude schopen označit oblast, kde se nacházejí očí viz. obr. 3.1. Naopak pokud zvolíme segment ke zpracování příliš velký a subjekt se bude během této doby výrazněji pohybovat oblasti změny budou velké s velkými hodnotami v jednotlivých pixelech viz. obr. 3.1.

Algoritmus je náchylný zvláště pokud se jedná o pohyb v horizontální ose, jelikož jako testovací video bylo použito zpravodajství, kde je tmavé pozadí, může stát, že jednotlivé oblasti se překryjí a stane se z nich jedna.

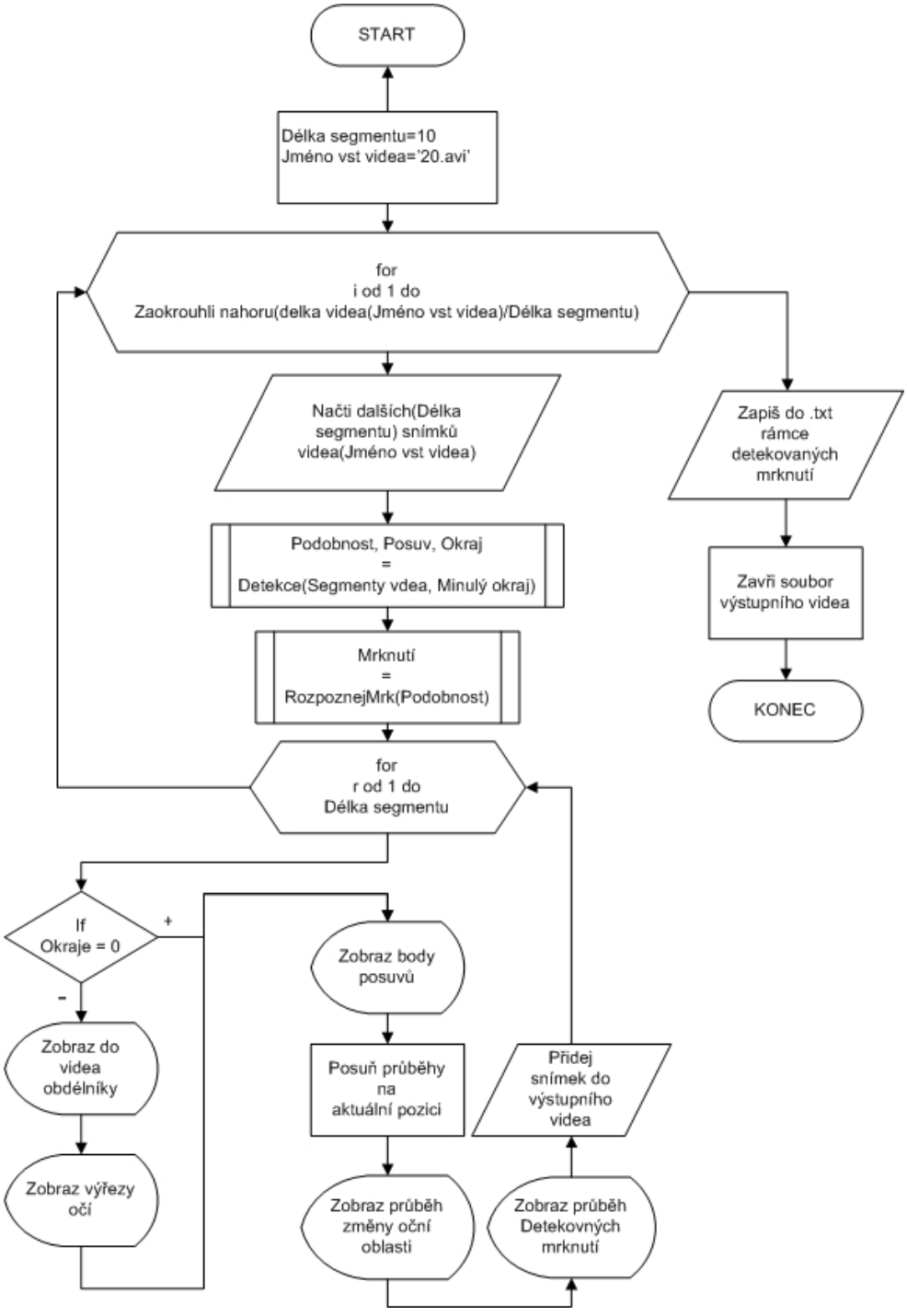
Obr. 3.1: Akumulační diferenční rámec se zvoleným a) malým segmentem, b) velkým segmentem

Další úlohou hlavního programu je prezentovat hodnoty vrácené funkcemi, jenž měly za úkol detekovat mrknutí. Prezentační plochu viz. obr. 3.2 jsem rozdělil na čtyři sloupce a čtyři řádky. V horní polovině je zpracovávané video s vyznačenou oblastí detekovaných očí z akumulační rámce. Dále jsou vlevo pod videem zobrazeny vyjmuté oční oblasti ve stupních šedi, vpravo je vyobrazen detekovaný posun levého a pravého oka. Vlevo dole je průběh velikosti podobnosti oblasti očí. A vpravo dole jsou vykreslena detekovaná mrknutí.



Obr. 3.2: Plocha prezentace výsledků programu

Data jsou předzpracována po segmentech, potom jsou zobrazena po jednotlivých snímcích a průběhy výsledných hodnot jsou posouvány, tak že na nule je vždy aktuální hodnota. Celý postup práce algoritmu je vyjádřen vývojovým diagramem viz. obr. 3.3.



Obr. 3.3: Vývojový diagram hlavního programu

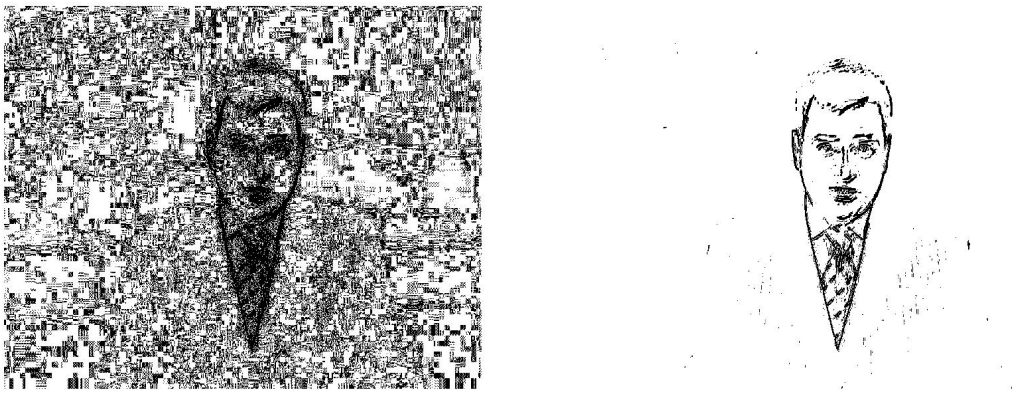
3.2 Podprogram Detekce

Další část skriptu shrnuje všechny detekční algoritmy celého programu. Na začátku skriptu se převede barevné video na čtyřdimenzionální matici normovanou na zlomek jednotlivých barevných složek, tzn. každou hodnotu dělenou maximální

možnou hodnotou jasu (255). Po inicializaci dalších proměnných převádíme matici na třídimenzionální se stupni šedi.

Největší část programu se zpracovává uvnitř smyčky, která prochází video od druhého snímku do snímku určeném velikostí vyjmutého segmentu. Prvním příkazem uvnitř smyčky je odčítání aktuálního snímku od předchozího, proto smyčka začíná druhým snímkem. Matice vzniklá odečtením dvou snímků se prahuje s hodnotou, kterou jsem přímo zvolil jako 4/255.

Hodnota prahu pro převod diferenční matice na binární obraz je dalším parametrem programu a pro jeho úplnou automatizaci by se tato hodnota měla v programu počítat pomocí funkcí v závislosti na míře pohybu ve videu, anebo za použití statistických metod aplikovaných na rozdíl dvou odečtených rámců. Nastavení správného prahu může velice urychlit práci a zvýšit procentuální šanci na správnou detekci. Nastavení vyšších hodnot prahu (2/255 a více) je nutné také kvůli odstranění šumu, který vznikl kvantizací a ve větší míře kompresí videa viz. obr. 3.4. Hodnotu prahu jsem zvolil na základě vzorků čtyř videí, tak aby byla co možná nejmenší kvůli velké ztrátě informací, ale dostatečně velkou na odstranění šumu nebo jiných nepatrných projevů změny v jednotlivých pixelech snímků. Rozdíl mezi správným a špatným nastavením je patrný z obr. 3.4.



Obr. 3.4: Obraz diferenční matice a) se špatným nastavením prahu, b) a dobrým nastavením

Po binarizaci rozdílové matice jsou dalším podprogramem OdstranMaleOblasti z obrazu odfiltrovány oblasti s menší plochou než 80 pixelů, což výrazně diferenční matici pročistí matici viz. obr. 3.5. Odstranění malých oblastí probíhá na základě sousedství kladných pixelů.

Obr. 3.5: Obraz diferenční matice a) před filtrací, b) po odstranění malých oblastí

Potom jsou binární data diferenční matice analyzována jako oblasti dotýkajících se pixelů. Pro každou oblast kladných hodnot jsou nalezeny sloupce, v kterých se nachází kladná hodnota a podle nich jsou určeny průměrné hodnoty sloupců pro tuto oblast. Tyto průměrné hodnoty sloupců jsou porovnávány s číselnou hodnotou levé a pravé hranice obličeje, a ty binární oblasti, které leží mimo hranici jsou odstraněny rozdíl mezi obr. 3.5b a obr. 3.6. Meze obličeje jsou určeny pomocí horizontální hustoty.

Obr. 3.6: Obraz diferenční matice po odstranění oblastí mimo oblast zájmu

Takto upravená binární matice se po normování počtem snímků přičítá k akumulační diferenční matici. Přičítání k akumulační diferenční matici probíhá tolikrát, kolik je do podprogramu na vstupu vloženo snímků. Pro testovaná videa jsem volil deset a dvacet snímků, podle míry pohybu. Po ukončení smyčky se zavolají podprogramy na hledání očí a detekci změny, jejichž vstupem je akumulační rámec pohybu. Vznik akumulačního rámce z jednotlivých diferenčních matic viz. obr. 3.7 je zobrazen na obr. 3.8. Celkový popis algoritmu skriptu Detekce je popsán pomocí vývojového diagramu viz. obr. 3.10.

Obr. 3.7: Jednotlivé nenormované diferenční snímky v pořadí 2 až 9

Obr. 3.8: Akumulační diferenční matice, která vznikla sečtením normovaných diferenčních snímku

Obr. 3.9: Prahovaná akumulační matice s prahem 0,4

Obr. 3.10: Vývojový diagram podprogramu Detekce

3.3 Podprogram na lokalizaci očí

Podprogram nazvaný NajdiOci je nejdůležitější částí celého programu na detekci mrknutí. Jeho úkolem je detekovat pozici očí ze vstupních dat, což je akumulační rámec a číslo obsahující velikost segmentu videa, které se zpracovává.

Prvním krokem programu je výpočet parametru pro práh, od kterého budeme pokládat hodnotu pixelu jako logickou jedničku. Tedy podobně jako u předešlé kapitoly, zde nedochází k binarizaci jednotlivých diferenčních snímků, ale k binarizaci celé akumulační matice. Tuto matici jsem v programu nazval jako bwSum. Dalším rozdílem je, že tato data neznázorňují jen velikost změny v pixelu, ale také v kolika snímcích se pixel nachází.

Jako dalšímu zpracování se bwSum podrobí filtraci, v které jsou odstraněny oblasti menší než 20 pixelů. A kvůli vyplnění a rozšíření oblastí jsem použil dilataci kruhem o průměru 2 pixely, která kruhové okolí logické jedničky vyplní jedničkami viz. obr. 3.11.

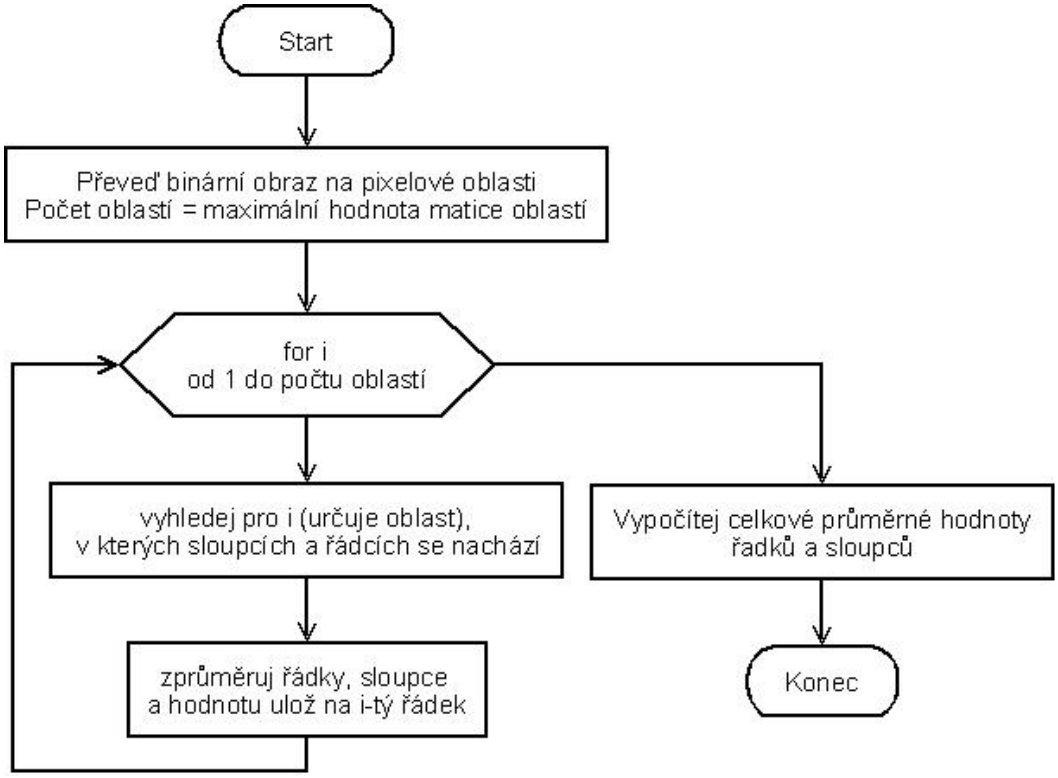
Obr. 3.11: Filtrace bwSum a) před dilatací, b) po dilataci

Odstraněním oblastí, které se pomocí dilatace nespojili s ostatními z obrazu odstraníme oblasti menší než 200 pixelů. Odstranění těchto oblastí také urychluje chod programu, jelikož potom není nutné procházet tak velké množství dat při jejich analýze. Po filtraci a odstranění malých oblastí z bwSum dostaneme data, která jsou už vhodná k rozpoznávání viz. obr. 3.12.

Analýza oblastí probíhá z toho důvodu, abychom zjistili, kde se nachází levá a pravá strana obličeje. Levou část zjistíme pomocí nejnižšího sloupce a pravou pomocí nejvyššího. Rozbor oblastí jsem už jednou použil v podprogramu Detekce k filtrování pixelových oblastí a je použit ještě několikrát, ale neudělal jsem z něj funkci,

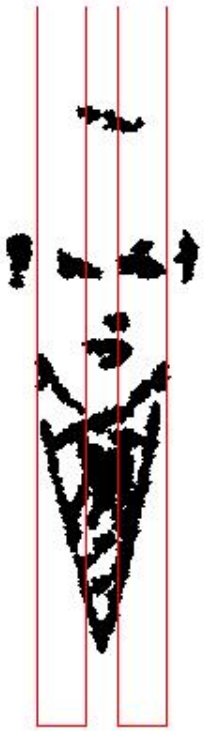
protože některá jeho výstupní data nejsou potřeba, takže jsem jen vynechal nepotřebné výpočty. V této sekci programu se ovšem používá v plném rozsahu a jeho funkce je vidět na vývojovém diagramu viz. obr. 3.13. Jeho vstupní data tvoří matice, která obsahuje hodnoty od jedné až do počtu jednotlivých pixelových oblastí podobně jako obr. 3.12, kde rozdílné hodnoty jsou převedeny na odlišné barevné odstíny. Analýza oblastí probíhá ve smyčce, která určuje s jakou oblastí se právě pracuje.Pomocí příkazu find v matlabu najdeme všechny sloupce a řádky, přes které se oblast rozprostírá. Hodnoty řádků a sloupců zprůměrujeme a uložíme do proměnné tak, aby hodnota byla na řádku odpovídající oblasti.

Obr. 3.12: Obraz rozdělených oblastí



Obr. 3.13: Vývojový diagram na analýzu oblastí

Dalším krokem k určení pozice očí je vzdálenostní poměřování vlastností jednotlivých oblastí, tedy zda se některá z oblastí nachází v rozmezí v určitých horizontálních mezích. Tyto proporce obličeje jsem odhadl na základě několika vzorků akumulačních map a vyšly mi hodnoty 2/12, 5/12 pro levé oko a symetricky pro pravé oko 7/12, 10/12 viz. obr. 3.14. Tyto hodnoty jsou pomocí zjištěných hodnot pravé a levé strany obličeje porovnávány s průměrnými sloupci oblastí. Jelikož dochází jen k porovnávání v horizontální ose může se stát, že budou do mezí spadat i jiné oblasti z horní části hlavy nebo takové, které vznikly pohybem šatů jak je vidět na obrázku část límce. Následně je oblast testována, zda její poměr výšky a šířky je v rozmezí jedna ku jedné až jedna ku dvěma.



Obr. 3.14: Zobrazení mezí pro určení očí

Špatná detekce zvláště vadí v případě, když je vybrána oblast jen v jednom rozmezí, levém či pravém a potom nedojde ke správnému určení osy obličeje. Hlavní idea této části programu je, že osa obličeje se nachází přesně mezi očima. A pomocí osy jsme schopni dále symetricky zpracovávat obraz.

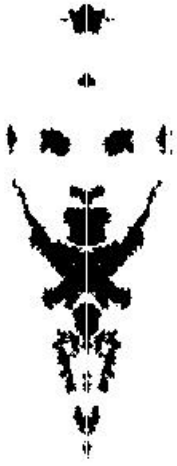
V prvním návrhu algoritmu jsem přímou detekci oblastí očí nepoužíval, protože osu obličeje jsem určoval pomocí celkového průměrného sloupce, což se při testování na víceru videosekvencích ukázalo jako nevhodné řešení viz. obr. 3.15. V konečné fázi programu se osa obličeje pořád určuje pomocí průměrného sloupce, ale pokud bude tento test na meze oblastí pozitivní, tak se hodnota osy upraví na střed mezi oblasti detekované jako oči. Další podmínkou pro úpravu hodnoty osy je, že levá oblast se výrazně neodlišuje od pravé oblasti svou průměrnou pozicí ve vertikální ose.

Obr. 3.15: První fáze určení středu obličeje, průměrným sloupcem

Pokud máme určen střed obličeje, tak pro další zpracování vypočítáme vzdálenost stran obličeje zleva i zprava od středu a otestujeme, zda obě hodnoty nepřesahují velikost matice. Následně testujeme, zda hodnoty osy obličeje a vzdálenost od ní vlevo a vpravo se vejdou do rozměrů obrazu, a když ano, zpracujeme obraz podle jeho souměrnosti. Když nedojde ke správnému výpočtu hodnot rozměrů obličeje je další zpracování matice přeskočeno.

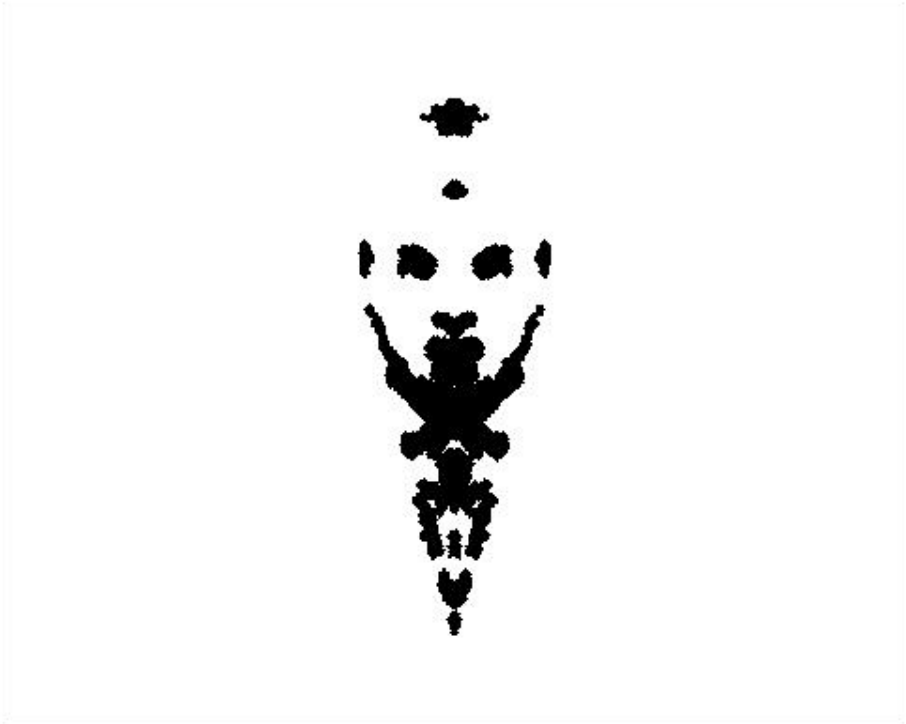
Další zpracování probíhá rozdělením obrazu na dvě části, tak jak byl obličej rozdělen jeho osou viz. obr. 3.16. Po rozdělení obrazu na dvě části se jeho levá polovina převrátí a vynásobí s pravou polovinou, čímž vznikne středově vyrovnaný obraz viz. obr. 3.17.

Obr. 3.16: Rozdělený obraz a) levá převrácená strana, b) pravá strana



Obr. 3.17: Složený souměrný obraz

Součin obou polovin obrazu se spojí v jeden binární obraz, který slouží jako binární maska původní akumulační matice. To znamená, že po úpravě středově souměrného obrazu tak, aby z něj vymizela mezera označující osu užitím dilatace, jím vynásobíme původní akumulační rámec viz. obr. 3.18. Použitím středové souměrnosti při zpracování obrazu docílíme odstranění nežádoucích výčnělků oblastí a také rozdělením nesouvisejících oblastí, což je nezbytné k přesnému vytyčení očí.



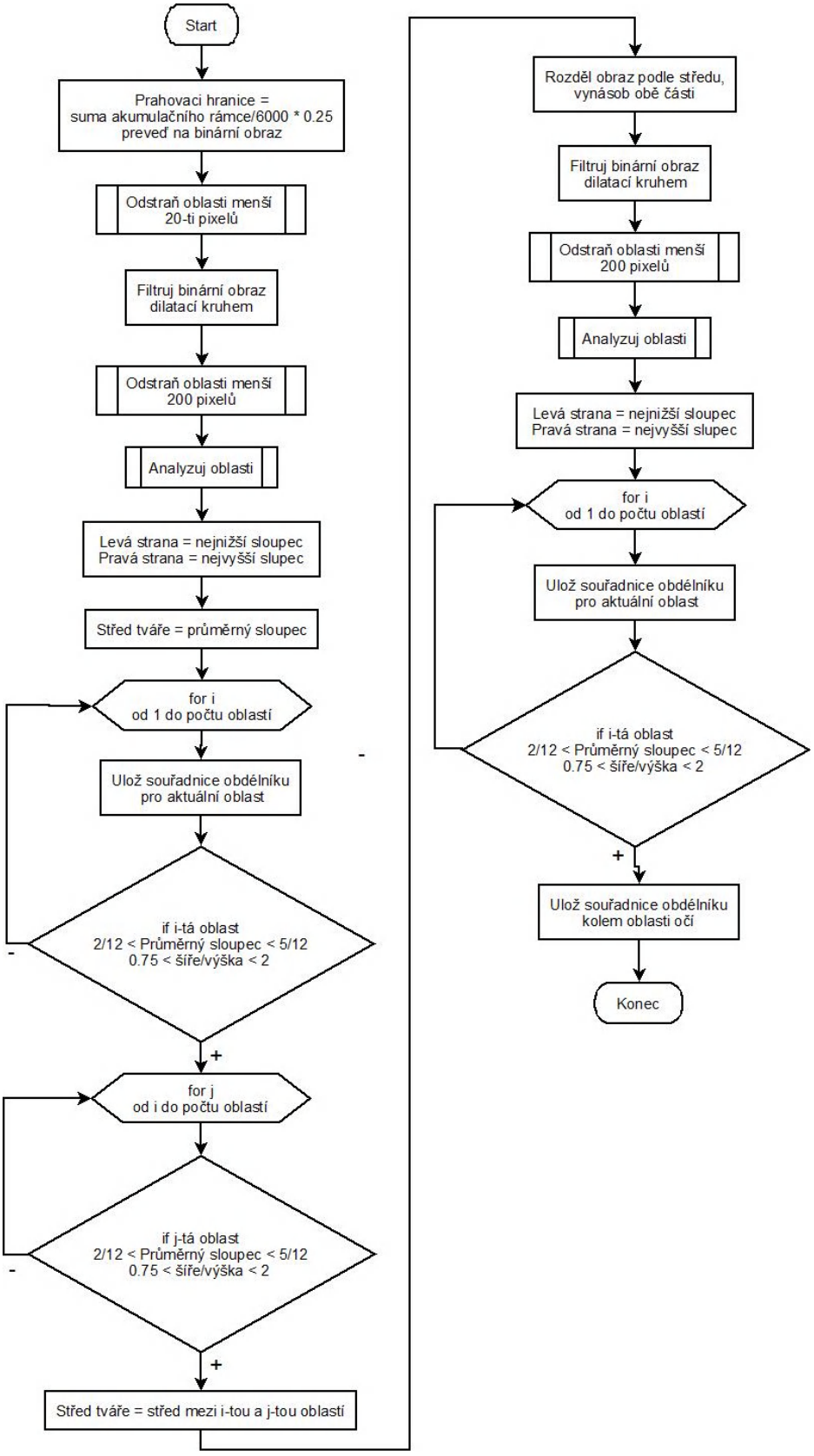
Obr. 3.17: Výsledek vynásobení dilatované masky a akumulačního rámce

Po získání binárního obrazu daného středově souměrnou maskou násobené akumulačním rámcem, tento obraz podrobíme další oblastní analýze, která vypočítá jejich průměrné sloupce. Potom se každá oblast podrobí testování zda jejich průměrný sloupec spadá do horizontální oblasti určené jako rozmezí 2/12 až 5/12 z celé šíře obličeje. Testuje se jen levé oko z toho důvodu, že obraz je symetrický, takže oblast pravého oka lze jen dopočítat.

V cyklu na testování oblastí se ukládají souřadnice obdélníku, který obkresluje hranice oblasti a při splnění podmínky pro oblast se z cyklu vyskočí. Získanou oblastí posléze násobíme původní akumulační rámec a s pomocí výsledných dat vzniklých

násobením je průnikem s původním rámcem určena oblast, která zachycuje celkovou pozici oka v průběhu části zpracovávaného videa. Okolí oka je definováno taktéž jako obdélník určený z oblasti původního akumulačního rámce nejnižším a nejvyšším sloupcem a řádkem.

Na závěr algoritmu se testuje, zda jsou výstupní souřadnice obdélníků správné. Pokud nejsou správné, jsou všechny vynulovány pro pozdější snazší určení toho, použít souřadnice z minulého hledání očí.



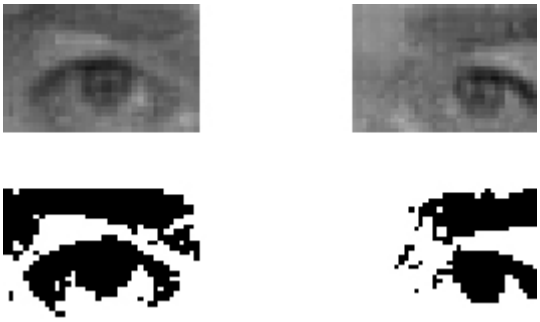
Obr. 3.18: Vývojový diagram podprogramu pro hledání očí

3.4 Podprogram na detekci změny oblasti očí

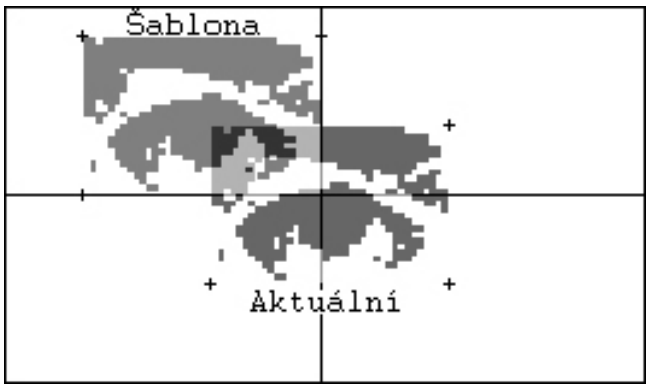
Úkolem tohoto podprogramu je sledovat k jaké změně došlo v okolí očí. Používá se k tomu detekce změny pixelů v oblasti pohybujícího se oka z aktuálního a předešlého snímku.

Prvním návrhem, jak detekovat mrknutí bylo určit kruh opisující duhovku a sledovat jeho úplnost. Ale při jeho implementaci na vzorek videa jsem usoudil, že pro tuto detekci je zapotřebí vysoce kvalitní video s rozlišením více jak 30 pixelů na šíři duhovky, což je nejméně trojnásobné rozlišení. Tak jako metodu detekce jsem zvolil porovnávání šablony a maticí reprezentující oblast oka.

Vstupní data jsou tvořena částí zpracovávaného videa ve stupních šedi, souřadnicemi čtverců obklopující oči a prahem, kterým převedeme šedý obraz na binární viz. obr. 3.19. V cyklu na detekci změny oproti šabloně dochází k porovnávání matice binarizovaného oka z aktuálního snímku z předchozím, tak že pohybujeme šablonou viz. obr. 3.20 po aktuální matici při tom obě násobíme a zaznamenáme nejvyšší hodnotu shody. Při nalezení maximální shody zaznamenáme vektor pohybu, který nastal posuvem šablony vzhledem k matici. Fáze pohybující se šablony probíhá jednou pro každé oko.

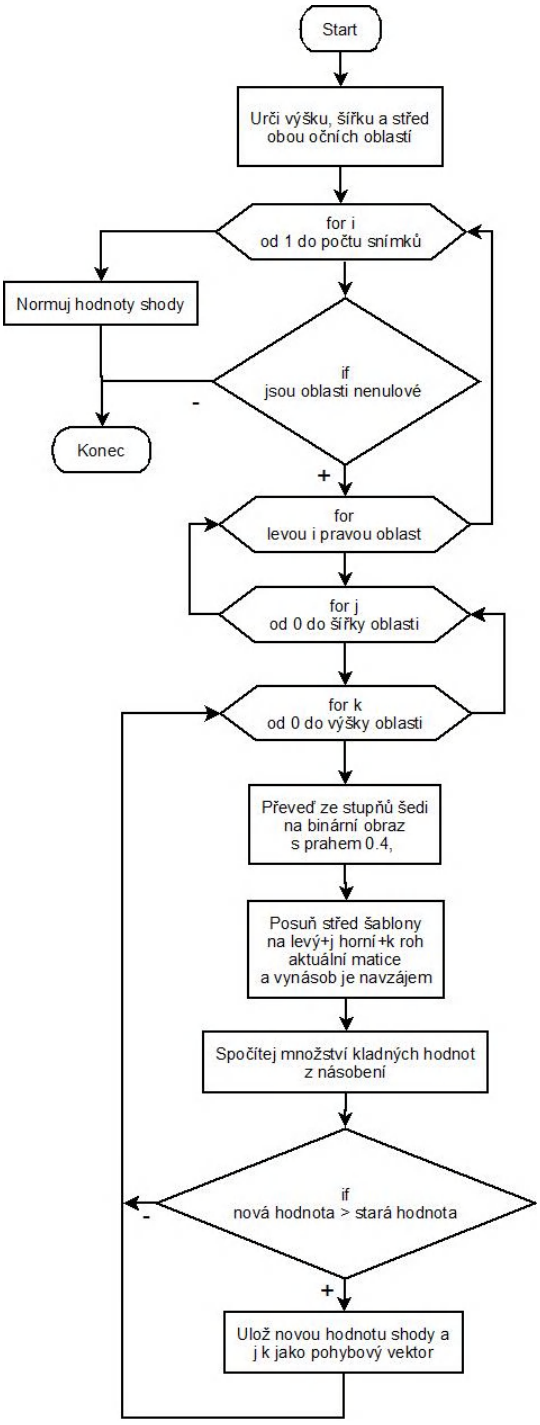


Obr. 3.19: Oblast očí a jejich binární podoba s prahovanou jasovou hodnotou 0.4

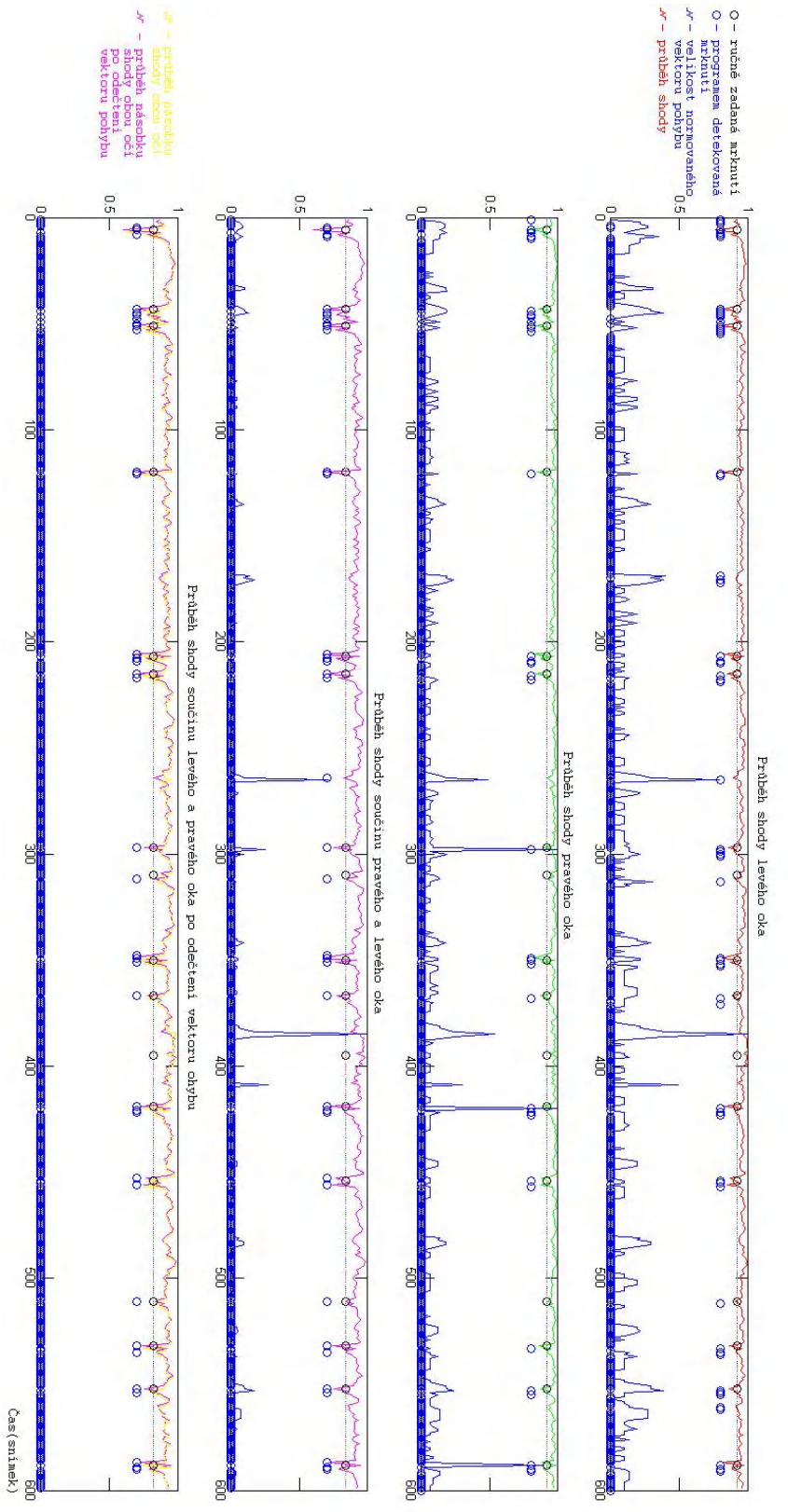


Obr. 3.20: Počátek překrytí šablony s aktuální oblastí

Výstup tedy tvoří hodnoty nejvyšší shody společně s vektorem pohybu aktuální matice oka vůči šabloně. Shoda je vyčíslená jako počet shodných pixelů a před ukončením algoritmu je normována celkovým počtem pixelů v matici. Funkce algoritmu je zobrazena ve vývojovém diagramu viz. obr. 3.21.



Obr. 3.21: Vývojový diagram detekce změny

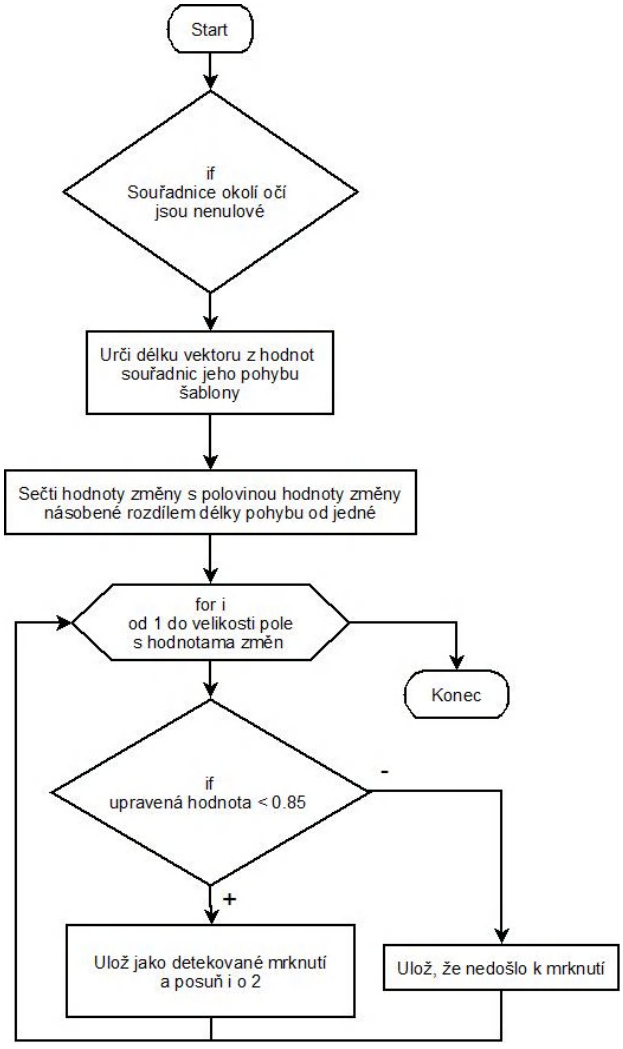


Obr. 3.22: Výstup podprogramu na detekci změny a jeho následné zpracování

3.5 Podprogram na vyhodnocení mrknutí z průběhu shody

Poslední částí programu na detekci mrkání je tento podprogram určený k vyhodnocení mrknutí z průběhu shody šablony s výřezem oka viz. obr. 3.22. Na obrázku se nachází výstup podprogramu pro určení změny v oblasti levého a pravého oka. Druhým průběhem je velikost pohybu levého a potom pravého oka. Po vynásobení průběhů změn levého a pravého oka se odečtou hodnoty velikosti vektoru pohybu také vzniklých vynásobením normovaných hodnot pro levé a pravé oko. Tento nový průběh se testuje na pokles pod 85-ti procentní hranici, a když tento stav nastane je detekován jako mrknutí.

Po vyhodnocení, že došlo k mrknutí se následující hodnoty pro další dva snímky automaticky ukládají jako nemrkání. Tím se ošetří chybná detekce vzniklá změnou ze zavřeného oka na otevřené.



Obr. 3.23: Vývojový diagram pro vyhodnocení mrknutí

Vyhodnocování mrknutí z průběhu pomocí testování poklesu pod určitou úroveň není vůbec robustní. Program je přizpůsoben k detekci rychlého a častého mrkání, které se v testovaném vzorku nachází. Pro detekci delších mrknutí by bylo třeba použít jednu šablonu k testování na více snímků. A místo detekce poklesu pod určitou úroveň by bylo lepší testovat tvar průběhu.

3.6 Podprogram na odstranění malých oblastí

Tento skript slouží k odstranění oblastí ze vstupní matice, které jsou menší než druhá vstupní hodnota funkce. Oblast je tvořena pixely, které se dotýkají aspoň v jednom z osmy směrů. Pro co nejrychlejší zpracování je skript plně vektorizován, tudíž nelze nakreslit jednoduchý vývojový diagram.

3.7 Časová náročnost programu

Účelem programu nebyla co nejrychlejší práce, to už bylo částečně dáno použitím prostředí MATLAB, u kterého načtení deseti snímků z videa trvá přibližně jednu sekundu na počítači s průměrnou konfigurací. Program byl testován na dvou počítačích s rozdílnou konfigurací, tedy i výkonem viz. tab. 3.1.

	Procesor	Paměť	Harddisk
Počítač 1	AMD Athlon 2500+	1 GB (333 MHz)	ULTRA ATA - 100
Počítač 2	Intel Dual Core 1.8GHz	2 GB (800 MHz)	SATA

Tab. 3.1: Přehled obou konfigurací použitých počítačů

Doba při zpracování videa o délce 600 snímků na počítači s první konfigurací trvala 2544 sekund z čehož vyplývá, že doba na zpracování jednoho snímku trvá přibližně 4 vteřiny. Na počítači s druhou konfigurací se doba zpracování 600 snímků snížila na 1320 sekund, což je přibližně 2 vteřiny na snímek, tzn. dvakrát rychlejší než u počítače s nižším výkonem.

Konkrétní čas spotřebovaný hlavním programem, než přejde k podprogramu detekce činí více jak jednu sekundu. Tento čas je z velké části způsoben načítáním úseku videa. Ostatní funkce v hlavním programu slouží jen k zobrazování výsledků a jejich popis k výkonu algoritmu není podstatný.

Část programu	Doba trvání [ms]	Doba trvání [%]
Hlavní program	1 000	2.56
Detekce	8 000	20.49
Najdi oči	22 000	56.35
Detekuj změnu	8 000	20.49
Vyhodnoť změnu	42	0.1
Celkem	39 042	100

Tab. 3.2: Přehled doby zpracování deseti snímků jednotlivými částmi programu

Časová náročnost podprogramu detekce tvoří polovinu celkové. Doba zpracování je zaprvé způsobena transformací dat z avi formátu do čtyřrozměrné matice, která je posléze převedena na třírozměrnou ve stupních šedi. Nejvíce časově náročnou části detekce je cyklus, kde probíhá zpracování snímků a ukládání jejich zpracovaných diferencí do akumulačního rámce. Cyklus tvoří polovinu časové spotřeby podprogramu detekce.

Čas spotřebovaný podprogramem určeným k hledání oka činí 2,2 sekundy. Takto velká hodnota je způsobena opakovaným použitím algoritmu na analýzu oblastí, u kterého je rychlost závislá na počtu a velikosti zpracovávaných oblastí, takže doba strávená hledáním očí je značně proměnlivá.

Podprogram na porovnávání výřezů oka pracuje ve smyčce, v které se prochází jednotlivými snímky, v tomto se nachází ještě další dva cykly pro posouvání šablony po výřezu. Uvnitř těchto cyklů je hlavní výkonná část na testování podobnosti. To znamená, že kubická náročnost algoritmu i s tak malou porcí dat zpracovávanými informacemi, jakými jsou matice o rozměru 40 pixelů znatelně prodlužuje dobu nutnou ke zpracování.

Vyšších jak kvadratických časových náročností je v programu několik a to se spojením s vektorizací a zpracováním velkého množství dat je důvodem k pomalé rychlosti programu.

Ostatní podprogramy jako ten na vyhodnocení mrkání z průběhu změny a skript na odstraňování malých oblastí jsou z časového hlediska vcelku zanedbatelné, protože první zpracovává jen krátké úseky jednorozměrných hodnot. U druhého se

čas potřebný na to, aby program odstranil malé oblasti pohybuje od 100 milisekund do 200 milisekund.

3.8 Účinnost programu

Dobrá identifikace mrknutí je závislá na dobrém nastavením parametrů, ale ani po dobrém nastavení parametrů stále není stoprocentní pravděpodobnost správné detekce.

Prvním místem důležitým pro správnou detekci je program na hledání očí. Je to také nejslabší článek celého programu. První možností jak může dojít k nesprávné detekci je, že oblast oka nebude vůbec v akumulačním rámci zastoupena potom dojde k vybrání jiné oblasti v horizontálním rozmezí. Dalším možným stavem, který může nastat je, že oblast oka bude spojena s další oblastí, což posune její těžiště mimo horizontální rozmezí, v kterém by měla být umístěna oblast oka. Pokud dojde k nesprávnému určení očí a označí se jiná oblast, která splní všechny podmínky program předá souřadnice této oblasti dál a podprogram na detekci změny potom hledá mrknutí tam kde se oko vůbec nenachází. Když nastane druhá možnost, že není identifikována žádná oblast podprogram detekce potom použije souřadnice z minulého úspěšného hledání oka. Obě tyto možnosti mohou nastat po sobě, to potom znamená, že program po neúspěšném hledání oblasti použije minulé souřadnice, které jsou vadné.

Podprogram určený k detekci změny zase byl nastaven na poměrně rychlé mrknutí, takže pokud došlo k pomalému mrknutí nedošlo k jeho detekci. Nastavení pevné hranice asi není nejlepším řešením, proto algoritmus není dostatečně robustní.

Vyhodnocování mrknutí nebylo zcela funkční při testování na videosekvenci s moderátorkou s tmavými očními stíny, protože při prahování výřezu okolí oka je jako kladná logická hodnota vyhodnoceno velké množství pixelů. Kvůli velkému počtu pixelů v šabloně i výřezu nedochází při mrknutí k výraznějšímu procentuálnímu skoku změny. A protože se podobnost šablony s výřezem pohybují nad devadesátí procenty, tak nejsou vyhodnocena mrknutí, protože nedojde k poklesu průběhu pod mnou definovanou hranici.

Při zkoušení přizpůsobeného algoritmu na první video byl algoritmus úspěšný při hledání na šedesáti segmentech ze stotřiceti, jinak byly použity souřadnice minulých detekcí. Při nastavení parametrů a testování na druhém videu byl algoritmus ještě

méně úspěšný, kvůli očním stínům byly oblasti pohybu očí velké a často se spojovaly s ostatními. V třetím videu se moderátor málo pohyboval a pohyb očí tak nebyl dostatečně zřetelný a špatně se detekovali.

Výsledky programu na vyhodnocení mrkání byly při správném určení okolí oka s osmdesáti procentní úspěšností.

4 Závěr

Kvůli realizaci programu jsem nastudoval různé techniky detekce lidí v obraze spolu s technikami na detekci mrkání a vybral ty, které budou nejvhodnější pro zvolené podmínky. Jako video vhodné na rozpoznávání mrknutí jsem zvolil televizní noviny, protože moderátoři se většinu doby co komentují dívají do kamery, což usnadní detekci a vyloučí nesprávnou detekci vlivem otočení hlavy. Dalším důvodem k použití televizních zpráv bylo jejich statické pozadí, na které jsou citlivé algoritmy založené na detekci pohybu. Scéna je také výborně osvětlena a kamera statická, tím odpadá úprava jasu nebo stabilizace obrazu.

Algoritmus na detekci mrkání jsem navrhl na základu metody detekce pohybu tak, aby byl co možná nejrobustnější. Toho jsem chtěl dosáhnout implementací parametrů, které se měli automaticky měnit na základě chování sledovaného člověka ve videu. Hlavním parametrem ovlivňujícím funkčnost programu je míra pohybu. Ovšem automatické nastavení parametrů se nepovedlo realizovat. Návrhy automatického nastavení parametrů nefungovaly, protože návrhy funkcí pro jejich výpočet neplatily v plném rozsahu. A testování parametrů při zpracování videa také nebylo možné implementovat, jelikož testování parametrů by se muselo dít ve smyčce, čím by se násobil čas potřebný ke zpracování počtem opakování, než by byl nalezen vhodná hodnota parametru. Takže parametry jsem zvolil ručně, pro každého moderátora.

Završením práce je několik videí, na kterých byl odzkoušen program. V každém videu byl jiný moderátor, což mělo otestovat robustnost algoritmu. Při nastavení parametrů ke každému videu, algoritmus s přibližně padesáti procentní pravděpodobností našel správně oči, pokud nedošlo ke správné lokaci mohly nastav dvě možnosti první byla, že nebyla žádná oblast vyhodnocena jako oko anebo byla vybrána špatná. První případ nastal ve čtyřiceti procentech detekce, ale s tímto se vypořádal program tak, že dosadil staré souřadnice oka. Když byla správně určena oblast, ve které se nacházejí oči, tak program detekoval mrknutí s osmdesáti procentní šancí, pokud tedy nebudeme brát v úvahu, že program detekoval vícenásobně jedno mrknutí. Ovšem takto vysoká úroveň správné detekce byla dosažena jen u prvního videa. U dalšího videa, v kterém se nacházela žena nebyla správně detekována skoro žádná mrknutí, to bylo způsobeno tmavými očními stíny, které zhoršovali podmínky správné detekce.

Problém spojený s detekcí mrknutí použitím šablony by mohl být kompletně vyřešen použitím dobrého algoritmu na lokalizaci duhovky. Ale tento algoritmus potřebuje k práci video s vysokým rozlišením. Použití videa s velkým datovým tokem by pro náš program bylo velice časově náročné. Použití detekce duhovky by zase v některých aspektech usnadnilo detekci, protože duhovka má velice specifický tvar a ještě se nachází dvakrát v jednom obraze.

Při porovnání rychlosti a frekvence mrkání zpravodajů na vzorku zpravodajství a hodnotou zjištěnou z literatury, jsem došel k závěru, že zpravodajové nejsou v normálním prostředí a jejich mrkaní je tím značně ovlivněno. Vyjádřeno čísly délka mrknutí moderátora činí průměrně 5 snímků nanejvýš 7 snímků, což je 125 a 175ms, tato hodnota se významně liší od hodnoty 300 milisekund z literatury. Tento rozdíl může být způsoben hlavně tím, že moderátoři se musí velice soustředit na rychlé mluvení a čtení textu z napovídacích zařízení.

5 Použitá literatura

[1] OGALE, N. *A survey of techniques for human detection from video*. Maryland: Department of Computer Science University of Maryland, 2006.

[2] Fleck, M., Forsyth D. *Naked People Skin Filter*. Seoul: School of Computer Engineering, Sejong University, 1996.

[3] Kapur, J. *Face Detection in Color Images*. Washington: University of Washington Department of Electrical Engineering, 1996.

[4] Beleznai, C., Fruhstuck B., Bischof H. *Human detection in groups using a fast mean shift procedure*. International Conference on Image Processing, 2004.

[5] Morris, T., Blenkhorn P., Zaidi F. *Blink detection for real-time eye tracking*. Manchester: Department of Computation UMIST, 2002.

[6] Comaniciu, D., Meer P. *Mean shift filtering*. [cit. 2008-28-04]. Dostupné z WWW: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/TUZEL1/MeanShift.pdf

[7] Guerrero S. *Model-Based Eye Detection and Animation*. Linköpings: Department of Electrical Engineering Universitat Linköpings, 2006.

[8] Lipinsky B. *Iris Recognition: Detecting the Iris*. [cit. 2008-29-04]. Dostupné z WWW: http://cnx.org/content/m12489/latest/

[9] Turk M., *Interactive-Time Vision: Face Recognition as a Visual Behaviour*, PhD Thesis, MIT, 1991.

[10] Yoon S., Kim H. *Real time multiple people detection using skin color, motion and apearance information*. South Korea: Human Comput. Interaction Lab., Samsung Adv. Inst. of Technology, 2004.

[11] Wikipedia. *Mahalanobis distance*. [cit. 2008-01-05]. Dostupné z WWW: http://en.wikipedia.org/wiki/Mahalanobis_distance

[12] Dalal N., Triggs B. *Histograms of oriented gradients for human detection.* Francie: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005.

[13] Grauman K., Betke M., Gips J., Bradski R. G. *Communication via Eye Blinks – Detecion and Duration Analysis in Real Time*. Boston: Vision Interface Group, MIT AI Lab, 2000.

[14] Wikipedia. *Blink*. [cit. 2008-10-05]. Dostupné z WWW: http://en.wikipedia.org/wiki/Blinkin

[15] Uzunova V. Master thesis: *An Eyelids nad Eye Corners Detection and Tracking Method for Rapid Iris Tracking*. Magdeburg: Otto von Guericke University of Magdeburg, 2005.

PROHLÁŠENÍ:

Prohlašuji, že svou diplomovou práci na téma „Detekce Mrkání a Rozpoznávání podle mrkání očí“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.“

V Brně dne …… ……
(podpis autora)

Přílohy

Podprogram Spust

```
% Viktor Tesárek, diplomová práce na téma Detekce mrkání
% a rozpoznávání podle mrkání očí.
% Vysoké učení technické v Brně
% Fakulta Elektrotechniky a komunikačních technologií
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
% Funkce Spust.m slouží jako spouštěcí skript celého programu.
% Je třeba definovat jméno vstupního videosouboru filename
% a výstupního příkazem avifile.

% aviobj = close(aviobj);
% Po nevhodném ukončení funkce je třeba zavřít výstupní soubor.
%clear all;

tic;
filename = '1390.avi';
info = aviinfo(filename);

ZprRamcu = 10;
aviobj = avifile ('vystup.avi', 'fps', 5);
ZOkraj(1:8) = 0;
CLNej = 0;
CPNej = 0;
CLPosuvX = 0;
CLPosuvY = 0;
CPPosuvX = 0;
CPPosuvY = 0;
CMrk = [0];
   scrsz = get(0,'ScreenSize');
   fig = figure('Position',[100 100 1052 748]);
%    fig =1;
for(i = 1:floor(info.NumFrames/ZprRamcu))
   if(i == 2)
      a =1;
   end
   if(i == 1)
      avi = aviread(filename, 1:i*ZprRamcu);
      video = {avi.cdata};
      [LNej, PNej, LPosuvX, LPosuvY, PPosuvX, PPosuvY, pixel, pixels, bw, bwSum, Okraj] =
Detekce(info, video, ZOkraj);
   else
      avi = aviread(filename, (i-1)*ZprRamcu:i*ZprRamcu);
      video = {avi.cdata};
```

```
    [LNej, PNej, LPosuvX, LPosuvY, PPosuvX, PPosuvY, pixel, pixels, bw, bwSum, Okraj] =
Detekce(info, video, ZOkraj);
  end
  [Mrk] = RozpoznejMrk(LNej(2:end), PNej(2:end), LPosuvX(2:end), LPosuvY(2:end),
PPosuvX(2:end), PPosuvY(2:end));
  CMrk = [CMrk Mrk];
  ZOkraj = Okraj;
  CLNej = [CLNej LNej(2:end)];
  CPNej = [CPNej PNej(2:end)];
  CLPosuvX = [CLPosuvX LPosuvX(2:end)];
  CLPosuvY = [CLPosuvY LPosuvY(2:end)];
  CPPosuvX = [CPPosuvX PPosuvX(2:end)];
  CPPosuvY = [CPPosuvY PPosuvY(2:end)];

  for(r = 1:ZprRamcu)
    disp([num2str(i) ', ' num2str(r) ', procent: ' num2str(100*i/floor(info.NumFrames/ZprRamcu))]);
    figure(fig);
    subplot(5,4,[1:12]);
    imshow(pixels(:,:,:,r))
    if((Okraj(2)-Okraj(1) > 0) && (Okraj(4)-Okraj(3) > 0))
      rectangle('Position',[Okraj(1) Okraj(3) (Okraj(2)-Okraj(1)) (Okraj(4)-Okraj(3))],'EdgeColor','r');
    end
    hold on;
    if((Okraj(6)-Okraj(5) > 0) && (Okraj(8)-Okraj(7) > 0))
      rectangle('Position',[Okraj(5) Okraj(7) (Okraj(6)-Okraj(5)) (Okraj(8)-Okraj(7))],'EdgeColor','r');
    end
    hold off;
    title(['Vstupující video: ' num2str(i) '. skupina, ' num2str(r) '. snímek' ]);
    subplot(5,4,13);
    if(sum(Okraj(1:4)) ~= 0)
      imshow(pixel(Okraj(3):Okraj(4),Okraj(1):Okraj(2),r));
    end
    title('Levé oko');
    subplot(5,4,14);
    if(sum(Okraj(5:8)) ~= 0)
      imshow(pixel(Okraj(7):Okraj(8),Okraj(5):Okraj(6),r));
    end
    title('Pravé oko');
    subplot(5,4,15);
    plot([0 -LPosuvX(r)], [0 LPosuvY(r)], 'bx');
    axis([-3 3 -3 3]);
    title('Pohyb l. oka');
    xlabel('{\itX}(pixel) \rightarrow');
    ylabel('{\itY}(pixel) \rightarrow');
    subplot(5,4,16);
    plot([0 -PPosuvX(r)], [0 PPosuvY(r)], 'bx');
    axis([-3 3 -3 3]);
```

```
        title('Pohyb p. oka');
        xlabel('{\itX}(pixel) \rightarrow');
        ylabel('{\itY}(pixel) \rightarrow');
        subplot(5,4,[17 18]);
        plot(1-r-((i-1)*ZprRamcu):size(CLNej,2)-r-((i-1)*ZprRamcu),CLNej,'r');
        hold on;
        plot(1-r-((i-1)*ZprRamcu):size(CPNej,2)-r-((i-1)*ZprRamcu),CPNej,'g');
        hold off;
        axis([-20 20 0.5 1]);
        title('Změna v oblasti oka');
        xlabel('{\itt}(snímek) \rightarrow');
        ylabel('{\itPodobnost}(-) \rightarrow');
        legend(['Levé oko'], ['Pravé oko']);
        subplot(5,4,[19 20]);
        stem(1-r-((i-1)*ZprRamcu):size(CPNej,2)-r-((i-1)*ZprRamcu),CMrk,'b');
        axis([-20 20 0 1]);
        title('Detekovaná mrknutí');
        xlabel('{\itt}(snímek) \rightarrow');
        frame = getframe(gcf);
        aviobj = addframe(aviobj, frame);
    end
end
[r,s,v] = find(CMrk);
dlmwrite('SnimkyMrknuti.txt', s);

aviobj = close(aviobj);
disp(toc);
%clear all;
%%%%%%%%%%%%%%%%%Konec skriptu Spust.m%%%%%%%%%%%%%%%%%
```

Podprogram Detekce

```
% Viktor Tesárek, diplomová práce na téma Detekce mrkání
% a rozpoznávání podle mrkání očí.
% Vysoké učení technické v Brně
% Fakulta Elektrotechniky a komunikačních technologií
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
function [LNej, PNej, LPosuvX, LPosuvY, PPosuvX, PPosuvY, pixel, pixels, bw, bwSum, Okraj] =
Detekce(info, video, ZOkraj)

% Funkce Detekce.m slouží ke zpracování formátu dat a tvorbě akumulovaného rámce.
% Výstupy LNej, PNej, LPosuvX, LPosuvY, PPosuvX, PPosuvY reprezentují výsledky podprogramu
% DetekceZmeny. Výstupy pixel, pixels jsou přeformátovaná data vstupního video souboru.
% bwSum je matice akumulovaného rámce. Okraj je vektor souřadnic obdélníku pravého i levého oka.
% Vstupnímy argumenty jsou info to je struktura představující parametry avi formátu.
% video je vícerozměrná matice barevného vstupního videa. ZOkraj jsou souřadnice oblasti očí z
% minulého hledání.
```

```
if ischar(video)
   avi = aviread(video);
   pixels = double(cat(4,avi(1:end).cdata))/255;
   clear avi
else
   pixels = double(cat(4,video{1:end}))/255;
   clear video
end


%PARAMETRY
   IntstChng = 4/255;   %hranice zmeny jasu pro prevod do binarniho obrazu
   MinVlkstOblasti = 80;      %velikost oblasti, ktere uz nebudou vycleneny

radku=info.Height;
sloupcu=info.Width;
ramcu=size(pixels,4);

bwSum = zeros(radku, sloupcu);
pixel = zeros(radku, sloupcu, ramcu);
d = zeros(radku, sloupcu, ramcu);
bw = zeros(radku, sloupcu, ramcu);

for j = 1:ramcu
   pixel(:,:,j) = (rgb2gray(pixels(:,:,:,j)));
end

for k = 2:ramcu
   d(:,:,k)=(abs(pixel(:,:,k)-pixel(:,:,k-1)));

   bw(:,:,k) = im2bw(d(:,:,k), IntstChng);
   [bw(:,:,k)] = OdstranMaleOblasti(bwlabel(bw(:,:,k)), MinVlkstOblasti);

   bw1=bwlabel(bw(:,:,k));
   oblasti2 = max(max(bw1));
%    oblast2 = sum(histc(bw1(:,:),1:oblasti2),2);
   for (ob = 1:oblasti2)
      [radky,sloupce,v] = find(bw1 == ob);
%       RakdyOblasti(1:size(radky),ob) = radky;
      SloupceOblasti(1:size(sloupce),ob) = sloupce;
      PrumRakdyOblasti(ob) = round(mean(radky));
      PrumSloupceOblasti(ob) = round(mean(sloupce));
%       if(PrumRakdyOblasti(ob) < 0.15*radku || PrumRakdyOblasti(ob) > 0.85*radku)
%          bw1( ismember(bw1(1:radku,1:sloupcu), ob)) = 0;
%       end
```

```
   end
%    PrumRadek = round(mean(PrumRakdyOblasti));
%    PrumSloupec = round(mean(PrumSloupceOblasti));

   for (ob = 1:oblasti2)
      [radek,sloupec,v] = find(SloupceOblasti(size(SloupceOblasti,1),:));
      if(((1.05*PrumSloupceOblasti(ob) < min(SloupceOblasti(:,round(mean(sloupec))))) ||
(0.95*PrumSloupceOblasti(ob) > max(SloupceOblasti(:,round(mean(sloupec))))))
         bw1( ismember(bw1(1:radku,1:sloupcu), ob)) = 0;
      end
   end

   clear oblasti2 oblast2 ob radky sloupce RakdyOblasti SloupceOblasti PrumRakdyOblasti
PrumSloupceOblasti PrumRadek PrumSloupec;

   bw(:,:,k) = im2bw(bw1,0.5);

   if(k == 2)
      bwSum(:,:) = zeros(radku, sloupcu);
   else
      bwSum(:,:) = bwSum(:,:) + bw(:,:,k)/ramcu;
   end
end
%       figure(3);
%       imshow(bwSum)

if(sum(sum(bwSum)) > 2000  || sum(ZOkraj) == 0)
   [Okraj] = NajdiOci(bwSum, ramcu);
   if((sum(Okraj - ZOkraj) > 160 || sum(Okraj) == 0) && sum(ZOkraj) ~= 0)
      Okraj = ZOkraj;
   end
   [LNej, PNej, LPosuvX, LPosuvY, PPosuvX, PPosuvY] = DetekceZmeny(pixel, 0.4, Okraj);
else
   Okraj = ZOkraj;
   [LNej, PNej, LPosuvX, LPosuvY, PPosuvX, PPosuvY] = DetekceZmeny(pixel, 0.4, Okraj);
%    LPosuvX(1:end) = 0;
%    LPosuvY(1:end) = 0;
%    PPosuvX(1:end) = 0;
%    PPosuvY(1:end) = 0;
end

% for l = 1:ramcu
%    figure(22);
%    subplot(4,2,1)
%    imshow(bw(HOkraj:DOkraj,LOkraj:POkraj,l));
%    subplot(4,2,2)
%    imshow(bw(HOkraj:DOkraj,PLOkraj:PPOkraj,l));
```

```
%     subplot(4,2,3)
%     imshow(pixel(HOkraj:DOkraj,LOkraj:POkraj,l));
%     subplot(4,2,4)
%     imshow(pixel(HOkraj:DOkraj,PLOkraj:PPOkraj,l));
%     subplot(4,2,5)
%     imshow(im2bw(pixel(HOkraj:DOkraj,LOkraj:POkraj,l),0.33));
%     subplot(4,2,6)
%     imshow(im2bw(pixel(HOkraj:DOkraj,PLOkraj:PPOkraj,l),0.33));
%     subplot(4,2,7)
%     imshow(im2bw(pixel(HOkraj:DOkraj,LOkraj:POkraj,l),0.33));
%     subplot(4,2,8)
%     imshow(im2bw(pixel(HOkraj:DOkraj,PLOkraj:PPOkraj,l),0.33));
% end

%%%%%%%%%%%%%%%Konec skriptu Detekce.m%%%%%%%%%%%%%
```

Podprogram NajdiOci

```
% Viktor Tesárek, diplomová práce na téma Detekce mrkání
% a rozpoznávání podle mrkání očí.
% Vysoké učení technické v Brně
% Fakulta Elektrotechniky a komunikačních technologií
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
function [Okraj] = NajdiOci(bwSum, ZprRamcu)

% Funkce NajdiOci slouží k určení souřadnic obdélníků obkreslující oční oblasti.
% Výstupem funkce je vektor souřadnic obou obdélníků.
% bwSum je matice představující akumulační rámec
% ZprRamcu udává velikost segmentu zpracovávaného videa ve snímcích.
DBG = 0;
%
%    Hrn = max(max(bwSum))/(1/3 * ZprRamcu);
Hrn = sum(sum(bwSum)/6000) * 1/4;
bw = im2bw(bwSum, Hrn);
radku=size(bw,1);

% BwObl = bwlabel(bw);
%
% oblasti = max(max(BwObl));
%
% for (ob = 1:oblasti)
%     [radky,sloupce,v] = find(BwObl == ob);
%     RakdyOblasti(1:size(radky),ob) = radky;
%     SloupceOblasti(1:size(sloupce),ob) = sloupce;
%     PrumRakdyOblasti(ob) = round(mean(radky));
%     PrumSloupceOblasti(ob) = round(mean(sloupce));
% end
```

```
% PrumSloupec = round(mean(PrumSloupceOblasti));

if(DBG == 1)
   figure(8);
   imshow(bw)
end
[bw] = OdstranMaleOblasti(bw, 20);
if(DBG == 1)
   figure(88);
   imshow(bw)
end

%  se = strel('disk',1);
%  bw = imerode(bw,se);
%  disp(sum(sum(bwSum)));
% if(sum(sum(bwSum)) < )
%
% else
%
% end

se = strel('disk',2);
bw = imdilate(bw,se);
if(DBG == 1)
   figure(888);
   imshow(bw)
end
[bw] = OdstranMaleOblasti(bw, 200);
bwF = bw;
if(DBG == 1)
   figure(9);
   imshow(bw)
end
% [radek,sloupec,v] = find(SloupceOblasti(size(SloupceOblasti,1),:));

% clear PrumSloupec PrumRadek PrumSloupceOblasti PrumRakdyOblasti RakdyOblasti radky
sloupce v oblasti oblast L P LOkraj POkraj HOkraj DOkraj PPOkraj PLOkraj; %nekde je tu chyba asi
kdyz clearnu sloupceoblasti
BwObl = bwlabel(bw);
oblasti = max(max(BwObl));
oblast = sum(histc(BwObl(:,:),1:oblasti),2);
for (ob = 1:oblasti)
   [radky,sloupce,v] = find(BwObl == ob);
   RakdyOblasti(1:size(radky),ob) = radky;
   SloupceOblasti(1:size(sloupce),ob) = sloupce;
   PrumRakdyOblasti(ob) = round(mean(radky));
   PrumSloupceOblasti(ob) = round(mean(sloupce));
```

```
end
PrumSloupec = round(mean(PrumSloupceOblasti));
L = SloupceOblasti(1,1);
P = max(max(SloupceOblasti(:,:)));
if(DBG == 1)
   hold on;
   rectangle('Position',[2/12*(P-L)+L 0 3/12*(P-L) size(bwSum,1)],'EdgeColor','r');
   rectangle('Position',[7/12*(P-L)+L 0 3/12*(P-L) size(bwSum,1)],'EdgeColor','r');
   hold off;
end

StredTvar = PrumSloupec;
RadekOci = 0;
for (ob = 1:oblasti)
   LOkraj = SloupceOblasti(1,ob);
   POkraj = max(SloupceOblasti(:,ob));
   [r,s,v] = find(RakdyOblasti(:,ob));
   HOkraj = min(RakdyOblasti(1:max(r),ob));
   DOkraj = max(RakdyOblasti(:,ob));
   PPOkraj = StredTvar + abs(StredTvar - LOkraj);
   PLOkraj = StredTvar + abs(StredTvar - POkraj);
   if((2/12*(P-L) < PrumSloupceOblasti(ob)-L) && (abs(PrumSloupceOblasti(ob)-L) < 5/12*(P-L)) &&
(((POkraj - LOkraj)/(DOkraj - HOkraj)) < 2 && ((POkraj - LOkraj)/(DOkraj - HOkraj)) > 0.75))
      for(ob2 = ob:oblasti)
        if((7/12*(P-L) < PrumSloupceOblasti(ob2)-L) && (abs(PrumSloupceOblasti(ob2)-L) < 10/12*(P-L))
&& (((POkraj - LOkraj)/(DOkraj - HOkraj)) < 2 && ((POkraj - LOkraj)/(DOkraj - HOkraj)) > 0.75) &&
(abs(PrumRakdyOblasti(ob) - PrumRakdyOblasti(ob2)) < 15))
           StredTvar = round((PrumSloupceOblasti(ob) + PrumSloupceOblasti(ob2))/2);
           RadekOci = PrumRakdyOblasti(ob);
        end
      end
   end
end

bw(:,round(StredTvar))=0;

      figure(111);
      imshow(bw);
BwObl = bwlabel(bw);

clear r s ob2 ob PrumSloupec PrumRadek PrumSloupceOblasti PrumRakdyOblasti RakdyOblasti
radky sloupce v oblasti oblast L P LOkraj POkraj HOkraj DOkraj PPOkraj PLOkraj;

oblasti = max(max(BwObl));

for (ob = 1:oblasti)
   [radky,sloupce,v] = find(BwObl == ob);
```

```
   SloupceOblasti(1:size(sloupce),ob) = sloupce;
end

delka = round(abs(min(SloupceOblasti(:,1))-max(max(SloupceOblasti(:,:))))/2);
if((StredTvar-delka > 0) && (StredTvar+delka < max(sloupce)))
   OsSoumJdnStr = BwObl(:,StredTvar:-1:StredTvar-delka).*BwObl(:,StredTvar:1:StredTvar+delka);
   OsSoumDvoStr = zeros(radku,2*delka);
   OsSoumDvoStr(:,1:delka) = OsSoumJdnStr(:,delka:-1:1);
   OsSoumDvoStr(:,delka:2*delka) = OsSoumJdnStr(:,:);
   if(DBG == 1)
      figure(10);
      imshow(bw);
   end
   bw(:,StredTvar-delka+1:StredTvar+delka) = bw(:,StredTvar-delka+1:StredTvar+delka) .*
OsSoumDvoStr(:,:);
   bw(:,1:StredTvar-delka-1) = 0;
   bw(:,StredTvar+delka+1:end) = 0;
    if(DBG == 1)
      figure(11);
      imshow(bw);
    end
end

clear SloupceOblasti OsSoumJdnStr OsSoumDvoStr sloupce radky v ob delka PrumSloupceOblasti
PrumRakdyOblasti RakdyOblasti oblasti oblast;

se = strel('disk',2);
bw = imdilate(bw,se);
[bw] = OdstranMaleOblasti(bw, 200);
BwObl = bwlabel(bw);
RGB = label2rgb(BwObl);
if(DBG == 1)
   figure(12);
   imshow(RGB)
end
oblasti = max(max(BwObl));
oblast = sum(histc(BwObl(:,:),1:oblasti),2);
for (ob = 1:oblasti)
   [radky,sloupce,v] = find(BwObl == ob);
   RakdyOblasti(1:size(radky),ob) = radky;
   SloupceOblasti(1:size(sloupce),ob) = sloupce;
   PrumRakdyOblasti(ob) = round(mean(radky));
   PrumSloupceOblasti(ob) = round(mean(sloupce));
end

clear v radky oblast ob sloupce;
```

```
%    [radek,sloupec,v] = find(SloupceOblasti(size(SloupceOblasti,1),:));
L = SloupceOblasti(1,1);
P = max(max(SloupceOblasti(:,:)));

if(DBG == 1)
   hold on;
   rectangle('Position',[2/12*(P-L)+L 0 3/12*(P-L) size(bwSum,1)],'EdgeColor','r');
   hold off;
end
for (ob = 1:oblasti)
   LOkraj = SloupceOblasti(1,ob);
   POkraj = max(SloupceOblasti(:,ob));
   [r,s,v] = find(RakdyOblasti(:,ob));
   HOkraj = min(RakdyOblasti(1:max(r),ob));
   DOkraj = max(RakdyOblasti(:,ob));
   PPOkraj = StredTvar + abs(StredTvar - LOkraj);
   PLOkraj = StredTvar + abs(StredTvar - POkraj);
   if((2/12*(P-L) < PrumSloupceOblasti(ob)-L) && (abs(PrumSloupceOblasti(ob)-L) < 5/12*(P-L)) &&
(((POkraj - LOkraj)/(DOkraj - HOkraj)) < 2 && ((POkraj - LOkraj)/(DOkraj - HOkraj)) > 0.75))
      if((abs(RadekOci - PrumRakdyOblasti(ob)) < 10) || (RadekOci == 0))
         break;
      end
   end
end

if(DBG == 1)
   figure(13);
   imshow(bwF);
   figure(14);
   imshow(bw);
end
BPozLOka = zeros(size(bwSum,1),size(bwSum,2));
BPozLOka(HOkraj:DOkraj,LOkraj:POkraj) = 1;
BPozPOka = zeros(size(bwSum,1),size(bwSum,2));
BPozPOka(HOkraj:DOkraj,PLOkraj:PPOkraj) = 1;
OkoliLOka = (bw .* bwF) .* BPozLOka;
OkoliPOka = (bw .* bwF) .* BPozPOka;
if(DBG == 1)
   figure(15);
   imshow(OkoliLOka);
   figure(16);
   imshow(OkoliPOka);
end
[r,s,v] = find(bwF .* OkoliLOka);
LOko = bwselect(bwF,s,r,4);
if(DBG == 1)
   figure(17);
```

```
   imshow(LOko);
end
if(size(s,1) == 0)
   Okraj(1) = 0;
   Okraj(2) = 0;
   Okraj(3) = 0;
   Okraj(4) = 0;
   Okraj(5) = 0;
   Okraj(6) = 0;
   Okraj(7) = 0;
   Okraj(8) = 0;
else
   Okraj(1) = min(s);
   Okraj(2) = max(s);
   Okraj(3) = min(r);
   Okraj(4) = max(r);
   [r,s,v] = find(bwF .* OkoliPOka);
   POko = bwselect(bwF,s,r,4);
end

%    if(DBG == 1)
%       figure(18);
%       imshow(POko);
%    end

if(size(s,1) == 0)
   Okraj(1) = 0;
   Okraj(2) = 0;
   Okraj(3) = 0;
   Okraj(4) = 0;
   Okraj(5) = 0;
   Okraj(6) = 0;
   Okraj(7) = 0;
   Okraj(8) = 0;
else
   Okraj(5) = min(s);
   Okraj(6) = max(s);
   Okraj(7) = min(r);
   Okraj(8) = max(r);
end

if(DBG == 1)
   figure(19);
   imshow(bwSum);
   if(Okraj(1) ~= 0 && Okraj(2) ~= 0 && Okraj(3) ~= 0 && Okraj(4) ~= 0 && Okraj(5) ~= 0 && Okraj(6) ~=
0 && Okraj(7) ~= 0 && Okraj(8) ~= 0)
      hold on;
```

```
    rectangle('Position',[Okraj(1) Okraj(3) (Okraj(2)-Okraj(1)) (Okraj(4)-Okraj(3))],'EdgeColor','r');
    rectangle('Position',[Okraj(5) Okraj(7) (Okraj(6)-Okraj(5)) (Okraj(8)-Okraj(7))],'EdgeColor','r');
    hold off;
  end
end

  clear rs ss vs r s v ob bwF BPozLOka BPozPOka BwObl DOkraj HOkraj PLOkraj POko POkraj
PPOkraj RGB bw radku se L LOko LOkraj RakdyOblasti OkoliLOka OkoliPOka P RadekOci
PrumRakdyOblasti PrumSloupceOblasti StredTvar SloupceOblasti PStredOka LStredOka oblasti;
%%%%%%%%%%%%Konec skriptu NajdiOci.m%%%%%%%%%%%%%
```

Podprogram DetekceZmeny

```
% Viktor Tesárek, diplomová práce na téma Detekce mrkání
% a rozpoznávání podle mrkání očí.
% Vysoké učení technické v Brně
% Fakulta Elektrotechniky a komunikačních technologií
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
function [LNej, PNej, LPosuvX, LPosuvY, PPosuvX, PPosuvY] = DetekceZmeny(pixel, thr, Okraj)

% Funkce DetekceZmeny slouží k porovnávání dvou výřezů definovaných souřadnicema.
% Výstupy LNej, PNej, LPosuvX, LPosuvY, PPosuvX, PPosuvY reprezentují jsou výsledky této
funkce.
% V pořadí představují míru podobnosti levého a pravého oka a jejich pohyb v obou osách.
% Vstup je tvořen černobílými snímky, prahem a souřadnicemi oblastí očí.

DBG = 0;
%
%thr = 0.4;

ramcu = size(pixel,3);
okno = 0;
LVyska = Okraj(4) - Okraj(3);
PVyska = Okraj(8) - Okraj(7);
LDelka = Okraj(2) - Okraj(1);
PDelka = Okraj(6) - Okraj(5);

LStrX = round((Okraj(2) - Okraj(1))/2);
LStrY = round((Okraj(4) - Okraj(3))/2);
PStrX = round((Okraj(6) - Okraj(5))/2);
PStrY = round((Okraj(8) - Okraj(7))/2);

% mrk = zeros(1,ramcu);
LNej = zeros(1,ramcu);
PNej = zeros(1,ramcu);

for l = (1:ramcu)
```

```
%     LOko = pixel(HOkraj-okno:DOkraj+okno,LOkraj-okno:POkraj+okno,l);
%     POko = pixel(HOkraj-okno:DOkraj+okno,PLOkraj-okno:PPOkraj+okno,l);
%     figure(22);
%     subplot(4,2,1)
%     imshow(LOko);
%     subplot(4,2,2)
%     imshow(POko);
%     subplot(4,2,3)
%     imshow(im2bw(LOko,thr));
%     subplot(4,2,4)
%     imshow(im2bw(POko,thr));

%     LOko(1,:) = 0;
%     LOko(vyska+1,:) = 0;
%     LOko(:,1) = 0;
%     LOko(:,delka+1) = 0;
%     POko(1,:) = 0;
%     POko(vyska+1,:) = 0;
%     POko(:,1) = 0;
%     POko(:,delka+1) = 0;
   if(sum(Okraj) ~= 0)
      if(l > 1)
         for(x = 0:LDelka)
            for(y = 0:LVyska)
               %              subplot(4,2,7)
               A = im2bw(pixel(Okraj(3)-LStrY+y:Okraj(4)-LStrY+y,Okraj(1)-LStrX+x:Okraj(2)-LStrX+x,l-
1),thr);
               %              imshow(A);
               %              subplot(4,2,8)
               B = im2bw(pixel(Okraj(3):Okraj(4),Okraj(1):Okraj(2),l),thr);
               %              imshow(B);
               Kontr = sum(sum(A == B));
               %              Proc = (vyska+1)*(delka+1);
               %              PProc = 0.90*Proc;
               if(LNej(l) < Kontr)
                  LPosuvX(l) = x-LStrX;
                  LPosuvY(l) = y-LStrY;
                  LNej(l) = Kontr;
               end
               %              if(Kontr > PProc)
               %                 mrk(l) = mrk(l) + 1;
               %                  disp([num2str(l) ': ' num2str(Kontr) ', x:' num2str(PosuvX(l)) ': '
num2str(PosuvY(l))]);
               %                  subplot(4,2,7)
               %                  imshow(A);
               %                  hhh = 1;
               %              end
```

```
            end
          end
        end

        if(l > 1)
          for(x = 0:PDelka)
            for(y = 0:PVyska)
              %             subplot(4,2,7)
              A = im2bw(pixel(Okraj(7)-PStrY+y:Okraj(8)-PStrY+y,Okraj(5)-PStrX+x:Okraj(6)-PStrX+x,l-
1),thr);
              %             imshow(A);
              %             subplot(4,2,8)
              B = im2bw(pixel(Okraj(7):Okraj(8),Okraj(5):Okraj(6),l),thr);
              %             imshow(B);
              Kontr = sum(sum(A == B));
              %             Proc = (vyska+1)*(delka+1);
              %             PProc = 0.90*Proc;
              if(PNej(l) < Kontr)
                PPosuvX(l) = x-PStrX;
                PPosuvY(l) = y-PStrY;
                PNej(l) = Kontr;
              end
              %             if(Kontr > PProc)
              %               mrk(l) = mrk(l) + 1;
              %                disp([num2str(l) ': ' num2str(Kontr) ', x:' num2str(PosuvX(l)) ': '
num2str(PosuvY(l))]);
              %                subplot(4,2,7)
              %                imshow(A);
              %                hhh = 1;
              %             end
            end
          end
        end
      else
        LNej(l) = 0;
        PNej(l) = 0;
        LPosuvX(l) = 0;
        LPosuvY(l) = 0;
        PPosuvX(l) = 0;
        PPosuvY(l) = 0;
      end
%     subplot(4,2,5)
%     imshow(im2bw(LOko,thr));
%     subplot(4,2,6)
%     imshow(im2bw(POko,thr));
%
%     subplot(4,2,7)
```

```
%    imshow(pixel(HOkraj-StrY+y:DOkraj-StrY+y,LOkraj-StrX+x:POkraj-StrX+x,l-1));
%    subplot(4,2,8)
%    imshow(pixel(HOkraj:DOkraj,LOkraj:POkraj,l));
end
if(sum(Okraj) ~= 0)
   LNej = LNej ./ ((LVyska+1) * (LDelka+1));
   PNej = PNej ./ ((PVyska+1) * (PDelka+1));
end
if(DBG == 1)
   figure(22);
   for (l = 1:ramcu)
      subplot(4,2,1)
      imshow(pixel(Okraj(3):Okraj(4),Okraj(1):Okraj(2),l));
      subplot(4,2,2)
      imshow(pixel(Okraj(7):Okraj(8),Okraj(5):Okraj(6),l));
      subplot(4,2,3)
      imshow(im2bw(pixel(Okraj(3):Okraj(4),Okraj(1):Okraj(2),l),thr));
      subplot(4,2,4)
      imshow(im2bw(pixel(Okraj(7):Okraj(8),Okraj(5):Okraj(6),l),thr));
      if(l > 1)
         subplot(4,2,5)
         plot(LNej(2:l))
         hold on;
         plot(0,0);
         plot(PNej(2:l))
         hold off;
      end
   end
end

%    G = not(im2bw(LOko,thr));
%    se = strel('disk',1);
%    G = imopen(G,se);
%    G = imclose(G,se);
%    imshow(G);

%    GG = not(im2bw(POko,thr));
%    se = strel('disk',1);
%    GG = imopen(GG,se);
%    GG = imclose(GG,se);
%    imshow(GG);

%    H = fspecial('disk',10);
%    blurred = imfilter(pixel(HOkraj-okno:DOkraj+okno,PLOkraj-okno:PPOkraj+okno,l),H,'replicate');
%    imshow(blurred);
%    imshow(im2bw(blurred,0.3));
%    disp(sum(sum(pixel(HOkraj-okno:DOkraj+okno,LOkraj-okno:POkraj+okno,l))));
```

```
%    disp(sum(sum(pixel(HOkraj:DOkraj,PLOkraj:PPOkraj,l))));

%%%%%%%%%%%%%%Konec skriptu DetekceZmeny.m%%%%%%%%%%%%%%
```

Podprogram RozpoznejMrk

```
% Viktor Tesárek, diplomová práce na téma Detekce mrkání
% a rozpoznávání podle mrkání očí.
% Vysoké učení technické v Brně
% Fakulta Elektrotechniky a komunikačních technologií
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
function [Mrk] = RozpoznejMrk(LNej, PNej, LPosuvX, LPosuvY, PPosuvX, PPosuvY);

% Funkce slouží na vyhodnocení mrkání z průběhu podobnosti výřezu očí.
% Výstupní binární vektor Mrk označuje rámce jedničkou, když bylo detekováno mrknutí.
% Vstupy LNej, PNej, LPosuvX, LPosuvY, PPosuvX, PPosuvY reprezentují hodnoty podobnosti a
pohybu šablony.

Kde = 0.85;
% ramcu = size(LNej,2);
if(sum(LNej(1:end)) ~= 0)
   pohyb = sqrt(LPosuvX.^2 + LPosuvY.^2) .* sqrt(PPosuvX.^2 + PPosuvY.^2);
   Mrk = ((PNej(1:end).*LNej(1:end)) + 0.5*((1 - (PNej(1:end).*LNej(1:end))) .*
(pohyb(1:end)./max(pohyb(1:end)) ))) < Kde;
   for(i = 1:size(Mrk,2))
      if(Mrk(i) == 1 && (size(Mrk,2) - i) > 1)
         Mrk(i+1:i+2) = 0;
      end
      if(Mrk(i) == 1 && (size(Mrk,2) - i) == 1)
         Mrk(i+1) = 0;
      end
   end
else
   Mrk(1:size(LNej,2)) = 0;
end
% hold on;
% plot((PNej(1:end).*LNej(1:end)) + ((1 - (PNej(1:end).*LNej(1:end))).*(pohyb(1:end) ./
max(pohyb(1:end)))),'y');
% plot(Kde-0.02);
% plot(0.7 * Mrk,'bo')
% hold off;
%%%%%%%%%%%%%%Konec skriptu RozpoznejMrk.m%%%%%%%%%%%%
```

Podprogram OdstranMaleOblasti

```
% Viktor Tesárek, diplomová práce na téma Detekce mrkání
% a rozpoznávání podle mrkání očí.
% Vysoké učení technické v Brně
```

```
% Fakulta Elektrotechniky a komunikačních technologií
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
function [ObrFilt] = OdstranMaleOblasti(Obr, VelikostOblasti)
% Funkce OdstranMaleOblasti odstraňuje oblasti menší než Vstupní
% argument VelikostOblasti z matice Obr, výstupem je pak filtrovaná matice
% ObrFilt.
  ObrOblasti = bwlabel(Obr);

  oblasti = max(max(ObrOblasti));
  oblast = sum(histc(ObrOblasti,1:oblasti),2);

  vyclen = oblast < VelikostOblasti;
  vyclen = unique(vyclen .* (1:oblasti)');
  if(size(vyclen,2) > 1)
    vyclen(1) = vyclen(2);
  end
  Obr( ismember(ObrOblasti, vyclen)) = 0;
  ObrFilt = Obr;
%%%%%%%%%%%%%%%%%Konec skriptu
OdstranMaleOblasti.m%%%%%%%%%%%%%%%%%
```