VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

DIPLOMOVÁ PRÁCE

2015 Jan Šlancar

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Rozpoznávací systém pro analýzu osob na základě obrazového signálu

Recognition System for Person Analysis by Image Signal Processing

2015 Jan Šlancar

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání diplomové práce

Student: **Bc. Jan Šlancar**

Studijní program: N2649 Elektrotechnika

Studijní obor: 3901T009 Biomedicínské inženýrství

Téma: Rozpoznávací systém pro analýzu osob na základě obrazového signálu
Recognition System for Person Analysis by Image Signal Processing

Zásady pro vypracování:

1. Rozbor problematiky rozpoznávacích systémů pro detekci osob.
2. Návrh systému pro analýzu osob na základě obrazového signálu.
3. Realizace systému pro analýzu osob na základě obrazového signálu.
4. Srovnání měřených a analyzovaných výsledků s teoretickými předpoklady.
5. Zhodnocení dosažených výsledků.

Seznam doporučené odborné literatury:

[1] PHILIPS, Dwayne. *Image processing in C.* 2. digitalizované vydání v roce 2000. Lawrence Kansas: R & D Publications, 1994. ISBN 0-13-104548-2. .
[2] HLAVÁČ, Václav a Miloš SEDLÁČEK. *Zpracování signálů a obrazů.* 2.přeprac.vyd. Praha: Nakladatelství ČVUT, 2007. 255 s. ISBN 978-80-01-03110-0.
[3] BOVIK, Alan C. *Handbook of Image and Video Processing.* 1.vyd. San Diego: Academic Press, c2000. 891 s. ISBN 0-12-119790-5.
[4] HALOUNOVÁ, Lena. *Zpracování obrazových dat.* 1. vyd. Praha: Česká technika - nakladatelství ČVUT, 2009. 102 s. ISBN 978-80-01-04253-3.

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

Vedoucí diplomové práce: **Ing. Zdeněk Macháček, Ph.D.**

Datum zadání: 01.09.2014

Datum odevzdání: 07.05.2015

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky OSTRAVA 1

doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení

„Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.“

V Ostravě dne 7.5.2015

podpis studenta

Poděkování

Chtěl bych poděkovat svému vedoucímu diplomové práce Ing. Zdeňku Macháčkovi, PhD., za vedení, rady, zájem a čas, který mi věnoval. Mé poděkování patří také mé rodině a blízkým přátelům za pomoc a podporu během studia.

Abstrakt

Diplomová práce se zabývá rozpoznáním osob na základě obrazového signálu. Je zde rozebrána problematika práce s obrazovým signálem - předzpracování signálu, digitalizace, segmentace, prahování. Na základě prahování obrazu jsou z obrazového signálu detekovány oči a ústa. Dalším parametrem, který je měřen je šířka a výška obličeje. K rozpoznávání osob je vytvořeno grafické uživatelské rozhraní, přes které lze jak rozpoznávat osoby již v databázi uložené, tak přidávat nové záznamy.

Klíčová slova: *obrazový signál, prahování, rozpoznávání obličeje, detekce očí, detekce úst, biometrie, rozpoznávací systém*

Abstract:

This graduation thesis deals with recognition of persons by image signal processing. It analyzes issues with image signal - signal preprocessing, digitalization, segmentation, thresholding. Based on thresholded image are detected eyes and mouth. Another parameter that is measured is the width and height of face. For recognizing of persons is designed graphical user interface, which allows recognition of person from stored database and also add new records.

Key words: *image signal, thresholding, face recognition, eye detection, mouth detection, biometry, recognition system*

Obsah

1. Úvod

Vnímání světa kolem sebe, tedy obrazových informací, je jednou z nejdůležitějších vlastností člověka. Zrak nám umožňuje rozpoznávat různé objekty, získat informace o našem okolí atd. Mozek informace, které dostává ze sítnice našeho oka, zpracovává a vyhodnocuje. Díky toho potom dokážeme určit co vidíme.

Cílem počítačového vidění je naučit počítač, aby zpracoval informaci obdobně jako mozek, tedy porozumět a interpretovat ho jako model reálného života. Pro počítačové zpracování se dá oko nahradit objektivem, sítnice potom snímačem obrazu. Mozkem je potom počítač, který informaci vyhodnocuje.

Počítačové vidění má oproti lidskému oku i jiné možnosti, kdy pomocí druhu snímače můžeme snímat obraz, který lidské oko není schopno vidět. Například pomocí termovize (snímání tepla), tomograf (snímání magnetického pole) nebo například sonar (snímání ultrazvuku).

Pokud se počítačové vidění optimalizuje pro danou činnost, může zautomatizovat činnosti, pro které bylo potřeba lidského zraku. V průmyslu se může jednat o různé automatizace, rozpoznávání kvality součástek, bezpečnostní kontrola pomocí kamery a jiné.

Cílem této práce je navrhnout vhodný algoritmus pro rozpoznávání osob na základě obrazového signálu, tedy rozpoznat osobu z obrazu. Algoritmus bude realizován pomocí programu MATLAB a jeho toolboxů. Popis programu a jednotlivých toolboxů bude uveden dále. V následujících kapitolách budou popsány biometrické systémy a jejich použití. Dále budou rozebrány operace na zpracování obrazu.

V následující části se budu zabývat operacemi na detekci očí a úst, což budou jedny ze stěžejních bodů této práce. Po definici algoritmů pro detekci biometrických údajů popíši vytvořené uživatelské prostředí, přes které je možné přidávat do databáze nové záznamy biometrických dat, nebo také rozpoznávat podle načtené fotografie záznam z databáze, pokud v ní daná osoba s biometrickými daty existuje.

V poslední části práce zobrazím výsledky detekce. Popsány jsou detekce jak správné, tak i nesprávné, jelikož systém je limitován hlavně kvalitou vstupních dat. Jsou zde také popsány požadavky na ideální vstupní obrazy.

2. Biometrické systémy

2.1 Historie

S biometrickými systémy se setkáváme na každém kroku běžného života. Rozpoznáváme osoby podle stylu chůze, hlasu, obličeji, výšce apod. Všechny tyto poznatky jsou prakticky signály, které nesou informaci o biometrických vlastnostech osoby a lze je proto i zaznamenávat a strojově zpracovávat.

První informace o použití biometrie sahá do 14. století n.l. a pochází z Číny. Kresby na stěnách zobrazovaly struktury, které se podobaly otiskům prstů. Ovšem průkazně se poprvé objevilo použití biometrie až v 19.století, a to při použití otisků prstů v kriminalistice.

Pro rozpoznávání osob se vyvinula antropometrie, která měřila 11 tělesných rozměrů:

- tělesná výška
- délka natažené paže
- výška v sedu
- délka hlavy
- šířka hlavy
- délka pravého ucha
- šířka levého ucha
- délka levé nohy
- délka levého prostředníčku
- délka levého malíčku
- délka levého předloktí

Antropometrie se používala k identifikaci nebo verifikaci osob, jelikož bylo prokázáno, že tělesné hodnoty u člověka se po 20. roku života nemění. Lepší spolehlivosti se dosáhne se zvyšujícím se počtem korektně změřených rozměrů. Poté je osobu možné jednoznačně identifikovat, případně verifikovat.[1]

2.2 Identita, identifikace, verifikace

Náš mozek nám umožňuje automaticky rozpoznat osobu podle postavy, obličeje, rtů, hlasu, intonace, pohybů při chůzi nebo například podpisu. Každý jedinec má svou jedinečnou identitu.

V reálném světě neexistují dva lidé se stejnou fyzickou identitou, což neplatí ve světe elektronickém. V elektronickém světe si můžeme vytvářet identity libovolně skrze různé účty, profily atd. Výjimkou jsou případy ve kterých je třeba další ověření než jen kliknutí myši, ale stále se nedá zaručit jedinečnost.

K identitě neodmyslitelně patří pojmy identifikace a verifikace. Identifikace zjišťuje identitu osoby tak, že uživatel zadá své biometrické vlastnosti. Daný systém potom zjistí identitu uživatele. Porovnávají se vstupní vzorky s databází. Identifikace je porovnávání 1:N, tedy jednoho vzorku s více vzorky.

Verifikace je potom proces, kdy uživatel zadá systému svou elektronickou identitu a poté dochází k ověření jeho fyzické identity. Systém vyhledává konkrétní biometrické vlastnosti uživatele, jelikož svou identitu zadal na začátku. Verifikace je porovnávání 1:1, tedy jednoho vzorku s jedním vzorkem z databáze.
Dalším pojmem je autentizace, činnost, při které se potvrzuje hodnověrnost daného uživatele. Jednoduchou autentizací je například porovnávání hesla, kdy je porovnáno zadané heslo s hesle v databázi. V případě porovnávání biometrických dat se jedná o složitou úlohu.
Bezpečí ověření identity má tři různé bezpečnostní úrovně. Ta nejnižší je založena na nějaké znalosti, něčem co víme, jako například heslo, PIN, tajné tlačítko apod. Je to lehce zapamatovatelný přístup, ale lze také jednoduše zneužít.
Bezpečnější možnost je založena na vlastnictví předmětu, který nám umožňuje se identifikovat. Může se jednat o čipovou kartu, RFID-tag, klíč. U tohoto přístupu nastává problém, že můžeme předmět například zapomenout doma nebo ztratit. Zneužít lze také okopírováním, ale už je zde nutnost fyzického kontaktu s předmětem.
Nejbezpečnější metodou ověřování identity je pomocí biometrických vlastností člověka. Tyto vlastnosti nikdy nezapomeneme, sami jsme svým způsobem klíčem. I zde se však dá systém obejít, například pomocí otisků prstů pomocí falešného prstu, vytisknutá fotografie apod. Kombinací více způsobů identifikace se systém stává bezpečnějším. [1]

2.3 Biometrie

Slovo biometrie vzniklo složení dvou řeckých slov: „bios“ (život) a „metron“ (měřítko). V překladu tedy jakési „měřítko života“. V IT oblasti takto označujeme systém nebo postup pro rozpoznávání lidských vlastností (obličeje, duhovky, sítnice, otisku prstů, chování, podpisu, chůze)
Výhody:

- vyšší bezpečnost
- nelze zapomenout či ztratit
- pohodlnost
- nelze lehce přenášet

Nevýhody:

- nezachovává soukromí
- při prozrazení nelze anulovat
- samotný systém je napadnutelný
- nutnost detekce živosti

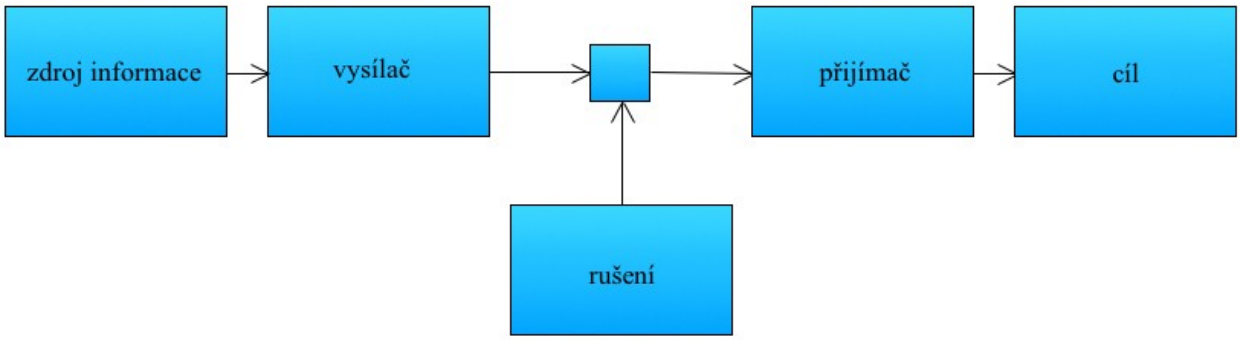
Každý biometrický systém se zjednodušeně skládá ze dvou částí. Jedna část určena pro snímání biometrických vlastností a ukládání do databáze. Druhá pro porovnávání vstupů s již vytvořenou databází.
Obecně se rozlišují unimodální a multimodální biometrické systémy. Unimodální využívají pouze jednu biometrickou vlastnost (nejvíce v praxi). Jsou levnější, ale také méně spolehlivé než systémy multimodální, které využívají více příznaků jedné vlastnosti (statická a dynamická

vlastnost podpisu) nebo více biometrických vlastností (rozpoznávání obličeje zároveň se sítnicí).
Podle jednotlivých nároků se volí systém na snímání určité biometrické vlastnosti. Tato práce se zabývá rozpoznáváním osob pomocí obličeje, proto tuto problematiku zde vynechám. [1]

3. Signál, informace, zpracování obrazu

3.1 Zpracování

Pokud chceme informaci zpracovávat, musíme ji nejdříve získat. Získávání probíhá mezi příjemcem a zdrojem. Jedná se o komunikační systém, který pomocí signálu přenáší informaci v určité formě.



Obr. 1: Obecný komunikační systém – schéma (převzato z [2])

Na Obr. 1 je zobrazeno zjednodušené schéma komunikačního systému. Zdroj informace vytváří zprávu, která je vysílána. Vysílač zpracuje zprávu a vytváří z ní signál vhodný k přenosu. K přenosu signálu od vysílače k přijímači se používá přenosový kanál – médium. Médium může být koaxiální kabel, drátový pár, paprsek světla atd. V kanálu bývá také často různé rušení. Přijímač z přijatého signálu rekonstruuje zprávu kterou vysílač zpracoval na signál. Nakonec je zpráva přijata v cíli, což může být osoba nebo určité zařízení.

Signál

Signál nese nějaký druh informace. Při zpracování signálu se snažíme extrahovat ze signálu danou informaci. Signál může být charakteru spojitého nebo diskrétního. Spojitý signál je takový signál, který může nabývat libovolné hodnoty z daného intervalu. Diskrétní signál je potom takový signál, který nabývá hodnot z konečné množiny čísel.[9] [10]

3.2 Filtrace signálu

Ze signálu potřebujeme pro určitou potřebu zesílit zajímavé složky a naopak potlačit složky, které analyzovat nepotřebujeme. K tomuto slouží filtrace. Elektronické filtry různých typů. Jedná se o aktivní nebo pasivní, digitální nebo analogové, lineární nebo nelineární, s konečnou nebo nekonečnou impulzní odezvou.

V oblasti filtrace jsou dominantními filtry lineární. Platí u nich princip superpozice, tedy sečteme-li odezvy filtru na jednotlivé signály, je výsledek stejný jako odezva součtu těchto signálů. Lineární filtry dělíme na filtry s konečnou impulsní charakteristikou (FIR – Finite Impulse Response) a s nekonečnou impulsní odezvou (IIR – Infinite Impulse Response).

I přes snazší realizaci lineárních filtrů je v oblasti zpracování obrazu vhodnější použití filtrů nelineárních. Do této filtrace spadá například mediánová filtrace (potlačuje impulsní šum a přitom zachová strmost hran signálu), filtry zajišťující erozi a dilataci, Kalmanův filtr a mnoho dalších filtrů vhodných pro zpracování obrazu. [15]

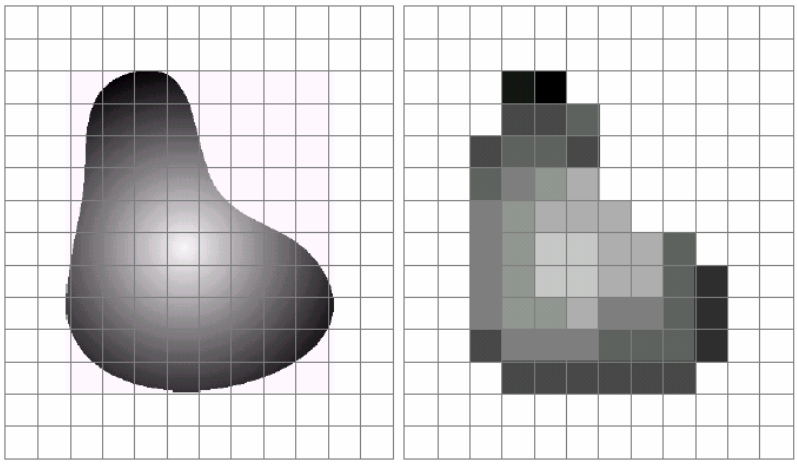
3.3 Zpracování obrazu

3.3.1 Digitalizace

Pokud chceme zpracovávat obraz pomocí výpočetní techniky, musíme nejprve převést jeho fyzickou (optickou) formu do formy elektrické. Tento proces se nazývá digitalizace a zahrnuje vzorkování, kvantování a kódování obrazového signálu.

Vzorkování je proces, kdy vytváříme matici velikosti M x N a diskretizujeme tak souřadnice obrazu. Jedná se o prostorové rozlišení.

Kvantování je rozdělení jasových úrovní, které jsou spojité, do K úrovní. Zde jde o jasové rozlišení.



Obr. 2: vlevo vstupní obraz, vpravo obraz po vzorkování a kvantování (převzato z [3])

Při zpracování obrazu bychom měli vzít v úvahu tyto pojmy:

Vzorkovací interval – pro dvojrozměrné obrazy bychom měli volit minimálně 1/2 velikosti nejmenšího detailu obrazu, avšak optimální je pro zpracování obrazu zvolit tuto hodnotu na 1/5.

Vzorkovací mřížka – plošné uspořádání bodů při vzorkování.

Kvantovací úroveň – jas musí být vyjádřen jako digitální údaj. Lidské oko rozliší asi 50 jasových úrovní, a proto musí být dostatečně velký počet kvantovacích úrovní, aby nedocházelo ke zkreslení obrazu. Používá se kvantování do k intervalů. Pokud obrazový element reprezentuje b bitů, pak je počet úrovní jasu $k = 2^b$.

Jeden obrazový element v digitalizovaném obrazu se nazývá pixel. Vzorkovací mřížku vyplňují jednotlivé pixely a pokrývají celý digitalizovaný obraz.[9] [10]

3.3.2 Vlastnosti digitálního obrazu

Monochromatický obraz popisují 2 souřadnice v rovině. U barevného obrazu má každá dvojice souřadnic ještě vektor, který reprezentuje danou chrominační složku (například model RGB má 3 chrominační složky, každá matice souřadnic představuje jas dané složky).

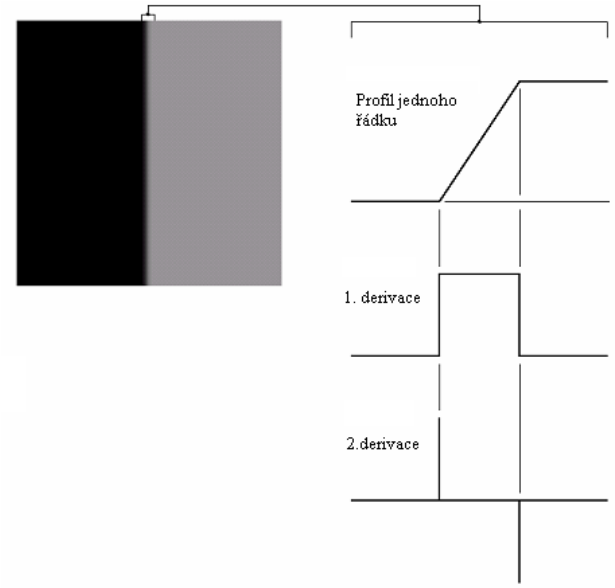
3.3.3 Histogram

Histogram nám dává informaci o tom, kolik které jasové úrovni připadá pixelů v obraze. Podává globální informaci o obraze. Často se stává, že nejsou využity všechny jasové hodnoty, ale jen úzká oblast. V takovýchto případech je vhodné v rámci předzpracování obrazu histogram ekvalizovat.

Ekvalizace histogramu znamená to, že se využijí všechny jasové úrovně a v ideálním případě má každá jasová úroveň stejnou četnost. Pro úrovně blízko maxim se jas zvyšuje, v blízkosti minim se jas snižuje.[13]

3.3.4 Detekce hran

Hrany v obraze vznikají tam, kde se skokově mění hodnota jasu.



Obr. 3: vlevo: detail skutečné hrany, vpravo shora: profil jednoho řádu, 1. a 2. derivace (převzato z [3])

První derivace je kladná v místech kde se mění hodnota jasu, když je jas konstantní, 1. derivace je nulová. Hranu lze určit z amplitudy 1. derivace.

Druhá derivace je kladná na přechodu z tmavé na světlou hranu na straně tmavé a záporná při přechodu na straně světlé. V místě přechodu je nulová. Pomocí znaménka 2. derivace můžeme určit, jestli bod leží na světlé nebo tmavé straně hrany. Pokud spojíme maximální hodnoty 2. derivace, protne nám pomyslná čára nulovou hodnotu v místě, které odpovídá středu čáry.

Derivace jsou však velmi citlivé na šum. 2. derivace je na šum ještě mnohem citlivější než derivace první.

Pro změnu dvou proměnných jsou vhodné parciální derivace, změnu funkce popisuje její gradient. Gradient je vektorová veličina, která určuje směr největšího růstu funkce (dáno směrem gradientu) a strmost tohoto růstu (velikost/modul gradientu). Pixely, které mají velký modul gradientu se dají nazývat hranami.
Gradient je dán vztahem (1):

$$\nabla f(x,y) = \left(\frac{\partial f(x,y)}{\partial x}, \frac{\partial f(x,y)}{\partial y}\right) \quad (1).$$

Velikost (modul) gradientu je dán (2):

$$|\nabla f(x,y)| = \sqrt{\left(\left(\frac{\partial f(x,y)}{\partial x}\right)^2 + \left(\frac{\partial f(x,y)}{\partial y}\right)^2\right)} \quad (2),$$

a směr gradientu popisuje vztah (3):

$$\varphi = arctan\left(\frac{\partial f(x,y)}{\partial x} / \frac{\partial f(x,y)}{\partial y}\right) \quad (3).$$

Nejjednodušší metodou výpočtu gradientu je aplikace Robertsova operátoru. Tento operátor detekuje především hrany, které mají sklon 45 stupňů. Popisují jej konvoluční masky:

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Velikost gradientu se potom počítá dle (4):

$$|g(x,y) - g(x+1,y+1)| + |g(x,y+1) - g(x+1,y)| \quad (4).$$

Robertsův operátor je však velmi citlivý na šum, jelikož aproximační oblast je velmi malá.
Přesnější metody založené také na první derivaci jsou Sobelův oprátor a operátor Prewittové. Tyto operátory nejsou tak citlivé na šum, jelikož aproximační plocha je větší – matice 3x3. Gradient je odhadován v osmi směrech, maska s největším modulem gradientu je poté vybrána jako nejvhodnější.
Sobelův operátor – ukázka konvolučních masek pro detekci vodorovných a svislých čar:

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}.$$

Operátor Prewittov - ukázka konvolučních masek pro detekci vodorovných a svislých čar:

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}, \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}.$$

Součet prvků v konvoluční matici musí být roven nule.
Dalším operátorem je Laplacián. Ten vychází z druhé derivace a má tvar $\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ pro čtyři okolní prvky a tvar $\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ pro osm okolních prvků.[16]

3.3.5 Segmentace

Segmentace je proces, kdy rozdělíme obraz na několik podobrazů, které chceme dále analyzovat. Segmentace může být realizována pomocí různých metod, například statistických, metod založených na detekci hran, regiony v obraze atd.

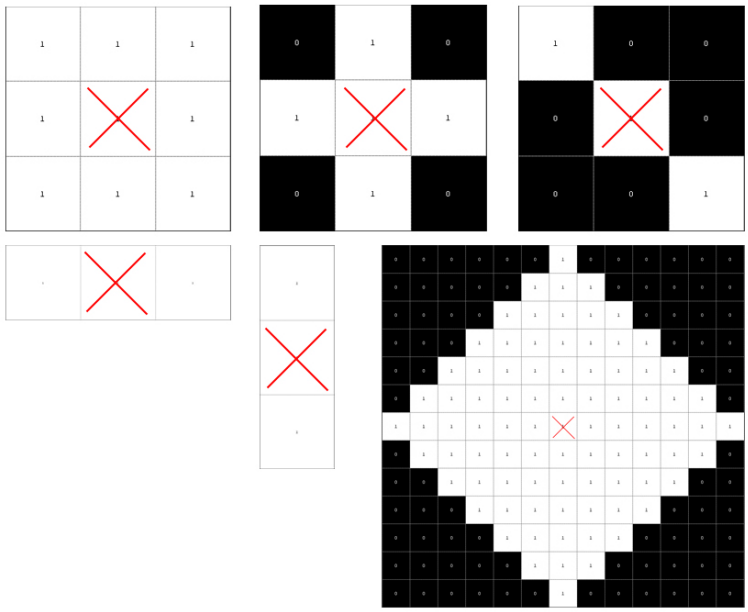
Úkolem segmentace je oddělit nezajímavé pozadí od popředí, které chceme dále analyzovat. Dokonalost segmentace nelze v této fázi dosáhnout, hlavním úkolem je zmenšit počet dat, se kterými se bude pracovat. Je například zbytečné pracovat s velkým obrazem, na kterém je malá osoba, když potřebujeme analyzovat právě tuto osobu. Segmentace nám pomůže ušetřit výpočetní výkon na důležitější úlohy. Různé metody segmentace mohou přinést různé výsledky, proto je třeba volit druh segmentace vždy podle daného problému. [11]

3.3.6 Prahování

Prahování je jednou z nejjednodušších a nejpoužívanějších druhů segmentací. Prahování je založeno na intenzitách jasu v obrazu. Podle zvolené meze (prahu) se porovnává každý pixel s touto hodnotou. Každý pixel, který má hodnotu menší než práh je určen jako pozadí.[12]

3.3.7 Morfologická transformace

Morfologická transformace je dána vztahem mezi vstupním obrazem a definovaným strukturním elementem. Tento element se vztahuje k tzv. reprezentativnímu bodu, což je jeho lokální počátek.



Obr. 4: Příklad strukturních elementů, počátky značeny červeným křížem (převzato z [6])

Dilatace

Výstupem je zvětšení objektů v binárním obraze o strukturní element. Rovnice dilatace:

$$A \oplus B = \left\{ z | \left(\hat{B}\right)_z \cap A \neq \emptyset \right\} \quad (5).$$

Rovnice (5) definuje, že dilatace A strukturním elementem B je množina všech pozic počátku strukturního elementu, kde zrcadlené a posunuté B překrývá alespoň jeden pixel A.

Eroze

Výstupem je zmenšení objektů v binárním obraze o strukturní element. Rovnice eroze:

$$A \ominus B = \{z|(B)_z \cap A^C \neq \emptyset\} \quad (6).$$

Rovnice (6) definuje, že eroze A strukturním elementem B je množina všech pozic počátku strukturního elementu, kde se B nepřekrývá s pozadím A.

Uzavření

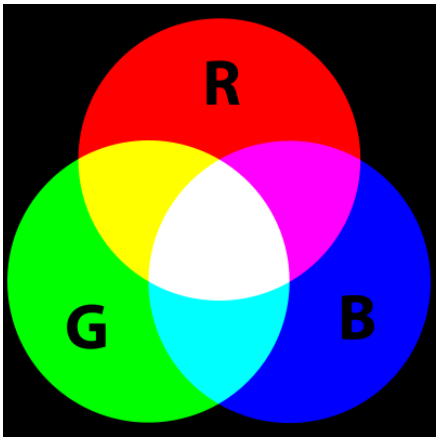
Provedení dilatace a následná eroze se stejným strukturním elementem. Rovnice uzavření:

$$A \cdot B = (A \oplus B) \ominus B \quad (7).$$

Dle rovnice (7) dojde k dilataci a následné erozi obrazu, což má za následek spojení blízkých objektů a zaplnění malých (v poměru k elementu) děr. [6]

3.4 RGB model

Zkratka RGB popisuje tři základní složky (barvy), ze kterých se tento model skládá. R = Red (červená), G = Green (zelená) a B = Blue (modrá). Model je aditivní, což znamená, že se jednotlivé složky míchají (sčítají). Sčítají se jednotlivá světelná záření, každé v rozmezí 0 až 255. Při sečtení všech složek v plné intenzitě vzniká bílá barva. Naopak při sečtení složek o nulové intenzitě vzniká barva černá (viz. Obr. 5).



Obr. 5: Barevný model RGB

3.5 YCbCr model

Tento model se používá u videa a digitálních fotografií. Y představuje luminanci (jas), Cb je modrý a Cr červený chrominační komponent. Tento model je způsob kódování RGB modelu pomocí určitých převodních vztahů. Tento model je bližší lidskému zraku, jelikož reaguje na jas, tak jako oko.

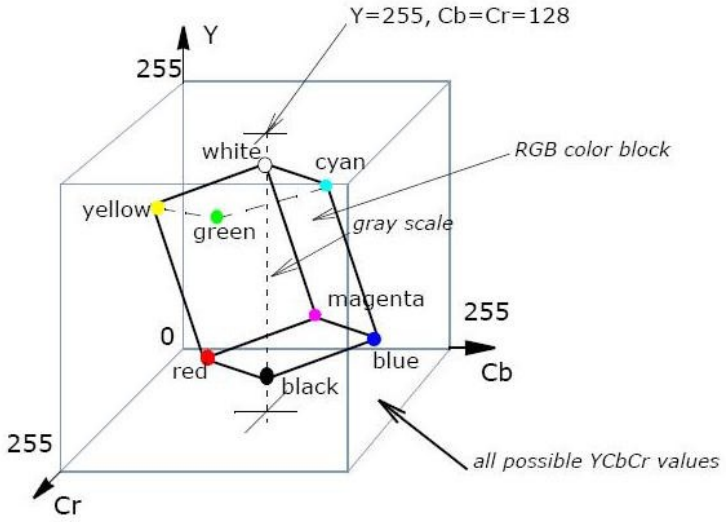
Převod z RGB do YCbCr je pomocí následujících rovnic (8) - (10):

$$Y = 0{,}299 \cdot R + 0{,}587 \cdot G + 0{,}114 \cdot B \quad (8),$$

$$Cb = 128 - 0{,}168736 \cdot R - 0{,}331264 \cdot G + 0{,}5 \cdot B \quad (9),$$

$$Cr = 128 + 0{,}5 \cdot R - 0{,}418688 \cdot G - 0{,}081312 \cdot B \quad (10).$$

Zobrazení modelu v prostoru je na Obr. 6 včetně jednotlivých barev a naznačení pozice RGB modelu v prostoru YCbCr.[14]



Obr. 6: Barevný model YCbCr

4. MATLAB

Tato kapitola bude obsahovat informace o programu MATLAB a jeho toolboxech, které budou potřeba pro realizace programu na rozpoznávání obličejů. Všechny předchozí zmíněné operace pro zpracování obrazu MATLAB umožňuje, některé z nich jsou funkce již vestavěné. Program se používá k řešení různých výpočetních problémů.

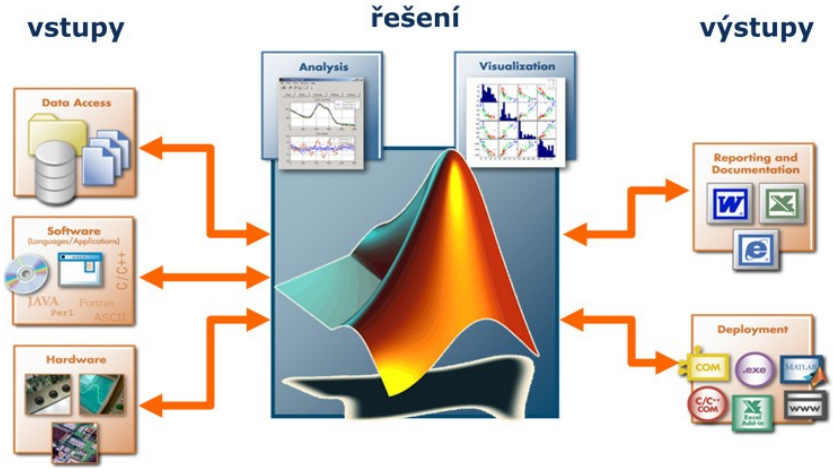
Výpočetní systém MATLAB se během uplynulých let stal celosvětovým standardem v oblasti technických výpočtů a simulací ve sféře vědy, výzkumu, průmyslu i v oblasti vzdělávání.

MATLAB poskytuje svým uživatelům nejen mocné grafické a výpočetní nástroje, ale i rozsáhlé specializované knihovny funkcí spolu s výkonným programovacím jazykem čtvrté generace. Knihovny jsou svým rozsahem využitelné prakticky ve všech oblastech lidské činnosti.

Díky své architektuře je MATLAB určen zejména těm, kteří potřebují řešit početně náročné úlohy a přitom nechtějí nebo nemají čas zkoumat matematickou podstatu problémů. Více než milion uživatelů po celém světě využívá možnosti jazyka MATLABu, který je mnohem jednodušší než například Fortran nebo C a který skýtá obrovský potenciál produktivity a tvořivosti. Za nejsilnější stránku MATLABu je považováno mimořádně rychlé výpočetní jádro s optimálními algoritmy, které jsou prověřeny léty provozu na špičkových pracovištích po celém světě. MATLAB byl implementován na všech významných platformách (Windows, Linux, Solaris, Mac).

Mezi klíčové vlastnosti Matlabu patří:

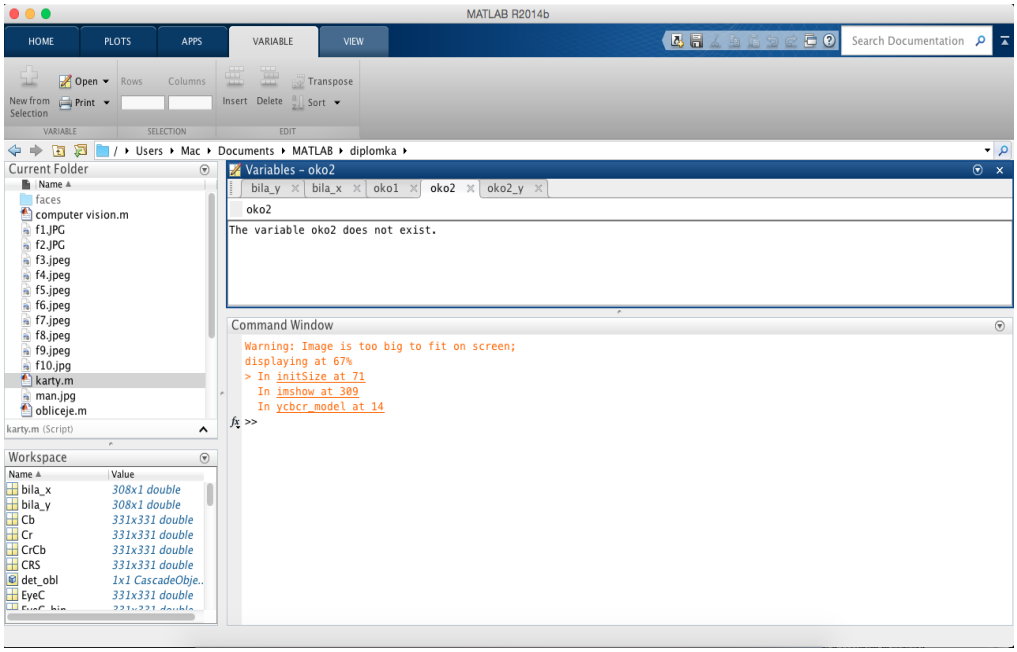
- vysokoúrovňový jazyk pro technické výpočty
- otevřený a rozšiřitelný systém
- velké množství aplikačních knihoven
- podpora vícerozměrných polí a datových struktur
- interaktivní nástroje pro tvorbu grafického uživatelského rozhraní
- import a export dat do mnoha formátů, znázorněno na obrázku 1
- komunikace s externími měřícími a monitorovacími přístroji v reálném čase
- rozšiřitelnost modulů jazyky C, C++, Fortra, Java



Obr. 7: Vizualizaee funkce MATLABu.

4.1 Popis prostředí MATLAB

Po spuštění programu vidíme hlavní okno. Na Obr. 8 je zobrazen MATLAB v operačním systému MAC OS X. Pracovní prostředí se skládá z několika částí, z nichž nejdůležitější je Command Window pomocí něhož probíhá pomocí příkazů komunikace mezi uživatelem a výpočetním jádrem MATLABu. Vypisují se zde také chybová hlášení, výsledky, proměnné a například historie použitých příkazů. V levé části dole je část Workspace, kde se ukládají proměnné které definujeme. Informaci o adresáři, ve kterém momentálně pracujeme, zobrazuje okno Current Folder vlevo nad Workspace. Jednotlivé rozložení těchto částí je záležitost uživatele a okna se dají libovolně přesouvat.



Obr. 8: Pracovní prostředí MATLAB

4.2 M-soubory (M-files)

Při komunikaci s jádrem MATLABu pomocí Command Window dostáváme výsledky ihned po potvrzení příkazu. To je výhodou při potřebě rychle vypočítat jednoduchý příklad, nebo pro jednoduchý příkaz. Ovšem chceme-li řešit komplexnější problém, je Command Window pro programování nevhodné. Proto jsou zde M-soubory (M-files). Tento soubor je v podstatě textový editor, kde můžeme psát složité kódy, různě je komentovat a hlavně je zde možnost uložení. Do těchto souborů si také můžeme uložit například naprogramované funkce, a pak je pomocí příkazů volat z M-souboru jiného.

4.3 Toolboxy

Toolboxy v MATLABu jsou knihovny, které obsahují vestavěné funkce pro určité operace. Jednotlivé toolboxy rozšiřují funkce MATLABu pro různá použití v různých odvětvích. Ve své podstatě jsou to M-soubory, které pomocí určitých parametrů voláme. Pro tuto práci bude

potřeba toolboxů pro zpracování obrazu (Image Processing Toolbox) a pro počítačové vidění (Computer Vision System Toolbox).

4.3.1 Image Processing Toolbox

Image Processing Toolbox je výkonný, pružný a snadno ovladatelný nástroj pro zpracování a analýzu obrazu. Na základě mohutného výpočetního potenciálu MATLABu jsou vybudovány nadstavby pro návrhy filtrů, rekonstrukci a analýzu obrazů, dále nadstavby pro manipulaci s barvami, geometrií a strukturou obrazů včetně 2-D transformací. Na technologii zpracování obrazu jsou založeny špičkové metody lékařské i průmyslové diagnostiky, analýzy dat a automatizace. Analýza obrazu je nepostradatelná v astronomii, geofyzice, ale i v ekologii a dalších oborech jako jsou například komunikace, vojenství nebo spotřební elektronika. Pro svou výpočetní mohutnost, otevřenost a strukturu aplikačních knihoven je MATLAB spolu s Image Processing Toolboxem optimálním nástrojem v tak mnohaoborovém prostředí jako je digitální zpracování obrazu. [5]

4.3.2 Computer Vision System Toolbox

Computer Vision System Toolbox poskytuje algoritmy a nástroje pro návrh a simulaci systémů zpracování videa a strojového vidění. Algoritmy jsou ve formě funkcí MATLABu, System objektů a knihoven bloků pro Simulink. Toolbox obsahuje algoritmy pro extrakci charakteristických vlastností, detekci pohybu, sledování objektů, stereo vision, zpracování videa a jeho analýzu. Nástroje zahrnují i načítání, zobrazení a ukládaní videa včetně vkládání grafických prvků. Algoritmy toolboxu také podporují výpočty v aritmetice s pevnou řádovou čárkou (fixed-point) a generování kódu v jazyce C, což lze využít při návrhu embedded zařízení nebo k návrhu metodou real-time rapid-prototyping.[4]
Tento toolbox obsahuje detektor Viola-Jones, který dokáže detekovat různé části obličeje. V práci je využit na výřez obličeje z původního obrazu do obrazu čtvercového tvaru, který je dále zpracováván a analyzován.

5. Metoda rozpoznávání očí a úst

5.1 Rozpoznání oblasti tváře

V první fázi musíme načíst obrázek pro zpracování. Obrázek načteme pomocí příkazu imread() a zmenšíme jej na velikost 640x480 pixelů, z důvodu ušetření paměti a výpočetní náročnosti. Deklarujeme vestavěný detektor obličeje, který je součástí Computer Vision System Toolboxu pomocí příkazu vision.CascadeObjectDetector. Dále si definujeme barvu rámečku, který použijeme na zvýraznění oblasti obličeje příkazem vision.ShapeInserter(). Na načtený obrázek aplikujeme detektor obličeje pomocí příkazu step(), což nám vrátí souřadnice detekovaného obličeje. Tyto souřadnice poté opět pomocí funkce step() zakreslíme do původního obrazu.

Obr. 9: Původní obrázek

Kód pro načtení, zmenšení a zobrazení obrázku je následující:

```
obr = imread('f2.jpg');              % nacteni obrazku
obr = imresize(obr,[640 480]);       % zmena rozmeru
imshow(obr)
```

Obrázek po detekci obličeje je na Obr. 10. Do původního obrazu je zakreslen čtverec, který vymezuje část obličeje, jež byl vypočítán pomocí detektoru Viola-Jones, který je součástí Computer Vision System Toolboxu.

Kód pro definici detektoru a tvaru s následným zakreslení do původního obrázku je:

```
det_obl = vision.CascadeObjectDetector;          %definice detektoru
tvar = vision.ShapeInserter('BorderColor','White'); %definice tvaru
```

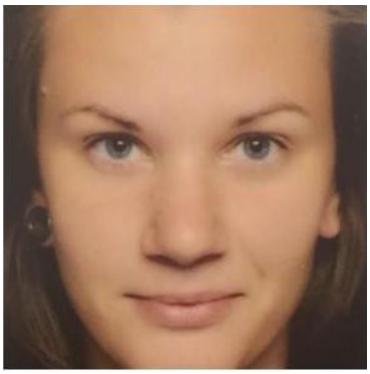
```
figure()
ob = step(det_obl,obr);                %aplikace detektoru na puv. obr.
obr_ob  =  step(tvar,obr,int32(ob));    %vykresleni tvaru dle souradnic
                                            detekovaneho obliceje
imshow(obr_ob)                         %zobrazeni tvaru v puv.obr.
```

Obr. 10: Původní obrázek s detekcí obličeje

Následuje krok, kdy si oblast s obličejem oddělíme od zbytku obrazu pomocí funkce imcrop(). Funkci imcrop() dáme jako parametry ořezání právě souřadnice, které vypočítal detektor. Výsledek operace je na Obr. 11.

Kód pro ořezání a zobrazení:

```
figure()
orez = imcrop(obr, ob);      %orezani obr.
imshow(orez)
```



Obr. 11: Výřez oblasti obličeje

5.2 Převod RGB do YCbCr modelu

5.2.1. Oči

Převod do YCbCr modelu z modelu RGB umožňuje vestavěná funkce MATLABu rgb2ycbcr(). V ní jsou obsaženy převodní vztahy dle rovnic (8) - (10). Převod aplikujeme na výřez s obličejem. Pomocí matematických operací dle (11) - (13) zvýrazníme tmavé jasové složky, kterým odpovídají oči, ale také například tmavé vlasy.

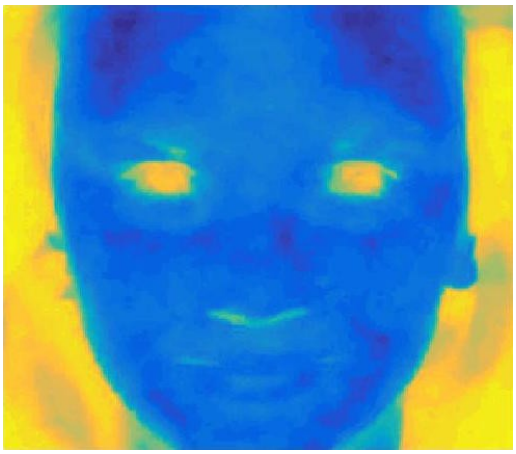
$$\boldsymbol{Q = Cb^2} \quad (11),$$

$$\boldsymbol{R = (255 - Cr)^2} \quad (12),$$

$$\boldsymbol{G = \frac{Cb}{Cr}} \quad (13).$$

Zprůměrováním hodnot Q, R a G dostaneme mapu očí EyeC.

$$\boldsymbol{EyeC = \frac{Q+R+G}{3}} \quad (14).$$



Obr. 12: Mapa očí EyeC z výřezu obličeje

Na Obr. 12 vidíme výsledný obraz z modelu YCbCr, kde jsou pomocí matematických operací zvýrazněny jasy očí.

Kód pro operace popsané výše:

```
ycbcr = rgb2ycbcr(orez);
[m n l]=size(ycbcr);
%Finding EyeMapC
y = double(ycbcr(:,:,1));
Cb = double(ycbcr(:,:,2));
Cr = double(ycbcr(:,:,3));

Q = Cb.^2;
R = (255-Cr).^2;
G = Cb./Cr;
%oci
EyeC=(Q+R+G)/3;
```

5.2.2. Ústa

Ve fázi detekce úst provádíme zvýraznění jasových složek dle rovnic:

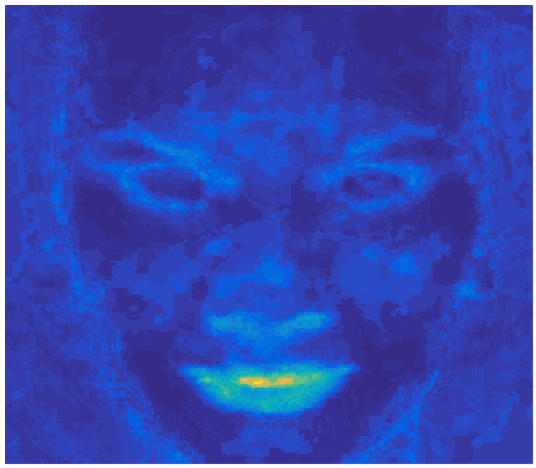
$$\boldsymbol{CRS = Cr^2} \quad (15),$$

$$\boldsymbol{eta = 0,95 \cdot \frac{\sum\sum CRS}{\sum\sum \frac{Cr}{Cb}}} \quad (16),$$

$$\boldsymbol{x = CRS - eta \cdot \frac{Cr}{Cb}} \quad (17),$$

$$\boldsymbol{MM = CRS \cdot x^2} \quad (18).$$

Výsledná mapa úst MM je na Obr. 13.



Obr. 13: Mapa úst MM z výřezu obličeje

Kód z MATLABu:

```
CRS = Cr.^2;
ssCRS = sum(sum(CRS));
ssCrCb=sum(sum(CrCb));
eta = 0.95 * ssCRS/ssCrCb;
x= CRS - eta * Cr./Cb;
MM = CRS.*x.*x;
```

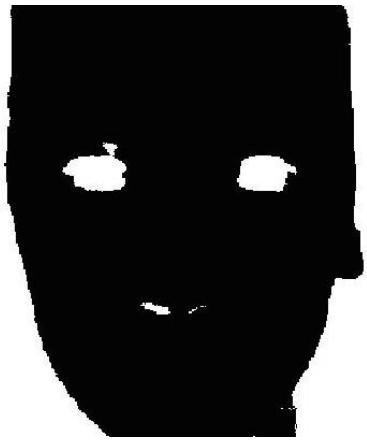
5.3 Prahování

5.3.1. Oči

Na Obr. 12 aplikujeme prahování tak, že jednotlivé pixely v obrazu, které mají hodnotu větší než 75 % hodnoty jasu budou odpovídat hodnotě 1 (bílá barva). Hodnoty nižší pak budou mít hodnotu 0 (černá barva). Operaci provádí kód:

```
prah = 0.75 * max(max(EyeC));     % nastaveni prahu na 80% max hodnoty
[r, s] = size(EyeC);              %rozmery obrazu EyeC
EyeC_bin = zeros(r,s);            %matice pro binarni obraz
```

```
for i = 1:r                    %cyklus pro prahovani
    for j = 1:s
        if EyeC(i,j) < prah
            EyeC_bin(i,j) = 0;
        else
            EyeC_bin(i,j) = 1;
        end
    end
end
imshow(EyeC_bin)
```



Obr. 14: Binární obraz po prahování

Výsledek prahování vidíme na Obr. 14. Výstupem je binární obraz, kde bílá je představována hodnotou 1, černá potom hodnotou 0.

5.3.2. Ústa

Obdobně jako v předchozím bodě aplikujeme prahování na Obr. 13 tak, že jednotlivé pixely v obrazu, které mají hodnotu větší než 55 % budou odpovídat hodnotě 1 (bílá barva). Hodnoty nižší pak budou mít hodnotu 0 (černá barva). Operaci provádí kód:

```
prah_M = 0.55 * max(max(MM));     % nastaveni prahu na 55% max hodnoty
[r, s] = size(MM);                %rozmery obrazu MM
M_bin = zeros(r,s);               %matice pro binarni obraz
for i = 1:r                       %cyklus pro prahovani
    for j = 1:s
        if MM(i,j) < prah_M
            M_bin(i,j) = 0;
        else
            M_bin(i,j) = 1;
        end
    end
end
imshow(M_bin)
```

Obr. 15: Binární obraz po prahování

Výsledek prahování vidíme na Obr. 15. Výstupem je binární obraz, kde bílá je představována hodnotou 1, černá potom hodnotou 0.

5.3.3. Obrys hlavy

Pro prahování obrysu hlavy se vychází z YCbCr barevného modelu. V tomto modelu odstíny lidské kůže nabývají hodnot červeného a modrého chrominančního komponentu v určitém rozmezí. V uváděném případě je minimální hranice modrého chrominančního komponentu nastavena na hodnotu 80 (v kódu cb_min) a horní potom na 100 (cb_max). Rozmezí červeného chrominančního komponentu je od 105 (cr_min) do 190 (cr_max). Na výsledný obraz je dále použita morfologická operace uzavření, což je dilatace následována erozí obrazu.

Obr. 16: Obrys hlavy po prahování

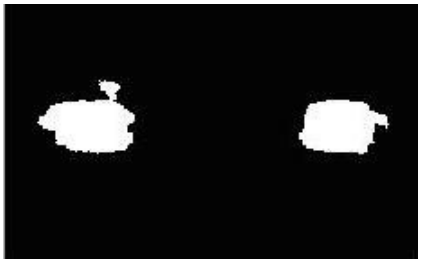
Na Obr. 16 je výsledek prahování. Rozmezí hodnot pro prahování se může měnit v závislosti na kvalitě fotografie a odstínu kůže, je proto třeba tyto hodnoty přizpůsobit dané fotografii.

Kód z MATLABu:

```
i = 1;
j = 1;
for i = 1:r1
    for j = 1:s1
        if Cb1(i,j)<cb_min || y1(i,j)>cb_max
            Cb2(i,j) = 0;
        else Cb2(i,j) = 1;
        end
        if Cr1(i,j)<cr_min || y1(i,j)>cr_max
            Cr2(i,j) = 0;
        else Cr2(i,j) = 1;
        end
    end
end
oblic = Cb2+Cr2;
SE = strel('disk',4);  %vytvoreni struktury pro dilataci/erozi
UP = imdilate(oblic,SE);    %dilatace
Down = imerode(UP,SE);   %eroze
```

5.4 Detekce očí

Při detekci očí z binárního obrazu budeme využívat znalosti, že ve výřezu obličeje se oči nenachází v oblasti 18 % šířky z každé strany a také v 20 % výšky shora a 40 % výšky zespod. Proto si pomocí funkce imcrop() uděláme výřez z binárního obrazu předpokládané oblasti s očima (viz Obr. 17).



Obr. 17: Binární obraz – výřez očí

Kód pro výřez očí z binárního obrazu:

```
vyrez_oci = imcrop(EyeC_bin,[0.18*s,0.2*r,0.64*s,0.4*r]);
imshow(vyrez_oci)
```

V následujícím kroku pomocí příkazu find() nalezneme souřadnice všech nenulových pixelů, tedy v obraze odpovídající bílé barvě (oči). V cyklu potom rozdělím tyto nenulové pixely, které odpovídají souřadnicím x, do dvou matic, každá představuje jedno oko. Cyklus probíhá tak, že bere souřadnici x nenulového pixelu a porovnává ji s polovinou obrazu. Pokud je souřadnice x

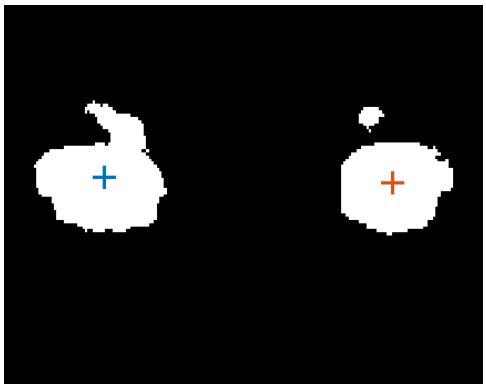
menší než polovina šířky obrazu, jedná se o oko vlevo (na obrázku levé, v reálu pravé). Naopak pokud je souřadnice x větší než polovina šířky obrazu, jde o oko vpravo. Na základě roztřídění x-ových souřadnic jim zpět přiřadíme odpovídající souřadnice y.
Operace pro roztřídění nenulových pixelů řeší kód:

```
[bila_y bila_x] = find(vyrez_oci);  %nalezeni nenulovych pixelu
[r_oci s_oci] = size(vyrez_oci);    %velikost vyrezu oci
i1 = 1;
j1 = 1;
for i = 1:length(bila_x)            %cyklus pro roztrideni x souradnic
    if bila_x(i) > s_oci/2
        oko2_x(i1) = bila_x(i);
        i1 = i1 + 1;
    else oko1_x(j1) = bila_x(i);
        j1 = j1 + 1;
    end
end

for i = 1:length(oko1_x)            % prirazeni y sour. odpovidajici x
    oko1_y(i) = bila_y(i);
end

j = 1;
for i = length(oko1_x)+1:length(bila_y)
    oko2_y(j) = bila_y(i);
    j = j + 1;
end
```

Po roztřídění x a y souřadnic jednotlivých očí se pro každé oko vypočte střed pomocí sečtení všech souřadnic daného oka (příkaz sum()) a dělíme počtem souřadnic, dostáváme souřadnici průměru. Tyto souřadnice pak vykreslíme do výřezu očí (viz Obr. 18).



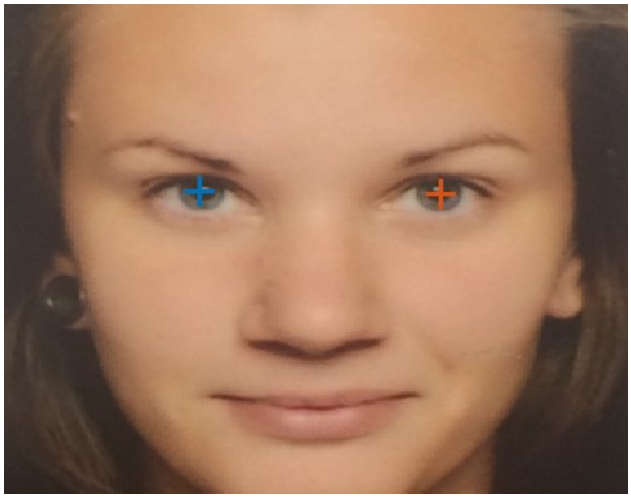
Obr. 18: Binární obraz – výřez očí se zakreslenými středy

Kód pro výpočet a zakreslení středů:

```
stred_oko1_x = sum(oko1_x)/length(oko1_x);
stred_oko2_x = sum(oko2_x)/length(oko2_x);
```

```
stred_oko1_y = sum(oko1_y)/length(oko1_y);
stred_oko2_y = sum(oko2_y)/length(oko2_y);

% zakresleni stredu do vyrezu
figure()
imagesc(vyrez_oci); axis off;
colormap(gray)
hold on
plot(stred_oko1_x,stred_oko1_y,'+')
plot(stred_oko2_x,stred_oko2_y,'+')
```



Obr. 19: Detekovány oči ve výřezu obličeje

Na Obr. 19 jsou zakresleny detekované oči do výřezu obličeje. Zjištěné středy očí v rámci výřezů je nutné přepočítat do původního výřezu. Přepočet souřadnic zpět do výřezu:

```
stred1_oko1_x = stred_oko1_x + 0.18*s;   %prepocet souradnic oci zpet
                                         do vyrezu obliceje
stred1_oko2_x = stred_oko2_x + 0.18*s;
stred1_oko1_y = stred_oko1_y + 0.2*r;
stred1_oko2_y = stred_oko2_y + 0.2*r;
```

5.5 Detekce úst

Při detekci úst z binárního obrazu budeme využívat znalosti, že ve výřezu obličeje se ústa nenachází v oblasti 30 % šířky z každé strany a také v 70 % výšky shora a 10 % výšky zespod. Pomocí funkce imcrop() uděláme výřez z binárního obrazu předpokládané oblasti s ústy (viz Obr. 20).

Obr. 20: Binární obraz – výřez úst

Kód pro výřez úst z binárního obrazu:

```
vyrez_M = imcrop(M_bin,[0.3*s,0.7*r,0.4*s,0.2*r]);
imshow(vyrez_M)
```

V následujícím kroku pomocí příkazu find() nalezneme souřadnice všech nenulových pixelů, tedy v obraze odpovídající bílé barvě (ústa) jako v předchozím bodě.
Operace pro nalezení nenulových pixelů řeší kód:

```
[bilaM_y bilaM_x] = find(vyrez_M);  %nalezeni nenulovych pixelu
[r_M s_M] = size(vyrez_M);     %velikost vyrezu pusy

stred_M_x = sum(bilaM_x)/length(bilaM_x);
stred_M_y = sum(bilaM_y)/length(bilaM_y);
```

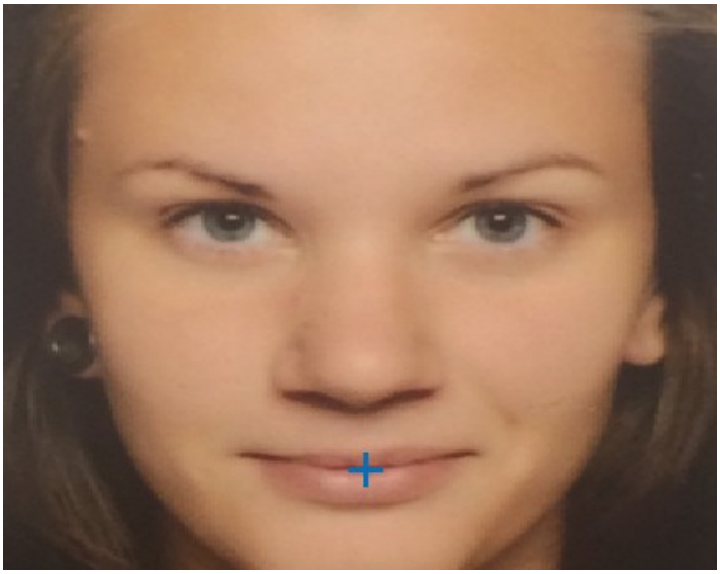
Po nalezení nenulových pixelů se vypočte střed úst pomocí sečtení všech souřadnic (příkaz sum()) a dělíme počtem souřadnic, dostáváme souřadnici průměru. Tyto souřadnice pak vykreslíme do výřezu úst (viz Obr. 21).

Obr. 21: Binární obraz – výřez úst se zakresleným středem

Kód pro zakreslení středu do výřezu:

```
figure()
```

```
imagesc(vyrez_M); axis off;
colormap(gray)
hold on
plot(stred_M_x,stred_M_y,'+')
```



Obr. 22: Detekována ústa ve výřezu obličeje

Na Obr. 22 jsou zakreslena detekována ústa do výřezu obličeje. Zjištěný střed úst v rámci výřezu je nutné přepočítat do původního výřezu. Přepočet souřadnic zpět do výřezu:

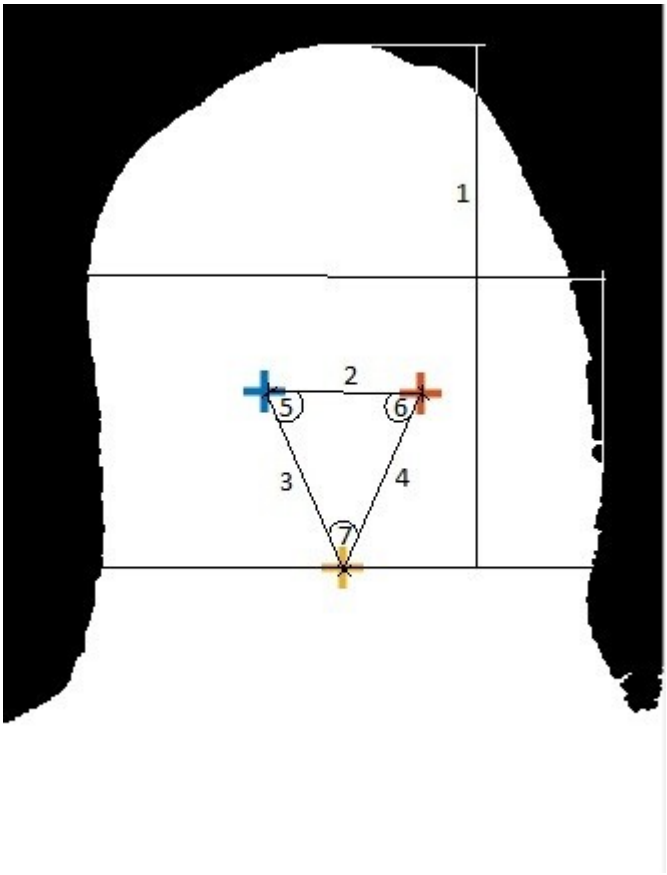
```
stred1_M_x = stred_M_x + 0.3*s;  %prepocet souradnic ust zpet do
                                    vyrezu obliceje
stred1_M_y = stred_M_y + 0.7*r;
```

Detekce očí a úst touto metodou vyžaduje kvalitní snímky, ideálně přímého pohledu. Jsou důležité jasové vlastnosti a nasvíceni fotografie.

6. Výpočet biometrických hodnot pro detekci

Výpočet biometrických údajů, které se vypočítají pro uložení do databáze nebo pro rozpoznání osoby z databáze sestává z následujících atributů:

1) poměr šířky a výšky obličeje,
2) poměr vzdálenosti očí k šířce obličeje,
3) poměr vzdáleností levého oka a úst k šířce obličeje,
4) poměr vzdáleností pravého oka a úst k šířce obličeje,
5) úhel mezi přímkami spojujících oči a levé oko s ústy,
6) úhel mezi přímkami spojujících oči a pravé oko s ústy,
7) úhel mezi přímkami spojujících levé oko s ústy a pravé oko s ústy.



Obr. 23: Měřené biometrické parametry

Na Obr. 23 jsou znázorněny měřené parametry, očíslované dle seznamu na obrázkem.

6.1 Poměr šířky a výšky

Prahovaný obrys obličeje se nejdříve ořízne zespodu v úrovni detekovaných úst. Poté se prochází obraz shora a detekuje se horní hranice hlavy. Následně je změřena vzdálenost v y-ové ose těchto dvou bodů. Tento údaj představuje výšku. Pro měření šířky se používá obdobný proces, taktéž počítaný z ořezaného obrazu podle úst. Zde se detekuje první bílý pixel zleva a

poté zprava. Po změření vzdálenosti x-ových souřadnic těchto bodů dostáváme šířku obličeje. Naměřená šířka s výškou se dají do poměru a to je první biometrický údaj (*sir_vys*).
Operace provádí tento kód z MATLABu:

```
orez1 = [0 0 s stred2_M_y];
obl_sirka = imcrop(Down,orez1);      % orezani zespodu podle ust
[y2end, x2end] = size(obl_sirka);
x1 = 1;
x2 = x2end;
y1 = 1;
y2 = y2end;
%zjisteni obl zleva
poc = 1;
while (sum(obl_sirka(:,poc)) == 0)
    x1 = x1 + 1;
    poc = poc + 1;
end
%zjisteni obl zhora
poc = 1;
while (sum(obl_sirka(poc,:)) == 0)
    y1 = y1 + 1;
    poc = poc + 1;
end
%zjisteni obl zprava
poc = x2;
while (sum(obl_sirka(:,poc)) == 0)
    x2 = x2 - 1;
    poc = poc - 1;
end
sirka_obl = x2-x1;
vyska_obl = y2-y1;
sir_vys = sirka_obl/vyska_obl;
```

6.2 Poměr vzdálenosti očí k šířce obličeje

Vzdálenost mezi levým a pravým okem je vypočítána klasicky Pythagorovou větou.

$$vzdálenost = \sqrt{(x\ levého\ oka - x\ pravého\ oka)^2 + (y\ levého\ oka - y\ pravého\ oka)^2} \quad (19).$$

Vzdálenost podle rovnice (19) se následně vydělí šířkou obličeje.
Kód z MATLABu:

```
vzd_oci = sqrt((stred2_oko2_x - stred2_oko1_x)^2 + (stred2_oko2_y -
stred2_oko1_y)^2);                  %vypocet vzdalenosti
vzd_oci_rel = vzd_oci/sirka_obl;    %pomer
```

6.3 Poměr vzdálenosti oko s ústy k šířce obličeje

Stejným způsobem jako v předchozím bodě se vypočítají dle rovnice (19) vzdálenosti levého a pravého oka s ústy. Tyto vzdálenosti se dále dají do poměru se šířkou hlavy.

6.4 Úhel mezi přímkami spojujících pravé a levé oko s ústy

Pro zjištění jednotlivých úhlů je třeba zjistit směrnice přímek spojujících jednotlivé body. Obecně lze zjistit směrový vektor dvou bodů A[x1,y1] a B[x2,y2] pomocí vztahu:

$$\vec{v}(x,y) = [x_2 - x_1, y_2 - y_1] \quad (20).$$

Ze směrového vektoru lze odvodit vektor normálový, který je ke směrovému kolmý.

$$\vec{n}(x,y) = [-\vec{v}(y), \vec{v}(x)] \quad (21).$$

Pomocí normálového vektoru z rovnice $\vec{n}(x,y) = [-\vec{v}(y), \vec{v}(x)]$ (21) lze vyjádřit směrnici přímky dosazením do následujícího vztahu:

$$k = \frac{-\vec{n}(x)}{\vec{n}(y)} \quad (22).$$

Směrnice přímky udává směr přímky vzhledem k x-ové ose. Jde o *tg(φ)*, kde *φ* je úhel mezi přímkou a kladnou poloosou x. Ze směrnice tedy získáme úhel inverzní operací k *tg*, kterou je *arctg*:

$$\varphi = arctg(k) \quad (23).$$

MATLAB používá souřadný systém trošku jiným způsobem než je běžně užíváno a to je třeba zohlednit. Hodnota x-ové souřadnice vzrůstá zleva doprava, hodnota souřadnice y vzrůstá shora dolů, jelikož vlastně přistupujeme k prvkům matice, jejichž řádkový index vzrůstá směrem dolů.[7]

Měření úhlu u úst provádí kód:

```
v13     =     [(stred2_M_x-stred2_oko1_x),(stred2_M_y-stred2_oko1_y)];
%smerovy vektor
n13 = [-v13(2), v13(1) ];      %normalovy vektor
k13 = (-n13(1))/n13(2);        %smernice primky
fi13 = (180*atan(k13))/pi;     %uhel sevreny primkou s kladnou poloosou
                               x prevedeny z radianu na stupne
v32 = [(stred2_oko2_x-stred2_M_x),(stred2_oko2_y-stred2_M_y)];
n32 = [-v32(2), v32(1) ];
k32 = (-n32(1))/n32(2);
fi32 = (180*atan(k32))/pi;
fi3 = 180 - (abs(fi32)+abs(fi13));  %uhel u ust
```

6.5 Úhel mezi přímkami spojující oči a levé/pravé oko s ústy

V předchozím případě úhlu u úst nabývají směrnice vždy hodnot se stejným znaménkem. Tomu však není stejně v tomto případě, kdy se znaménko směrnice mění v závislosti na tom, které z obou očí je výš než druhé. To je zohledněno v následujícím kódu:

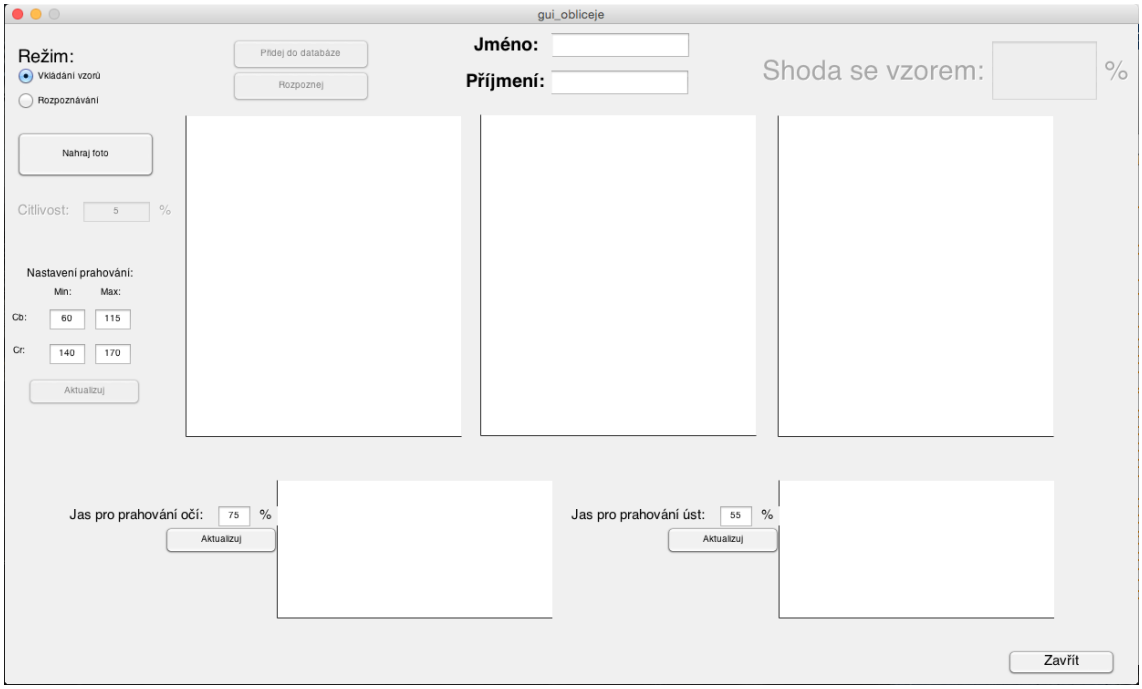
```
v12 = [(stred2_oko2_x-stred2_oko1_x),(stred2_oko2_y-stred2_oko1_y)];
n12 = [-v12(2), v12(1)];
k12 = (-n12(1))/n12(2);
fi12 = (180*atan(k12))/pi;
```

```
if k12 < 0
    fi12_pom = 180 - (abs(fi12) + abs(fi13));
    fi1 = abs((360 - 2*fi12_pom)/2);
else
    fi1 = abs(fi13 - fi12);
end
```

Výsledkem jsou tedy úhly u jednotlivých očí a úst.

7. Uživatelské rozhraní GUI

Pro vkládání a rozpoznávání osob je vytvořeno uživatelské rozhraní. Skrze toto rozhraní lze vkládat vzory do databáze, přiřadit ke vzoru jméno a upravovat hodnoty pro prahování jednotlivých obrazů. Druhým režimem je rozpoznání osoby z vloženého obrázku, který se porovná s již uloženými vzory. Je možno nastavit citlivost rozpoznávání a výsledek shody je potom uveden v procentuální shodě. Výsledkem rozpoznání je poté jméno osoby v příslušných textových polích.

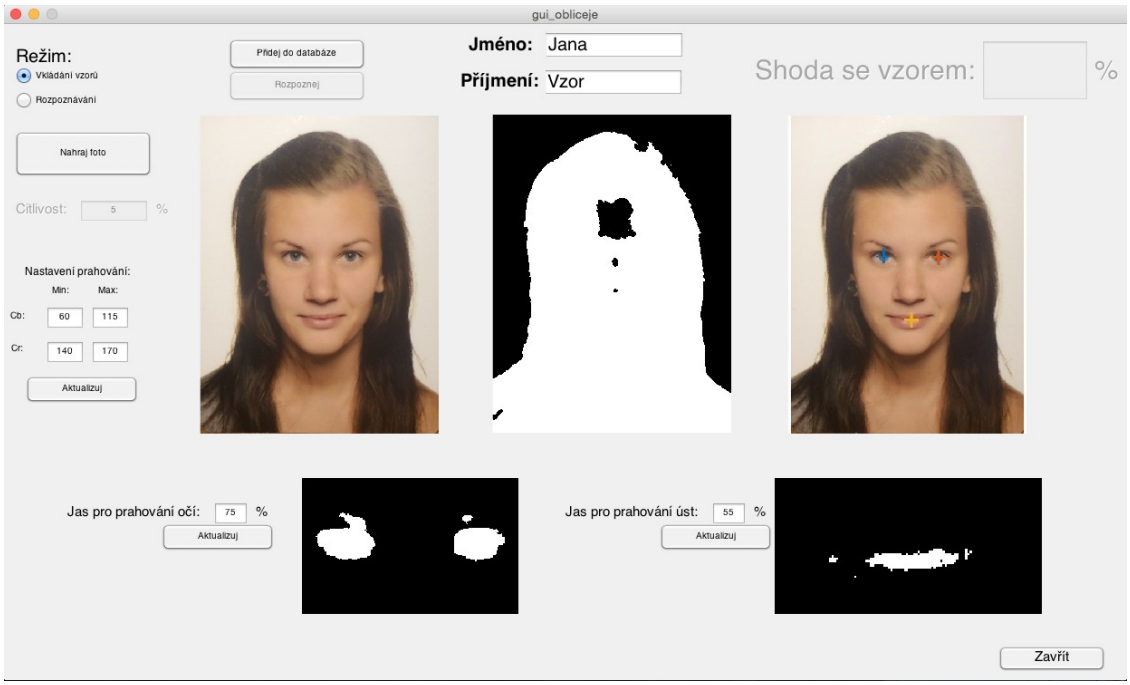


Obr. 24: Ukázka uživ. prostředí po spuštění

Režim se volí v levé horní části GUI (viz Obr. 24) a lze jej zvolit až po nahrání fotky do programu. Uživatelské rozhraní bylo řešeno za pomocí literatury [8]

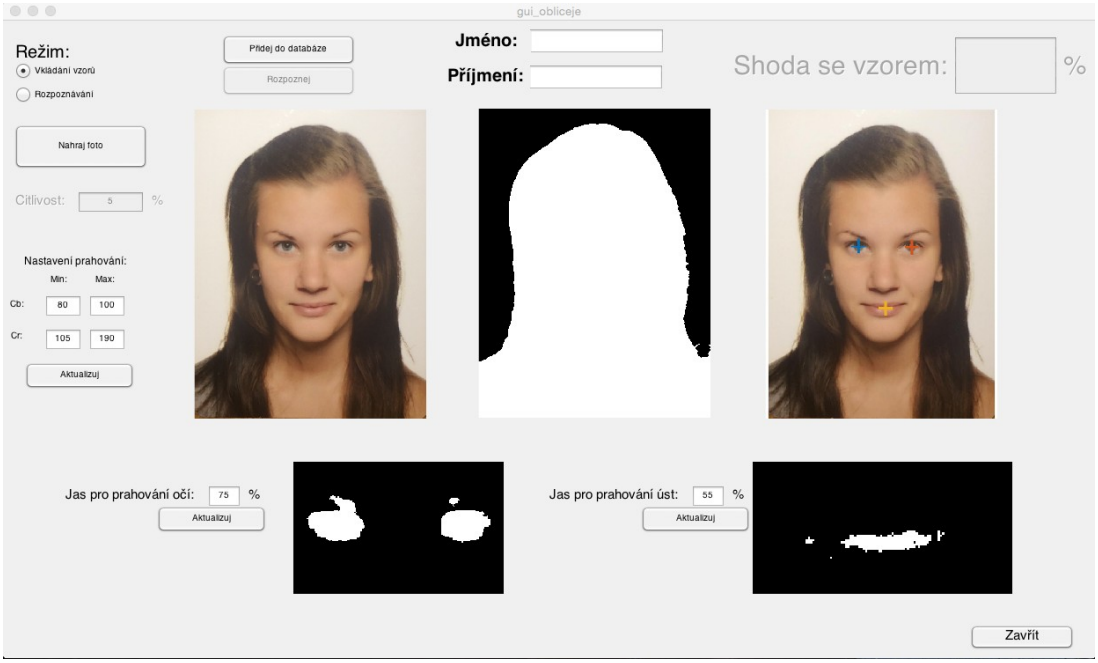
7.1 Nahrání fotografie

Po stisknutí tlačítka "Nahraj foto" se otevře okno pro výběr fotografie. Podporovány jsou fotografie ve formátu JPG. Poté se provede výpočet všech biometrických hodnot na základě obrázků prahovaných podle hodnot pro prahování v jednotlivých textových polích.



Obr. 25: GUI po nahrání fotografie

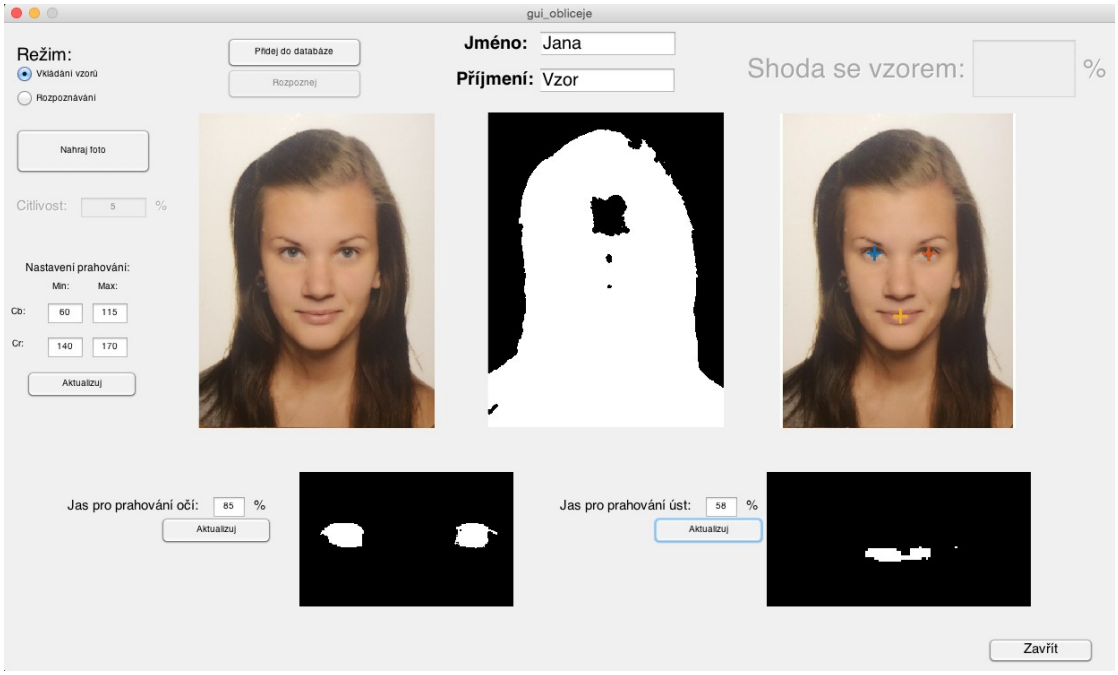
Při spuštění je přednastaveno prahování obrysu obličeje podle modrého a červeného chrominančního komponentu modelu YCbCr v rozmezích od 60 do 115 hodnoty modrého komponentu a od 140 do 170 pro komponent červený. Výstup prahování obrysu obličeje je zobrazen uprostřed. Hodnoty pro prahování tohoto obrysu se dají měnit v levé části v textových polích. Po každé změně je třeba stisknout tlačítko aktualizuj, které je pod těmito textovými polemi. Přepočítaný obraz podle nových mezí je poté zobrazen opět uprostřed pro kontrolu. To, že jsou uvnitř obrysu černá místa nemá vliv na hodnotu šířky a výšky obrysu, jelikož se hodnoty zjišťují "zvenčí", tedy od okraje obrazu k obrysu.



Obr. 26: Upravené kritéria prahování obrysu hlavy

Na Obr. 26 jsou správně upravena kritéria pro prahování obrysu. Hodnota modrého chrominančního komponentu je zde od 80 do 100 hodnoty. Červený komponent potom od 105 do 190.

V dolní části jsou potom zobrazeny prahované obrazy očí a úst. Hodnoty maximálních jasů pro prahování jsou zde přednastaveny na 75 % pro oči a 55 % pro ústa. Zde je také vhodné tyto prahovací kritéria upravit, dle kvality detekce.



Obr. 27: Upravené kritéria pro prahování očí a úst

Na Obr. 27 je upraveno prahování na 85 % maximální hodnoty jasu pro oči a 58 % max. hodnoty jasu pro ústa.
Celkovou detekci očí a úst vidíme na obrázku vpravo nahoře. Po stisknutí kteréhokoliv tlačítka aktualizuj je i tento obraz přepočítán a zobrazen aktualizovaný. Po každé změně jsou všechny biometrické vlastnosti (viz kapitola 6) přepočítávány a ukládány do "workspace" MATLABu pod určitými proměnnými. Pokud jsme spokojeni s pozicemi detekovaných očí a úst, máme na výběr ze dvou režimů, a to zda chceme tuto osobu uložit do databáze, nebo zjistit a jakou osobu se jedná, pokud už v databázi uložena někdy byla.

7.2 Režim "Vkládání vzorů"

Pro vkládání vzorů je třeba mít tento režim zvolen vlevo nahoře. Do textových polí v horní části uprostřed se napíše jméno a příjmení. Po stisknutí tlačítka "Přidej do databáze" se do souboru pod identifikačním číslem ID uloží jednotlivé biometrické hodnoty a jméno s příjmením.
Proces provádí následující kód:

```
EOF = size(data);        %konec souboru
if EOF(1) == 0           %vytvoreni ID
    ID = 1;
else
    ID = data(EOF(1))+1;
end
assignin('base','ID',ID)
data_oblicej(1,1) = ID;
data_oblicej(1,2) = sir_vys;
data_oblicej(1,3) = vzd_oci_rel;
data_oblicej(1,4) = vzd_oko1_pusa_rel;
data_oblicej(1,5) = vzd_oko2_pusa_rel;
data_oblicej(1,6) = fi1;
data_oblicej(1,7) = fi2;
data_oblicej(1,8) = fi3;
data(ID,:) = data_oblicej;
assignin('base','data',data);
data_jm(ID,:) = {ID,jmeno,prijm};
save('obliceje.mat','data','-ascii')
save('jmena','data_jm');
```

Biometrická data se společne s ID ukládají do matice, jména s příjmením se pod ID ukládají zvlášť do pole typu buňky. Situace je takto řešena z důvodu nemožnosti uložit do jedné matice proměnné typu "double" a "string".

7.3 Režim "Rozpoznávání"

Pokud už je vytvořena databáze se vzory, lze použít režim pro rozpoznávání. Rozpoznávání používá biometrická data vypočtená z dané fotografie a porovnává je postupně s daty z databáze s tolerancí, která je nastavena v textovém poli pod tlačítkem "Nahraj foto". Tato tolerance je na počátku nastavena na 5 % a lze ji měnit.

Ukázka kódu pro rozpoznání:

```
EOF = size(data);
if EOF == [0,0]
    konec = 1
else konec = max(data(:,1)) + 1;
end
k = 0;
global err
if konec > 1
for i = 1:konec-1
    if data(i,2) > (sir_vys-sir_vys*koef) && data(i,2) < (sir_vys+sir_vys*koef)
        if data(i,3) > (vzd_oci_rel-vzd_oci_rel*koef) && data(i,3) <
                                                    (vzd_oci_rel+vzd_oci_rel*koef)
            if data(i,4) > (vzd_oko1_pusa_rel-vzd_oko1_pusa_rel*koef) && data(i,4) <
                                                    (vzd_oko1_pusa_rel+vzd_oko1_pusa_rel*koef)
                if data(i,5) > (vzd_oko2_pusa_rel-vzd_oko2_pusa_rel*koef) && data(i,5) <
                                                    (vzd_oko2_pusa_rel+vzd_oko2_pusa_rel*koef)
                    if data(i,6) > (fi1-fi1*koef) && data(i,6) < (fi1+fi1*koef)
                        if data(i,7) > (fi2-fi2*koef) && data(i,7) < (fi2+fi2*koef)
                            if data(i,8) > (fi3-fi3*koef) && data(i,8) < (fi3+fi3*koef)
                                shoda(k+1) = i;
                                k = k+1;
                            end
                        end
                    end
                end
            end
        end
    end
end
end
if k == 0
    err = errordlg('Shoda nenalezena','Error');
    assignin('base','err',err);
end
shoda = shoda';
assignin('base','shoda',shoda)
```

Při procházení matice z databáze se tedy postupně berou biometrické hodnoty dle zadané tolerance. Pokud všechny atributy splní podmínku tolerance, do matice shody se uloží hodnota řádku na kterém došlo ke shodě. Pokud nedojde k žádné shodě, zobrazí se dialogové okno s informací, že shoda nebyla nalezena.

Výstupem tohoto rozpoznání je matice, ve které je jeden sloupec s hodnotami řádků na kterých došlo ke shodě. Tyto řádky jsou v matici "data" včetně ID, které je dále potřeba k přiřazení ke jménu. Následující kód provádí naplnění matice "prirad_id" řádky s biometrickými daty včetně ID:

```
n = length(shoda);
m = 0;
for i = 1 : n
    m = shoda(i,1);
    prir_id(i,:) = data(m,:);
end
```

V matici "prirad_id" jsou po provedení všechny udáje z databáze, ve kterých byla nalezena shoda. Následně se u všech hodnot provede výpočet odchylky od biometrické hodnoty z fotografie, která je momentálně nahrána v GUI a porovnávána. Aby se našla nejlepší shoda, odchylky se sečtou a ID u nejmenší výsledné odchylky je nejpravděpodobnější shoda hledané osoby.

```
prir_id(:,2) = abs(prir_id(:,2) - sir_vys);
prir_id(:,3) = abs(prir_id(:,3) - vzd_oci_rel);
prir_id(:,4) = abs(prir_id(:,4) - vzd_oko1_pusa_rel);
prir_id(:,5) = abs(prir_id(:,5) - vzd_oko2_pusa_rel);
prir_id(:,6) = abs(prir_id(:,6) - fi1);
prir_id(:,7) = abs(prir_id(:,7) - fi2);
prir_id(:,8) = abs(prir_id(:,8) - fi3);
for i = 1 : n
    nej_shoda(i,1) = prir_id(i,1);
    nej_shoda(i,2) = sum(prir_id(i,2:8));
end
nej_shoda = sortrows(nej_shoda,2);
shoda_id = nej_shoda(1,1);
```

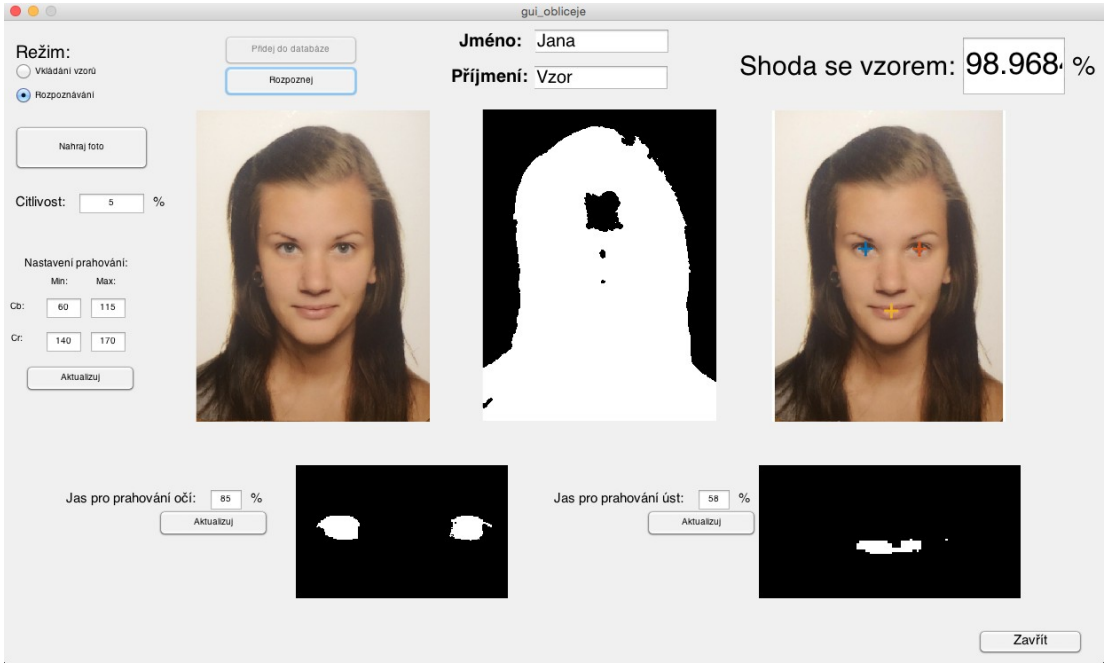
Podle ID poté přiřadíme z buňky se jmény odpovídající jméno rozpoznané osoby:

```
jmeno_rozp = data_jm(shoda_id,2);
prijm_rozp = data_jm(shoda_id,3);
```

Poslední fází rozpoznávání je výpočet procentuální shody se vzorem:

```
procento = prir_id(find(shoda_id),:);
procento(:,2) = 1 - (procento(:,2) / sir_vys);
procento(:,3) = 1 - (procento(:,3) / vzd_oci_rel);
procento(:,4) = 1 - (procento(:,4) / vzd_oko1_pusa_rel);
procento(:,5) = 1 - (procento(:,5) / vzd_oko2_pusa_rel);
procento(:,6) = 1 - (procento(:,6) / fi1);
procento(:,7) = 1 - (procento(:,7) / fi2);
procento(:,8) = 1 - (procento(:,8) / fi3);
shoda_procento = (sum(procento(1,2:8))/7)*100;
```

Procentuální shody jednotlivých parametrů se zprůměrují a tato výsledná hodnota se zobrazí v GUI vpravo nahoře v textovém poli "Shoda se vzorem"



Obr. 28: Ukázka režimu rozpoznávání

Na Obr. 28 je výsledek rozpoznávání se vzorem. V ukázce bylo použito stejné fotografie, ale s jiným prahováním než u vzoru, který byl uložen do databáze.

8. Testování na fotografiích

Všechny obrázky v této kapitole mají stejné rozmístění a jsou složeny z 5 obrazů ve dvou řadách:

- původní vstupní fotografie - horní řada vlevo
- prahovaný obrys hlavy - horní řada uprostřed
- obraz po detekci očí a úst - horní řada vpravo
- prahovaný obraz očí - dolní řada vlevo
- prahovaný obraz úst - dolní řada vpravo

Pod každým obrázkem je uvedeno, za jakých hodnot prahování proběhlo.

8.1 Úspěšné detekce a měření

Obr. 29: Detekce č.1

Rozmezí modrého chrominančního komponentu Cb	Rozmezí červeného chrominančního komponentu Cr	Práh jasu pro detekci očí v YCbCr [%]	Práh jasu pro detekci úst v YCbCr [%]
60 - 115	140 - 170	75	55

Tabulka 1: Parametry pro detekci č.1

Obr. 30: Detekce č.2

Rozmezí modrého chrominančního komponentu Cb	Rozmezí červeného chrominančního komponentu Cr	Práh jasu pro detekci očí v YCbCr [%]	Práh jasu pro detekci úst v YCbCr [%]
60 - 115	140 - 170	75	55

Tabulka 2: Parametry pro detekci č.2

Obr. 31: Detekce č.3

Rozmezí modrého chrominančního komponentu Cb	Rozmezí červeného chrominančního komponentu Cr	Práh jasu pro detekci očí v YCbCr [%]	Práh jasu pro detekci úst v YCbCr [%]
60 - 115	140 - 170	81	55

Tabulka 3: Parametry pro detekci č.3

Obr. 32: Detekce č.4

Rozmezí modrého chrominančního komponentu Cb	Rozmezí červeného chrominančního komponentu Cr	Práh jasu pro detekci očí v YCbCr [%]	Práh jasu pro detekci úst v YCbCr [%]
60 - 115	140 - 170	82	55

Tabulka 4: Parametry pro detekci č.4

V detekci na obrázku Obr. 32 uprostřed lze vidět lehký bílý pruh způsobený vlasy. Program tento bílý pixel vyhodnotí jako okraj obličeje. Oči a ústa jsou detekovány dobře, ale šířka obličeje bude v databázi zkreslená. Tento artefakt lze odstranit buď změnou rozmezí pro prahování nebo zvětšením elementu pro erozi (ve zdrojovém kódu programu).

Obr. 33: Detekce č.5

Rozmezí modrého chrominančního komponentu Cb	Rozmezí červeného chrominančního komponentu Cr	Práh jasu pro detekci očí v YCbCr [%]	Práh jasu pro detekci úst v YCbCr [%]
60 - 115	140 - 170	80	55

Tabulka 5: Parametry pro detekci č.5

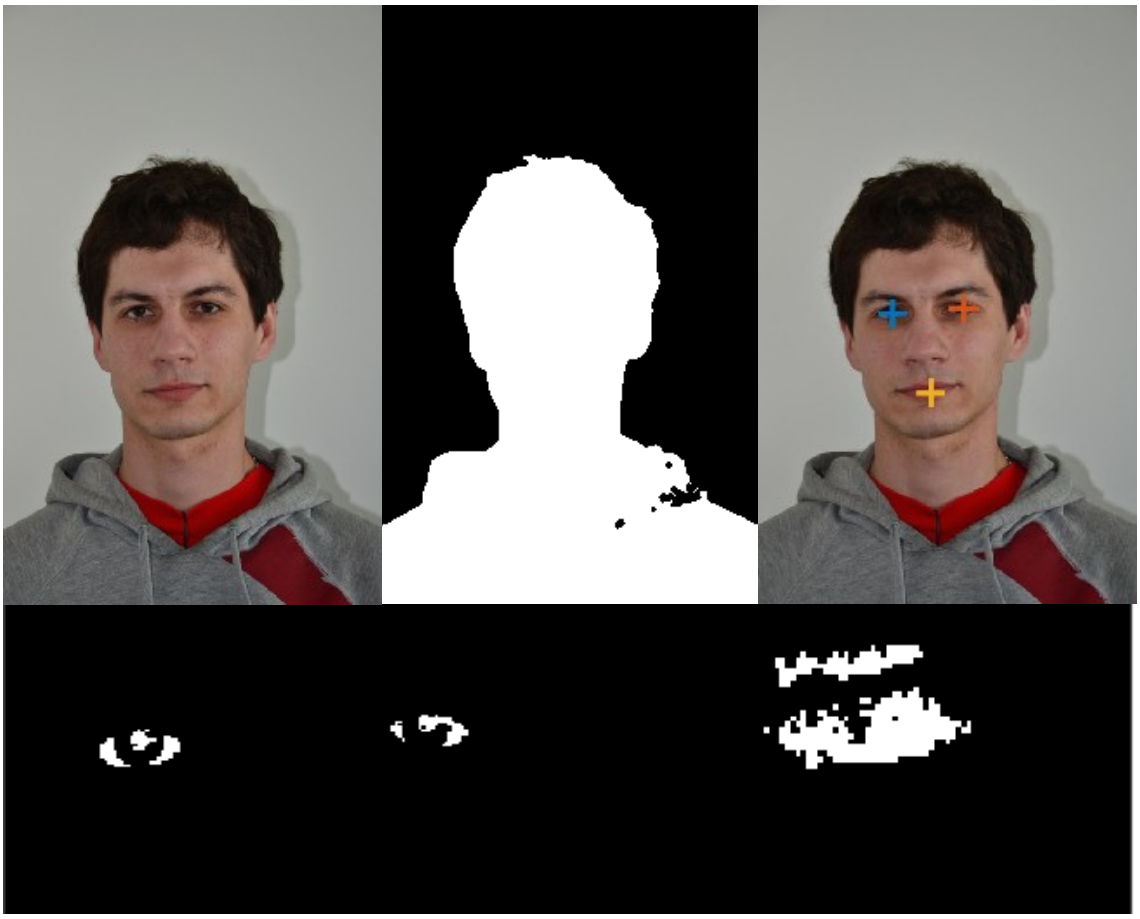
Na Obr. 33 v prostředním obrázku nahoře černá místa nijak neovlivní výsledky dat, jelikož jsou mimo oblast se kterou se dále pracuje.

Obr. 34: Detekce č.6

Rozmezí modrého chrominančního komponentu Cb	Rozmezí červeného chrominančního komponentu Cr	Práh jasu pro detekci očí v YCbCr [%]	Práh jasu pro detekci úst v YCbCr [%]
60 - 115	140 - 170	80	48

Tabulka 6: Parametry pro detekci č.6

Na Obr. 34 vpravo dole lze vidět ne úplně přesné prahování úst. Bílá oblast však není významně velká, a proto neovlivňuje razantně pozici detekovaných úst.



Obr. 35: Detekce č.7

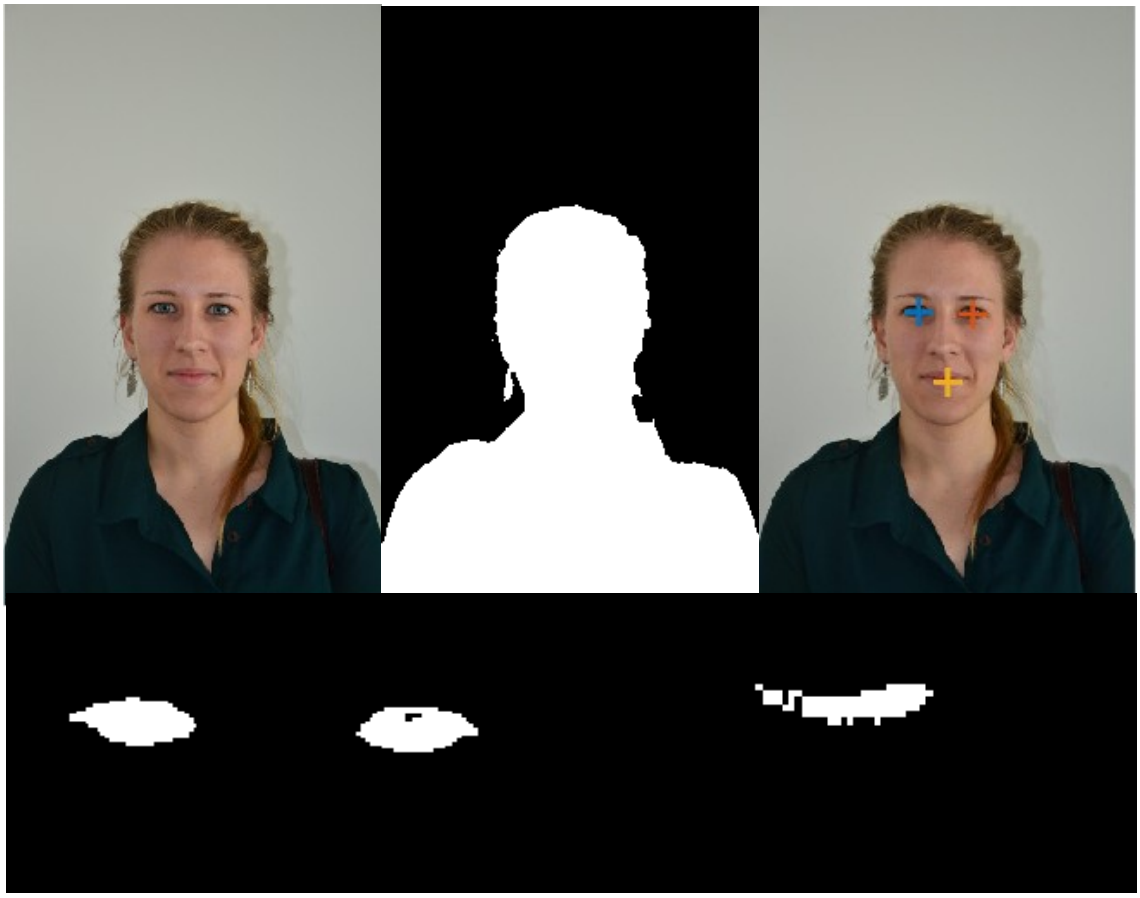
Rozmezí modrého chrominančního komponentu Cb	Rozmezí červeného chrominančního komponentu Cr	Práh jasu pro detekci očí v YCbCr [%]	Práh jasu pro detekci úst v YCbCr [%]
60 - 115	140 - 170	82	55

Tabulka 7: Parametry pro detekci č.7

Obr. 36: Detekce č.8

Rozmezí modrého chrominančního komponentu Cb	Rozmezí červeného chrominančního komponentu Cr	Práh jasu pro detekci očí v YCbCr [%]	Práh jasu pro detekci úst v YCbCr [%]
60 - 115	130 - 170	85	57

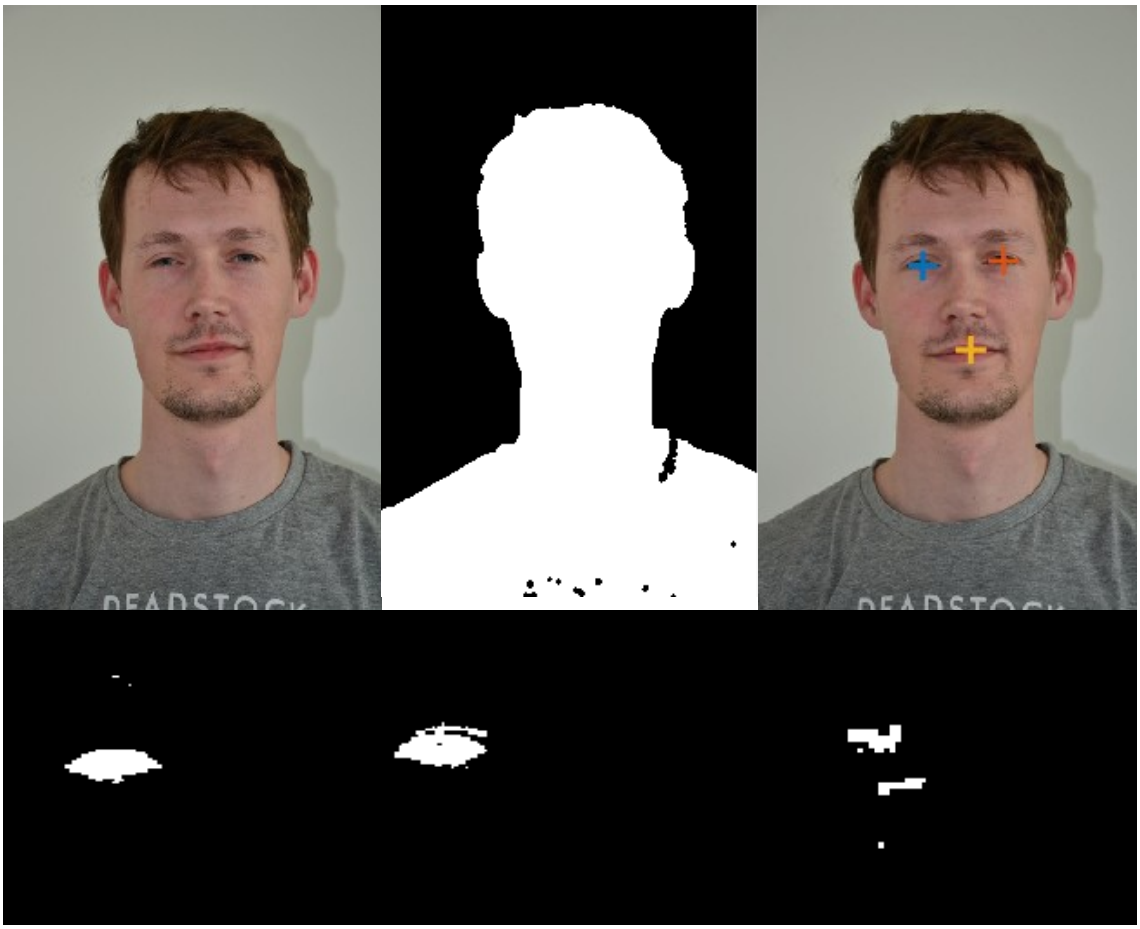
Tabulka 8: Parametry pro detekci č.8



Obr. 37: Detekce č.9

Rozmezí modrého chrominančního komponentu Cb	Rozmezí červeného chrominančního komponentu Cr	Práh jasu pro detekci očí v YCbCr [%]	Práh jasu pro detekci úst v YCbCr [%]
60 - 115	140 - 170	75	55

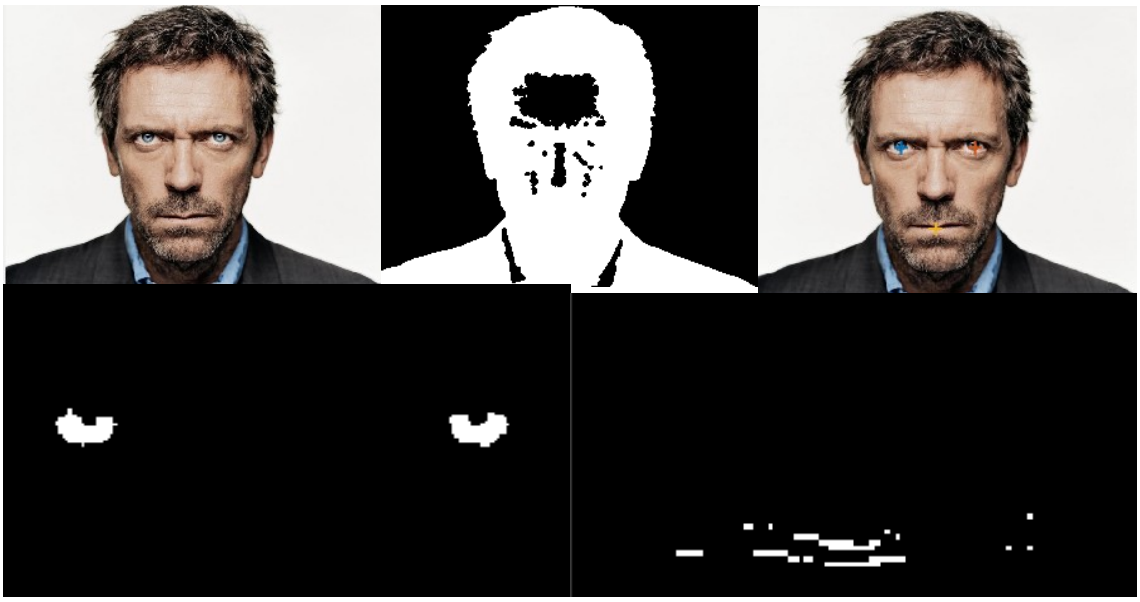
Tabulka 9: Parametry pro detekci č.9



Obr. 38: Detekce č.10

Rozmezí modrého chrominančního komponentu Cb	Rozmezí červeného chrominančního komponentu Cr	Práh jasu pro detekci očí v YCbCr [%]	Práh jasu pro detekci úst v YCbCr [%]
60 - 115	140 - 170	82	45

Tabulka 10: Parametry pro detekci č.10



Obr. 39: Detekce č.11

Rozmezí modrého chrominančního komponentu Cb	Rozmezí červeného chrominančního komponentu Cr	Práh jasu pro detekci očí v YCbCr [%]	Práh jasu pro detekci úst v YCbCr [%]
60 - 115	140 - 170	75	55

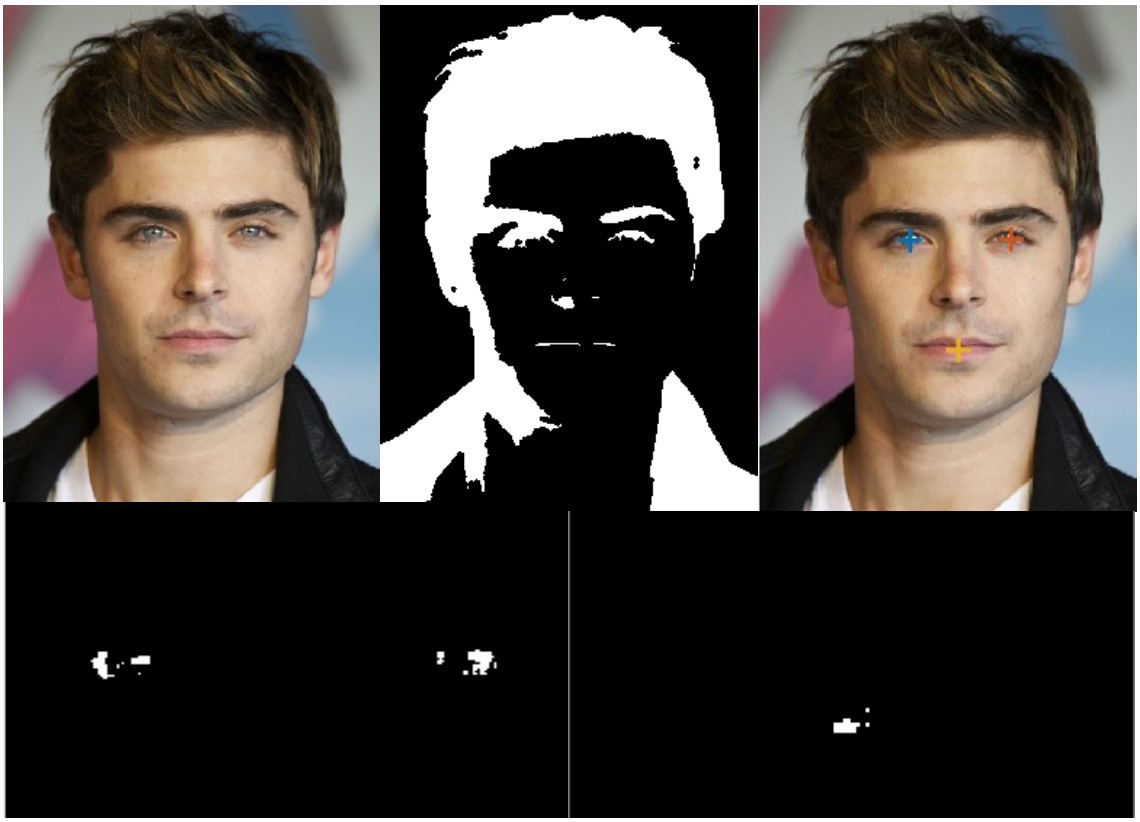
Tabulka 11: Parametry pro detekci č.11

Obr. 40: Detekce č.12

Rozmezí modrého chrominančního komponentu Cb	Rozmezí červeného chrominančního komponentu Cr	Práh jasu pro detekci očí v YCbCr [%]	Práh jasu pro detekci úst v YCbCr [%]
60 - 115	130 - 200	87	45

Tabulka 12: Parametry pro detekci č.12

U postavy na Obr. 40 muselo dojít k velkým upravám u hodnot prahování, což je způsobeno odstínem pleti. Ústa se zde špatně detekují z důsledku velkého odlesku.

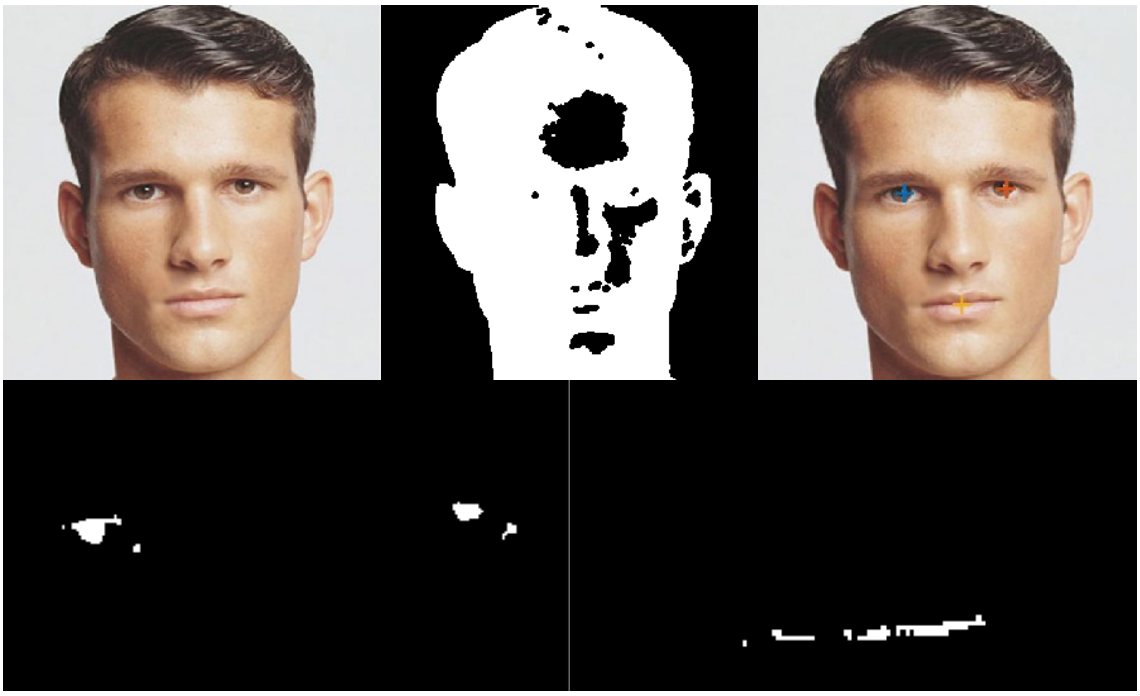


Obr. 41: Detekce č.13

Rozmezí modrého chrominančního komponentu Cb	Rozmezí červeného chrominančního komponentu Cr	Práh jasu pro detekci očí v YCbCr [%]	Práh jasu pro detekci úst v YCbCr [%]
50 - 90	170 - 210	75	55

Tabulka 13: Parametry pro detekci č.13

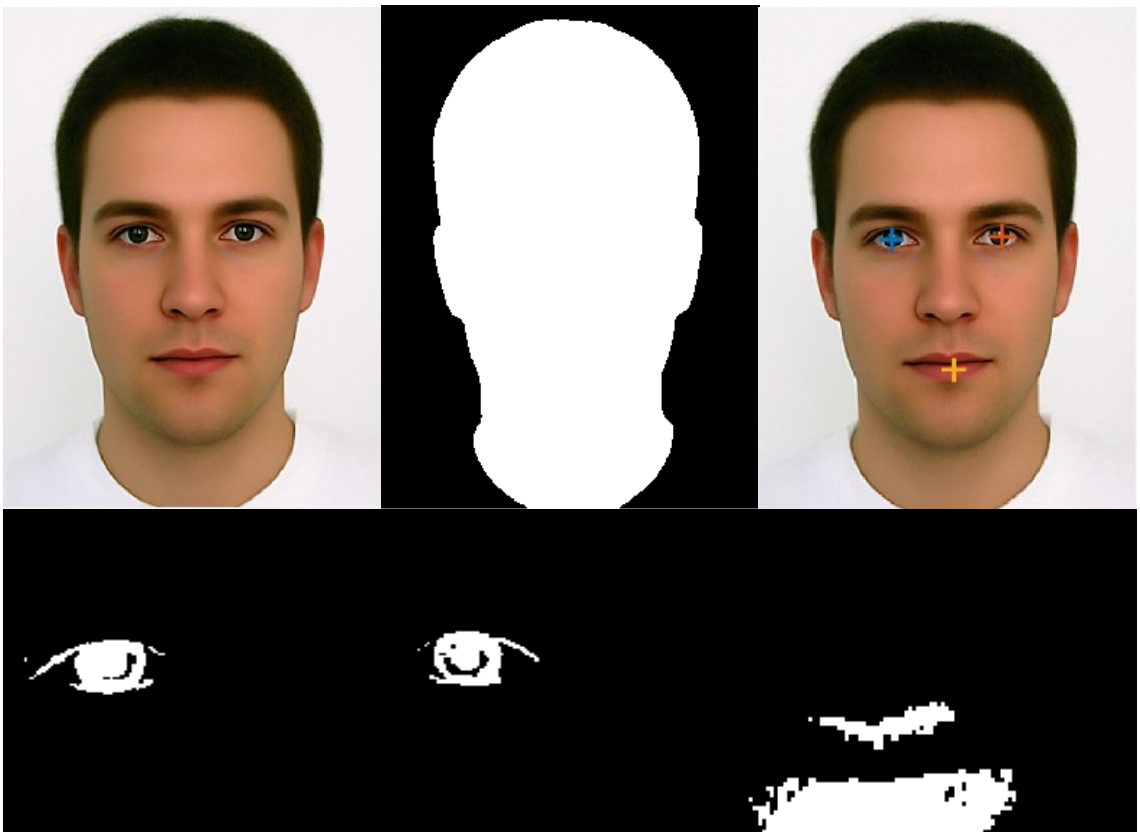
U Obr. 41 je třeba důkladně nastavit meze modrého a červeného chrominančního komponentu v důsledku barevného pozadí. Některé barvy v pozadí se v jednotlivých komponentech YCbCr modelu shodují s hodnotami lidské kůže.



Obr. 42: Detekce č.14

Rozmezí modrého chrominančního komponentu Cb	Rozmezí červeného chrominančního komponentu Cr	Práh jasu pro detekci očí v YCbCr [%]	Práh jasu pro detekci úst v YCbCr [%]
60 - 115	140 - 170	75	75

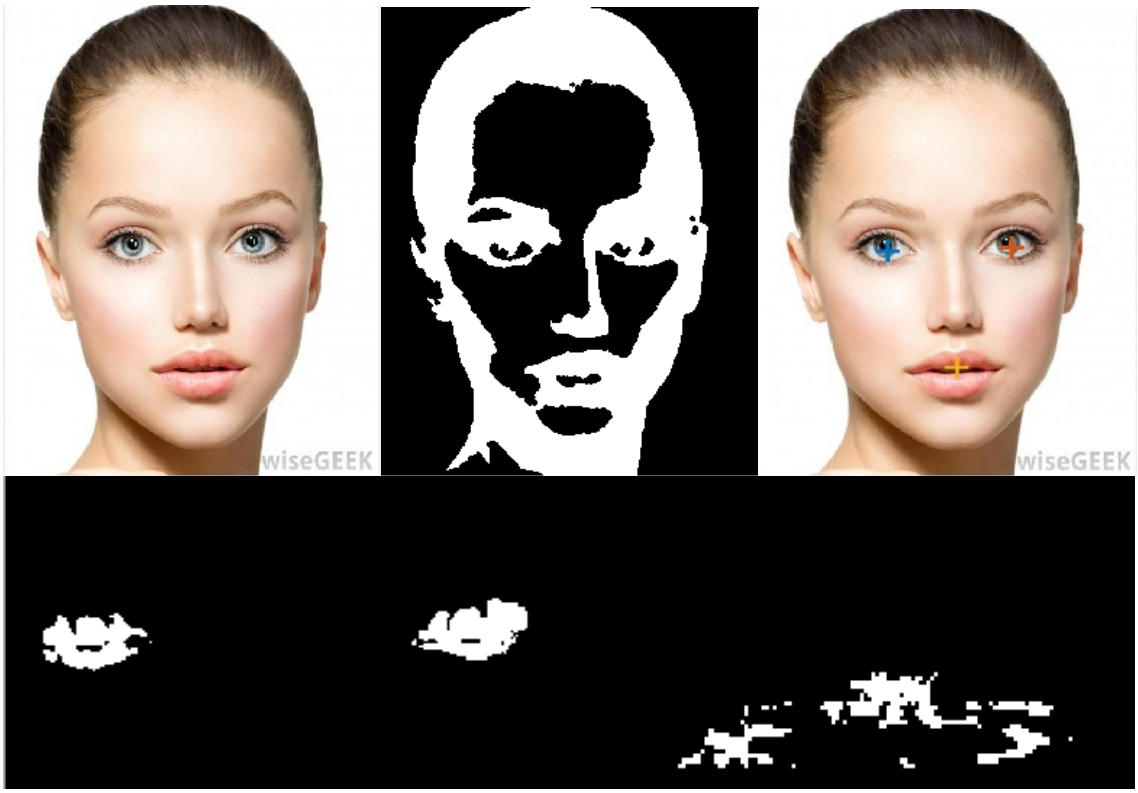
Tabulka 14: Parametry pro detekci č.14



Obr. 43: Detekce č.15

Rozmezí modrého chrominančního komponentu Cb	Rozmezí červeného chrominančního komponentu Cr	Práh jasu pro detekci očí v YCbCr [%]	Práh jasu pro detekci úst v YCbCr [%]
60 - 115	140 - 170	79	50

Tabulka 15: Parametry pro detekci č.15

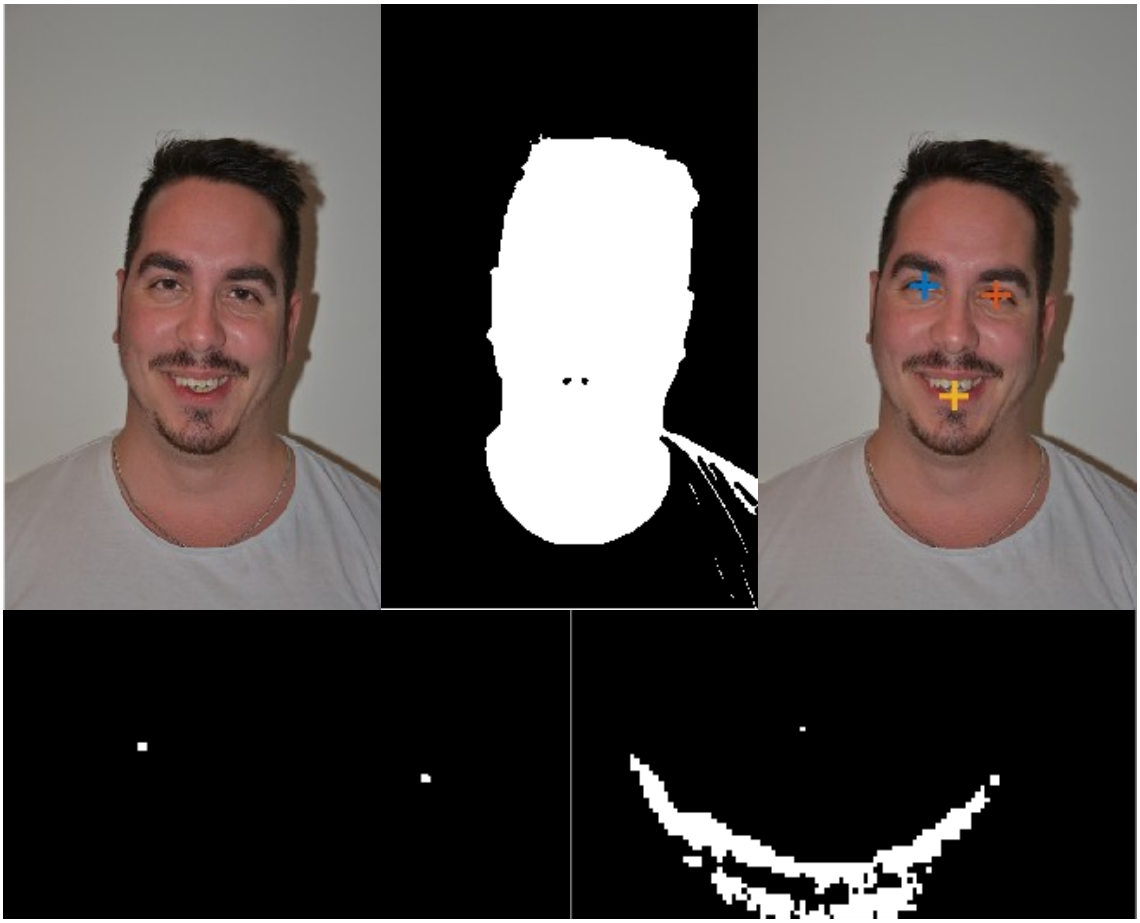


Obr. 44: Detekce č.16

Rozmezí modrého chrominančního komponentu Cb	Rozmezí červeného chrominančního komponentu Cr	Práh jasu pro detekci očí v YCbCr [%]	Práh jasu pro detekci úst v YCbCr [%]
60 - 115	140 - 170	83	50

Tabulka 16: Parametry pro detekci č.16

8.2 Detekce s chybami



Obr. 45: Detekce č.17

Rozmezí modrého chrominančního komponentu Cb	Rozmezí červeného chrominančního komponentu Cr	Práh jasu pro detekci očí v YCbCr [%]	Práh jasu pro detekci úst v YCbCr [%]
60 - 115	140 - 170	75	40

Tabulka 17: Parametry pro detekci č.17

U postavy na Obr. 45 nelze ani úpravami hodnot prahování úst docílit přesné detekce úst. Detekuje se spodní ret, který v důsledku úsměvu má výraznější jasová vlastnosti oproti rtu hornímu. Do prahování se vkládají také světlé části jako jsou zuby. Pro fotografie na detekci tedy jsou vhodné neutrální výrazy v obličeji.

Obr. 46: Detekce č.18

Rozmezí modrého chrominančního komponentu Cb	Rozmezí červeného chrominančního komponentu Cr	Práh jasu pro detekci očí v YCbCr [%]	Práh jasu pro detekci úst v YCbCr [%]
60 - 115	140 - 170	81	55

Tabulka 18: Parametry pro detekci č.18

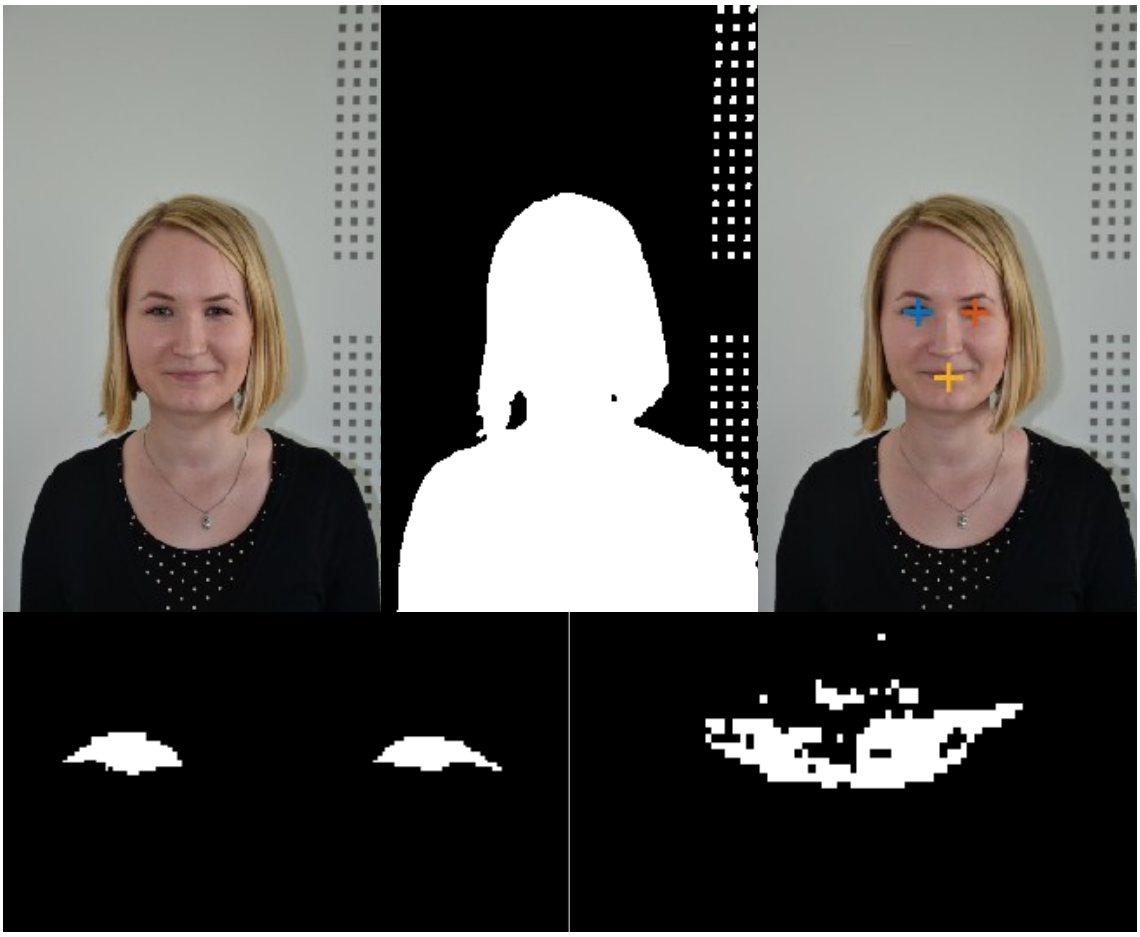
Na Obr. 46 v horní řadě uprostřed byl detekován stín, který zanáší chybu do hodnoty šířky a výšky obličeje, ke kterým se vztahují veškerá biometrická data. Proto tato fotografie není vhodná pro další rozpoznávání. Řešením je z fotografie odstranit stín v rohu například ořízntutím.

Obr. 47: Detekce č.19

Rozmezí modrého chrominančního komponentu Cb	Rozmezí červeného chrominančního komponentu Cr	Práh jasu pro detekci očí v YCbCr [%]	Práh jasu pro detekci úst v YCbCr [%]
60 - 115	140 - 170	80	48

Tabulka 19: Parametry pro detekci č.19

Na Obr. 47 dochází k podobnému problému jako v předchozím případě u Obr. 46. Opět zde špatné světelné podmínky způsobí nekorektní prahování obrazu obrysu hlavy, což má za důsledek špatně změřené hodnoty šířky a výšky hlavy. Řešením zde opět oříznutí obrazu na nezbytně nutnou velikost, pokud možno beze stínu. Další možností je pořídit fotografii znovu s lepšími světelnými podmínkami.

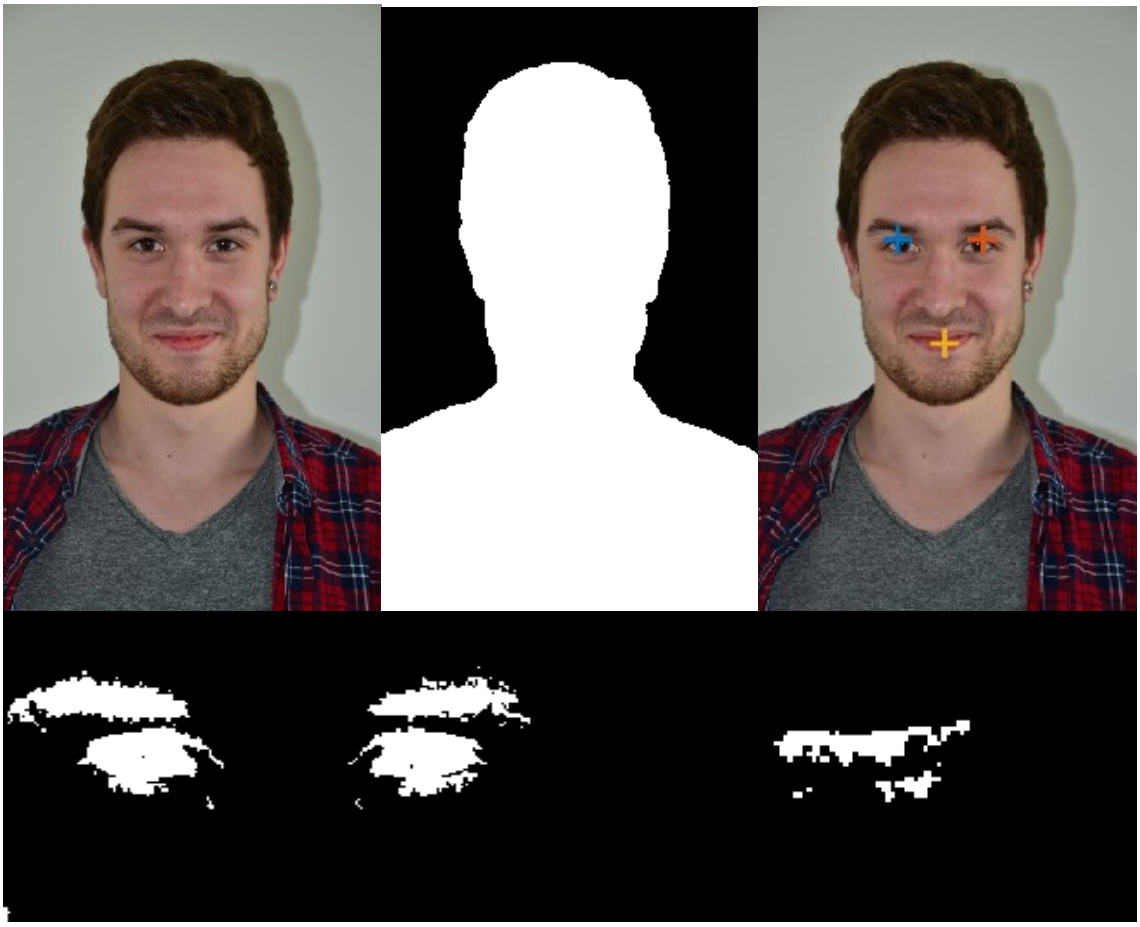


Obr. 48: Detekce č.20

Rozmezí modrého chrominančního komponentu Cb	Rozmezí červeného chrominančního komponentu Cr	Práh jasu pro detekci očí v YCbCr [%]	Práh jasu pro detekci úst v YCbCr [%]
60 - 115	140 - 170	70	55

Tabulka 20: Parametry pro detekci č.20

V případě Obr. 48 je nevhodně zvolené pozadí pro fotografii. Tmavá místa jsou při prahování nesprávně rozpoznána jako popředí a na tomto pozadí dochází k chybnému měření biometrických dat.

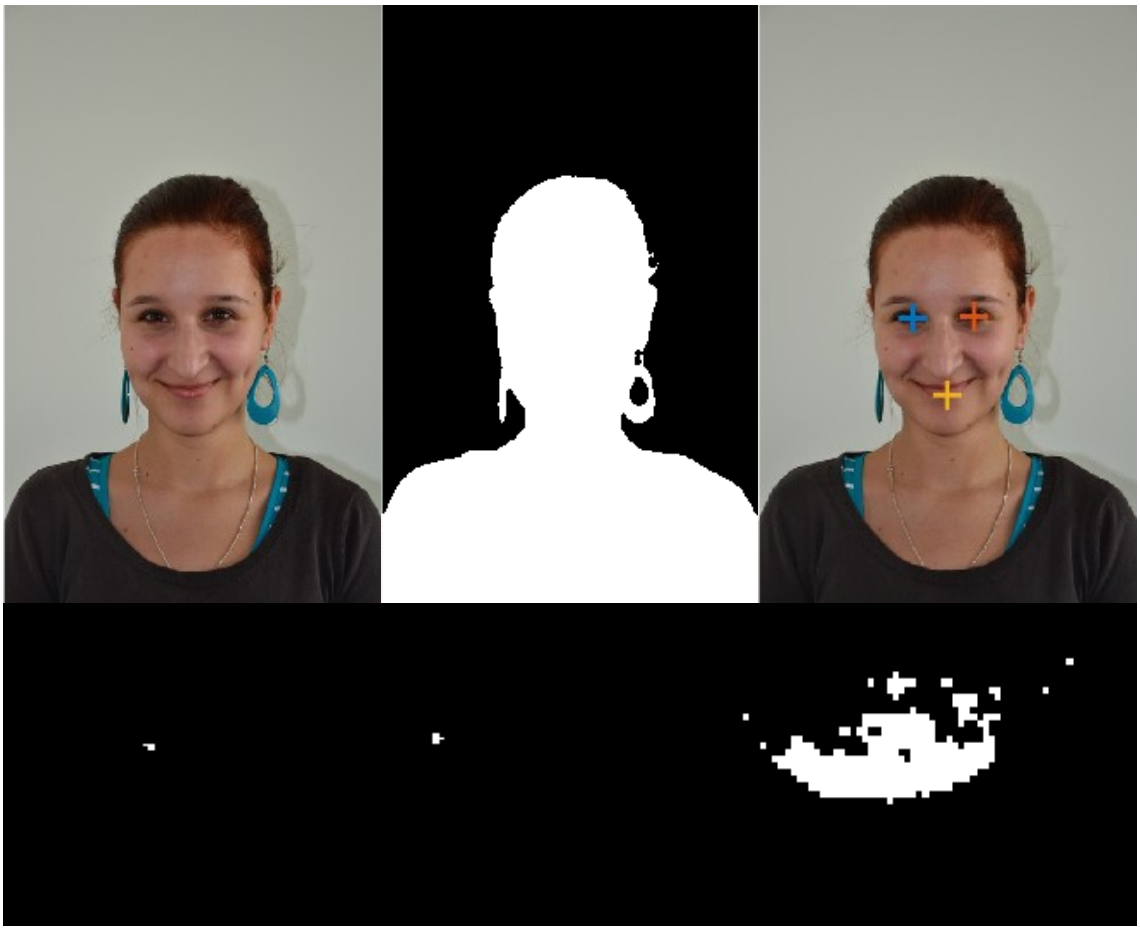


Obr. 49: Detekce č.21

Rozmezí modrého chrominančního komponentu Cb	Rozmezí červeného chrominančního komponentu Cr	Práh jasu pro detekci očí v YCbCr [%]	Práh jasu pro detekci úst v YCbCr [%]
60 - 115	140 - 170	75	55

Tabulka 21: Parametry pro detekci č.21

Na Obr. 49 je znázorněn případ nevhodně zvolené úrovně prahování očí. Elimovat tuto chybu lze zvýšením hodnoty pro prahování oblasti očí tak, aby zustaly detekovány pouze oči bez obočí.



Obr. 50: Detekce č.22

Rozmezí modrého chrominančního komponentu Cb	Rozmezí červeného chrominančního komponentu Cr	Práh jasu pro detekci očí v YCbCr [%]	Práh jasu pro detekci úst v YCbCr [%]
60 - 115	140 - 170	65	45

Tabulka 22: Parametry pro detekci č.22

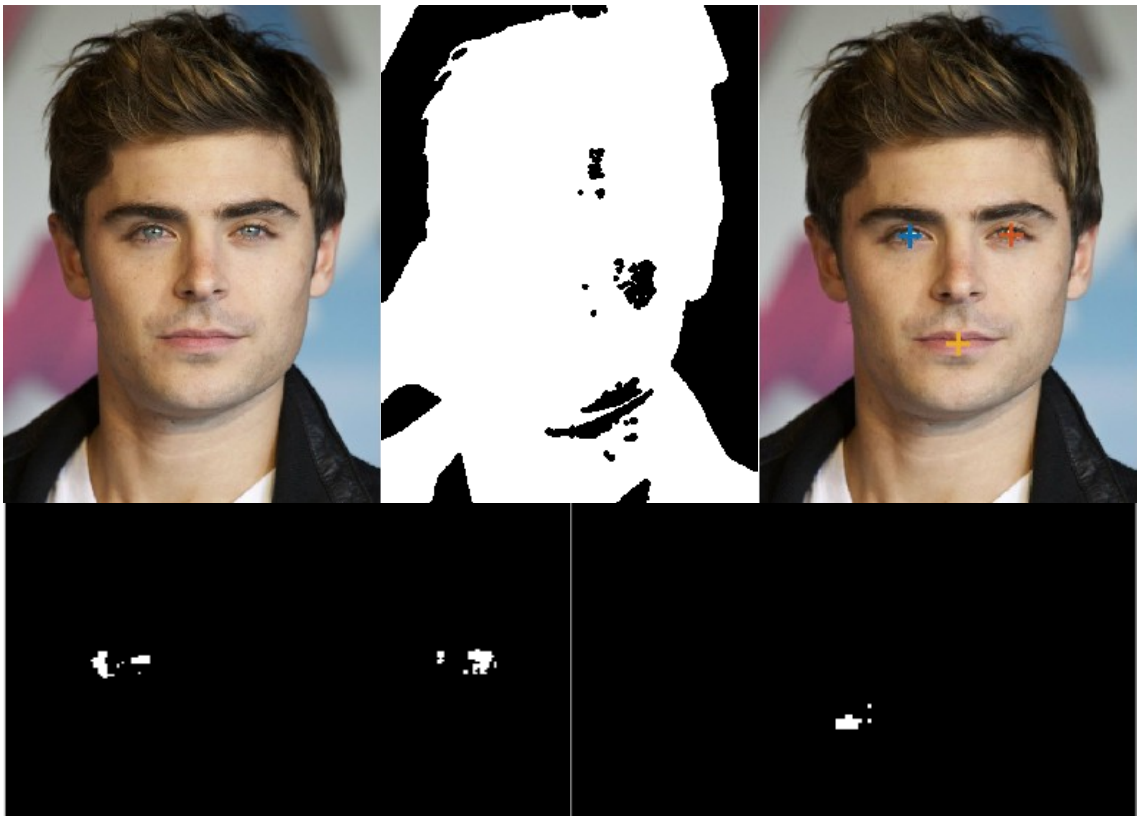
Na Obr. 50 je případ, kdy pozadí i prahování je v pořádku, avšak chybu do měření můžou zanést náušnice, které ční mimo šířku obličeje jako je tomu u tohoto obrázku. Osoba focená pro rozpoznávání tímto programem by měla ideálně být bez těchto módních doplňku, které mohou meření hodnot zkreslit.

Obr. 51: Detekce č.23

Rozmezí modrého chrominančního komponentu Cb	Rozmezí červeného chrominančního komponentu Cr	Práh jasu pro detekci očí v YCbCr [%]	Práh jasu pro detekci úst v YCbCr [%]
60 - 115	120 - 190	78	56

Tabulka 23: Parametry pro detekci č.23

Na Obr. 51 je nevhodnými vlasy způsobena chyba měření výšky. Tato chyba nemá vliv na ostatní parametry, jako například špatně určená šířka hlavy, ale je v tomto případě obtížnější najít shodu této osoby v databázi, pokud na jiné fotografii bude mít razantně odlišný účes. To je způsobeno hlavně také tím, že osoba není focena přesně zepředu, ale je mírně natočena. Ideální fotografie by měla být pořízená přesně zepředu.

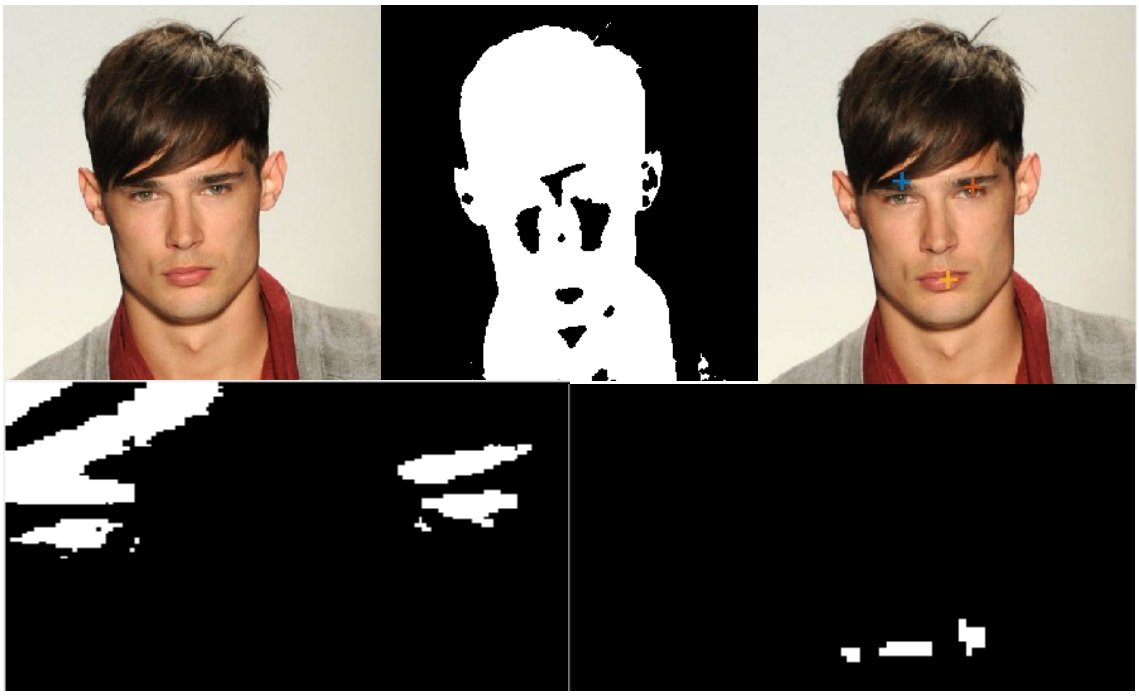


Obr. 52: Detekce č.24

Rozmezí modrého chrominančního komponentu Cb	Rozmezí červeného chrominančního komponentu Cr	Práh jasu pro detekci očí v YCbCr [%]	Práh jasu pro detekci úst v YCbCr [%]
60 - 115	140 - 170	75	55

Tabulka 24: Parametry pro detekci č.24

Obr. 52 je příkladem špatného pozadí. Takto špatně prahovaný obraz obrysu hlavy (horní řada uprostřed) lze v mnoha případech zachránit zvolením vhodných rozmezím pro prahování jako například u detekce č.13 na Obr. 41.



Obr. 53: Detekce č.25

Rozmezí modrého chrominančního komponentu Cb	Rozmezí červeného chrominančního komponentu Cr	Práh jasu pro detekci očí v YCbCr [%]	Práh jasu pro detekci úst v YCbCr [%]
60 - 115	140 - 170	80	55

Tabulka 25: Parametry pro detekci č.25

Na Obr. 53 dochází ke špatné detekci očí vlivem výskytů vlasů v bezprostřední blízosti očí.

8.3 Požadavky na ideální detekci

Pro to, aby detekce očí a úst proběhla v pořádku je třeba mít vhodné vstupní fotografie/obrazy. Ideální fotografie by měla splňovat co nejvíce z těchto požadavků:

- dobré nasvětlení (beze stínů)
- vlasy by neměly zakrývat oblast očí
- vlasy by měly co nejméně zasahovat do okolí
- bez módních doplňků jako např. velké náušnice apod.
- bílé rovnoměrně nasvícené pozadí
- osoba by měla být zpříma, bez natáčení hlavy v jakémkoli směru
- vážný výraz (bez úsměvu)

Čím více kritérií z předchozí výčtu je splněno, tím kvalitnější detekci je program schopen vykonat. Některé atributy lze doladit při nahrávání fotografií skrze uživatelské rozhraní změnou prahovacích kritérií, některé úpravou fotky v externím programu na PC.

9. Závěr

Cílem této diplomové práce bylo vytvoření rozpoznávacího systému pro analýzu osob na základě obrazového signálu. Program byl realizován v programovém prostředí MATLAB.

Vytvořený program umožňuje detekci biometrických dat u analyzovaných osob z fotografií. Měřenými parametry jsou zde na základě detekce očí a úst vzdálenosti mezi jednotlivýma očima a ústy. K tomu je ještě měřena šířka a výška hlavy, ke kterým se vzdálenosti oči a úst vztahují. Dalším parametrem je měření úhlů u jednotlivých očí a úst, které svírají pomyslné přímky spojující detekované body.

Vytvořil jsem v MATLABu uživatelské prostředí, pomocí kterého lze nahrávat fotografie z PC. Po nahrání fotografie dochází k detekcím a měřením biometrických hodnot. Po vypočtení všech parametrů má uživatel možnost zvolit ze dvou režimů. Jedním režimem je přidávání osob do databáze, druhým je rozpoznávání osoby na základě fotografie. Pro režim rozpoznávání je zde informace o procentuální shodě s hledaným vzorem (pokud v databázi existuje).

V rámci ostatních diplomových prací byly řešeny další projekty, které využívají biometrická data člověka k identifikaci. Například analýza zvukového signálu, analýza otisku prstu a jiné. Moje práce v kombinaci s jinými pracemi by mohla být využita společně pro spolehlivější a přesnější systém pro rozpoznávání a identifikaci osob skrze výpočetní techniku.

10. Zdroje

[1] DRAHANSKÝ, Martin a Filip ORSÁG. *Biometrie*. 1. vyd. [Brno: M. Drahanský], 2011, 294 s. ISBN 978-80-254-8979-6.

[2] SHANNON, C.E. *A Mathematical Theory of Communication*. The Bell System Technical Journal. 1948, číslo 27, s. 279-423

[3] HÁJOVSKÝ, Radovan, Radka PUSTKOVÁ a František KUTÁLEK. *Zpracování obrazu v měřící a řídící technice*. Vyd. 1. Ostrava: Vysoká škola báňská - Technická univerzita Ostrava, 2012, 1 DVD-ROM. ISBN 978-80-248-2596-0.

[4] Computer Vision System Toolbox. *Humusoft*. [online]. [cit. 2015-01-19]. Dostupné z: http://www.humusoft.cz/produkty/matlab/aknihovny/computer-vision/

[5] Image Processing Toolbox. *Humusoft*. [online]. [cit. 2015-01-20]. Dostupné z: http://www.humusoft.cz/produkty/matlab/aknihovny/image/

[6] *Matematická morfologie. Dilatace. Eroze. Otevření a uzavření. Hit or miss transformace. Základní morfologické algoritmy na binárních a šedotónových obrazech.* [online]. [cit. 2015-05-05]. Dostupné z: http://blade1.ft.tul.cz/elearning/Media/File/5/123/P7.pdf

[7] KRYNICKÝ, Martin. *Směrnicový tvar rovnice přímky* [online]. [cit. 2015-05-05]. Dostupné z: http://www.ucebnice.krynicky.cz/Matematika/07_Analyticka_geometrie/3_Analyticka_geometrie_v_rovine/7309_Smernicovy_tvar_rovnice_primky.pdf

[8] ZAPLATÍLEK, Karel a Bohuslav, DOŇAR.*MATLAB: tvorba uživatelských aplikací.* 1.vyd. Praha: BEN, 2004. 215 s. ISBN 80-7300-133-0

[9] BOVIK, Alan C. *Handbook of Image and Video Processing.*1. vyd. San Diego: Academic press, c2000. 891 s. ISBN 0-12-119790-5

[10] HALOUNOVÁ, Lena. *Zpracování obrazových dat.* 1. vyd. Praha: Česká technika - nakladatelství ČVUT, 2009. 102 s. ISBN 978-80-01-04253-3

[11] GOLDBERGER, J. a H. GREENSPAN. 2006. Context-based segmentation of image sequences. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* [online]. s. 463-468 [cit. 2015-03-03]. DOI: 10.1109/TPAMI.2006.47. ISSN 0162-8828. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1580490

[12] FENG, Wenying a Feng GAO. 2007. *Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing: SNPD 2007 : [in conjunction with 3rd ACIS International Workshop on Self-Assembling Networks : SAWN 2007] : proceedings : 30 July-1 August 2007, Qingdao, China* [online]. Los Alamitos, CA: IEEE Computer Society [cit. 2015-02-03]. ISBN 0769529097.

[13] IGARASHI, Masaki, Akira MIZUNO a Masayuki IKEBE. 2013. Accuracy improvement of histogram-based image filtering. *2013 IEEE International Conference on Image Processing* [online]. IEEE, : 1217-1221 [cit. 2015-05-07]. DOI:

10.1109/ICIP.2013.6738251. ISBN 978-1-4799-2341-0. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6738251

[14] LEE, Sooyeon, Youngshin KWAK, Youn KIM, Sehyeok PARK a Jaehyun KIM. 2012. Contrast-preserved chroma enhancement technique using YCbCr color space. *IEEE Transactions on Consumer Electronics* [online]. IEEE, **58**(2): 641-645 [cit. 2015-01-07]. DOI: 10.1109/TCE.2012.6227471. ISBN 978-1-4799-2341-0. ISSN 0098-3063. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6227471

[15] *Eighth World Congress on Intelligent Control and Automation, WCICA 2010, 7-9 July 2010* [online]. 2010. Piscataway, N.J.: IEEE [cit. 2014-12-12]. ISBN 9781424467129.

[16] MADHULIKA,, Divakar YADAV, MADHURIMA, Pritee GUPTA, Gurpreet KAUR, Jyoti SINGH, Mallika GANDHI a Ajeet SINGH. 2013. Implementing Edge Detection for Medical Diagnosis of a Bone in Matlab. *2013 5th International Conference on Computational Intelligence and Communication Networks* [online]. IEEE, : 270-274 [cit. 2015-05-07]. DOI: 10.1109/CICN.2013.64. ISBN 978-0-7695-5069-5. Dostupné z: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6657998

11. Seznam příloh

Přílohy v textu:

Přílohy CD:

1) Diplomová práce "Rozpoznávací systém pro analýzu osob na základě obrazového signálu" ve formátu PDF/A
2) Složka "původní fotografie"
3) Složka "detekované fotografie"
4) Složka "GUI": - gui_obliceje.m
   - gui_obliceje.fig
   - bio.m
   - detekce.m
   - prirad_id.m
   - rozp.m
   - uloz.m
   - obliceje.mat
   - data_jm.mat

Přílohy

A. Zdrojové kódy

I. Grafické uživatelské rozhraní - soubor gui_obliceje.m

```
function varargout = gui_obliceje(varargin)
% GUI_OBLICEJE MATLAB code for gui_obliceje.fig
%         GUI_OBLICEJE, by itself, creates a new GUI_OBLICEJE or raises the
existing
%      singleton*.
%
%      H = GUI_OBLICEJE returns the handle to a new GUI_OBLICEJE or the handle
to
%      the existing singleton*.
%
%      GUI_OBLICEJE('CALLBACK',hObject,eventData,handles,...) calls the local
%          function named CALLBACK in GUI_OBLICEJE.M with the given input
arguments.
%
%         GUI_OBLICEJE('Property','Value',...) creates a new GUI_OBLICEJE or
raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before gui_obliceje_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to gui_obliceje_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help gui_obliceje

% Last Modified by GUIDE v2.5 28-Apr-2015 17:47:20

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @gui_obliceje_OpeningFcn, ...
                   'gui_OutputFcn',  @gui_obliceje_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
```

```
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before gui_obliceje is made visible.
function gui_obliceje_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to gui_obliceje (see VARARGIN)
global data
global stat
global data_jm

stat = 0;
data = load('obliceje.mat','-ascii');
data_jm = load('jmena');
data_jm = data_jm.data_jm;

assignin ('base','data',data)
assignin ('base','data_jm',data_jm)

set(handles.radio1,'Value',1)
set(handles.radio2,'Value',0)
set(handles.axes_puv,'xtick',[],'ytick',[])
set(handles.axes_prah,'xtick',[],'ytick',[])
set(handles.axes_det,'xtick',[],'ytick',[])
set(handles.axes_oci,'xtick',[],'ytick',[])
set(handles.axes_usta,'xtick',[],'ytick',[])
set(handles.btn_pridej,'enable','off')
set(handles.btn_rozp,'enable','off')
set(handles.edit1,'string',5,'enable','off')
set(handles.text3,'enable','off')
set(handles.text4,'enable','off')
set(handles.text18,'enable','off')
set(handles.text19,'enable','off')
set(handles.edit_proc,'enable','off')
set(handles.edit_cb_min,'String',60)
set(handles.edit_cb_max,'String',115)
set(handles.edit_cr_min,'String',140)
set(handles.edit_cr_max,'String',170)
set(handles.edit_oci,'String',75)
set(handles.edit_usta,'String',55)
set(handles.btn_akt,'enable','off')
```

```
% Choose default command line output for gui_obliceje
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes gui_obliceje wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = gui_obliceje_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;


% --- Executes on button press in btn_load.
function btn_load_Callback(hObject, eventdata, handles)
% hObject    handle to btn_load (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clear Down
global cb_min
global cb_max
global cr_min
global cr_max

cb_min = str2double(get(handles.edit_cb_min,'String'));
assignin('base','cb_min',cb_min)
cb_max = str2double(get(handles.edit_cb_max,'String'));
assignin('base','cb_max',cb_max)
cr_min = str2double(get(handles.edit_cr_min,'String'));
assignin('base','cr_min',cr_min)
cr_max = str2double(get(handles.edit_cr_max,'String'));
assignin('base','cr_max',cr_max)
global jas_oci
global jas_usta
jas_oci = str2double(get(handles.edit_oci,'String'))/100;
assignin('base','jas_oci',jas_oci)
jas_usta = str2double(get(handles.edit_usta,'String'))/100;
assignin('base','jas_usta',jas_usta)

set(handles.btn_akt,'enable','on')
set(handles.edit_jmeno,'String','')
set(handles.edit_prijm,'String','')
set(handles.edit_proc,'String','')
```

```
set(handles.text18,'enable','off')
set(handles.text19,'enable','off')
set(handles.edit_proc,'enable','off')

global obr
[FileName,PathName] = uigetfile('*.jpg','Select an image');
obr = imread(strcat(PathName,FileName));
%obr = imrotate(obr,90);
[r_obr s_obr ch] = size(obr);
pomer_stran = r_obr/s_obr;
if (1.45 > pomer_stran) && (pomer_stran > 1.05)
    obr = imresize(obr,[640 480]);
elseif (1.7 > pomer_stran) && (pomer_stran > 1.45)
    obr = imresize(obr,[768 480]);
elseif pomer_stran > 1.7
    obr = imresize(obr,[853 480]);
elseif (1.05 > pomer_stran) && (pomer_stran > 0.95)
    obr = imresize(obr,[480 480]);
elseif (0.95 > pomer_stran) && (pomer_stran > 0.685)
    obr = imresize(obr,[480 640]);
elseif (0.685 > pomer_stran) && (pomer_stran > 0.59)
    obr = imresize(obr,[480 768]);
elseif 0.59 > pomer_stran
    obr = imresize(obr,[480 853]);
end
assignin('base','obr',obr)

[r_zmen s_zmen ch_zmen] = size(obr);

imshow(obr, 'parent', handles.axes_puv)
run('detekce.m')

imshow(obr_det_final, 'parent', handles.axes_det)
imshow(Down, 'parent', handles.axes_prah)

run('bio.m')

global vyrez_oci
global vyrez_M
imshow(vyrez_oci, 'parent', handles.axes_oci)
imshow(vyrez_M, 'parent', handles.axes_usta)


global stat
stat = 1;

if get(handles.radio1,'Value') == 1
    set(handles.btn_pridej,'enable','on')
else
    set(handles.btn_rozp,'enable','on')
end
```

```
% --- Executes on button press in radio1.
function radio1_Callback(hObject, eventdata, handles)
% hObject    handle to radio1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radio1
set(handles.radio1,'Value',1)
set(handles.radio2,'Value',0)
set(handles.edit1,'enable','off')
set(handles.text3,'enable','off')
set(handles.text4,'enable','off')
global stat
if stat ==1
    set(handles.btn_pridej,'enable','on')
    set(handles.btn_rozp,'enable','off')
end

% --- Executes on button press in radio2.
function radio2_Callback(hObject, eventdata, handles)
% hObject    handle to radio2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hint: get(hObject,'Value') returns toggle state of radio2
set(handles.radio1,'Value',0)
set(handles.radio2,'Value',1)
set(handles.edit1,'enable','on')
set(handles.text3,'enable','on')
set(handles.text4,'enable','on')
global stat
if stat ==1
    set(handles.btn_pridej,'enable','off')
    set(handles.btn_rozp,'enable','on')
end


function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%           str2double(get(hObject,'String')) returns contents of edit1 as a
double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if           ispc           &&           isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%           str2double(get(hObject,'String')) returns contents of edit2 as a
double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if           ispc           &&           isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in btn_pridej.
function btn_pridej_Callback(hObject, eventdata, handles)
% hObject    handle to btn_pridej (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

global jmeno
global prijm
global data_jm
jmeno = get(handles.edit_jmeno,'string');
prijm = get(handles.edit_prijm,'string');
assignin('base','jmeno',jmeno);
assignin('base','prijm',prijm);
```

```
run('uloz.m')
pause(0.01)
data_jm = load('jmena');
data_jm = data_jm.data_jm;
assignin('base','data_jm',data_jm);

guidata(hObject, handles);

% --- Executes on button press in btn_rozp.
function btn_rozp_Callback(hObject, eventdata, handles)
% hObject    handle to btn_rozp (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global koef
global shoda_id
global jmeno_rozp
global prijm_rozp
global shoda_procento
global err

koef = str2double(get(handles.edit1,'string'));
koef = koef * 0.01;
assignin('base','koef',koef)

run('rozp.m')

set(handles.edit_jmeno,'string',jmeno_rozp);
set(handles.edit_prijm,'string',prijm_rozp);

set(handles.text18,'enable','on')
set(handles.text19,'enable','on')
set(handles.edit_proc,'enable','on')
set(handles.edit_proc,'String', num2str(shoda_procento))

if err ~= 0
    set(err, 'WindowStyle', 'modal');
    uiwait(err);
end

function edit_cb_min_Callback(hObject, eventdata, handles)
% hObject    handle to edit_cb_min (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_cb_min as text
%        str2double(get(hObject,'String')) returns contents of edit_cb_min as
a double


% --- Executes during object creation, after setting all properties.
function edit_cb_min_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to edit_cb_min (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if           ispc           &&           isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit_cb_max_Callback(hObject, eventdata, handles)
% hObject    handle to edit_cb_max (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_cb_max as text
%         str2double(get(hObject,'String')) returns contents of edit_cb_max as
a double


% --- Executes during object creation, after setting all properties.
function edit_cb_max_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_cb_max (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if           ispc           &&           isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit_cr_min_Callback(hObject, eventdata, handles)
% hObject    handle to edit_cr_min (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_cr_min as text
%         str2double(get(hObject,'String')) returns contents of edit_cr_min as
a double


% --- Executes during object creation, after setting all properties.
function edit_cr_min_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_cr_min (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if           ispc           &&           isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit_cr_max_Callback(hObject, eventdata, handles)
% hObject    handle to edit_cr_max (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_cr_max as text
%         str2double(get(hObject,'String')) returns contents of edit_cr_max as
a double


% --- Executes during object creation, after setting all properties.
function edit_cr_max_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_cr_max (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if           ispc           &&           isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in btn_akt.
function btn_akt_Callback(hObject, eventdata, handles)
% hObject    handle to btn_akt (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

cb_min = str2double(get(handles.edit_cb_min,'String'));
assignin('base','cb_min',cb_min)
cb_max = str2double(get(handles.edit_cb_max,'String'));
assignin('base','cb_max',cb_max)
cr_min = str2double(get(handles.edit_cr_min,'String'));
assignin('base','cr_min',cr_min)
cr_max = str2double(get(handles.edit_cr_max,'String'));
assignin('base','cr_max',cr_max)
```

```
run('detekce.m')
pause(0.01)
imshow(obr_det_final, 'parent', handles.axes_det)
imshow(Down, 'parent', handles.axes_prah)
run('bio.m')

global stat
if stat == 1
    if get(handles.radio1,'Value') == 1
     set(handles.btn_pridej,'enable','on')
    else
        set(handles.btn_rozp,'enable','on')
    end
end



function edit_jmeno_Callback(hObject, eventdata, handles)
% hObject    handle to edit_jmeno (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_jmeno as text
%        str2double(get(hObject,'String')) returns contents of edit_jmeno as a
double


% --- Executes during object creation, after setting all properties.
function edit_jmeno_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_jmeno (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if           ispc           &&           isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit_prijm_Callback(hObject, eventdata, handles)
% hObject    handle to edit_prijm (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_prijm as text
%        str2double(get(hObject,'String')) returns contents of edit_prijm as a
double
```

```
% --- Executes during object creation, after setting all properties.
function edit_prijm_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_prijm (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if           ispc           &&           isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in btn_konec.
function btn_konec_Callback(hObject, eventdata, handles)
% hObject    handle to btn_konec (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(gcf)

function edit_oci_Callback(hObject, eventdata, handles)
% hObject    handle to edit_oci (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_oci as text
%          str2double(get(hObject,'String')) returns contents of edit_oci as a
double


% --- Executes during object creation, after setting all properties.
function edit_oci_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_oci (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if           ispc           &&           isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end



function edit_usta_Callback(hObject, eventdata, handles)
% hObject    handle to edit_usta (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit_usta as text
%         str2double(get(hObject,'String')) returns contents of edit_usta as a
double


% --- Executes during object creation, after setting all properties.
function edit_usta_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_usta (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if           ispc           &&           isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in btn_oci.
function btn_oci_Callback(hObject, eventdata, handles)
% hObject    handle to btn_oci (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global jas_oci
jas_oci = str2double(get(handles.edit_oci,'String'))/100;
assignin('base','jas_oci',jas_oci)

run('detekce.m')
run('bio.m')
imshow(vyrez_oci, 'parent', handles.axes_oci)
imshow(obr_det_final, 'parent', handles.axes_det)


% --- Executes on button press in btn_usta.
function btn_usta_Callback(hObject, eventdata, handles)
% hObject    handle to btn_usta (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global jas_usta
jas_usta = str2double(get(handles.edit_usta,'String'))/100;
assignin('base','jas_usta',jas_usta)

run('detekce.m')
run('bio.m')
imshow(vyrez_M, 'parent', handles.axes_usta)
imshow(obr_det_final, 'parent', handles.axes_det)
```

```
function edit_proc_Callback(hObject, eventdata, handles)
% hObject    handle to edit_proc (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit_proc as text
%        str2double(get(hObject,'String')) returns contents of edit_proc as a
double


% --- Executes during object creation, after setting all properties.
function edit_proc_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit_proc (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if          ispc          &&          isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

II. Výpočet biometrických údajů - soubor bio.m

```
global sir_vys
global vzd_oci_rel
global vzd_oko1_pusa_rel
global vzd_oko2_pusa_rel
global fi1
global fi2
global fi3
%% vzdalenost oci a pusy
vzd_oci   =   sqrt((stred2_oko2_x   -   stred2_oko1_x)^2   +   (stred2_oko2_y   -
stred2_oko1_y)^2);
vzd_oko1_pusa   =   sqrt((stred2_M_x   -   stred2_oko1_x)^2   +   (stred2_M_y   -
stred2_oko1_y)^2);
vzd_oko2_pusa   =   sqrt((stred2_M_x   -   stred2_oko2_x)^2   +   (stred2_M_y   -
stred2_oko2_y)^2);

%% uhly mezi primkami z bodu oci a ust
%uhel u ust
v13 = [(stred2_M_x-stred2_oko1_x),(stred2_M_y-stred2_oko1_y)];
n13 = [-v13(2), v13(1) ];
k13 = (-n13(1))/n13(2);
fi13 = (180*atan(k13))/pi;

v32 = [(stred2_oko2_x-stred2_M_x),(stred2_oko2_y-stred2_M_y)];
n32 = [-v32(2), v32(1) ];
k32 = (-n32(1))/n32(2);
fi32 = (180*atan(k32))/pi;
```

```
fi3 = 180 - (abs(fi32)+abs(fi13));

%uhel u oko1
v12 = [(stred2_oko2_x-stred2_oko1_x),(stred2_oko2_y-stred2_oko1_y)];
n12 = [-v12(2), v12(1)];
k12 = (-n12(1))/n12(2);
fi12 = (180*atan(k12))/pi;

if k12 < 0
    fi12_pom = 180 - (abs(fi12) + abs(fi13));
    fi1 = abs((360 - 2*fi12_pom)/2);
else
    fi1 = abs(fi13 - fi12);
end

%uhel u oko2
if k12 > 0
    fi32_pom = 180 - (abs(fi12) + abs(fi32));
    fi2 = abs((360 - 2*fi32_pom)/2);
else
    fi2 = abs(fi32 - fi12);
end

%% sirka obliceje
orez1 = [0 0 s stred2_M_y];
obl_sirka = imcrop(Down,orez1);
% figure()
% imshow(obl_sirka)

[y2end, x2end] = size(obl_sirka);
x1 = 1;
x2 = x2end;
y1 = 1;
y2 = y2end;

%zjisteni obl zleva
poc = 1;
while (sum(obl_sirka(:,poc)) == 0)
    x1 = x1 + 1;
    poc = poc + 1;
end

%zjisteni obl zhora
poc = 1;
while (sum(obl_sirka(poc,:)) == 0)
    y1 = y1 + 1;
    poc = poc + 1;
end

%zjisteni obl zprava
```

```
poc = x2;
while (sum(obl_sirka(:,poc)) == 0)
    x2 = x2 - 1;
    poc = poc - 1;
end
sirka_obl = x2-x1;
vyska_obl = y2-y1;

% relativni vzdalenosti


sir_vys = sirka_obl/vyska_obl;
vzd_oci_rel = vzd_oci/sirka_obl;
vzd_oko1_pusa_rel = vzd_oko1_pusa/sirka_obl;
vzd_oko2_pusa_rel = vzd_oko2_pusa/sirka_obl;
% fi1
% fi2
% fi3
assignin('base','sir_vys',sir_vys)
assignin('base','vzd_oci_rel',vzd_oci_rel)
assignin('base','vzd_oko1_pusa_rel',vzd_oko1_pusa_rel)
assignin('base','vzd_oko2_pusa_rel',vzd_oko2_pusa_rel)
assignin('base','fi1',fi1)
assignin('base','fi2',fi2)
assignin('base','fi3',fi3)
```

III. Přiřazení ID ke jménu a výsledku shody - soubor prirad_id.m

```
global shoda
global data
global data_jm
global shoda_id
global jmeno_rozp
global prijm_rozp
global shoda_procento

n = length(shoda);
m = 0;
for i = 1 : n
    m = shoda(i,1);
    prir_id(i,:) = data(m,:);
end

prir_id(:,2) = abs(prir_id(:,2) - sir_vys);
prir_id(:,3) = abs(prir_id(:,3) - vzd_oci_rel);
prir_id(:,4) = abs(prir_id(:,4) - vzd_oko1_pusa_rel);
prir_id(:,5) = abs(prir_id(:,5) - vzd_oko2_pusa_rel);
prir_id(:,6) = abs(prir_id(:,6) - fi1);
prir_id(:,7) = abs(prir_id(:,7) - fi2);
prir_id(:,8) = abs(prir_id(:,8) - fi3);
```

```
for i = 1 : n
    nej_shoda(i,1) = prir_id(i,1);
    nej_shoda(i,2) = sum(prir_id(i,2:8));
end

nej_shoda = sortrows(nej_shoda,2);
shoda_id = nej_shoda(1,1);
assignin('base','shoda_id',shoda_id);

jmeno_rozp = data_jm(shoda_id,2);
prijm_rozp = data_jm(shoda_id,3);

procento = prir_id(find(shoda_id),:);
procento(:,2) = 1 - (procento(:,2) / sir_vys);
procento(:,3) = 1 - (procento(:,3) / vzd_oci_rel);
procento(:,4) = 1 - (procento(:,4) / vzd_oko1_pusa_rel);
procento(:,5) = 1 - (procento(:,5) / vzd_oko2_pusa_rel);
procento(:,6) = 1 - (procento(:,6) / fi1);
procento(:,7) = 1 - (procento(:,7) / fi2);
procento(:,8) = 1 - (procento(:,8) / fi3);

shoda_procento = (sum(procento(1,2:8))/7)*100;
```

IV. Rozpoznávání z databáze - soubor rozp.m

```
global data
global EOF
global sir_vys
global vzd_oci_rel
global vzd_oko1_pusa_rel
global vzd_oko2_pusa_rel
global fi1
global fi2
global fi3
global koef
global shoda

EOF = size(data);
if EOF == [0,0]
    konec = 1
else konec = max(data(:,1)) + 1;
end

%koef = koef_gui * 0.01;    %koeficient presnosti
k = 0;
shoda = [];
global err
if konec > 1
for i = 1:konec-1
    if     data(i,2)     >     (sir_vys-sir_vys*koef)     &&     data(i,2)     <
(sir_vys+sir_vys*koef)
```

```
        if   data(i,3)   >   (vzd_oci_rel-vzd_oci_rel*koef)   &&   data(i,3)   <
(vzd_oci_rel+vzd_oci_rel*koef)
            if   data(i,4)   >   (vzd_oko1_pusa_rel-vzd_oko1_pusa_rel*koef)   &&
data(i,4) < (vzd_oko1_pusa_rel+vzd_oko1_pusa_rel*koef)
                if  data(i,5)  >  (vzd_oko2_pusa_rel-vzd_oko2_pusa_rel*koef)  &&
data(i,5) < (vzd_oko2_pusa_rel+vzd_oko2_pusa_rel*koef)
                    if    data(i,6)    >    (fi1-fi1*koef)    &&    data(i,6)    <
(fi1+fi1*koef)
                        if   data(i,7)   >   (fi2-fi2*koef)   &&   data(i,7)   <
(fi2+fi2*koef)
                            if  data(i,8)  >  (fi3-fi3*koef)  &&  data(i,8)  <
(fi3+fi3*koef)
                                shoda(k+1) = i;
                                k = k+1;
                            end
                        end
                    end
                end
            end
        end
    end
end
end
if k == 0
    err = errordlg('Shoda nenalezena','Error');
    assignin('base','err',err);
end
shoda = shoda';
assignin('base','shoda',shoda)

run('prirad_id.m')
```

V. Ukládání do databáze - soubor uloz.m

```
global data
global jmeno
global prijm
global sir_vys
global vzd_oci_rel
global vzd_oko1_pusa_rel
global vzd_oko2_pusa_rel
global fi1
global fi2
global fi3
global data_oblicej
global ID
global data_jm


EOF = size(data);
if EOF(1) == 0
```

```
    ID = 1;
else
    ID = data(EOF(1))+1;
end
assignin('base','ID',ID)

data_oblicej(1,1) = ID;
data_oblicej(1,2) = sir_vys;
data_oblicej(1,3) = vzd_oci_rel;
data_oblicej(1,4) = vzd_oko1_pusa_rel;
data_oblicej(1,5) = vzd_oko2_pusa_rel;
data_oblicej(1,6) = fi1;
data_oblicej(1,7) = fi2;
data_oblicej(1,8) = fi3;
%assignin('base','data_oblicej',data_oblicej)
data(ID,:) = data_oblicej;
assignin('base','data',data);

data_jm(ID,:) = {ID,jmeno,prijm};

save('obliceje.mat','data','-ascii')
save('jmena','data_jm');
```

VI. Detekce obličeje, očí, úst - soubor detekce.m

```
global obr
%% detektor obliceje
det_obl = vision.CascadeObjectDetector;              %definice detektoru
tvar = vision.ShapeInserter('BorderColor','White'); %definice tvaru

%figure()
ob = step(det_obl,obr);              %aplikace detektoru na puvodni obr.
obr_ob   =   step(tvar,obr,int32(ob));      %vykresleni   tvaru   dle   souradnic
detekovaneho obliceje
%imshow(obr_ob)                       %zobrazeni tvaru v puv.obr.
%% orezani obliceje
%figure()
orez = imcrop(obr, ob);      %orezani obr.
%imshow(orez)

%% prahovani obliceje v celem obr
ycbcr1 = rgb2ycbcr(obr);
y1 = double(ycbcr1(:,:,1));
Cb1 = double(ycbcr1(:,:,2));
Cr1 = double(ycbcr1(:,:,3));
[r1 s1 v1] = size(ycbcr1);

clear oblic;
global cb_min
global cb_max
global cr_min
```

```
global cr_max
i = 1;
j = 1;
for i = 1:r1
    for j = 1:s1
        if Cb1(i,j)<cb_min || y1(i,j)>cb_max
            Cb2(i,j) = 0;
        else Cb2(i,j) = 1;
        end
        if Cr1(i,j)<cr_min || y1(i,j)>cr_max
            Cr2(i,j) = 0;
        else Cr2(i,j) = 1;
        end
    end
end
%figure()
oblic = Cb2+Cr2;
% imshow(oblic)
SE = strel('disk',4);  %vytvoreni struktury pro dilataci/erozi
UP = imdilate(oblic,SE);    %dilatace
Down = imerode(UP,SE);   %eroze
assignin('base','Down',Down);
% figure()
% imshow(UP)
figure()
imshow(Down)
%%
ycbcr = rgb2ycbcr(orez);
[m n l]=size(ycbcr);

%nalezeni EyeMapC
y = double(ycbcr(:,:,1));
Cb = double(ycbcr(:,:,2));
Cr = double(ycbcr(:,:,3));

Q = Cb.^2;
R = (255-Cr).^2;
G = Cb./Cr;
CrCb = Cr./Cb;

%oci
EyeC=(Q+R+G)/3;
CRS = Cr.^2;

%pusa
ssCRS = sum(sum(CRS));
ssCrCb=sum(sum(CrCb));
eta = 0.95 * ssCRS/ssCrCb;
x= CRS - eta * Cr./Cb;
MM = CRS.*x.*x;
global jas_oci
```

```
prah = jas_oci * max(max(EyeC));    % nastaveni prahu na 80% max hodnoty
[r, s] = size(EyeC);            %rozmery obrazu EyeC
EyeC_bin = zeros(r,s);          %matice pro binarni obraz
for i = 1:r                     %cyklus pro prahovani
    for j = 1:s
        if EyeC(i,j) < prah
            EyeC_bin(i,j) = 0;
        else
            EyeC_bin(i,j) = 1;
        end
    end
end
%imshow(EyeC_bin)
global vyrez_oci
vyrez_oci = imcrop(EyeC_bin,[0.18*s,0.2*r,0.64*s,0.4*r]);
assignin('base','vyrez_oci', vyrez_oci);
%imshow(vyrez_oci)
[bila_y bila_x] = find(vyrez_oci);  %nalezeni nenulovych pixelu
[r_oci s_oci] = size(vyrez_oci);    %velikost vyrezu oci
i1 = 1;
j1 = 1;
for i = 1:length(bila_x)            %cyklus pro roztrideni x souradnic
    if bila_x(i) > s_oci/2
        oko2_x(i1) = bila_x(i);
        i1 = i1 + 1;
    else oko1_x(j1) = bila_x(i);
        j1 = j1 + 1;
    end
end

for i = 1:length(oko1_x)            % prirazeni y souradnic odpovidajici x
    oko1_y(i) = bila_y(i);
end

j = 1;
for i = length(oko1_x)+1:length(bila_y)
    oko2_y(j) = bila_y(i);
    j = j + 1;
end

stred_oko1_x = sum(oko1_x)/length(oko1_x);
stred_oko2_x = sum(oko2_x)/length(oko2_x);
stred_oko1_y = sum(oko1_y)/length(oko1_y);
stred_oko2_y = sum(oko2_y)/length(oko2_y);

% zakresleni stredu do vyrezu
% figure()
% imagesc(vyrez_oci); axis off;
% colormap(gray)
% hold on
```

```
% plot(stred_oko1_x,stred_oko1_y,'+', 'MarkerSize',20, 'LineWidth',3)
% plot(stred_oko2_x,stred_oko2_y,'+', 'MarkerSize',20, 'LineWidth',3)

stred1_oko1_x = stred_oko1_x + 0.18*s;  %prepocet souradnic zpet do vyrezu obl
stred1_oko2_x = stred_oko2_x + 0.18*s;
stred1_oko1_y = stred_oko1_y + 0.2*r;
stred1_oko2_y = stred_oko2_y + 0.2*r;

global jas_usta

prah_M = jas_usta * max(max(MM));    % nastaveni prahu na 55% max hodnoty
[r, s] = size(MM);            %rozmery obrazu MM
M_bin = zeros(r,s);          %matice pro binarni obraz
for i = 1:r                     %cyklus pro prahovani
    for j = 1:s
        if MM(i,j) < prah_M
            M_bin(i,j) = 0;
        else
            M_bin(i,j) = 1;
        end
    end
end
%imshow(M_bin)
global vyrez_M
vyrez_M = imcrop(M_bin,[0.3*s,0.7*r,0.4*s,0.2*r]);
assignin('base','vyrez_M',vyrez_M);
%imshow(vyrez_M)
[bilaM_y bilaM_x] = find(vyrez_M);  %nalezeni nenulovych pixelu
[r_M s_M] = size(vyrez_M);    %velikost vyrezu pusy

stred_M_x = sum(bilaM_x)/length(bilaM_x);
stred_M_y = sum(bilaM_y)/length(bilaM_y);

% % zakresleni stredu do vyrezu
% figure()
% imagesc(vyrez_M); axis off;
% colormap(gray)
% hold on
% plot(stred_M_x,stred_M_y,'+', 'MarkerSize',20, 'LineWidth',3)

stred1_M_x = stred_M_x + 0.3*s;  %prepocet souradnic zpet do vyrezu obl
stred1_M_y = stred_M_y + 0.7*r;
stred2_oko1_x = stred1_oko1_x + ob(1);  %prepocet souradnic zpet do vyrezu obl
stred2_oko2_x = stred1_oko2_x + ob(1);
stred2_oko1_y = stred1_oko1_y + ob(2);
stred2_oko2_y = stred1_oko2_y + ob(2);
stred2_M_x = stred1_M_x + ob(1);  %prepocet souradnic zpet do vyrezu obl
stred2_M_y = stred1_M_y + ob(2);

fig = figure('Visible','off')
imshow(obr); axis off
```

```
hold on
plot(stred2_oko1_x,stred2_oko1_y,'+', 'MarkerSize',20, 'LineWidth',4)
plot(stred2_oko2_x,stred2_oko2_y,'+', 'MarkerSize',20, 'LineWidth',4)
plot(stred2_M_x,stred2_M_y,'+', 'MarkerSize',20, 'LineWidth',4)
set(gca,'position',[0 0 1 1],'units','normalized')
saveas(fig,'obr_det.jpg')
obr_det = imread('obr_det.jpg');
obr_det_1 = obr_det(:,:,1);
[y2end, x2end ch] = size(obr_det);

for i = 1:y2end
    for j = 1:x2end
    if obr_det_1(i,j) == 255
        obr_det_1(i,j) = 0;
    end
    end
end

x1 = 1;
x2 = x2end;
y1 = 1;
y2 = y2end;
%zjisteni obl zleva
poc = 1;
while (sum(obr_det_1(:,poc)) == 0)
    x1 = x1 + 1;
    poc = poc + 1;
end

%zjisteni obl zprava
poc = x2;
while (sum(obr_det_1(:,poc)) == 0)
    x2 = x2 - 1;
    poc = poc - 1;
end

[s r ch] = size(obr);
obr_det_final = imresize(imcrop(obr_det,[x1 0 x2-x1 y2end]),[s r]);
% figure()
% imshow(obr_det_final)
```