TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Modulární systém pro předzpracování a analýzu digitálních obrazových dat

Modular system for preprocessing and analysis of digital image data

Bakalářská práce

Studijní program: B2646 – Informační technologie
Studijní obor: 1802R007 / Informační technologie

Autor práce: **Michaela Grusmanová**
Vedoucí práce: doc. Ing. Josef Chaloupka, Ph.D.

Liberec 2021

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání bakalářské práce

Modulární systém pro předzpracování a analýzu digitálních obrazových dat

Jméno a příjmení: **Michaela Grusmanová**
Osobní číslo: M18000073
Studijní program: B2646 Informační technologie
Studijní obor: Informační technologie
Zadávající katedra: Ústav informačních technologií a elektroniky
Akademický rok: **2020/2021**

Zásady pro vypracování:

1. Seznamte se s problematikou zpracování a rozpoznávání obrazu.
2. Navrhněte a realizujte systém (v Pythonu) pro poloautomatické zpracování a rozpoznávání obrazu.
3. Tento systém by měl obsahovat graficky přívětivé prostředí, ve kterém budou použity jednotlivé algoritmy pro zpracování a rozpoznávání obrazu z knihovny OpenCV.
4. Každý algoritmus bude realizována v rámci samostatného modulu. Moduly by mělo být možné propojovat do složitějších celků.
5. Celý systém musí být navržen tak, aby byl „otevřený", tj. aby se v budoucnu daly do programu přidávat nové moduly.

Rozsah grafických prací:	Dle potřeby dokumentace
Rozsah pracovní zprávy:	30-40 stran
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	Čeština

Seznam odborné literatury:

[1] ŠONKA, M., Hlaváč, V., Boyle, R.: Image processing, analysis, and machine vision. Fourth Edition. Australia: Cengage Learning, ISBN 978-1-133-59369-0, 2015.
[2] GONZALEZ, Rafael C. a Richard E. WOODS. Digital image processing. Global edition. New York: Pearson, ISBN 978-1-292-22304-9, 2017.
[3] HLAVÁČ, V., Sedláček, M.: Zpracování signálů a obrazů. 2. přeprac. vyd. Praha: ČVUT, 255 s. ISBN 978-80-01-03110-0, 2007.

Vedoucí práce:	doc. Ing. Josef Chaloupka, Ph.D. Ústav informačních technologií a elektroniky
Datum zadání práce:	19. října 2020
Předpokládaný termín odevzdání:	17. května 2021

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

prof. Ing. Ondřej Novák, CSc.
vedoucí ústavu

V Liberci dne 19. října 2020

Prohlášení

Byla jsem seznámena s tím, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé bakalářské práce a prohlašuji, že **s o u h l a s í m** s případným užitím mé bakalářské práce (prodej, zapůjčení apod.).

Jsem si vědoma toho, že užít své bakalářské práci či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Bakalářskou práci jsem vypracovala samostatně s použitím uvedené literatury a na základě konzultací s vedoucím bakalářské práce.

10. května 2021 Michaela Grusmanová

Chtěla bych poděkovat panu doc. Ing. Josefu Chaloupkovi, Ph.D. za odborné vedení, cenné rady, vstřícnost a hlavně trpělivost při zpracovávání této bakalářské práce.

Abstrakt

Tato bakalářská práce se zabývá problematikou zpracování a rozpoznávání obrazu. V rámci ní byl vytvořen modulární systém s jednotlivými metodami z knihovny OpenCV v programovacím jazyce Python.

Vzhledem k tomu, že se systém skládá z velkého množství funkcí, tak byla pro přehlednost spojena teoretická a praktická část do jednotlivých podkapitol. V každé podkapitole je podrobněji popsána metoda, která je v systému použita, a je v ní uveden i obrázek po zavolání této metody. V úvodní části se nachází blokový diagram, který upřesní představu o struktuře systému.

Klíčová slova:

počítačové vidění, OpenCV, předzpracování a analýza digitálních obrazových dat, modulární systém, barevné prostory, geometrické transformace, jasové transformace, filtrace šumu, hranové detektory, segmentace obrazu, binární morfologie, Houghova transformace, detekce a identifikace objektů

Abstract

This bachelor thesis deals with the issue of image processing and recognition. Within it, the modular system with individual methods from the OpenCV library in the Python programming language was created.

Since that the system consists of a large number of functions, the theoretical and practical part was combined into individual subchapters for clarity. Each subchapter describes in more detail the method that is used in the system, and it also contains a picture after calling this method. In the introductory part, there is a block diagram, which will specify the idea of the structure of the system.

Keywords:

computer vision, OpenCV, preprocessing and analysis of digital image data, modular system, color spaces, geometric transformations, brightness transformations, noise filtering, edge detectors, image segmentation, binary morphology, Hough transform, object detection and identification

Obsah

Seznam obrázků

Seznam zkratek

BSD	*Berkeley Software Distribution, licence pro svobodný software*
JPEG	*Joint Photographic (Experts) Group*
PNG	*Portable Network Graphics*
BMP	*Windows Bitmap*
RGB	*Red (červená), Green (zelená), Blue (modrá)*
HSV	*Hue (odstín), Saturation (sytost), Value (jas)*
YCbCr	*Y (jas), Cb (modrý) a Cr (červený) chrominanční komponent*
2D	*dvojdimenzionální*
3D	*trojdimenzionální*
cv	*Computer Vision*
OpenCV	*knihovna s metodami pro počítačové vidění*
Dlib	*knihovna s algoritmy strojového učení*
AdaBoost	*Adaptive Boosting, klasifikační algoritmus*

1 Úvod

Díky rychlému vývoji číslicové techniky a výpočetního výkonu si dnes lidé mohou dovolit vytvářet a využívat takových programových technologií, které byly dříve pouhými představami.

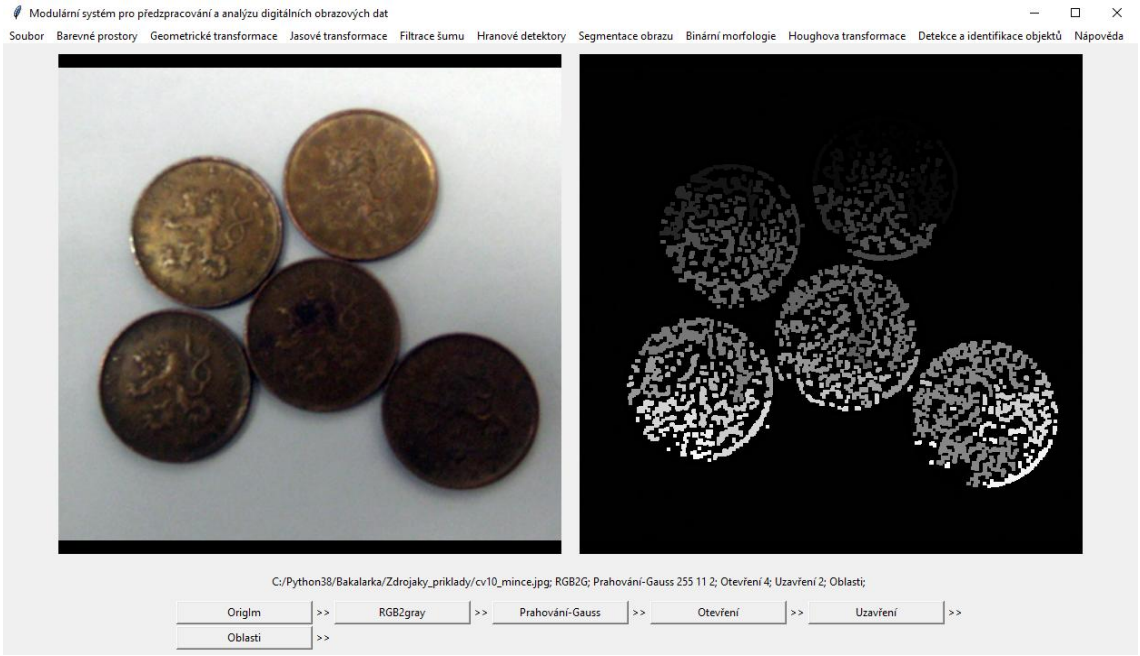
Jedna z těchto technologií je obor počítačového vidění. To umožňuje počítačům analyzovat a zpracovávat obrazové informace. Využití takové technologie lze uplatnit např. v průmyslu k automatizaci procesů, v kriminalistice k monitorování osob a jejich identifikaci, v lékařství pro detekci maligních změn tkáně a jiných informací, při verifikaci (ověření identity) v průběhu přihlašování atd. Dokonce je tento obor vnímán za část oboru umělé inteligence, kdy se používá jako zrakový senzor robota.

Pro samotnou realizaci se využívají různé knihovny, ve kterých jsou metody počítačového vidění implementovány. Nejznámější je knihovna OpenCV, jejímž autorem je společnost Intel, je napsána v programovacím jazyce C, všechny algoritmy běží velmi rychle, a je šířena pod BSD licencí.

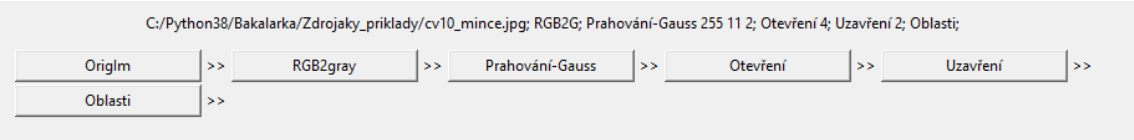
Cílem této práce je přiblížit metody z knihovny OpenCV používaných právě v počítačovém vidění a to v samostatné aplikaci naprogramované v jazyce Python. Aplikace resp. systém nabídne mj. různé modulace obrazu nebo identifikaci konkrétních útvarů v obraze.

2 Systém pro předzpracování a analýzu digitálních obrazových dat

Systém se skládá z jednotlivých modulů pro dané metody – akce. Otevírá se v samostatném okně, kde si uživatel musí nejprve vybrat obrázek (JPG/JPEG, PNG, BMP) a zvolit barevný prostor (R-RGB, G-RGB, B-RGB, stupě šedi, Cb-YCbCr, Cr-YCbCr, H-HSV a S-HSV). Pokud uživatel aplikuje na vstupní obrázek některé metody, má možnost uložit si nejen obrázek, ale také i samotný řetězec akcí, který se dá zpětně načíst, a tím pádem může v práci pokračovat později.



Obrázek 2.1: Náhled aplikace



Obrázek 2.2: Přiblížení dolní části (řetězec a tlačítka metod)

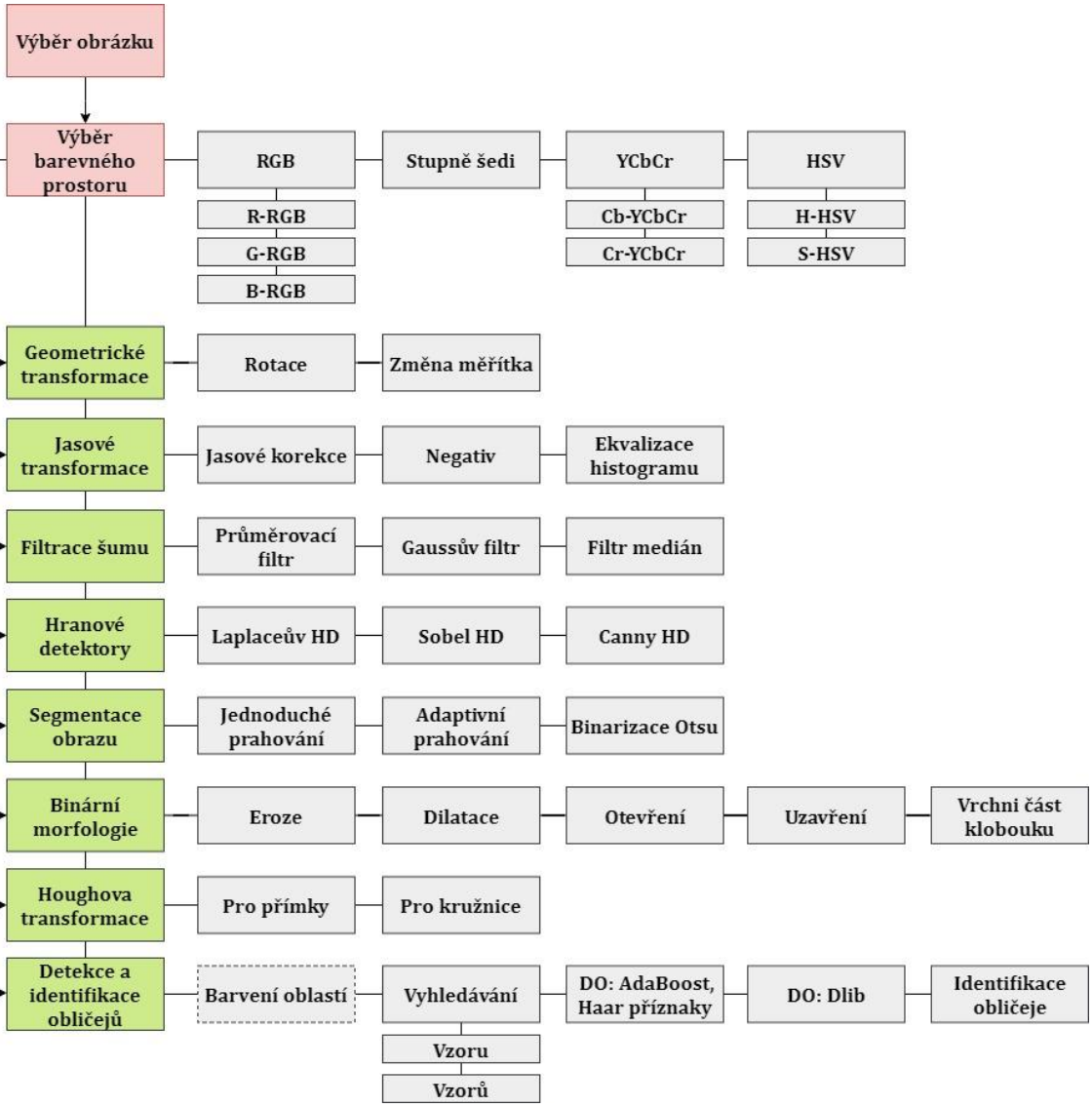
Aplikace je představena na obrázkách 2.1 a 2.2, podrobně bude popsána v dalších podkapitolách.

Na obrázku 2.3 je blokový diagram tohoto systému s jednotlivými moduly. Červeně značené moduly jsou povinné. Uživatel jimi musí projít před tím, než bude moci provést další akce. Zelené označení mají názvy hlavních skupin, do kterých jsou metody rozděleny. Tyto metody jsou pak šedé. Přerušovaná čára rámečku metody – Barvení oblastí znamená, že uživatel musí nejprve zvolit některou metodu ze segmentace obrazu.

Pro práci s tímto systémem je uživateli k dispozici i nápověda, která mu v rychlosti popíše význam jednotlivých metod a vysvětlí mu jejich parametry.

V případě, že uživatel není spokojený s navolením parametrů, může si je za chodu kdykoli pozměnit kliknutím pravým tlačítkem myši na tlačítko pro konkrétní metodu.

Vzhledem k tomu, že se systém skládá z velkého množství funkcí, tak byla pro přehlednost spojena teoretická a praktická část do jednotlivých podkapitol.



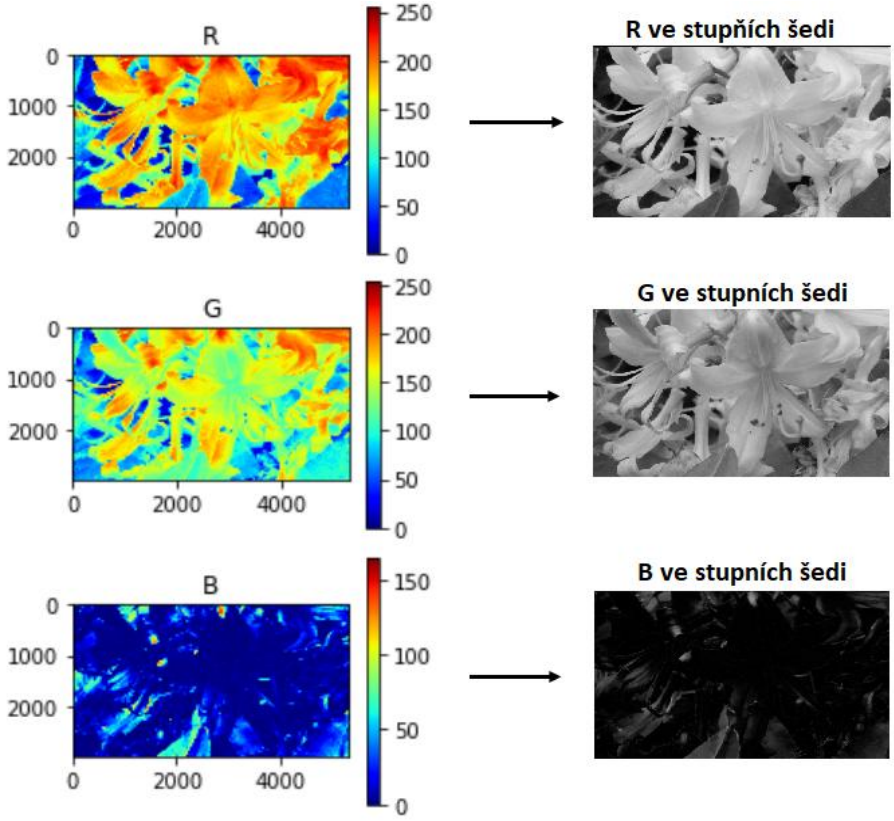
Obrázek 2.3: Blokový diagram systému

2.1 Barevné prostory

2.1.1 RGB

Jedná se o nejznámější uložení barevného obrazu. [1] Vychází z principu, že světlo je vyzařováno do okolí. [2] RGB je kombinace barevných složek, R – red *(červená)*, G – green *(zelená)*, B – blue *(modrá)*. Rozsah jednotlivých složek barev může být od 0 do 255, např. (0…255, 0…255, 0…255) pro uložení jako uint8, případně může být hodnota normována, tzn. (0…1, 0…1, 0…1). Kombinací složek R, G, B vznikne libovolná barva. [1] Proto se tomuto prostoru také říká aditivní, jelikož se jednotlivé složky barev sčítají a tím vytváří světlo větší intenzity. RGB prostor používají počítačové monitory, dataprojektory apod. [2]

V praktické části bakalářské práce je převod do jednotlivých složek RGB umístěn v modulu *modul_torgb*. V modulu se nachází funkce *method_torrgb(required_size, path, number),* která obrázek načte *(parametr path)*, změní se jeho velikost na požadovanou *(parametr required_size)*, dle parametru *number (0 – red, 1 – green, 2 – blue)* se vybere pouze jedna složka, která se následně zobrazí šedotónově.



Obrázek 2.4: Převod obrázku do jednotlivých složek RGB (ve stupních šedi)

2.1.2 Převod do stupňů šedi

Převod RGB-barevného obrazu se provádí pomocí vztahu:

$$Y = 0.299R + 0.587G + 0.114B. \quad (2.1)$$

Jedná se o ztrátový lineární převod, jelikož se z původní 3D matice (RGB) vytvoří 2D matice.

Převod do stupňů šedi je realizován v modulu *modul_togray*. Funkce *method_togray(required_size, path)* obdobně jako předchozí metoda načte obrázek, změní jeho velikost a převede jej do stupňů šedi. Tento převod je realizován pomocí funkce z OpenCV *cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)*. Výsledek je na obrázku níže.

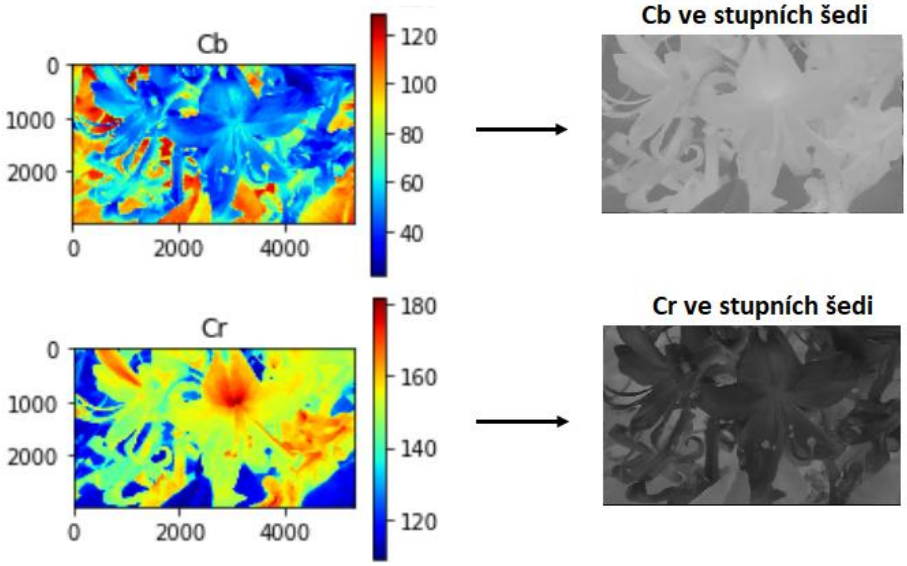
Obrázek 2.5: Převod obrázku do stupňů šedi

2.1.3 YCbCr

Složky prostoru YCrCb vzniknou lineárním převodem složek prostoru RGB. Složka Y se podobá převodu do stupňů šedi, Cb je "doplněk" k modré barvě a Cr zase "doplněk" k barvě červené. Vztah pro převod:

$$\begin{aligned} Y &= +0.299R + 0.587G + 0.114B, \\ Cb &= -0.168736R - 0.331264G + 0.5B, \\ Cr &= +0.5R - 0.418688G - 0.081312B. \end{aligned} \quad (2.2)$$

Modul *modul_toycbcr* obsahuje metodu *method_ycbcr(required_size, path, number)*. Vykonávání začátku této metody se provádí stejně jako u předešlých, poté se obrázek převede do YCbCr pomocí funkce z OpenCV *cv2.cvtColor(image, cv2.COLOR_RGB2YCrCb)* a následně se pomocí parametru *number* vybere složka, která se má zobrazit *(1 – Cr, 2 – Cb).*

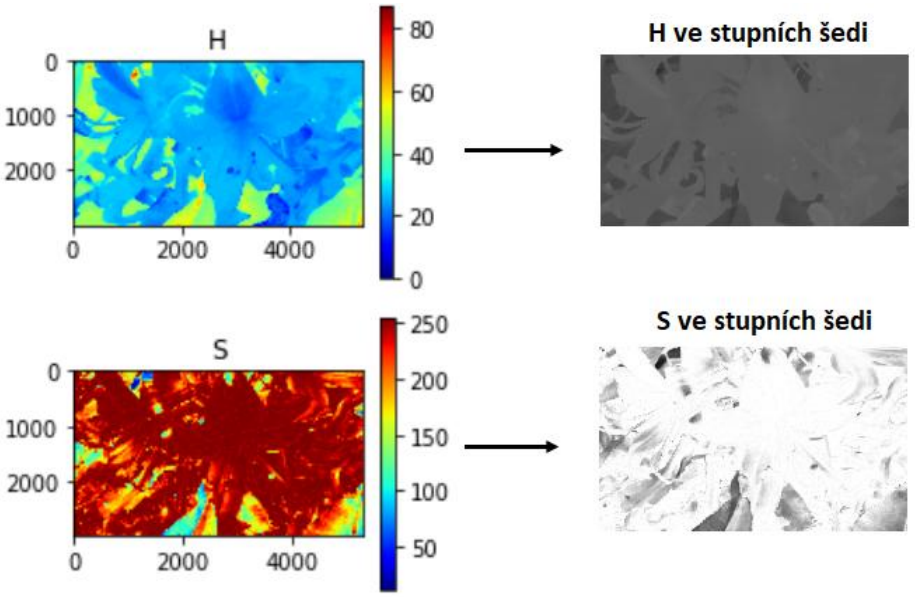


Obrázek 2.6: Převod obrázku do složek Cb a Cr

2.1.4 HSV

Mezi používané barevné prostory patří také HSV, kde H – hue *(barevnost, barevný odstín)*, S – saturation *(saturace barvy)*, V – value *(hodnota vytvářející šedotónový odstín)*. Pro monochromatický převod v různých algoritmech pro zpracování a rozpoznávání obrazu se využívá především složka barevnosti *(H)*. Hodnoty této složky H jsou od 0 až do 360 stupňů. [1]

Modul *modul_tohsv* obsahuje metodu *method_tohsv(required_size, path, number).* Je zde využita funkce z OpenCV *cv2.cvtColor(image, cv2.COLOR_RGB-2HSV).* Pro parametr *number* platí, že *0 – H, 1 – S.*



Obrázek 2.7: Převod obrázku do složek H a S

2.2 Geometrické transformace

Geometrické transformace vypočítají ze souřadnic bodů vstupního obrazu souřadnice bodů výstupního obrazu. Skládají se z transformace souřadnic bodů a aproximace jasové funkce.

Transformace souřadnic bodů nalezne k diskrétním souřadnicím bodu vstupního obrazu odpovídající bod výstupního obrazu.

Aproximace jasové funkce hledá celočíselnou hodnotu jasu, která nejlépe odpovídá nově nalezené poloze. [3]

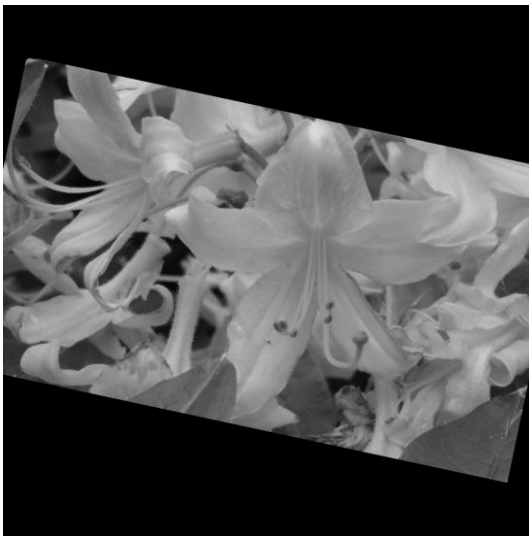
Geometrické transformace se využívají zejména pro zvětšování, posouvání, pootáčení, zkosení 2D obrazů a pro odstraňování geometrických zkreslení. [4]

2.2.1 Rotace

Rotace představuje otočení obrazu kolem počátku souřadné soustavy o daný úhel ϕ proti směru hodinových ručiček proti originálnímu obrazu.

$$x' = x \cdot cos\phi + y \cdot sin\phi \quad y' = -x \cdot cos\phi + y \cdot sin\phi \,. \tag{2.3}$$

Funkce *method_rotate(previous, angles)*, která zajišťuje otočení obrazu, se nachází v modulu *modul_rotate.* Samotná implementovaná funkce pro rotaci je v balíčku *imutils* pod názvem *rotate_bound(previous, angles).* Parametrem *previous* je předchozí obrázek, který se nadále upravuje a předává se i v dalších metodách. Parametr *angles* si nastaví sám uživatel zadáním hodnoty v dialogovém okně. Tato hodnota říká, o kolik stupňů se daný obrázek má otočit.



Obrázek 2.8: Otočený obrázek o 12 °

2.2.2 Změna měřítka

Změna měřítka umožní změnit velikost obrazu ve směru souřadnicových os.

$$x' = a \cdot x \quad y' = b \cdot x\,. \tag{2.4}$$

Kde *a, b* jsou koeficienty změny měřítka, *a* ve směru souřadnicové osy x a *b* ve směru souřadnicové osy y. [1]

Změna měřítka v modulu *modul_scale*, funkce *method_scale(previous, percent)* pracuje s předchozím obrázkem *previous*, kterému změní velikost dle uživatelem zadaného parametru *percent.* Z OpenCV je využita metoda *cv2.resize (previous, dsize, fx=0.1, fy=0.1, interpolation = cv2.INTER_CUBIC)*.

Obrázek 2.9: Zmenšený obrázek o 50 %

2.3 Jasové transformace

Jasové transformace lze rozdělit na dvě skupiny, a to jasové korekce a modifikace jasové stupnice. [3]

2.3.1 Jasové korekce

U jasových korekcí závisí jas v bodě výstupního obrazu na jasu odpovídajícího bodu vstupního obrazu. Pomocí jasových korekcí je možné potlačit různé poruchy, jako je:

- vinětace, kde světlo procházející dále od optické osy je více zeslabováno,
- porucha, kdy snímací prvek není citlivý ve všech bodech,
- nadměrné osvětlení scény,
- prachové částice na optice nebo na snímacím prvku. [1]

Nejčastěji se předpokládá jednoduchý model porušení obrazu multiplikativním koeficientem *e (x, y)*. Pro každý bod původního obrazu *g (x, y)* je vypočtena výstupní hodnota jasu zkresleného obrazu *f (x, y)*.

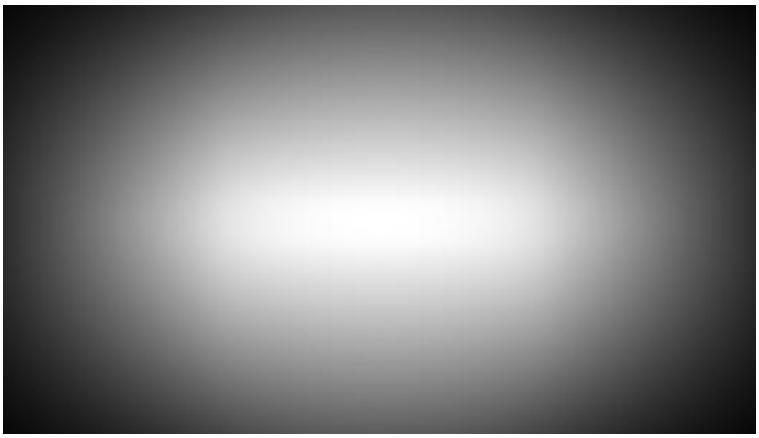
$$f(x,y) = e(x,y) \cdot g(x,y) \tag{2.5}$$

Při systematické degradaci *e* je možné za stálých podmínek nasnímat obraz o známém průběhu jasové funkce *g (x, y)*. Pro jednoduchost se použije jako etalon obraz o konstantním jasu *c* (etalonová šedá plocha), který se po sejmutí a digitalizaci značí f_c *(x, y)*. [3]

$$g(x,y) = \frac{f(x,y)}{e(x,y)} = \frac{c \cdot f(x,y)}{f_c(x,y)} \tag{2.6}$$

Tato metoda je implementována v modulu *modul_correction* v metodě *method_correct(previous, etalon, c)*. V této metodě je použit vzorec pro jasovou korekci. Prvním parametrem je vstupní obrázek, druhým je etalonový šedý obraz a třetím konstantní jas. Na obrázku 8 je vstupní obrázek zatížen chybou, pod ním je použitý etalon a na obrázku 10 je výsledný obrázek po převodu do stupňů šedi a po provedení jasové korekce s příslušným etalonem a hodnotou konstantního jasu 255.

Obrázek 2.10: Vstupní obrázek zatížený chybou



Obrázek 2.11: Etalon

Obrázek 2.12: Výstupní obrázek po jasové korekci

2.3.2 Negativ

U negativu *(inverzi)* není třeba znát informace z předchozí analýzy obrazu. Intenzita jasu každého pixelu v obraze $f(x, y)$ je nahrazena novou hodnotou intenzity jasu $g(x, y)$. [2]

$$g(x, y) = 1 - f(x, y) \tag{2.7}$$

Implementace negativu se nachází v modulu *modul_negate* v metodě *method_negate(previous)*, kde *previous* je vstupní obraz. Zde jsou odečítány od hodnoty 255 jednotlivé obrazové body.

Obrázek 2.13: Negativ

2.3.3 Ekvalizace histogramu

Ekvalizace *(vyrovnávání)* histogramu zvýší kontrast blízko maxim histogramu a sníží kontrast blízko minim histogramu. Využívá se pro zvýšení kontrastu monochromatického obrazu. Ideálem je, že všechny jasy jsou v histogramu zastoupeny stejně četně.

$$q = \frac{q_k - q_0}{N^2} \cdot \sum_{i=p_0}^{p} H(i) + q_0. \quad (2.8)$$

Kde $H(p)$ je histogram výchozího obrazu, $p = \langle p_0, p_k \rangle$ je výchozí jasová stupnice, q monotónní transformace jasové stupnice, $q = \langle q_0, q_k \rangle$ je výstupní interval jasů, N^2 počet řádků a sloupců.

Obrázek 2.14: Ekvalizace histogramu

V systému je ekvalizace histogramu realizována ve funkci *method_equalize(previous)* v modulu *modul_equalize*. Parametrem je pouze vstupní obrázek.

Z knihovny OpenCV je využita funkce *cv2.equalizeHist(previous, 0)*, kde parametr *0* zajišťuje šedobílý obrázek.

2.4 Filtrace šumu

2.4.1 Statistický princip filtrace šumu

Nechť je každý obrazový pixel zatížen náhodným aditivním šumem ν, nezávislým na obrazové funkci, s nulovou směrodatnou odchylkou σ a střední hodnotou μ. N-násobným sejmutím statické scény při stejných podmínkám je možné získat více realizací takových obrazů. Vybrané pixely o stejných souřadnicích z každého sejmutého obrázku jsou označeny g_i, kde i značí, k jakému obrazu pixel patří. Aritmetickým průměrem pixelů $g_1,\ldots, g_n$ se šumem $\nu_1,\ldots, \nu_n$ lze získat odhad správné hodnoty.

$$\frac{g_1 + \ \ldots \ + \ g_n}{n} + \frac{\nu_1 + \ \ldots \ + \ \nu_n}{n} \tag{2.9}$$

2.4.2 Lokální filtrace

Lokální filtrace je taková filtrace, která k výpočtu nové hodnoty pixelu využívá malé okolí O právě zpracovávaného pixelu – nejčastěji se jedná o malý obdélník. Výsledek analýzy je poté zapsán jako hodnota tohoto pixelu ve výsledném obrazu. Tento způsob se zakládá na představě, že se celý obraz prochází systematicky (po řádcích nebo sloupcích).

Metody lokální filtrace se rozdělují do dvou skupin podle toho, k jakému účelu slouží. První skupinou jsou metody pro vyhlazování. Ty usilují o potlačení šumu a osamocených fluktuací hodnot obrazové funkce a jsou příbuzné s dolnofrekvenčními propustěmi. Metody v druhé skupině se užívají k detekci hran.

Dalším hlediskem pro třídění metod lokální filtrace mohou být matematické vlastnosti příslušné transformace. Lineární metody počítají hodnotu ve výstupním obraze *g (x, y) pomocí* lineární kombinace hodnot vstupního obrazu *f* v okolí zpracovávaného pixelu *(x, y)*. Druhou skupinou jsou nelineární metody.

Lineární filtry lze realizovat pomocí konvoluce. Rovnice níže popisuje diskrétní konvoluci s jádrem *h*, nazývaném jako konvoluční maska.

$$f(x,y) = \sum_{(m,n)} \sum_{\in \mathrm{O}} h(x-m, y-n) g(m,n) \tag{2.10}$$

2.4.3 Průměrovací filtr

Jedná se o základní typ filtru pro vyhlazování obrazu. Ke každému pixelu je přiřazen nový jas, který vznikne aritmetickým průměrem původních jasů ve zvoleném okolí. Např. pro okolí 3 × 3 vypadá konvoluční maska *h* následovně:

$$h = \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}. [3] \tag{2.11}$$

Průměrovací filtr je implementován v modulu *modul_average* ve funkci *method_average(previous, number)*. Prvním argumentem *previous* je vstupní obrázek a druhým parametrem *number* je velikost jádra. V OpenCV je využita metoda *cv2.blur(previous, (numb,numb))*. Z obrázků níže vyplývá, že čím větší je zvolená hodnota jádra větší, tím větší je vyhlazení.

Obrázek 2.15: Průměrovací filtr s jádrem 5x5

Obrázek 2.16: Průměrovací filtr s jádrem 13x13

2.4.4 Gaussův filtr

Gaussův filtr je dán vztahem níže, x a y jsou obrazové souřadnice a σ je směrodatná odchylka rozdělení.

$$G(x,y) = e^{-(x^2+y^2)/2\sigma^2} \tag{2.12}$$

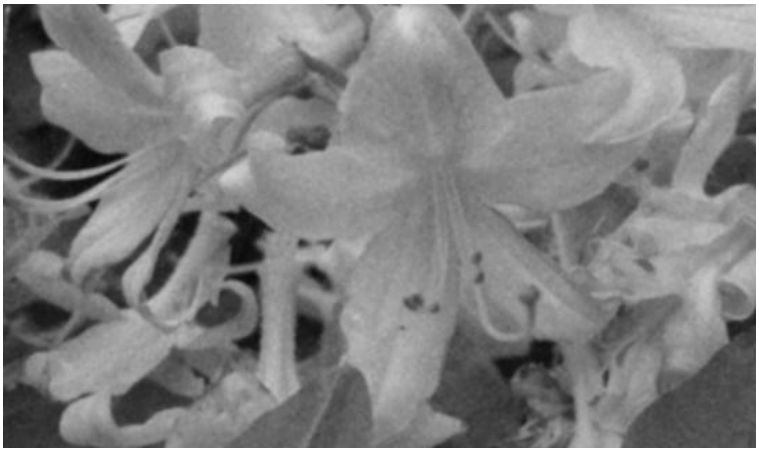
Někdy se vyjadřuje tento vzorec pomocí normalizačního faktoru.

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \tag{2.13}$$

Směrodatná odchylka je jediným parametrem Gaussova filtru a je úměrná velikosti okolí, ve kterém filtr pracuje. [1]

Tento filtr je v systému realizován v modulu *modul_gaussian* v metodě *method_gaussian(previous, number). Previous* je vstupní obrázek a *number* velikost jádra. Metoda je implementována pomocí funkce *cv2.GaussianBlur(previous, (numb,numb), 0)* z OpenCV. Parametr *0* je směrodatná odchylka ve směru x a y. Tak jako u průměrovacího filtru, i tady platí, že čím větší je zvolená hodnota jádra, tím je obraz vyhlazenější.

Obrázek 2.17: Zašuměný obrázek



Obrázek 2.18: Obrázek po Gaussově filtraci (velikost jádra: 5) ve stupních šedi

2.4.5 Filtr medián

Mediánový filtr patří, na rozdíl od předešlých filtrů, do skupiny nelineárních filtrů. Jas výsledného bodu je stanoven jako medián, který je určený z hodnot jasu bodů ve zvoleném okolí vstupního obrazu. Výhodou je redukce stupně rozmazání hran a potlačení impulsního šumu, naopak nevýhodou může být právě zvolené obdélníkové okolí, jež porušuje ostré rohy a tenké čáry.

Z OpenCV je využita funkce pro tento typ filtru *cv.medianBlur(previous, numb)*. Tato funkce je realizována v modulu *modul_median* ve funkci *method_median(previous, number).* Prvním parametrem je vstupní obrázek a druhým je velikost okolí.

Obrázek 2.19: Zašuměný obrázek

Obrázek 2.20: Filtr medián s okolím 3 ve stupních šedi

2.5 Hranové detektory

Pro lidské vnímání jsou důležitá místa v obraze, kde se mění jas, tedy hrany. Hrana je určena tím, jak se mění hodnota obrazové funkce *f (x, y).* Pro zjištění změn funkce dvou proměnných se využívá parciálních derivací. Změna funkce je daná jejím gradientem ∇, který určuje směr největšího růstu funkce, a strmostí růstu gradientu ψ. Pokud mají pixely velkou strmost gradientu, pak to jsou hrany.

$$|\nabla f(x,y)| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \tag{2.14}$$

$$\psi = \arg\left(\frac{\partial g}{\partial x}, \frac{\partial g}{\partial y}\right)^{1} \tag{2.15}$$

2.5.1 Laplaceův hranový detektor

Tento detektor využívá pro odhad velikosti gradientu *(též velikost hrany)* všesměrový lineární Laplaceův operátor – Laplacián ∇^2, vycházející z druhých parciálních derivací. V tomto případě záleží pouze na velikosti gradientu, nikoli na jeho směru.

$$\nabla^2 g(x,y) = \frac{\partial^2 g(x,y)}{\partial x^2} + \frac{\partial^2 g(x,y)}{\partial y^2} \tag{2.16}$$

Laplacián je v digitálním obraze aproximován diskrétní konvolucí. Nejpoužívanější jádra pro okolí 3×3 jsou:

$$h = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{2.17}$$

$$h = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{2.18}$$

Nevýhodou je citlivost na šum a dvojité odezvy na hrany, které odpovídají tenkým liniím v obraze.

V praktické části je Laplaceův hranový detektor implementován v modulu *modul_laplace* v metodě *method_laplace(previous)*, kde parametrem previous je vstupní obraz. V této metodě je využita funkce z knihovny OpenCV *cv2.Laplacian(img, cv2.CV_64F).* Druhým parametrem je použité jádro. Detekované hrany jsou přesné a jemné.

[1] Pro vzorec platí, že *arg(x, y)* je úhel mezi osou *x* a radiusvektorem k bodu *(x, y)*.

Obrázek 2.21: Laplaceův hranový detektor

2.5.2 Sobelův hranový detektor

Sobelův hranový detektor se využívá často pro detekci svislých a vodorovných hran, využitím následujících masek.

$$h_1 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{2.19}$$

$$h_2 = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix} \tag{2.20}$$

Sobelův hranový detektor se nachází v modulu *modul_sobel* v metodě *method_sobel(previous, number)*. Parametr *previous* je vstupní obrázek a parametr *number* je velikost jádra, kterou si uživatel může zvolit v dialogovém okně. Z knihovny OpenCV je využita funkce *cv2.Sobel(previous, cv2.CV_8U, 1, 0, ksize=number)*. Dle této funkce se obrazové body zpracovávají v ose *x*.

Obrázek 2.22: Sobelův hranový detektor s velikostí jádra 5

2.5.3 Cannyho hranový detektor

Základní myšlenkou Cannyho hranového detektoru je představa toho, že filtr hledá skokovou hranu. Existují tři kritéria pro detekci hran dle J. Cannyho.

Prvním z nich je *detekční kritérium.* To požaduje, aby pro jednu hranu nevznikaly vícenásobné odezvy a aby nebyly přehlédnuty významné hrany.

Druhým požadavkem *(lokalizační kritérium)* je, aby byl rozdíl mezi skutečnou a nalezenou polohou hrany minimální.

Posledním kritériem je *požadavek jedné odezvy*, který zajišťuje, že detektor nereaguje víckrát na jednu hranu v obraze. [3]

Z knihovny OpenCV je v modulu *modul_canny* v metodě *method_canny (previous, number)* využita metoda *cv2.Canny(previous, number, 200)*. Parametr *previous* je vstupní obrázek, *number* je minimální prahová hodnota pro detekci hran a číslo *200* je naopak maximální prahová hodnota, která říká, že všechny hrany s gradientem intenzity větším než je tato hodnota, jsou hranami.

Obrázek 2.23: Cannyho hranový detektor s minimální prahovou hodnotou 2

Obrázek 2.24: Cannyho hranový detektor s minimální prahovou hodnotou 60

2.6 Segmentace obrazu

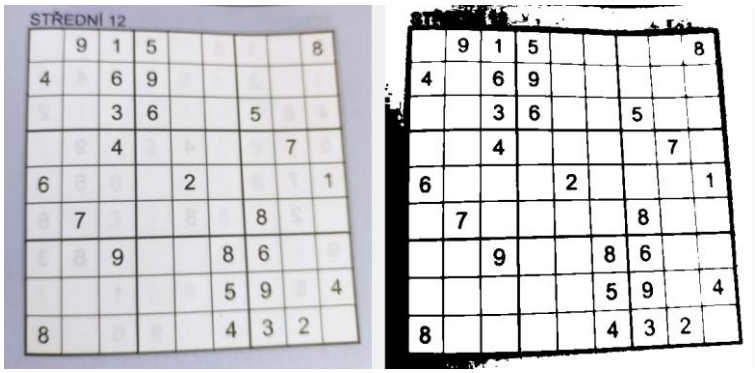
Segmentace obrazu v sobě zahrnuje několik metod. Tyto metody rozdělují daný obraz na části, které mají společné vlastnosti, tím dochází k redukci objemu zpracovávaných dat.

Existují dva typy segmentace, kompletní a částečná. Výsledkem kompletní segmentace je sada nesouvislých oblastí odpovídající jednoznačně objektům ve vstupním obraze, naopak výsledkem částečné segmentace je to, že regiony přímo neodpovídají objektům obrazu, rozdělují se podle jasu, barvy, textury atd.

Metody segmentace obrazu se dají rozdělit do tří skupin. První skupinou jsou metody využívající globální znalost, druhou skupinou jsou metody pro určování hranic mezi oblastmi a sledování průběhu hranice a poslední skupinou jsou metody pro vytváření oblastí na základě podobného jasu, barvy, textury atd. Metody druhé a třetí skupiny se dají kombinovat.

2.6.1 Jednoduché prahování

Prahování je nejstarší, nejjednodušší a nejrychlejší metoda segmentace. Tudíž se často používá v jednoduchých aplikacích. V této metodě se předpokládá, že objekty jsou tvořeny barvou (jasem), která se liší od ostatních barev (jasu). Základem je dobré stanovení prahu. [1] Pokud je hodnota pixelu větší, než zvolený prah, přiřadí se pixelu např. hodnota 0 (bílá barva), pokud je menší, přiřadí se hodnota 1 (černá barva). [5]



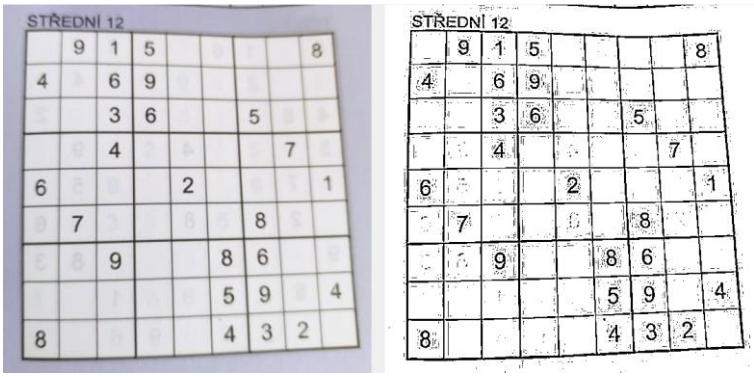
Obrázek 2.25: Jednoduché binární prahování (s parametry 180 - prah, 255)

Tato metoda je v praktické části implementována v modulu *modul_simple_thres* v metodě *method_simple_thresh(previous, n1, n2, method)*. Z OpenCV knihovny se využita funkce *cv2.threshold(previous, n1, n2, method)*. Prvním argumentem je vstupní obraz, druhým je prahová hodnota, třetím je hodnota, se kterou se má

počítat, když je hodnota pixelu větší než prahová hodnota a posledním parametrem je druh prahování. Uživatel si může vybrat z binárního, binárního inverzního, zkráceného a z nultého prahování.

2.6.2 Adaptivní prahování

Při pořízení obrazu často dochází k nerovnoměrnému osvětlení nebo drobným odchylkám vstupního zařízení, takže je velice nízká pravděpodobnost, že bude zvolení jednoho prahu pro celý obraz úspěšný. Adaptivní prahování se snaží tento problém vyřešit pomocí proměnných prahových hodnot. Ty se mění v závislosti na místních charakteristikách obrazu. Jednou z možností, jak získat tyto lokální prahové hodnoty je rozdělení obrazu na dílčí části a určení prahu nezávisle v každé z nich, pokud lokální prahovou hodnotu v nějaké dílčí části určit nelze, tak se tato hodnota interpoluje ze sousedních dílčích částí. [1]



Obrázek 2.26: Gaussovo adaptivní prahování (maxV = 255, blockS = 5, C = 2)

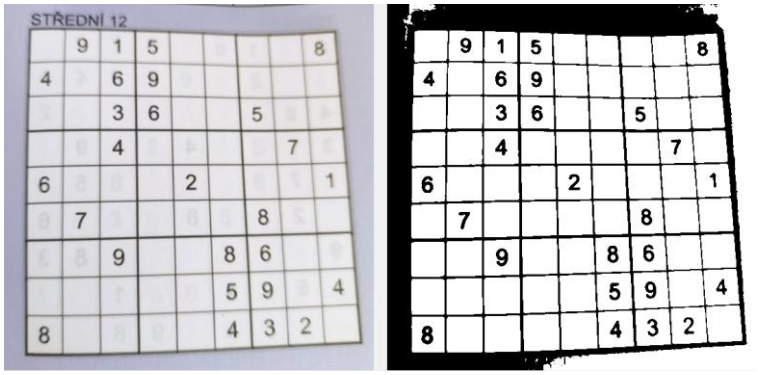
V praktické části je adaptivní prahování implementováno v modulu *modul_adaptive_thresh* v metodě *method_adaptive_thresh(previous, maxV, blockS, C, method)*. V OpenCV se tato metoda nazývá *cv2.adaptiveThreshold.* Prvním parametrem je vstupní obrázek, *maxV* je maximální prahová hodnota, *blockS* představuje velikost bloku, *C* je pouze konstanta, která se odečítá od vypočítaného průměru nebo od váženého průměru a poslední parametr rozhoduje o tom, zda se provede průměrné adaptivní prahování, kdy prahová hodnota se získá jako průměr ze sousedních oblastí, nebo Gaussovo adaptivní prahování, kdy se prahová hodnota získává jako vážený součet sousedních hodnot, kde váhy jsou gaussovské okno. [5]

2.6.3 Binarizace Otsu

Binarizace Otsu patří také mezi prahovací metody. Pro tuto metodu je používán tzv. Otsův algoritmus, který předpokládá, že zpracovávaný obraz má bimodální histogram. Tento typ histogramu obsahuje dvě lokální maxima, kdy jedno je pro tmavou barvu a druhé je pro světlou barvu. Následně je z histogramu odvozena prahová hodnota, která leží mezi dvěma lokálními maximy.

Výhodou této metody je velká rychlost, nevýhodou je to, že obrázek musí mít bimodální histogram, aby byla binarizace přesná. [1]

Tato metoda je realizovaná v modulu *modul_otsu_thresh* v metodě *method_otsu_thresh(previous, thresh, maxValue)*. Z knihovny OpenCV je využita funkce *cv2.threshold(previous, thresh, maxValue, cv2.THRESH_BINARY + cv2.THRESH_OTSU)*. Prvním parametrem je vstupní obrázek, druhým je prahová hodnota, třetí je hodnota, která se má použít při překročení prahu a posledním parametrem je kombinace binárního prahování a binarizace Otsu.



Obrázek 2.27: Binarizace Otsu s prahem 0 a max. hodnotou 255

2.7 Binární morfologie

Matematická morfologie využívá vlastnosti bodových množin, výsledky z integrální geometrie a topologie. Předpokladem je představa možnosti vymodelovat reálné obrázky pomocí bodových množin libovolné dimenze. [3]

Binární (šedotónový) obraz lze vymodelovat v euklidovském prostoru pomocí bodové množiny. Spojitý binární obraz je definován na množině racionálních čísel a diskrétní binární obraz zase na množině celých čísel. [6]

2.7.1 Eroze

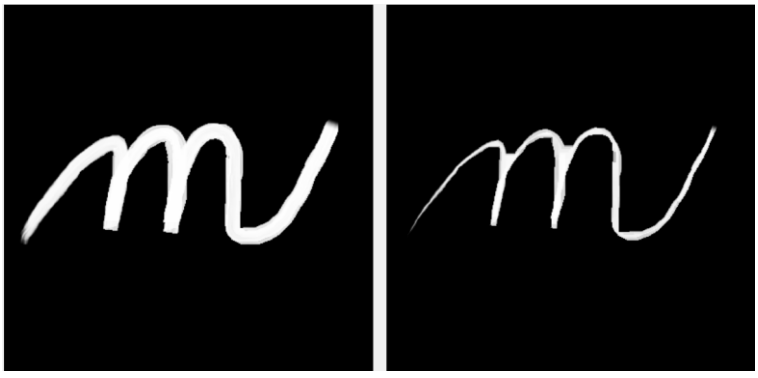
Eroze skládá dvě množiny, dle následujícího předpisu:

$$X \ominus^{2} B = \{p \in \mathcal{E}^2 : p + b \in X \text{ pro každé } b \in B\}. \quad (2.21)$$

Tento předpis říká, že pro každý obrazový bod p se ověří, zda pro všechna možná $p + b$ leží výsledek v X. Pokud výsledek v X leží, tak se v reprezentativním bodě výsledného obrázku zapíše *1* a v opačném případě *0*.

Eroze se využívá pro zjednodušení struktury objektů a není komutativní. [3]

V praktické části je tato metoda implementována v modulu *modul_erosion* v metodě *method_erosion(previous, kernel, iteration)*. Z knihovny OpenCV je využita funkce *cv2.erode(previous, final_kernel, iterations = iteration)*, kde prvním parametrem *previous* je vstupní obrázek, druhým parametrem *final_kernel* je velikost jádra a *iterations* je počet iterací.



Obrázek 2.28: Eroze, vlevo: původní obrázek, vpravo: výsledný obrázek po erozi (velikost jádra 5, počet iterací: 3)

[2] Minkowský rozdíl – pomocí tohoto rozdílu se dá určit, zda je možné umístit jeden objekt do druhého, výsledek určuje množinu všech posunutí, kterými je možné toto umístění provést. [13]

2.7.2 Dilatace

Dilatace je opakem eroze. [5] Jedná se o bodovou množinu všech možných vektorových součtů pro dvojice pixelů, pokaždé pro jeden z množiny X a jeden z množiny B. Předpis pro dilataci je:

$$X \oplus^{3} B = \{p \in \mathcal{E}^2 : p = x + b, x \in X, b \in B\}. \text{[3]} \quad (2.22)$$

Hodnota reprezentativního pixelu ve výsledném obrázku je *1*, pokud je hodnota pixelu pod jádrem jedna. Rozdíl oproti erozi je tedy v tom, že eroze postupně ztenčuje bílou oblast a zmenšuje velikost objektu v popředí, zatímco dilatace je zvětšuje. Oproti erozi je dilatace komutativní.

Dilataci by se dalo využít např. pro spojení rozbitých částí objektu. [5]

Tato metoda je realizována v systému v modulu *modul_dilation* v metodě *method_dilation(previous, kernel, iteration)*. Z knihovny OpenCV je použita funkce *cv2.dilate(previous, final_kernel, iterations = iteration)*. Prvním parametrem je vstupní obrázek, druhým je velikost jádra a posledním je počet iterací.

Obrázek 2.29: Dilatace, vlevo: původní obrázek, vpravo: obrázek po dilataci (velikost jádra: 5, počet iterací: 5)

2.7.3 Otevření

Kombinací eroze a dilatace vzniknou další významné morfologické transformace – otevření a uzavření. Výsledkem otevření a uzavření je zjednodušený obraz, obsahující méně detailů.

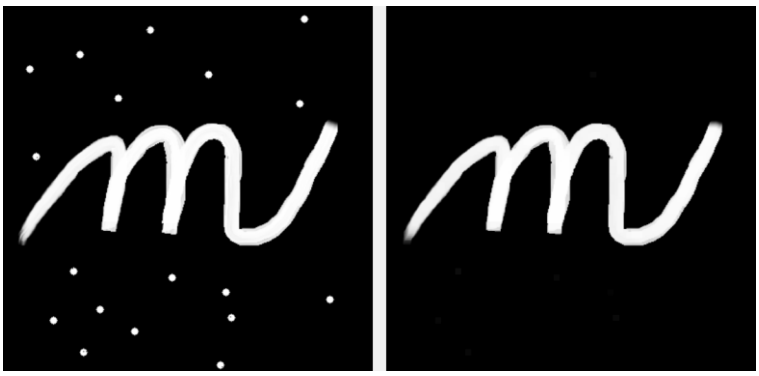
Otevření vznikne, když se provede eroze a následně po ní dilatace. Předpis pro tuto transformaci je:

$$X \circ^{4} B = \{X \ominus B\} \oplus B. \quad (2.23)$$

Otevření se využívá při odstraňování šumu. Jedná se od antiextenzivní operaci (některé pixely jsou z obrazu odstraněny).

[3] Minkowský množinový součet.
[4] Otevření množiny X strukturním elementem B. [3]

Tato transformace se v systému nachází v modulu *modul_opening* v metodě *method_opening(previous, kernel)*. Z OpenCV je využita funkce *cv2.morphology-Ex(previous, cv2.MORPH_OPEN, final_kernel).* Parametrem *previous* je vstupní obrázek, *cv2.-MORPH_OPEN* říká, že se jedná o otevření, a parametrem *final_kernel* je velikost jádra.



Obrázek 2.30: Otevření, vlevo: původní obrázek zatížený šumem, vpravo: obrázek po otevření (velikost jádra: 9)

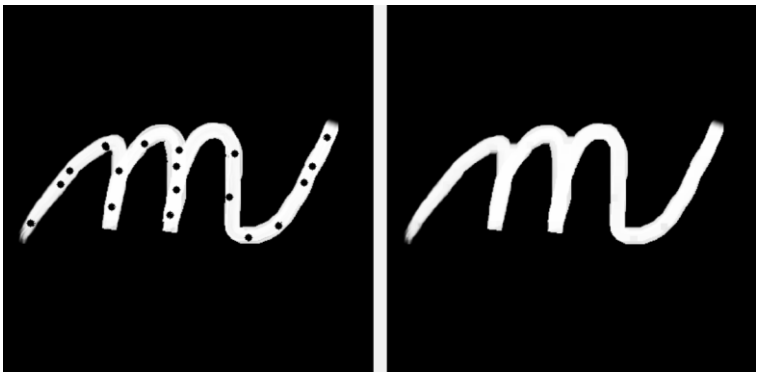
2.7.4 Uzavření

Uzavření vznikne, když se nejprve provede dilatace a následně po ní eroze. Je tedy opakem otevření. Předpis pro tuto morfologickou transformaci je:

$$X \bullet^{5} B = \{X \oplus B\} \ominus B. \tag{2.24}$$

Uzavření se využívá pro spojení objektů, které jsou blízko u sebe, zaplnění malých děr a vyhlazení obrysu zaplněním úzkých zálivů. Je extenzivní operací (některé pixely v obrazu přibudou).

Podobně jako dilatace a eroze, tak i otevření a uzavření jsou duálními transformacemi. Další vlastností otevření i uzavření je idempotentnost, což znamená, že opakované použití těchto operací nemění předchozí výsledek.



Obrázek 2.31: Uzavření, vlevo: původní obrázek, vpravo: obrázek po uzavření (velikost jádra: 10)

[5] Uzavření množiny *X* strukturním elementem *B*. [3]

Pro implementaci této transformace v praktické části je z knihovny OpenCV využita funkce *cv2.morphologyEx(previous, cv2.MORPH_CLOSE, final_kernel)*. Nachází se v modulu *modul_closing* v metodě *method_closing(previous, kernel)*. Parametrem *previous je* vstupní obrázek, *cv2.MORPH_CLOSE* říká, že se jedná o uzavření, a parametrem *final_kernel* je velikost jádra.

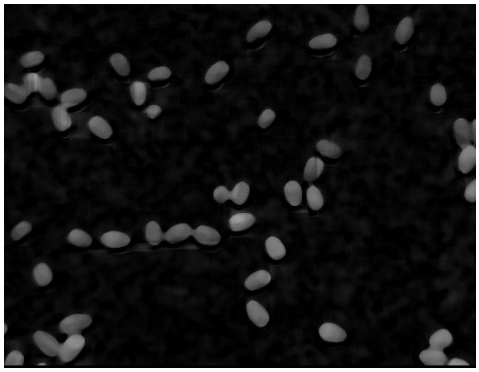
2.7.5 Vrchní část klobouku

Operace vrchní část klobouku je množinovým rozdílem mezi otevřením a původním obrazem, tj.:

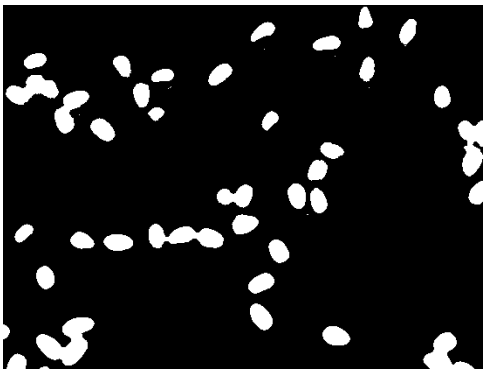
$$X \setminus (X \circ K). \tag{2.25}$$

Tato transformace se používá jako jednoduchý nástroj pro segmentaci objektů, které se v obrazu ve stupních šedi liší od pozadí, i když se jas pozadí mění. Části obrazu, které se nevejdou do strukturního elementu *K* použitého pro otevření, se odstraní. Po odečtení otevřeného obrazu od původního se ukážou jen odstraněné části obrazu. Ty pak lze v obrazu dohledat pomocí jednoduchého prahování.

Pro tuto transformaci je implementován modul s názvem *modul_top_hat* a metoda *method_top_hat(previous, kernel)*. Z knihovny OpenCV je využita funkce *cv2.morphologyEx(previous, cv2.MORPH_TOPHAT, final_kernel)*. *Previous* je vstupní obrázek, *cv2.MORPH_TOPHAT* značí, že se jedná o transformaci vrchní část klobouku a *final_kernel* je velikost jádra.



Obrázek 2.32: Vrchní část klobouku, vysegmentované části (velikost jádra: 30)



Obrázek 2.33: Jednoduché binární prahování (prahová hodnota: 50) předešlého obrázku

2.8 Houghova transformace

Houghova transformace byla navržena k detekci parametricky popsaných objektů, dnes se ale již dá použít i pro detekci oblastí, kde jsou známy rovnice jejich hraničních křivek. Je zvláštním případem Radonovy transformace. [3] Velkou výhodou je, že Houghova transformace není citlivá na zašuměná data. Praktické využití je např. nalezení spoje na deskách plošných spojů nebo předmětů konkrétních tvarů v leteckých či satelitních datech atd. [1]

2.8.1 Pro přímky

Přímka je definována následující rovnicí:

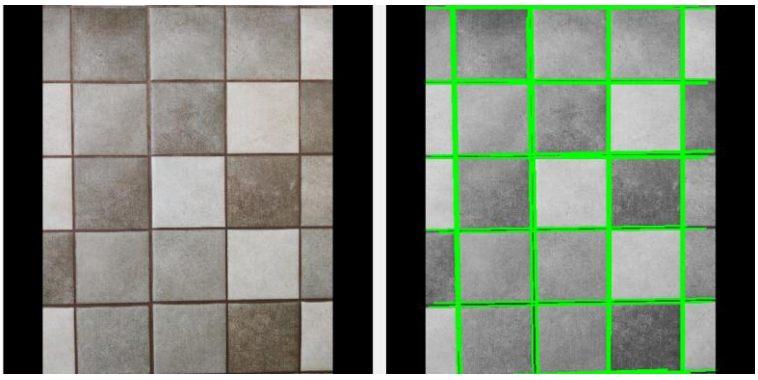
$$y = kx + q, \tag{2.26}$$

nebo parametricky:

$$r = xcos(\theta) + ysin(\theta). \tag{2.27}$$

Neznámá *r* je kolmá vzdálenost od počátku k přímce a θ je úhel tvořený touto kolmou čarou a vodorovnou osou měřenou proti směru hodinových ručiček. Pokud úhel θ nabývá hodnot 0 ° až 360 °, tak r je kladné, pokud 0 ° až 180 °, tak může být *r* i záporné.

Před provedením Houghovy transformace je definován akumulátor (zásobník), kde jsou parametry, které popisují objekt nebo objekty. Dále se obrázek na vstupu po řádcích (a sloupcích) prochází, pokud se nalezne hodnota 1, tak se za úhel θ dosadí hodnoty 0 až 360 a dopočítá se vzdálenost *r*. Na pozice *r* a θ se přičte do akumulátoru 1. Tyto proměnné se vybírají z lokálních maxim akumulátoru. [5]



Obrázek 2.34: Houghova transformace pro přímky (linegap = 10, maxLineGape = 200, threshold = 90)

V praktické části v modulu *modul_hough* je tato část Houghovy transformace pro přímky implementovaná v *method_hough_transform(previous, linelength, linegap, threshold)*. V této metodě se využívá pravděpodobnostní funkce *cv2.-*

HoughLinesP(edges, 1, np.pi/180, threshold, minLineLength=linelength, maxLineGap=linegap). Prvním parametrem je vstupní obraz, druhým a třetím jsou již zmiňované hodnoty *r (1 pixel)* a θ *(1 radián)*, třetím je prahová hodnota, čtvrtým je minimální počet bodů, které mohou tvořit linii a posledním pátým parametrem funkce z knihovny OpenCV je maximální mezera mezi dvěma body na stejném řádku, která by měla být brána v úvahu.

2.8.2 Pro kružnice

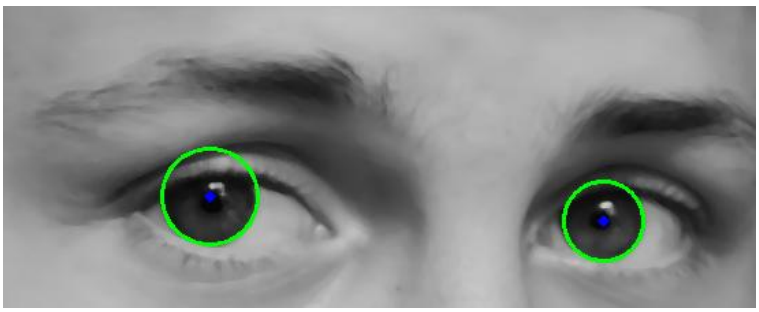
Kružnice je parametricky popsána vztahem:

$$r^2 = (x - x_{center})^2 + (y - y_{center})^2. \quad (2.28)$$

Souřadnice *x, y* jsou souřadnice bodu v prostoru obrazu, body x_{center}, y_{center} představují souřadnice středu kružnice a *r* je poloměr kružnice. Parametrický prostor bude mít tři souřadnice $[x_{center}, y_{center}\ r]$.

Tato transformace se provádí tak, že se pro každý hranový bod *(tj. bílý bod v binárním obraze)* se mění x_{center}, y_{center} a dopočítává se *r.* To znamená, že za potenciální střed kružnice je považován každý nehranový bod. Pokud se často vyskytuje potencionální střed $[x_{center}, y_{center}]$ v daném prostoru pro konkrétní *r*, je velmi pravděpodobné, že se v obrazu vyskytuje kružnice s daným středem $[x_{center}, y_{center}]$ a poloměrem *r.*

Akumulátor má při detekci kružnic tři hodnoty. Tím pádem je nutné vytvořit datovou strukturu, ve které se bude inkrementovat hodnota odpovídající trojici hodnot $[x_{center}, y_{center}, r]$.[7]



Obrázek 2.35: Houghova transformace pro kružnice (vyšší prahová hodnota: 180, prahová hodnota akumulátoru 28)

I Houghova transformace pro kružnice se nachází v modulu *modul_hough* ve funkci *method_hought_circle(previous, param1, param2).* Z knihovny se využívá metody *cv2.HoughCircles().* První parametr je vstupní obraz, druhým parametrem je vyšší prahová hodnota ze dvou předaných Cannymu hranovému detektoru

a třetím parametrem je prahová hodnota akumulátoru pro středy kružnic ve fázi detekce (čím menší je, tím více jsou detekovány falešné kružnice). [5]

2.9 Detekce a identifikace objektů

2.9.1 Barvení oblastí

Barvení oblastí je jednou z metod pro identifikaci oblastí. Výsledkem je, že každá oblast v obraze je označená unikátním číslem. Nejvyšší číslo může, ale nemusí, udávat počet oblastí. Požadavkem pro algoritmus barvení oblastí je, že dvě různé oblasti nesmí být označeny stejným číslem. Před barvením oblastí se musí provést některá z metod ze segmentace obrazu. [1]

Algoritmus pro barvení oblastí se nazývá *Connected Component Labeling (CCL)*. Ten získává počet objektů v obraze pomocí sekvenčního označování regionů. V tomto algoritmu se k obarvení oblastí obrázku využívá dvou průchodů o třech procesech.

1) Pro každý pixel objektu v obrazu se vybere na základě jeho sousedních pixelů dočasný zástupce.
2) Tento zástupce je následně nahrazen značkou, která reprezentuje všechny ekvivalentní zástupce všech spojených komponent.
3) Značka je přiřazena k danému pixelu. [8]

Obrázek se prochází zleva doprava a shora dolů, postupně se každému pixelu objektu přiřazuje prozatímní označení v závislosti na okolních bodech. Tyto body jsou určeny maskou, která vybírá čtyři, šest nebo osm připojených komponent (pixelů). [9]

Z knihovny OpenCV je využita funkce *label(image)* z balíčku *scipy.ndimage*. Realizace metody barvení oblastí se nachází v modulu *modul_coloring* v metodě *method_coloring(image)*. Parametrem *image* je vstupní obrázek.

Obrázek 2.36: Barvení oblastí, vlevo: původní obrázek, vpravo: obrázek po jednoduchém binárním prahování a po barvení oblastí

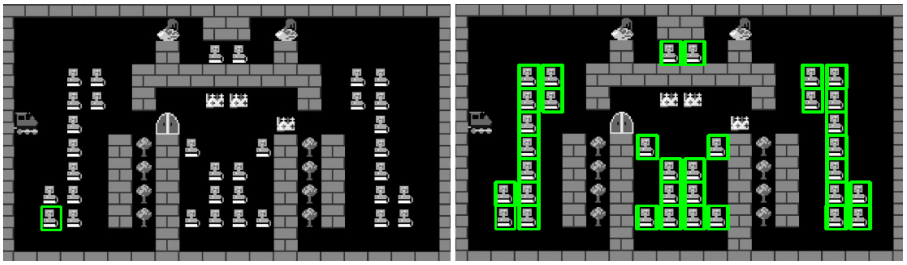
2.9.2 Vyhledávání vzorů

Vyhledávání vzorů je technika strojového vidění na vysoké úrovni, která identifikuje části obrazu, odpovídající předdefinované šabloně. [6] V obrazu se tedy hledá pixelová kopie šablony, jednotlivé hodnoty pixelů vzoru se porovnávají s hodnotami pixelů šablony (pixel po pixelu). [1] Šablona by měla být menší než obraz, který je analyzován.

Pokročilejší algoritmy vyhledávání vzorů umožňují najít šablonu v obrázku bez ohledu na orientaci a jas. Použití takových algoritmů je ale omezeno dostupnou výpočetní silou. [6]

V praktické části si uživatel může vybrat, zda chce najít pouze jeden nebo více výskytů šablony v obraze. Obě dvě možnosti jsou implementovány v modulu *modul_searching* v metodě *method_searching(previous, template, method)*. Poslední parametr v této metodě určuje, jestli se bude hledat pouze jeden výskyt šablony nebo více. Z knihovny OpenCV je využitá funkce *cv2.matchTemplate(img, template, method)*, kde prvním parametrem je vstupní obrázek, druhým je šablona a třetím je metoda porovnávání, jež je defaultně nastavená na *cv2.TM_CCOEFF_NORMED*, kvůli poměrně dobrým výsledkům. Princip této funkce je v tom, že je šablona postupně posouvána přes vstupní obrázek a porovnávána.

Obrázek 2.37: Příklad šablony



Obrázek 2.38: Vyhledávání vzorů, vlevo: vyhledávání jednoho vzoru, vpravo: vyhledávání více vzorů

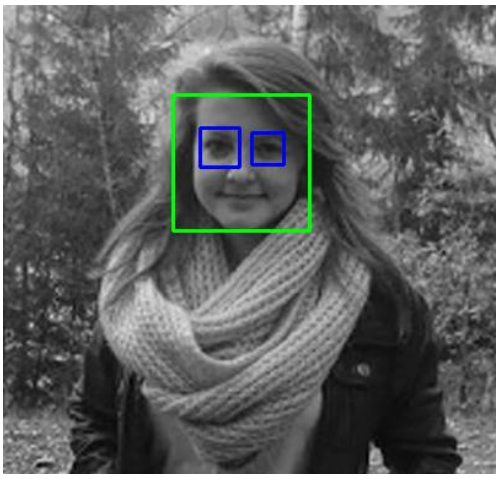
2.9.3 Detekce obličeje pomocí AdaBoost a Haar příznaků

Detekce obličeje pomocí AdaBoost a Haar příznaků je efektivní metoda, založená na strojovém učení, kde je kaskádová funkce trénovaná z mnoha pozitivních a negativních obrazů. Pozitivními obrazy jsou takové obrazy, kde se vyskytují tváře a negativními obrazy jsou obrazy bez tváří. Tím vzniknou proškolené kaskádové klasifikátory, které jsou následně využity k detekci obličejů na jiných obrázcích. [5]

Tato metoda byla navržená Paulem Violou a Michaelem Jonesem v roce 2001. AdaBoost (Adaptive Boosting) je klasifikační algoritmus, který využívá pro učení slabé klasifikátory $h_t(x)$, které jsou vybírány z množiny klasifikátorů H, a jejichž lineární kombinací vzniká nelineární silný klasifikátor $H(x)$. Haar příznaky jsou skupinou příznaků. Čím je tato množina příznaků obsáhlejší, tím existuje větší pravděpodobnost výběru slabého klasifikátoru s vyšší mírou přesnosti.

Výhodou tohoto detektoru je rychlost, dostatečná spolehlivost, nezávislost na osvětlení a velikosti sledovaného objektu. [10] Nemusí se trénovat pouze na obličejích, ale i na různých jiných objektech. [5]

Tato metoda je v systému implementována v modulu *modul_adaboost_haar* v metodě *method_face_recognition(previous)*. Z knihovny OpenCV jsou nadefinované klasifikátory pro obličej a pro oči (pro obličej: *cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml'),* obdobně pro oči) a funkce *face_cascade.detectMultiScale(gray, 1.3, 5)*. Prvním argumentem je vstupní obrázek, druhým je *minNeighbors*, které ovlivňuje množství pozitivních výsledků, a třetím parametrem je *scaleFactor*, který určuje kompromis mezi přesností a rychlostí detekce.

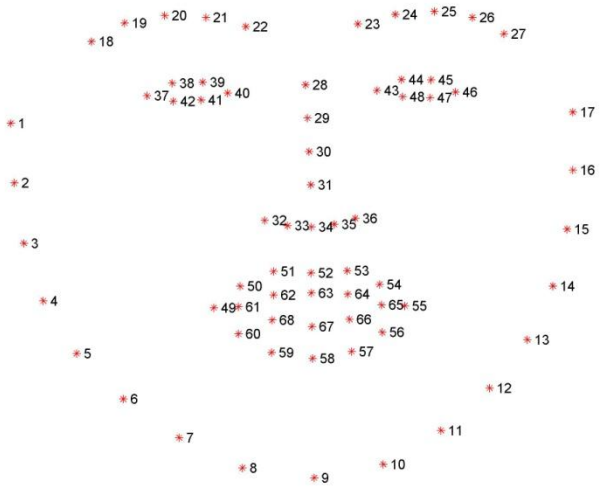


Obrázek 2.39: Detekovaný obličej pomocí AdaBoost a Haar příznaků

2.9.4 Detekce obličeje pomocí Dlib

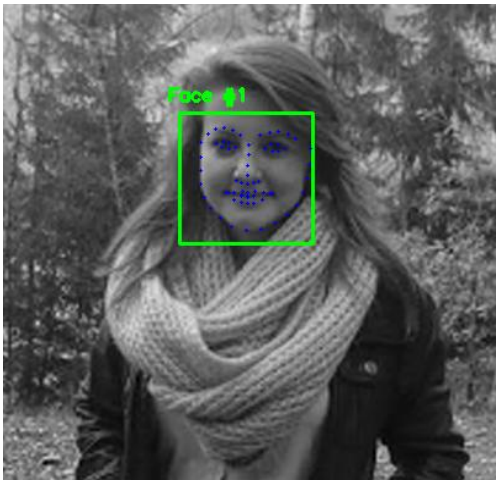
Dlib je univerzální multiplatformní softwarová knihovna napsaná v programovacím jazyce C++. Obsahuje širokou škálu algoritmů strojového učení. [11] Tvůrcem této knihovny je Davis King.

Metoda detekce obličeje pomocí Dlib hledá orientační body tváře. Ty se používají k lokalizaci a reprezentaci hlavních částí obličeje (oči, nos, ústa, obočí). V knihovně Dlib se nachází proškolený detektor těchto hraničních bodů a k odhadu jejich polohy využívá 68 *(x, y)* souřadnic. [12]



Obrázek 2.40: Vizualizace 68 souřadnic orientačních bodů tváře (datová sada: iBUG300-W) [12]

V praktické části je tato metoda realizována v modulu *modul_dlib* v metodě *method_dlib_face_recognition(previous)*. Z knihovny Dlib je definován detektor *dlib.get_frontal_face_detector()* a prediktor *dlib.shape_predictor("dat/shape_predictor_68_face_landmarks.dat")*, dále jsou využity metody z knihovny *imutils.face_utils.*



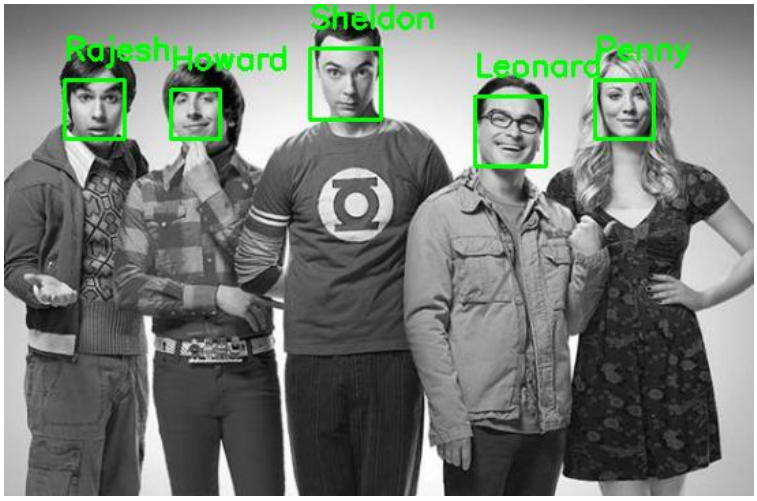
Obrázek 2.41: Detekovaný obličej pomocí knihovny Dlib

2.9.5 Identifikace obličeje pomocí Face_recognition

Identifikace obličeje funguje na základě hlubokého metrického učení. Autorem modulu Face_recognition je Adam Geitgey.

Místo souřadnic, které ohraničují určité oblasti jako tomu je např. u Detekce obličeje pomocí Dlib, toto učení vytváří vektor. Identifikace se skládá ze čtyř kroků. Prvním krokem je detekce všech tváří v obrázku. Druhým je nalezení orientačních bodů obličejů a vycentrování daného obličeje pomocí těchto bodů. Vycentrování se provádí pomocí Geometrické transformaceí obrazu. Třetím krokem je rozeznání tváří od sebe. Zde přichází na řadu trénování sítě. Síť bude při trénování generovat 128 měření pro každou tvář. Proces trénování funguje tak, že se síť dívá na tři obrázky najednou (dva obrázky jsou se stejnou tváří a jeden obrázek s neznámou tváří). Posledním krokem je vyhledání jména osoby, které se nachází v databázi známých obličejů. [14]

V praktické části je tato metoda implementována v modulu *mdul_face_recognition* v *method_face_recognition(previous, dataset, method)*. Prvním parametrem je vstupní obrázek, druhým parametrem je vytvořený dataset, který si uživatel může vytvořit pomocí skriptu *encode_faces.py*, a třetím je typ metody pro rozpoznávání. Uživatel si může vybrat mezi metodami HOG *(histogram orientovaných gradientů)* a CNN *(konvoluční neuronové sítě).* Je zde naimportován modul *face_recognition*, ve kterém jsou metody pro rozpoznávání obličeje. Pro ověření funkčnosti modulu *face_recognition* byl vytvořen dataset *encodings.pickle* z fotografií pěti postav ze seriálu Teorie velkého třesku.



Obrázek 2.42: Face_recognition – identifikované obličeje

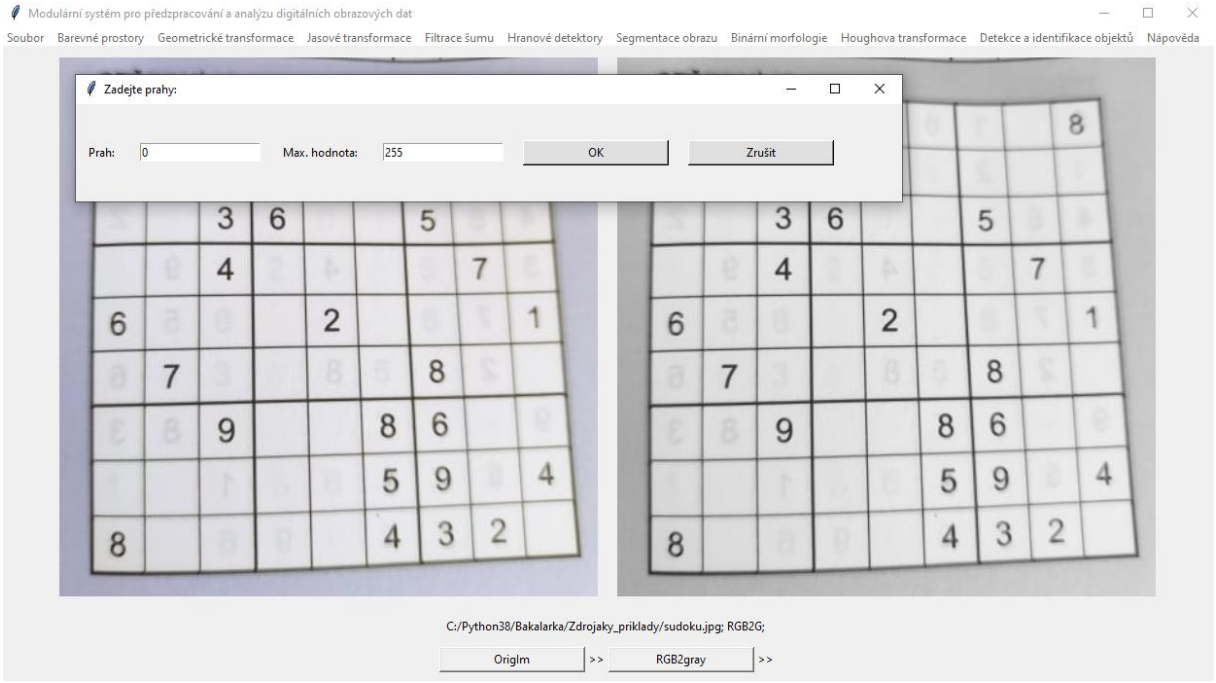
3 Využití systému

Tento systém by se mohl využít jako výuková aplikace pro studenty a ukázka toho, co vše se dá realizovat s knihovnou OpenCV.

3.1 Ukázka – práce se systémem

V této podkapitole je ukázka toho, jak lze se systémem pracovat.

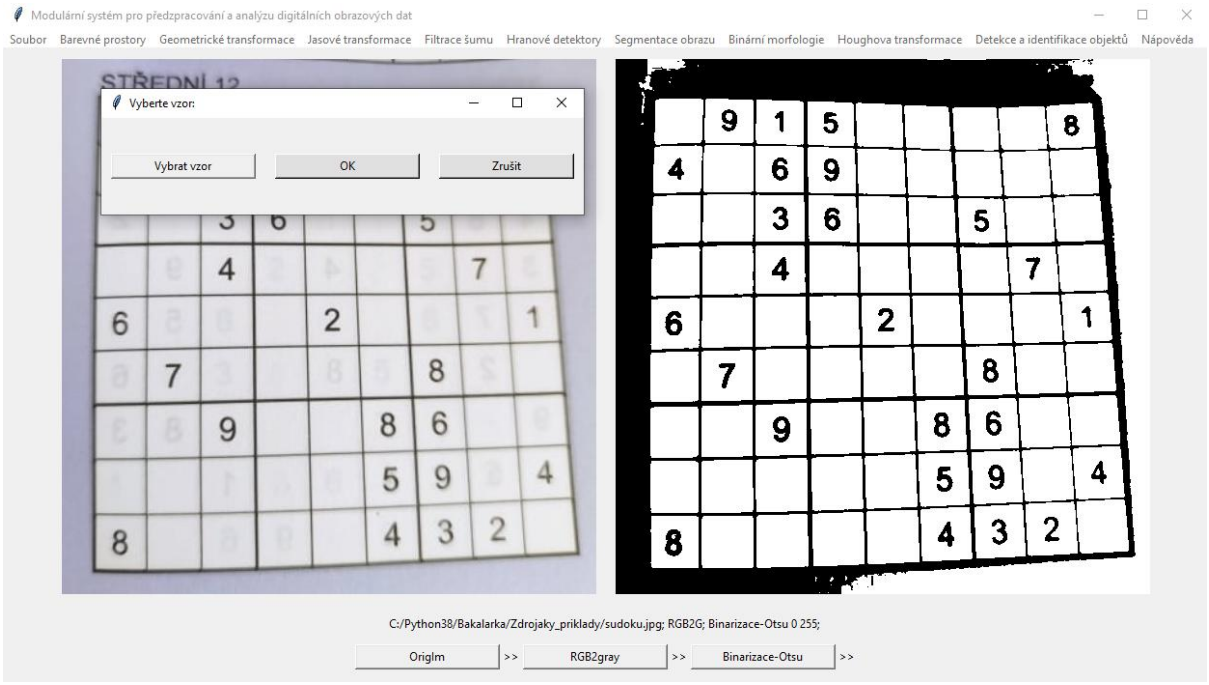
Nejprve je vybrán obrázek, který je následně převeden do stupňů šedi. Další zvolenou operací je binarizace Otsu, kde musí uživatel zadat prahovou a maximální hodnotu (viz o binarizaci Otsu na stránce 3131).



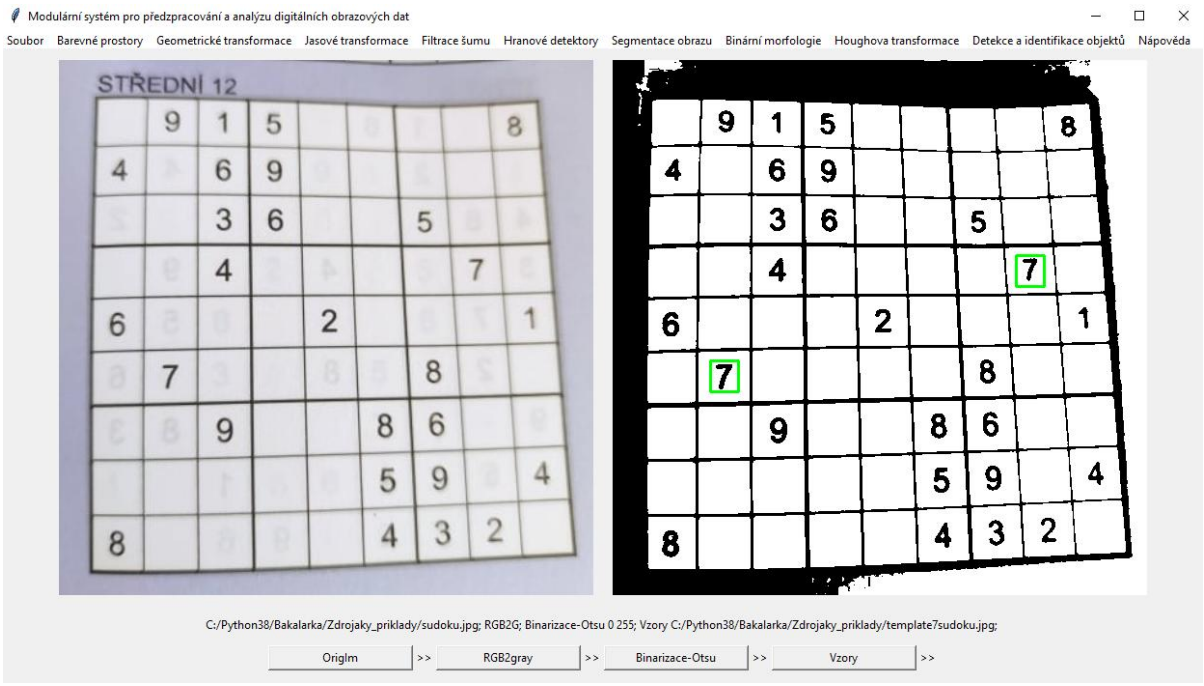
Obrázek 3.1: Ukázka I

Na obrázku 3.1 v pravé části je výsledný obrázek. Dále je vybrána operace vyhledání vzorů. Uživateli se otevře dialogové okno, pomocí něhož vybere požadovanou šablonu vzoru (obrázek 3.2), který chce v obrázku vyhledat. Detekované objekty jsou na obrázku 3.4.

Obrázek 3.2: Ukázka II



Obrázek 3.3: Ukázka III



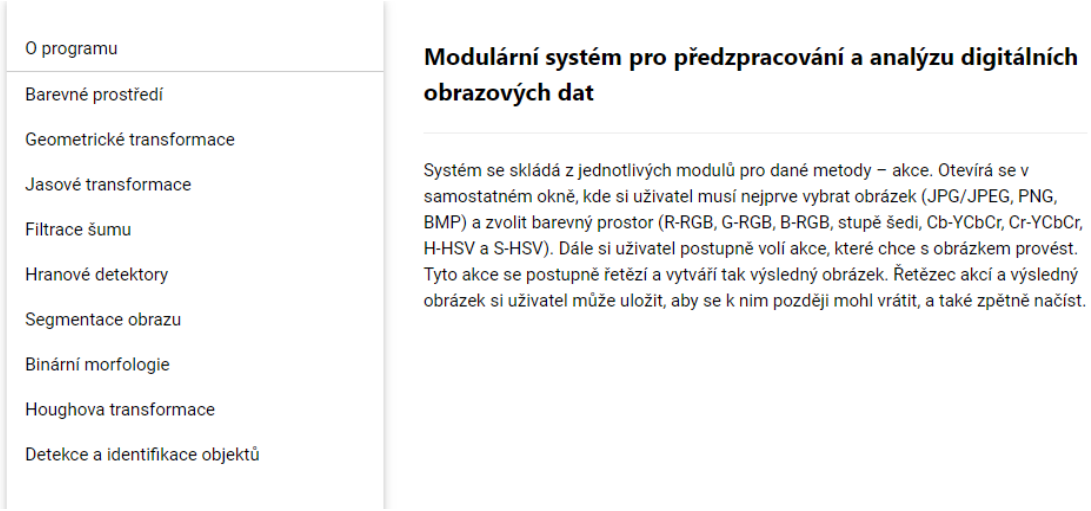
Obrázek 3.4: Ukázka IV

Pokud si tento řetězec akcí uživatel uloží, pak se tento řetězec ukládá do souboru *list_of_actions.xml* s touto strukturou:

```
<?xml version='1.0' encoding='UTF-8'?>
<methods>
  <Method ID="1" Name="C:/Python38/Bakalarka/Zdrojaky_priklady/sudoku.jpg"/>
  <Method ID="2" Name="RGB2G"/>
  <Method ID="3" Name="Binarizace-Otsu" Number="0" Temp="255"/>
  <Method ID="4" Name="Vzory" Number="C:/Python38/Bakalarka/Zdrojaky_priklady/template7sudoku.jpg"/>
  <Method ID="5" Name=""/>
</methods>
```

Obrázek 3.5: Ukázka V

K dispozici jsou i uživatelské stránky s nápovědou.



Obrázek 3.6: Ukázka VI

Závěr

V rámci této bakalářské práce byl vytvořen systém, který se skládá z jednotlivých modulů pro zpracování a rozpoznávání obrazu z knihovny OpenCV.

Práce s tímto systémem je jednoduchá a intuitivní, neboť jsou u některých metod (pokud je třeba) předvyplněny přibližné hodnoty k zobrazení relevantního výsledku. Dále je v tomto systému možnost otevření uživatelských stránek s nápovědou k jednotlivým metodám. Výsledný obrázek nebo řetězec si uživatel může uložit a zpětně načíst.

Systém by bylo možné vylepšit o dynamické posuvníky pro změnu parametrů z důvodu rychlejší a pohodlnější práce. Dále by se dal rozšířit o další moduly a tím tak rozšířit jeho funkcionalitu o další metody počítačového vidění.

Práce na systému byla velice zajímavá. Nejen při samotné tvorbě kódu, ale i při objevování možností, které dnešní technologie v oboru umělé inteligence nabízejí.

Využití popsaných metod je velmi široké. Některé lze uplatnit prakticky každý den, např. při vytváření fotografií v mobilních přístrojích. Ihned po pořízení těchto obrazových souborů může totiž docházet k jejich ukládání podle různých algoritmů tak, aby je bylo možné následně vyhledávat podle požadovaných kritérií např. podle obličeje, tvaru atp.

Jinou možností může být úprava obrazu při digitalizaci a následného zveřejňování fotografií v mapových systémech. Zde je žádoucí, aby neexistovala citlivá data, jakým bezesporu jsou konkrétní obličeje lidí, registrační značky aut atp. Při takovém zveřejňování musí dojít k automatizovanému znehodnocení identifikačních prvků.

Je až fascinující, k čemu všemu se dnes standardně metody počítačového vidění používají.

Vytvořený systém lze snadno použít pro názorné ukázky činností, které v oblasti umělé inteligence využíváme každý den. Především je vhodný pro výukové účely, pro rychlé zobrazení jednotlivých metod, které se nachází v knihovně OpenCV, pro rychlou analýzu obrazu a zároveň pro vytvoření řetězce jednotlivých událostí, kde si člověk může velice rychle vyzkoušet návrh řetězce metod zpracování a rozpoznávání obrazu, které by měly vést k finálnímu rozpoznávání.

Zdroje

[1] ŠONKA, Milan, Václav HLAVÁČ a Roger BOYLE. Image Processing, analysis, and machine vision. 4th ed. Stamford: Cengage Learning, 2015. ISBN 1-133-59360-7.

[2] NAVRÁTIL, Pavel. Počítačová grafika a multimédia. Kralice na Hané: Computer Media, 2007. ISBN 978-80-86686-77-6.

[3] HLAVÁČ, Václav a Miloš SEDLÁČEK. Zpracování signálu a obrazu. Praha, 1999. Pracovní verze skripta v tisku pro studenty. Elektrotechnická fakulta ČVUT v Praze.

[4] HLAVÁČ, Václav a Jan KYBIC. Předzpracování v prostoru obrazů: Geometrické transformace. Center for Machine Perception @ CTU in Prague [online]. Praha: Václav Hlaváč, c1996–2021 [cit. 2021-5-1].
Dostupné z:
http://cmp.felk.cvut.cz/cmp/courses/33DZOzima2005/slidy/geometrickeTransformace.pdf

[5] OpenCV-Python Tutorials: OpenCV [online]. Alexander Mordvintsev, c2013 [cit. 2021-5-1].
Dostupné z: https://opencv-python-tutroals.readthedocs.io/en/latest/

[6] Puneet and Naresh Garg. Article: Binarization Techniques used for Grey Scale Images. International Journal of Computer Applications 71(1):8-11, June 2013 [cit. 2021-5-2]. Dostupné z: https://research.ijcaonline.org/volume71/number1/pxc3888533.pdf

[7] BRADSKI, Gary R. a Adrian KAEHLER. Learning OpenCV. Sebastopol: O'Reilly, c2008. ISBN 978-0-596-51613-0.

[8] ALAWI, Mahmoud, Othman KHALIFA a Rafiqul ISLAM. Performance Comparison of Background Estimation Algorithms for Detecting Moving Vehicle. DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING, International Islamic University Malaysia. World Applied Sciences Journal 21 (Mathematical Applications in Engineering) [online]. 2013 [cit. 2021-5-1].
Dostupné z: http://www.idosi.org/wasj/WASJ21(mae)13/20.pdf

[9] GABBUR, Prasad, Hong HUA a Kobus BARNARD. A fast connected components labeling algorithm and its application to real-time pupil detection. Machine Vision and Applications [online]. 2010, 21(5), 779-787 [cit. 2021-5-1]. ISSN 0932-8092. Dostupné z: doi:10.1007/s00138-009-0183-1

[10] GEITGEY, Adam. Machine Learning is Fun!: Part 4: Modern Face Recognition with Deep Learning. Adam Geitgey [online]. Adam Geitgey, 2016, 2016 [cit. 2021-5-1].
Dostupné z: https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78

[11] FRIEDMAN, Jerome, Trevor HASTIE a Robert TIBSHIRANI. Additive logistic regression: a statistical view of boosting (With discussion and a rejoinder by the authors). The Annals of Statistics [online]. 2000, 28(2) [cit. 2021-5-2]. ISSN 0090-5364.
Dostupné z: doi:10.1214/aos/1016218223

[12] Template Matching. Adaptive Vision [online]. Poland: Adaptive Vision Sp. z o.o., c2007-2017 [cit. 2021-5-2]. Dostupné z: https://docs.adaptive-vision.com/4.7/studio/machine_vision_guide/TemplateMatching.html

[13] TOMICZKOVÁ, Světlana. Pokroky matematiky, fyziky a astronomie: Minkowského množinové operace a jejich aplikace. Czech Digital Mathematics Library [online]. Plzeň: Světlana Tomiczková, 2007 [cit. 2021-5-1].
Dostupné z: https://dml.cz/bitstream/handle/10338.dmlcz/141371/PokrokyMFA_52-2007-4_6.pdf

[14] Dlib-ml: A Machine Learning Toolkit [online]. Davis E. King, 2009 [cit. 2021-5-2]. Dostupné z: http://dlib.net/ml.html

Obsah přiloženého CD

- text bakalářské práce ve formátu PDF
- použitá online literatura ve formátu PDF
- zdrojové kódy k systému, k jednotlivým modulům a nápovědě
- obrázky a dataset pro práci s modulem