TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Systém pro vizualizaci metod a algoritmů z OpenCV knihovny

Bakalářská práce

Studijní program: B2646 Informační technologie
Studijní obor: Informační technologie

Autor práce: **Václav Halama**
Vedoucí práce: doc. Ing. Josef Chaloupka, Ph.D.
Ústav informačních technologií a elektroniky

Liberec 2021

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání bakalářské práce

Systém pro vizualizaci metod a algoritmů z OpenCV knihovny

Jméno a příjmení: **Václav Halama**
Osobní číslo: M18000075
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 a knihovnou OpenCV.
2. Navrhněte a realizujte systém (v Pythonu) pro vizualizaci výsledků metod a algoritmů z knihovny OpenCV.
3. Tento systém by měl obsahovat graficky přívětivé prostředí.
4. Pro každou metodu (algoritmus) by mělo být možné z grafického prostředí nastavovat její parametry a další vlastnosti.

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

Prohlašuji, že svou bakalářskou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé bakalářské práce.

Jsem si vědom toho, že na mou bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé bakalářské práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li bakalářskou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS/STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má bakalářská práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

Datum: .....

Podpis: .....

Poděkování

Rád bych poděkoval vedoucímu mé práce panu doc. Ing. Josefu Chaloupkovi, Ph.D. za jeho čas, cenné rady, vstřícnost při pravidelných konzultacích, a hlavně za jeho trpělivost.

Abstrakt

Tato bakalářská práce se zabývá metodami a algoritmy používanými v oblasti zpracování obrazu a počítačového vidění. Zejména je kladen důraz na metody a algoritmy implementované v knihovně s otevřeným zdrojovým kódem pro počítačové vidění a strojové učení OpenCV. Hlavním cílem práce bylo vytvořit systém s uživatelsky přívětivým grafickým rozhraním, který umožní vizualizaci výsledků zkoumaných metod a algoritmů z knihovny OpenCV. Pokud metoda obsahuje volitelné parametry je uživateli umožněno je měnit z grafického prostředí.

Práce je rozdělena do dvanácti kapitol. První kapitola obsahuje krátký přehled o vytvořeném systému a jeho částech. Ve zbylých jedenácti kapitolách je vždy teoretický popis metod a algoritmů obsažených v dané části systému, který je následován stručným popisem praktické implementace této části systému. Vytvořený systém je napsaný v programovacím jazyce Python.

Klíčová slova

Zpracování obrazu, OpenCV, Počítačové vidění, Segmentace obrazu, Detekce hran

Abstract

This bachelor thesis deals with methods and algorithms used in image processing and computer vision. The emphasis is mainly on methods and algorithms implemented in the open-source library for computer vision and machine learning OpenCV. The main goal of this work was to create a system with a user-friendly graphical interface that would allow the user to visualize the results of examined methods and algorithms from the OpenCV library. If any method contains optional parameters, the user can change them from the graphical interface.

The work consists of twelve chapters. The first chapter contains a brief overview of the created system and its parts. Each of the remaining eleven chapters consists of a theoretical description of the methods and algorithms contained in a given segment of the system, followed by a brief description of the practical implementation of this part of the system. The system is written in the Python programming language.

Keywords

Image processing, OpenCV, Computer vision, Image segmentation, Edge detection

Obsah

Úvod **10**

1 Systém pro vizualizaci metod a algoritmů **11**

2 Barevné prostory **13**

3 Geometrické transformace **15**

4 Jasové transformace **17**

5 Filtrace šumu **19**

5.1 Průměrovací filtr 19

5.2 Filtr medián 19

5.3 Gaussův filtr 19

5.4 Bilaterální filtr 20

5.5 Okno Filtrace šumu 21

6 Hranové detektory **22**

6.1 Sobelův hranový detektor 22

6.2 Laplaceův hranový detektor 22

6.3 Cannyho hranový detektor 23

6.4 Okno Hranové detektory 23

7 Segmentace obrazu **25**

7.1 Barvení oblastí 25

7.2 Algoritmus Rozvodí 26

7.3 Okno Segmentace obrazu 27

8 Morfologické transformace **28**

8.1 Dilatace a eroze 28

8.2 Otevření a uzavření 30

8.3 Okno Morfologické transformace 31

9 Houghova transformace **32**

10 Hledání vzorů **34**

11 Parametrizace objektů **36**

11.1 Harrisův detektor rohů 36

11.2 Shi-Tomasův detektor rohů 36

11.3 SIFT příznaky 37

11.4 SURF příznaky 37

11.5 Okno Parametrizace objektů 39
12 Detekce a identifikace obličeje 41
12.1 Viola-Jones detektor 41
12.2 Detektor z knihovny dlib 42
12.3 Identifikace pomocí knihovny face_recognition 43
12.4 Okno Detekce a identifikace obličeje 43
Závěr 45
Literatura 46

Úvod

Zpracování obrazu a počítačové vidění je důležitou součástí spousty reálných systémů, u kterých je nutné strojové zpracování obrazových dat. Z tohoto důvodu existuje obrovské množství metod a algoritmů, které se v této oblasti využívají. Aby bylo možné tyto metody rychle aplikovat ve vlastních, specializovaných systémech, vznikají knihovny pro různé programovací jazyky, které obsahují efektivní implementace těchto metod. Jednou z těchto knihoven je OpenCV.

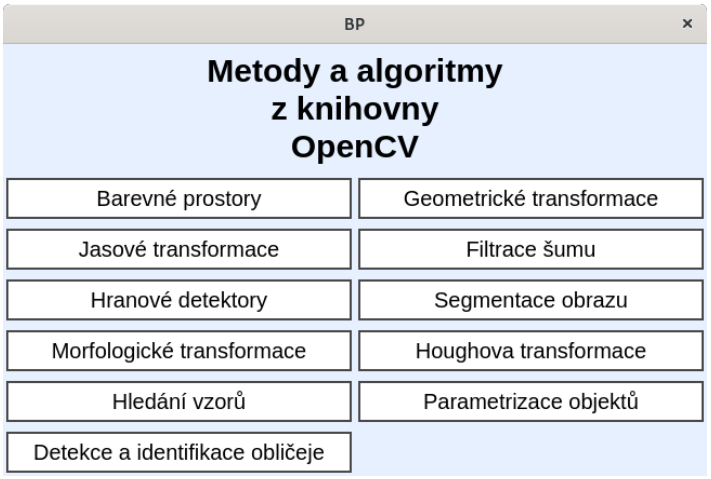
OpenCV (Open Source Computer Vision Library) je velmi rozsáhlá knihovna implementující metody a algoritmy zejména z oblasti počítačového vidění a strojového učení. V současnosti obsahuje celkem přes 2500 optimalizovaných algoritmů. Knihovna je s otevřeným zdrojovým kódem a každý si jí tedy může stáhnou a zdarma používat i pro komerční účely. Byla napsána v jazyce C++, ale obsahuje rozhraní pro jazyky Java, Matlab a Python. Nachází reálná uplatnění například v dopravě při rozpoznávání dopravního značení nebo detekci chodců, dále pak u zobrazovacích metod v medicíně, v automatizované výrobě nebo v bezpečnostních systémech pro sledování osob.

Obsahem knihovny jsou, jak klasické metody pro zpracování obrazu jako je například filtrace šumu, detekce hran, nebo segmentace obrazu, tak i nejmodernější algoritmy z oblasti počítačového vidění a strojového učení, které lze využít například pro detekci a identifikaci obličejů, sledování pohybujících se objektů, vytváření 3D modelů objektů z obrazu atd.

Cílem této práce je seznámit se s různými metodami a algoritmy z oblasti zpracování obrazu a počítačového vidění, dále pak vytvořit systém, pomocí kterého bude možné vizualizovat výsledky těchto metod a algoritmů implementovaných zejména knihovnou OpenCV.

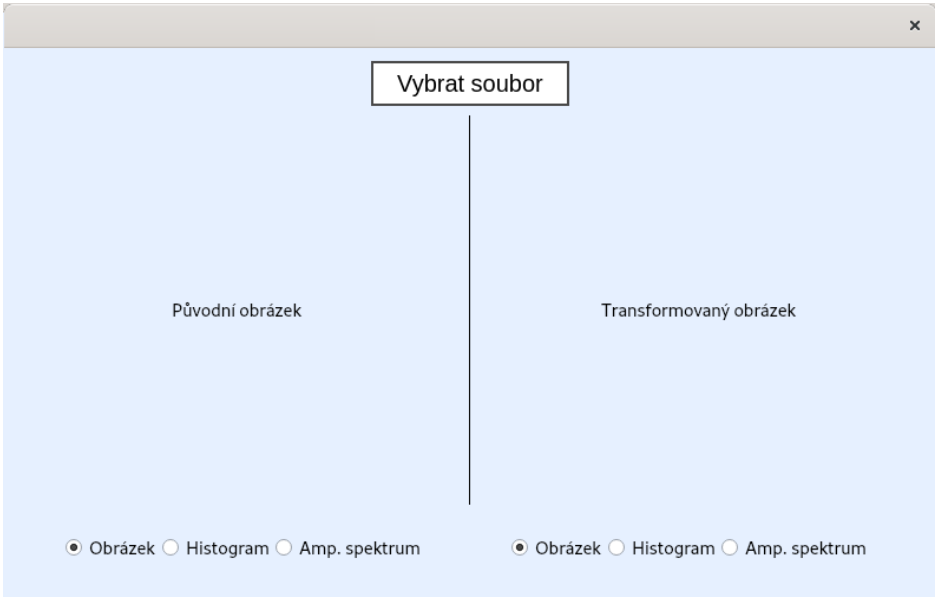
1 Systém pro vizualizaci metod a algoritmů z OpenCV

Vytvořený systém se skládá celkem z 11 částí. Z důvodu, že je systém takto rozsáhlý, je teoretická i praktická část práce v jednotlivých kapitolách spojena. Každá část systému je zastoupena vlastním oknem, které obsahuje prostředí pro vizualizaci metod zkoumaných v rámci jednotlivých kapitol této práce. K dílčím oknům lze přistupovat z jednoduchého menu (Obrázek 1.1).



Obrázek 1.1: Menu systému

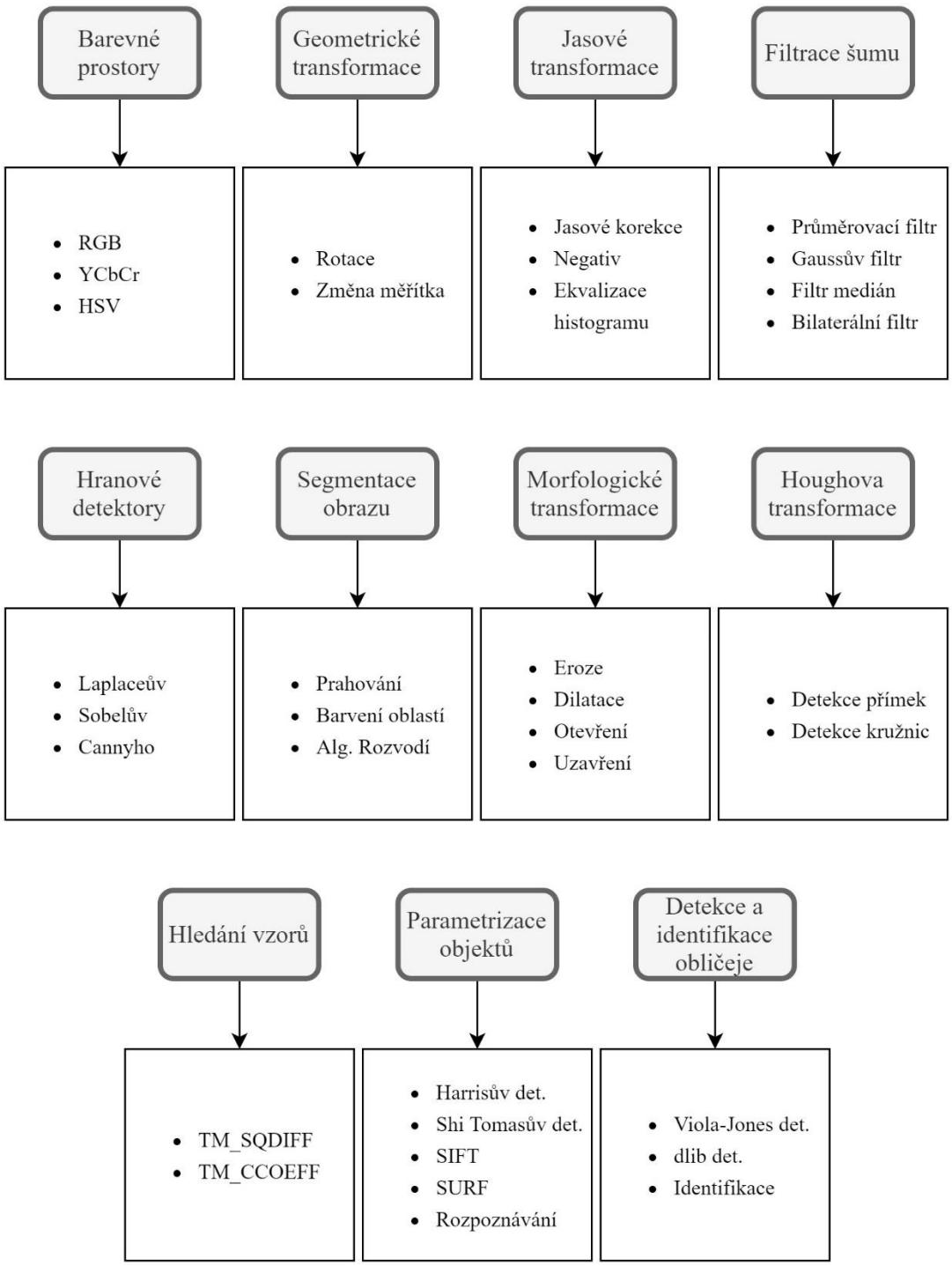
Všechna okna mají podobné rozložení, vždy je rozděleno do dvou částí. V levé části je zobrazen původní obrázek a v pravé obrázek po transformaci (Obrázek 1.2). Výběr obrázku pro aplikaci metod probíhá přes tlačítko „Vybrat soubor“ v horní části okna. Dále pak lze přepínat režim zobrazení pro každý z obrázků mezi obrázkem, jeho



Obrázek 1.2: Rozvržení oken

histogramem nebo jeho amplitudovým spektrem pomocí přepínačů umístěných pod obrázky.

Obsah jednotlivých oken je naznačen na blokovém diagramu na Obrázku 1.3. Detailnější popisy a ukázky jednotlivých oken jsou uvedeny na konci každé z následujících kapitol.



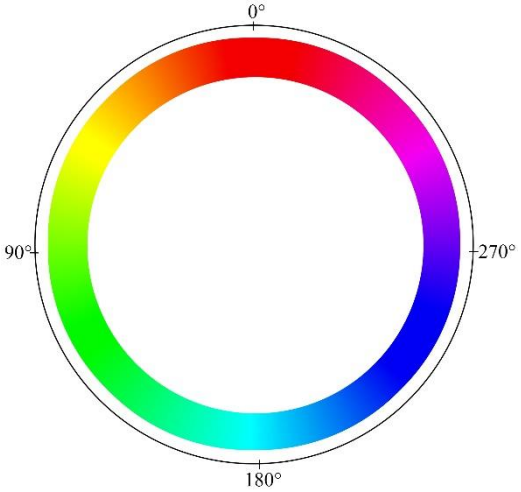
Obrázek 1.3: Obsah jednotlivých oken

2 Barevné prostory

Barevný prostor představuje způsob, jakým lze barvy reprezentovat numericky v počítači. Mezi jednotlivými prostory lze převádět, avšak každý barevný prostor má různý gamut. Gamut je množina barev, které v tomto barevném prostoru můžeme zobrazit. Pokud se při transformaci ocitne barva mimo gamut cílového prostoru, je tato informace ztracena a barva je zobrazena pouze přibližně [1].

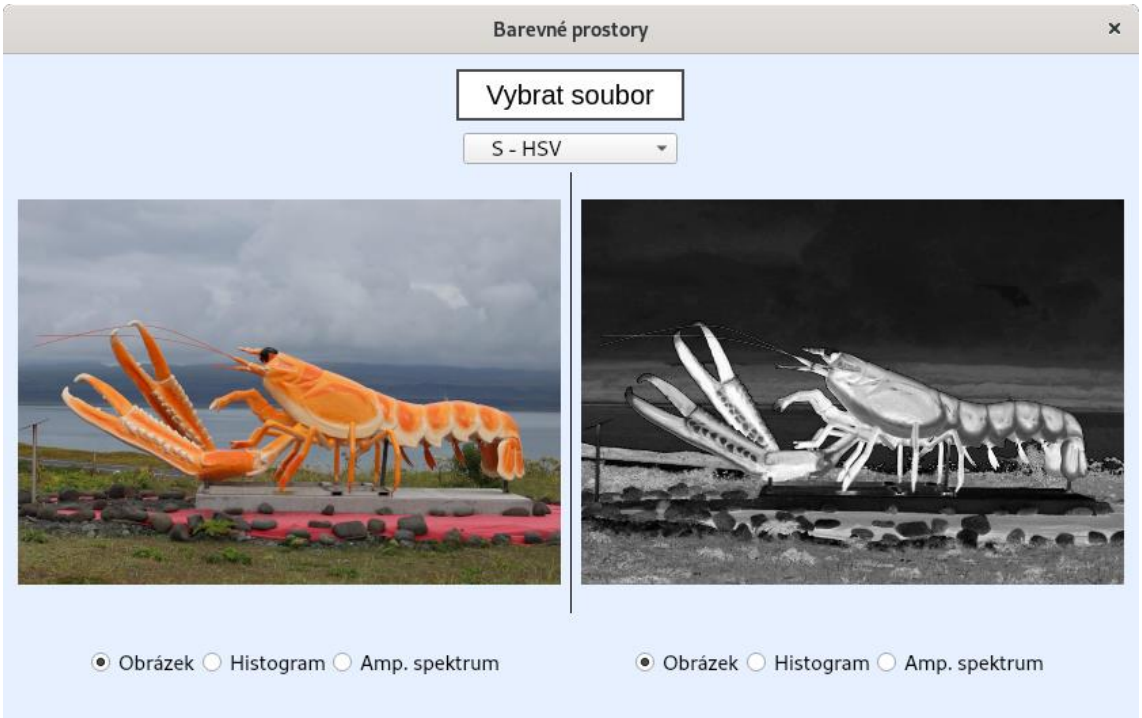
Nejznámější a nejpoužívanější z barevných prostorů je RGB. Hlavní složky tohoto prostoru jsou R – červená (red), G – zelená (green) a B – modrá (blue). Hodnota jednotlivých barev je vyjádřena jako vektor tří čísel, z nichž každé reprezentuje intenzitu dané složky. Mezi nejpoužívanější verze RGB patří její 24bitová implementace. Pro každou z jednotlivých složek je tedy vyhrazeno 8 bitů. Její hodnota je typicky reprezentována hodnotou z rozsahu 0-255 nebo 0-1 v případě normovaných hodnot. Potom vektor (0, 0, 0) představuje černou, (255, 255, 255) bílou, (255, 0, 0) obyčejnou červenou atd.

Dalším z často používaných barevných prostorů je YCbCr. Tento prostor vznikne lineární transformací složek RGB. Složka Y představuje intenzitu jasu a podobá se převodu RGB obrazu do stupňů šedi. Cb a Cr je modrá, resp. červená chromatická složka.



Obrázek 2.1: Vizuální reprezentace složky H (HSV)

Barevný prostor HSV popisuje barvy způsobem, který je nejbližší lidskému vnímání barev. Jeho složky tvoří H – odstín/barevná složka (hue), S – sytost barvy (saturation) a V – intenzita jasu (value). Odstín se typicky vyjadřuje ve stupních (Obrázek 2.*1*), radiánech, nebo je hodnota normovaná [2].



Obrázek 2.2: Okno Barevné prostory

V okně Barevné prostory (Obrázek 2.2) je po vybrání obrázku možné zobrazit jednotlivé barevné složky z barevných prostorů RGB, HSV a YCbCr. Tato složka je navolena v comboboxu pod tlačítkem pro načtení souboru a je ihned zobrazena v pravé části okna.

3 Geometrické transformace

Geometrické transformace slouží k přepočtu původních souřadnic pixelů na souřadnice nové. V principu není žádný rozdíl mezi jednotlivými geometrickými transformacemi jako jsou například rotace nebo změna měřítka. U digitálních obrazů navíc umožňují odstranění některých geometrických zkresleních vzniklých při pořízení snímku.

Geometrická transformace je funkce T, která zobrazuje původní souřadnice bodu (x, y) do bodu (x', y'). Při výpočtu transformace musíme počítat se spojitým výstupem z transformační funkce. Výstupní souřadnice nemusí být celá čísla a nemusí tedy odpovídat konkrétním souřadnicím v transformovaném obraze.

Abychom mohli geometrické transformace zapsat pomocí jednoduchého maticového vztahu, je nutné souřadnice převést do tzv. homogenních souřadnic [3]. Toho lze docílit převedením původních souřadnic bodu do prostoru o jednu dimenzi vyšší. Bod (x, y) se převede na bod (x, y, w), přičemž se obvykle pro jednoduchost volí $w = 1$. Obecná transformace do nových souřadnice se poté dá zapsat tímto maticovým vztahem:

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a_1 & a_2 & a_0 \\ b_1 & b_2 & b_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{3.1}$$

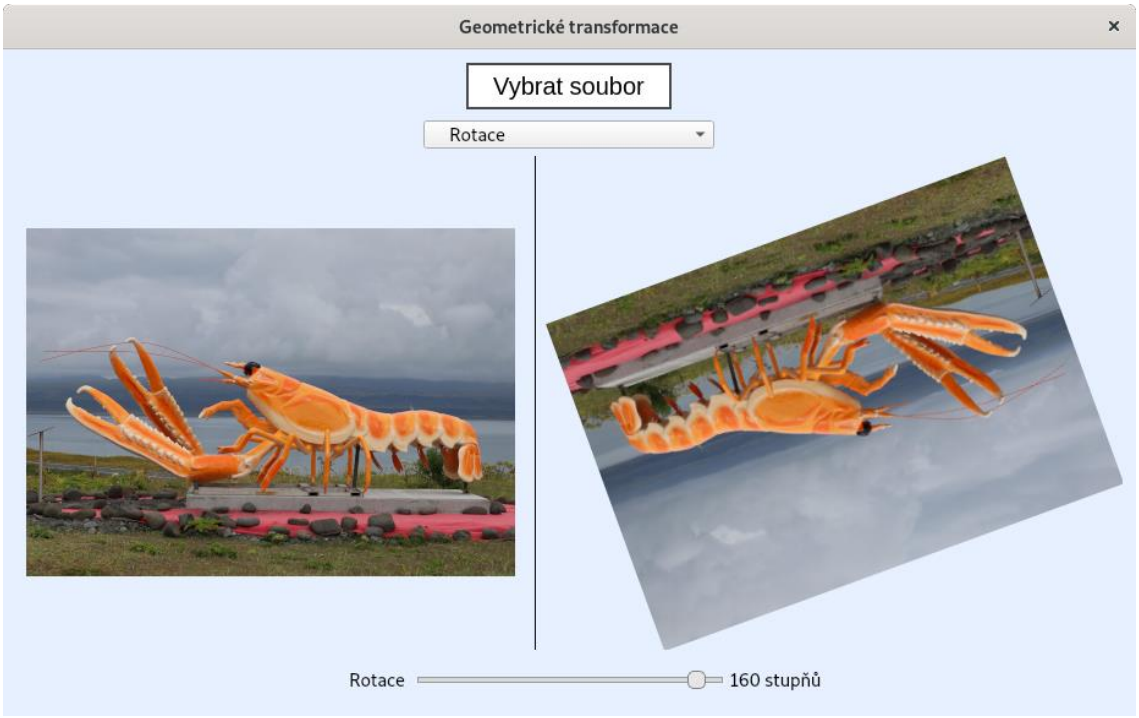
Transformační rovnice (Rovnice 3.1) je známa již předem v některých případech, jako je například rotace, posun, či změna měřítka. V jiných případech je možné její nalezení díky znalosti původního i transformovaného obrazu. Transformační matice pro rotaci o úhel α kolem středu soustavy souřadnic má tvar:

$$R = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.2}$$

Matici pro změnu měřítka lze zapsat takto:

$$S = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.3}$$

Kde s_x představuje změnu měřítka ve směru x a s_y ve směru y.



Obrázek 3.1: Okno Geometrické transformace

V systému jsou k dispozici dvě geometrické transformace, rotace a změna měřítka. Po načtení obrázku je možné si v comboboxu v horní části okna navolit požadovanou transformaci a horizontálním posuvníkem v dolní části měnit rozsah této transformace (Obrázek 3.1).

4 Jasové transformace

Jasová transformace mění intenzitu jasu daného pixelu. Tyto transformace se dají rozdělit do dvou skupin: jasové korekce a transformace jasové stupnice. U jasové korekce závisí hodnota transformovaného pixelu na jeho původní hodnotě a jeho pozici v obraze. Při transformaci jasové stupnice je hodnota pixelu pouze transformována na jinou a nezáleží při ní na poloze původního pixelu.

Pokud je degradace obrazu systematická, je možné ji odstranit pomocí jasové korekce. Předpokládejme, že $f(x,y)$ je degradovaný obraz a koeficient $e(x,y)$ představuje odchylku od ideálního obrazu $g(x,y)$, pro každý pixel *(x, y)* potom platí

$$f(x,y) = e(x,y) \cdot g(x,y). \tag{4.1}$$

Koeficient $e(x,y)$ je možné získat v případě, kdy je pořízen obraz $g(x,y)$ se známou jasovou funkcí, v nejjednodušším případě se bude jednat o obraz s konstantním jasem c (šedá plocha). Pořízený obraz této plochy o známém jasu označíme $f_c(x,y)$. Potom je možné potlačit chybu v degradovaném obraze následujícím způsobem:

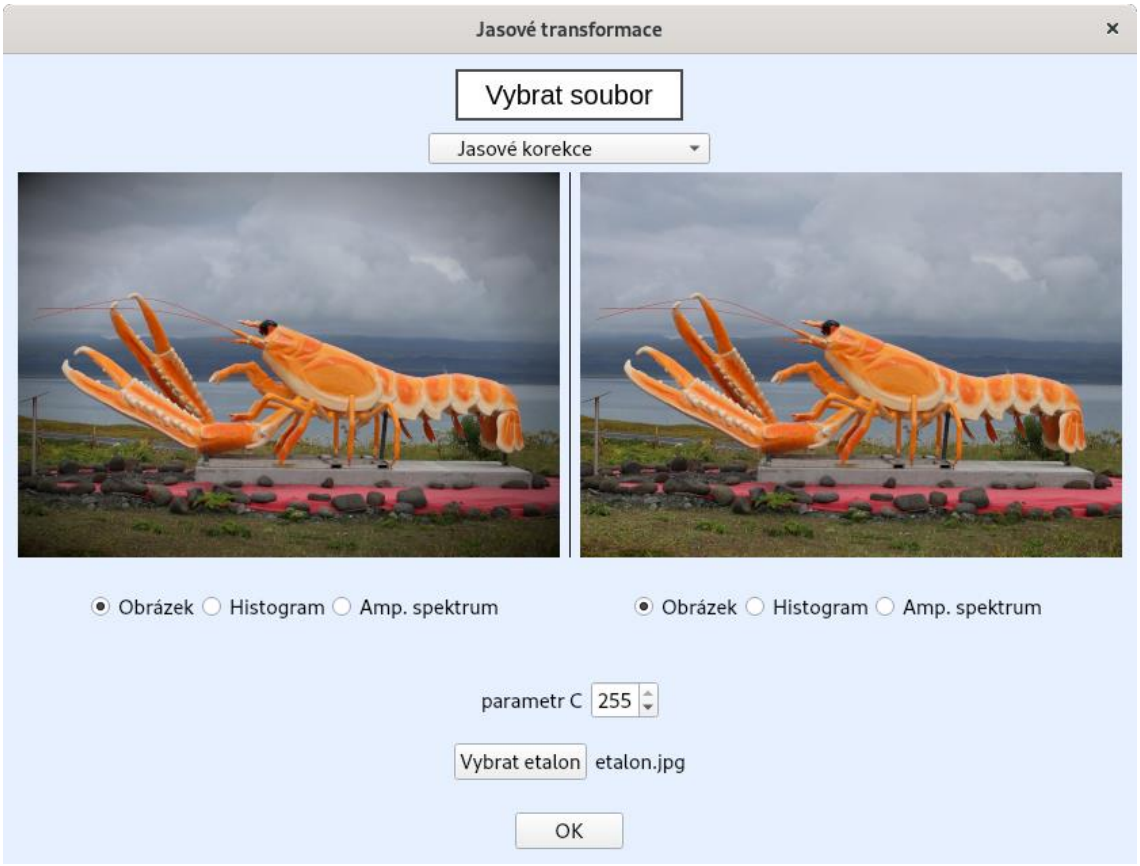
$$g(x,y) = \frac{f(x,y)}{e(x,y)} = \frac{c \cdot f(x,y)}{f_c(x,y)}. \tag{4.2}$$

Transformace jasové stupnice jsou využívány zejména v případech, kdy je potřeba zvýšit kontrast, aby byl obraz čitelnější pro člověka. Zvýšení kontrastu může být docíleno metodou ekvalizace histogramu. Cílem této metody je vytvořit nový obraz s rovnoměrně rozloženými četnostmi jednotlivých jasových úrovní z celé jasové stupnice. Algoritmus pro ekvalizaci histogramu [1] vypadá následovně:

1. Pro obraz o rozměrech $N \times M$ s G jasovými úrovněmi inicializujeme prázdný histogram H o délce G s každým prvkem rovným nule.
2. Projdeme každý pixel p, pokud má intenzitu rovnou g_p provedeme
$$H[g_p] = H[g_p] + 1\,. \tag{4.3}$$
3. Vytvoříme histogram kumulativních četností H_c a nastavíme $H_c[0] = H[0]$.
$$H_c[g] = H_c[g-1] + H[g], \;\; pro\; g = 1, 2, \ldots, G-1 \tag{4.4}$$
4. Potom
$$T[g] = round\left(\frac{H_c[g] - H_{min}}{MN - H_{min}}(G-1)\right), \tag{4.5}$$
kde $H_{min} = H_c[g_{min}]$ a g_{min} je intenzita s nejnižší četností výskytu v původním obraze.

5. Obraz s ekvalizovaným histogramem sestavíme takto:

$$g_p = T[g_p] \tag{4.6}$$



Obrázek 4.1: Okno Jasové transformace

V okně s jasovými transformace (Obrázek 4.1) jsou k výběru tři transformace, jasové korekce, negativ obrazu a ekvalizace histogramu. Požadovanou transformace lze zvolit v horním comboboxu a v případě jasových korekcí je možné dále vybrat etalon a zadat hodnotu jasu etalonové plochy.

5 Filtrace šumu

Filtrace šumu se používá v případě, když chceme z obrazu odebrat nežádoucí šum a zároveň se snažíme zachovat co největší množství původní informace. K odebrání šumu se využívá různých technik, většinou k tomu využívají lineárních kombinací hodnot okolí zkoumaného pixelu. Lineární kombinace vstupního obrazu $f(x, y)$ lze spočítat diskrétní konvolucí s jádrem h. Hodnota pixelu ve výstupním obraze $g(x, y)$ v okolí O se pak vypočítá takto:

$$g(x, y) = \sum_{(m,n)} \sum_{\epsilon O} h(x - m, y - n) f(m, n) \tag{5.1}$$

Konvolučnímu jádru $h(x, y)$ se také říká konvoluční maska. [3] Výběr filtru záleží na typu obrazu a chování filtru.

5.1 Průměrovací filtr

Průměrovací filtr jednoduše spočítá průměr původních jasů v daném okolí a hodnotu prostředního pixelu nastaví na výslednou hodnotu. Například konvoluční jádro průměrovacího filtru o rozměrech 3×3 je

$$h = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}. \tag{5.2}$$

Nevýhodou této metody je rozmazávání hran. Je vhodné jej tedy použít, pouze v případě, kdy nám rozmazání hran nevadí.

5.2 Filtr medián

Další jednoduchou metodou odstranění šumu je filtrace mediánem. Principem této metody je nalezení mediánu v daném okolí pixelu původního obrazu. Hodnota pixelu ve výstupním obraze je poté nastavena na hodnotu mediánu. Pro nalezení mediánu stačí vzestupně uspořádat hodnoty pixelů v okolí a medián zvolit jako prvek uprostřed. Aby bylo zvolení prostředního prvku jednoznačné, volí se čtvercové okolí s lichými rozměry.

Stupeň rozmazání hran je u filtrace mediánem v porovnání s obyčejným průměrováním nižší. Nevýhodou může být porušení tenkých čar a ostrých rohů v obraze.

5.3 Gaussův filtr

Koeficienty konvolučního jádra Gaussova filtru se řídí normálním rozdělením. Hodnotě prostředního pixelu je přisuzována nejvyšší váha a ostatním nižší čím více jsou

vzdálené od prostředního pixelu. Jednotlivé koeficienty konvolučního jádra G se spočítají podle

$$G(x,y) = \alpha e^{-(x^2+y^2)/2\sigma^2} \tag{5.3}$$

, kde α je normalizační člen, x udává horizontální posun od středu, y vertikální posun od středu a σ je směrodatná odchylka. Členy musí být normalizovány na součet 1, aby nedošlo ke změně jasu výstupního obrazu oproti originálu.

$$h = \frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix} \tag{5.4}$$

Stejně jako u průměrovacího filtru jsou i Gaussovým filtrem rozmazávány hrany. Rozmazání ale není tak výrazné, protože prostřední pixel dostává větší váhu než pixely okolní. Jedná se tedy o vážený průměr okolních pixelů. Rovnice 5.4 zobrazuje příklad Gaussova konvolučního jádra o rozměrech 3×3.

5.4 Bilaterální filtr

Bilaterální filtr je stejně jako Gaussův filtr definovaný jako vážený průměr okolních pixelů. Rozdíl je v tom, že bilaterální filtr zohledňuje i rozdíl hodnoty zkoumaného pixelu od hodnot pixelů v daném okolí [4]. Tímto způsobem je dosaženo zachování hran. Konvoluční jádro pro obraz *g(x, y)* se středem v pixelu p můžeme získat tímto způsobem:

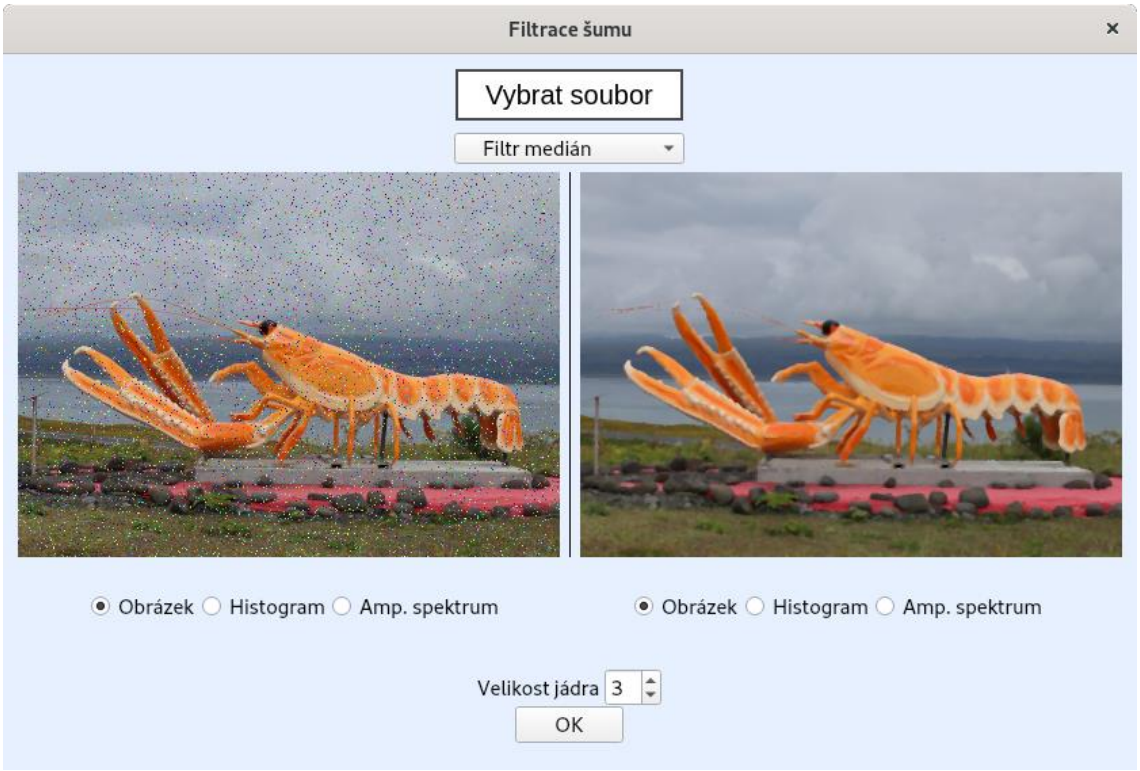
$$BF(p) = \alpha \sum_{q \epsilon S} G_{\sigma_s}(\|p-q\|)\ G_{\sigma_r}(|g(p)-g(q)|)\ g(q), \tag{5.5}$$

kde

- α je normalizační člen
- $\|p-q\|$ představuje vzdálenost mezi pixely p a q
- Parametry σ_s a σ_r udávají směrodatné odchylky Gaussova filtru G_{σ_s} resp. G_{σ_r}

5.5 Okno Filtrace šumu

V systému jsou zahrnuty všechny filtry popsané v této kapitole. Požadovaný filtr je zvolen v comboboxu pod tlačítkem pro výběr obrázku (Obrázek 5.1). Po zvolení obrázku a filtru je možné měnit parametry daného filtru v dolní části okna. Filtrovaný obrázek je jako v ostatních oknech zobrazen v pravé časti okna.



Obrázek 5.1: Okno filtrace šumu

6 Hranové detektory

Hranové detektory slouží k nalezení oblastí obrazu s ostrými změnami v intenzitách sousedních pixelů. Hranové pixely jsou pixely, u kterých se intenzita oproti ostatním pixelům v dané oblasti náhle mění. Hrany jsou množiny hranových pixelů.

Náhlé změny obrazové funkci lze nalézt pomocí parciálních derivací. Tyto změny mohou být popsány jako gradient. Gradient je dvousložková vektorová veličina, která udává směr největšího růstu funkce a také velikost změny. U digitálního obrazu se parciální derivace odhaduje jako diference.

$$\begin{aligned} \Delta_x f(x,y) &= f(x,y) - f(x-1,y) \\ \Delta_y f(x,y) &= f(x,y) - f(x,y-1) \end{aligned} \tag{6.1}$$

Nejběžnější hranové detektory se dají vyjádřit jako jádro pro diskrétní konvoluci. Některé detektory, které dokáží rozlišit orientaci hrany, jsou složeny z více konvolučních jader, z nichž každé odpovídá jednomu směru.

6.1 Sobelův hranový detektor

Sobelův hranový detektor je jednoduchý hranový detektor určený k detekci horizontálních a vertikálních hran. K tomu používá diskrétní konvoluci například s jádry

$$h_1 = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}, \qquad h_2 = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \tag{6.2}$$

Velikost G a směr Θ gradientu se pak dá vyjádři jako:

$$G = \sqrt{x^2 + y^2}, \qquad \Theta = \tan^{-1}(y/x), \tag{6.3}$$

kde x je výstup po konvoluci s jádrem h_2 a y je výstup po konvoluci s jádrem h_1.

6.2 Laplaceův hranový detektor

Principem Laplaceova hranového detektoru je aproximace druhé parciální derivace. Je tedy neměnný vzhledem k orientaci hrany a udává pouze její velikost. K výpočtu se obvykle používá konvoluce s jádrem o rozměrech 3x3. Jádro h_1 pro čtyřokolí a h_2 pro osmiokolí.

$$h_1 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}, \qquad h_2 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{6.4}$$

Někdy se používají i jádra, která kladou větší důraz na prostřední pixel nebo jeho okolí. V tomto případě ale neplatí neměnnost vůči orientaci hrany.

$$h_1 = \begin{pmatrix} 2 & -1 & 2 \\ -1 & -4 & -1 \\ 2 & -1 & 2 \end{pmatrix}, \qquad h_2 = \begin{pmatrix} -1 & 2 & -1 \\ 2 & -4 & 2 \\ -1 & 2 & -1 \end{pmatrix} \tag{6.5}$$

6.3 Cannyho hranový detektor

Cannyho hranový detektor je považován za jeden z nejlepších hranových detektorů. Algoritmus stojící za Cannyho detektorem zajišťuje tři kritéria [2], která ostatní detektory zaručit nedokáží:

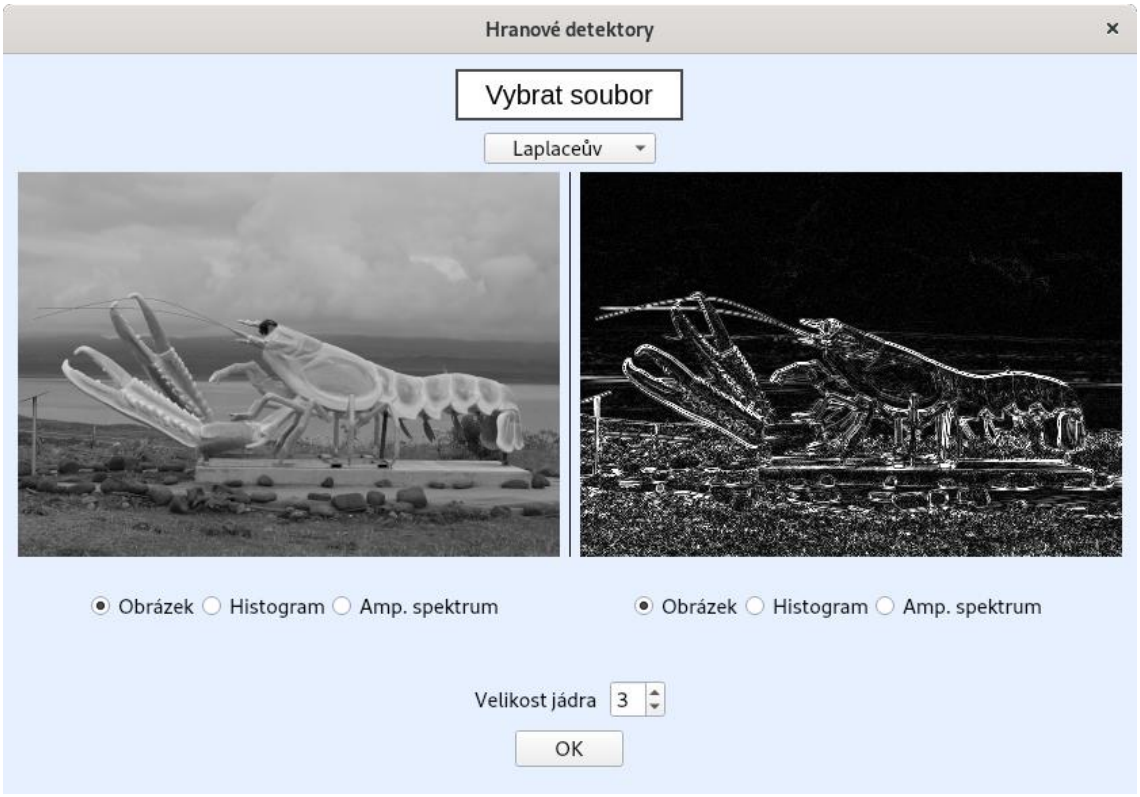
- *Detekční kritérium*. Všechny hrany by měly být detekovány a zároveň by neměly být detekovány místa, co nejsou hranami.
- *Lokalizační kritérium*. Vzdálenost mezi detekovanými hranami a skutečnými hranami by měla být minimální.
- *Kritérium jednoznačnosti*. Každá hrana by měla být detekována pouze jednou.

Samotný algoritmus se skládá z několika kroků.

1. Filtrace šumu Gaussovým filtrem (viz kapitola 5.3).
2. Výpočet velikosti a směru gradientu. Vhodné je za tímto účelem využít například Sobelův hranový detektor (viz kapitola 6.1).
3. Nalezení lokálních maxim gradientů kolmých na hranu. Díky tomu je hrana ztenčena a jsou vybrány pouze hrany v místě největšího gradientu.
4. Z celého obrazu jsou za využití prahování odstraněny nevýznamné hrany nebo detekované oblasti, které hranami nejsou. Výstupem je tedy binární obraz.

6.4 Okno Hranové detektory

Všechny popsané hranové detektory lze aplikovat na zvolený obrázek v okně s hranovými detektory (Obrázek 6.1). Detektor je opět zvolen v comboboxu v horní části okna. Po zvolení obrázku a detektoru se v dolní části objeví měnitelné parametry daného detektoru.



Obrázek 6.1: Okno Hranové detektory

7 Segmentace obrazu

Segmentace obrazu je jeden z nejdůležitějších kroků analýzy obrazových dat. Cílem segmentace je rozdělení obrazu do částí, které korespondují s objekty nebo oblastmi z reálného světa. Důvodem k segmentaci může být například redukce objemu zpracovávaných dat nebo další využití segmentovaných dat v algoritmech pro detekci, či klasifikaci objektů. V této práci budou předvedeny zejména techniky segmentace obrazu založené na základě prahování.

Prahování je nejjednodušší a výpočetně nenáročná metoda segmentace obrazu [2]. Jedná se o transformaci vstupního obrazu f do výstupního binárního obrazu g:

$$g(x,y) = \begin{cases} 1, & pokud\ f(x,y) > T \\ 0, & pokud\ f(x,y) \leq T, \end{cases} \quad (7.1)$$

kde T je práh, $g(x,y) = 1$ pro pixely korespondující objektům a $g(x,y) = 0$ pro pixely odpovídající pozadí. Hodnoty lze zvolit i opačně. Možné je zvolit i prahy dva, poté bude

$$g(x,y) = 1 \quad pokud\ T_1 < f(x,y) < T_2, \quad jinak\ g(x,y) = 0 \quad (7.2)$$

Volba správného prahu je pro výslednou segmentaci nezbytná. Lze ho nalézt experimentálně či jednou z metod pro detekci prahu, například analýzou histogramu. Je možné zvolit i více prahů a přiřadit různou intenzitu jasu jednotlivým oblastem či objektům v obraze. V tomto případě už obraz nebude binární.

Použití jednoho prahu k prahování celého obrazu může být v některých případech dostačující, ale ve většině případů mají různé oblasti obrazu jiné osvětlení a jeden práh nemusí vést k dobrému výsledku segmentace. V těchto případech je vhodné využít tzv. adaptivního prahování, u kterého hodnoty prahu závisí na poloze v obrazu. Práh je potom spočítán pro každý pixel zvlášť v závislosti na jeho okolí. Může to být například průměr, či vážený průměr hodnot pixelů v okolí.

7.1 Barvení oblastí

Barvení oblastí je metoda sloužící k popisu jednotlivých oblastí v obraze. Vstupem je binární obraz f. Výstupem je obraz g, ve kterém hodnota každého pixelu oblasti odpovídá číslu dané oblasti. Postup pro barvení oblastí segmentovaného obrazu f vypadá takto:

1. Řádek po řádku projdeme celý obraz a každému pixelu $f(x,y)$ přiřadíme číslo oblasti na základě čísel jeho sousedních pixelů. Sousedství je definováno maskou na Obrázku 7.1.

- Pokud jsou všechny sousední pixely nulové, pixelu $f(x, y)$ je přiřazeno nové, ještě nepoužité číslo oblasti.
- Pokud je v sousedství jeden nenulový pixel, pixelu $f(x, y)$ je přiřazeno číslo oblasti tohoto pixelu
- Pokud je v sousedství více nenulových pixel, pixelu $f(x, y)$ je přiřazeno číslo oblasti některého z nich. Pokud se čísla oblastní některých sousedních pixelů liší, jsou tato čísla uložena do tabulky ekvivalencí.

2. Znovu projdeme celý obraz f. Po prvním průchodu byly očíslovány všechny pixely oblastí, ale některé oblasti mají pixely s různými čísly. Pixely jsou očíslovány znovu, ale podle tabulky ekvivalencí, například nejnižším číslem z třídy ekvivalencí.

f(x-1, y-1)	f(x-1, y)	f(x-1, y+1)
f(x, y-1)	f(x, y)	

Obrázek 7.1: Maska použitá při barvení oblastí

7.2 Algoritmus Rozvodí

Algoritmus Rozvodí je klasický algoritmus k segmentaci obrazu, který je vhodné použít zejména v případě, kdy se snažíme oddělit překrývající se objekty.

Na šedotónový obraz lze nahlížet jako na topografickou mapu, ve které vysoké intenzity pixelů představují kopce a malé intenzity údolí. Myšlenkou za algoritmem Rozvodí je začít plnit každé údolí vodou a v místech, kde se vody slévají, postavit hráz a pokračovat, dokud nejsou všechny kopce pod vodou. Tyto hráze pak udávají hranice objektů v segmentovaném obrazu.

Tímto způsobem ale získáme přesegmentovaný obraz, proto je účinnější nejprve nalézt značky vyskytující se v údolích, která chceme zalít a začít zalévat pouze od těchto značek [6]. Tyto značky můžeme zadat ručně nebo je nalézt například pomocí vzdálenostní transformace v binárním obraze.

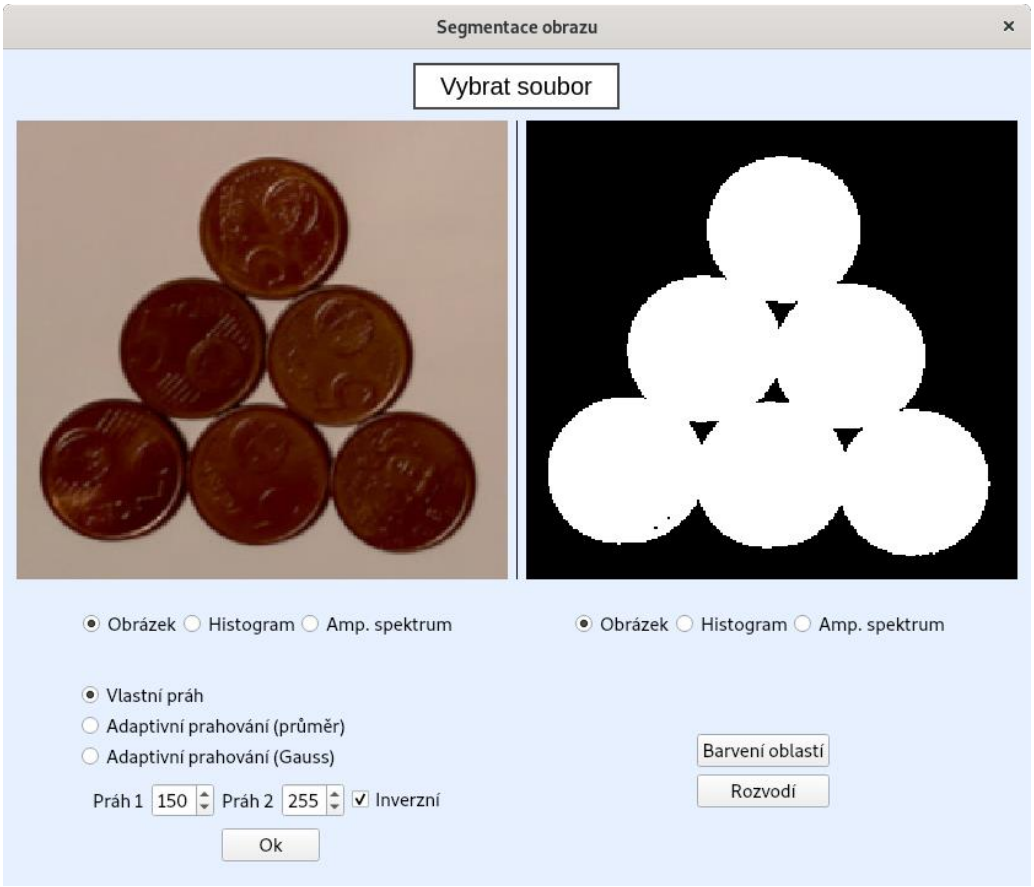
Celý algoritmus Rozvodí s použitím značek pak vypadá následovně:

1. Jsou zvoleny pixely, od kterých chceme začít zalévat. Každému z nich je přiřazena jiná třída.

2. Pixely z osmiokolí počátečních pixelů jsou přidány do prioritní fronty s prioritou odpovídající velikosti gradientu na stejném místě v odpovídajícím gradientním obraze.
3. Z fronty je vybrán pixel s nejvyšší prioritou. Pokud mají všechny sousední pixely tohoto pixelu přiřazenou třídu, je mu přiřazena stejná třída. Všechny neoznačené sousední pixely jsou přidány do fronty (pokud tam už nejsou).
4. Bod číslo 3 je opakován, dokud není fronta prázdná.
5. Neoznačené pixely představují hranice segmentovaných objektů.

7.3 Okno Segmentace obrazu

V okně Segmentace obrazu (Obrázek 7.2) lze provést prahování s vlastními prahovými hodnotami, či adaptivní prahování, kde práh je průměr okénka nebo vážený průměr s váhami z normálního rozdělení. Po provedení prahování se zpřístupní tlačítka pro barvení oblastí podle 7.1 a algoritmus rozvodí, popsán v sekci 7.2.

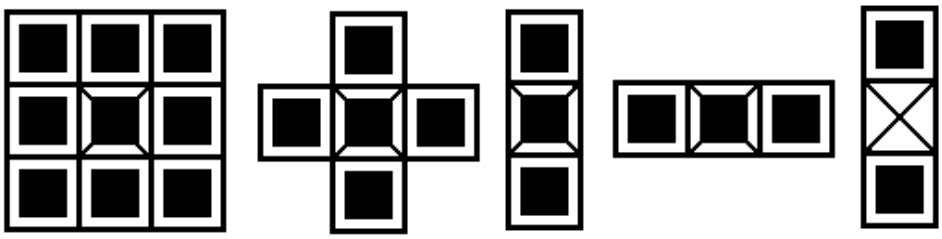


Obrázek 7.2: Okno Segmentace obrazu

8 Morfologické transformace

Matematická morfologie je založená na teorii množin. Při práci s digitálním obrazem představují objekty v obrazu množiny. V případě binárních obrazů je každá množina elementem prostoru $\mathbb{Z}^2$, kde každý pixel množiny je dvourozměrný vektor, jehož prvky představují souřadnice x a y pixelu v obrazu. Šedotónový obraz může být vyjádřen pomocí množin, jejichž prvky patří do $\mathbb{Z}^3$. Dva prvky každého vektoru budou představovat souřadnice pixelu v obraze a třetí bude intenzita jasu pixelu. Ve vyšších prostorech je možné přidat i intenzity jednotlivých barevných složek.

Morfologické operace jsou prováděny mezi obrazem a druhou malou množinou pixelů zvanou strukturní element. Strukturní element je vztažen k lokálnímu počátku v jednom ze svých bodů, v ilustracích se běžně označuje pomocí symbolu × (Obrázek 8.1).



Obrázek 8.1: Typické strukturní elementy

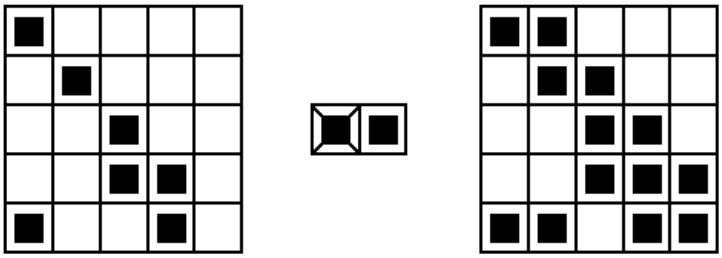
Morfologická transformace je aplikována tak, že je strukturní element systematicky posouván po celém obraze, přičemž pro transformaci jsou uvažovány pouze černé nebo pouze bílé pixely. Výsledek transformace je poté zapsán do výstupního binárního obrazu.

8.1 Dilatace a eroze

Dilatace a eroze jsou hlavními morfologickými operacemi. Vycházejí z nich další, složitější operace jako je například otevření nebo uzavření.

Dilatace je morfologická transformace, která kombinuje dvě množiny bodů pomocí vektorového součtu. Značí se operátorem $\oplus$ a $X \oplus B$ je množina každého možného vektorového součtu dvojic pixelů z množiny X a množiny B (Rovnice 8.1), X typicky představuje binární obraz a B strukturní element.

$$X \oplus B = \{p \in \varepsilon^2 : p = x + b, x \in X, b \in B\} \qquad (8.1)$$



Obrázek 8.2: Dilatace

Dilatace má několik vlastností:

- Komutativnost

$$X \oplus B = B \oplus X \tag{8.2}$$

- Asociativnost

$$X \oplus (B \oplus D) = (X \oplus B) \oplus D \tag{8.3}$$

- Lze jí vyjádřit i jako sjednocení posunutých bodových množin:

$$X \oplus B = \bigcup_{b \epsilon B} X_b \, , \tag{8.4}$$

  kde X_b představuje obraz X posunutý o vektor b.

- Jedná se o rostoucí transformaci

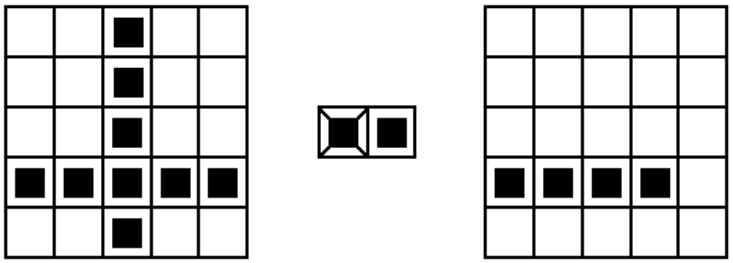
$$Pokud\ X \subseteq Y, potom\ X \oplus B \subseteq Y \oplus B \tag{8.5}$$

- Je invariantní vůči posunu:

$$X_h \oplus B = (X \oplus B)_h \tag{8.6}$$

Dilatace se používá k zaplnění malých děr a úzkých zálivů v objektech. Zvětšuje také rozměry objektů v obraze. Pro zachování původních rozměrů se kombinuje s erozí.

Eroze kombinuje dvě množiny bodů pomocí vektorového rozdílu. Značí se operátorem $\ominus$ a $X \ominus B$ je množina každého možného vektorového rozdílu dvojic pixelů z množiny X a množiny B. Eroze se používá ke zjednodušení struktury objektů v obraze.



Obrázek 8.3: Eroze

Komplexní objekty se po transformaci rozloží na vícero jednodušších.

$$X \ominus B = \{p \in \varepsilon^2 : p + b \in X \; pro \; každé \; b \in B\} \tag{8.7}$$

Rovnice 8.7 říká, že výsledek eroze je dán pouze těmi pixely p obrazu X, pro které platí, že všechny možné kombinace $p + b$ leží v obraze X.

Eroze může být také vyjádřena jako průnik posunutých bodových množin

$$X \ominus B = \bigcap_{b \epsilon B} X_{-b}. \tag{8.8}$$

Stejně jako dilatace je invariantní vůči posunu

$$X_h \ominus B = (X \ominus B)_h \tag{8.9}$$

$$X \ominus B_h = (X \ominus B)_{-h} \tag{8.10}$$

a znovu se jedná o rostoucí transformaci.

$$Pokud \; X \subseteq Y, potom \; X \ominus B \subseteq Y \ominus B \tag{8.11}$$

Na rozdíl od dilatace není eroze komutativní operace

$$X \ominus B \neq B \ominus X \tag{8.12}$$

Eroze a dilatace jsou duální operace. Platí pro ně následující vztah

$$(X \ominus Y)^C = X^C \oplus \check{Y}. \tag{8.13}$$

$\check{Y}$ značí symetrickou množinu k množině Y, podle Rovnice 8.14

$$\check{B} = \{-b : b \epsilon B\} \tag{8.14}$$

a X^C značí doplněk k množině X.

8.2 Otevření a uzavření

Eroze ani dilatace nejsou invertovatelné, když obraz erodujeme a následně dilatujeme, nezískáme původní obraz. Získáme pouze zjednodušenou, méně detailnější verzi původního obrazu. Eroze následovaná dilatací je další morfologickou operací a nazývá se otevření a značí se operátorem ○. Otevření obrazu X strukturním elementem B je definováno následovně:

$$X \circ B = (X \ominus B) \oplus B \tag{8.15}$$

Otevření je antiextenzivní:

$$X \circ B \subseteq X \tag{8.16}$$

Dilatace následovaná erozí se nazývá uzavření a značí se operátorem ●. Uzavření spojí blízké objekty a zaplní malé díry a zálivy. Je definováno následovně:

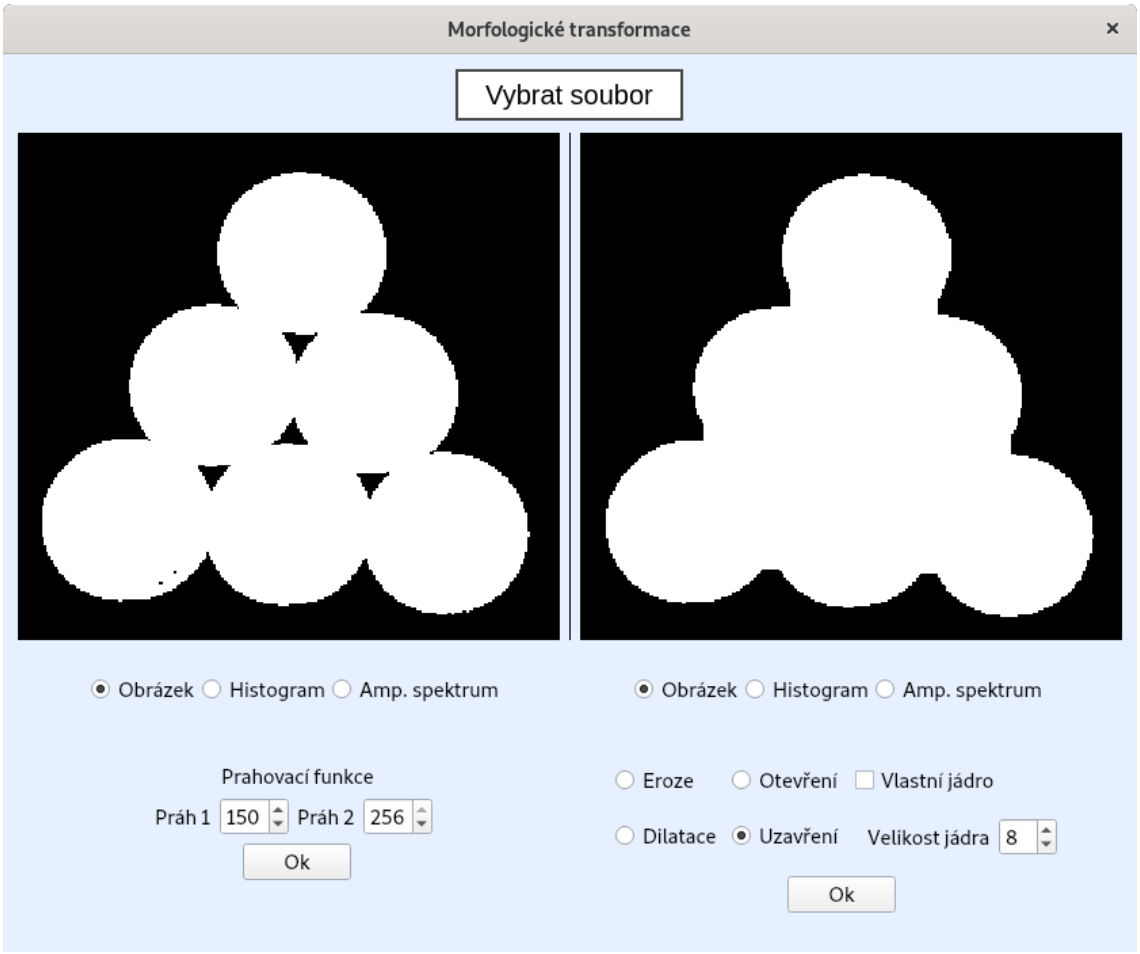
$$X \bullet B = (X \ominus B) \oplus B \tag{8.17}$$

Uzavření je extenzivní:

$$X \subseteq X \bullet B \tag{8.18}$$

8.3 Okno Morfologické transformace

V okně Morfologické transformace (Obrázek 8.4) je možné vizualizovat některé binární morfologické transformace, jako jsou dilatace, eroze, otevření a uzavření. V levé části se nejdříve nastaví prahy a provede se jednoduché prahování. Na výsledný binární obraz lze aplikovat některou z binárních morfologických transformací. Jako strukturní element lze zvolit čtvercové jádro se samými jedničkami nebo vlastní binární obraz.



Obrázek 8.4: Okno Morfologické transformace

9 Houghova transformace

Houghova transformace je algoritmus sloužící k detekci čar, či jiných křivek v obraze, které lze vyjádřit analyticky. Výhodou algoritmu je, že není příliš citlivý na neúplné tvary a šum. Algoritmus funguje podobně pro detekci všech křivek, proto zde bude uveden pouze příklad detekce přímek v obraze.

Přímku lze vyjádřit ve tvaru

$$y = kx + b. \tag{9.1}$$

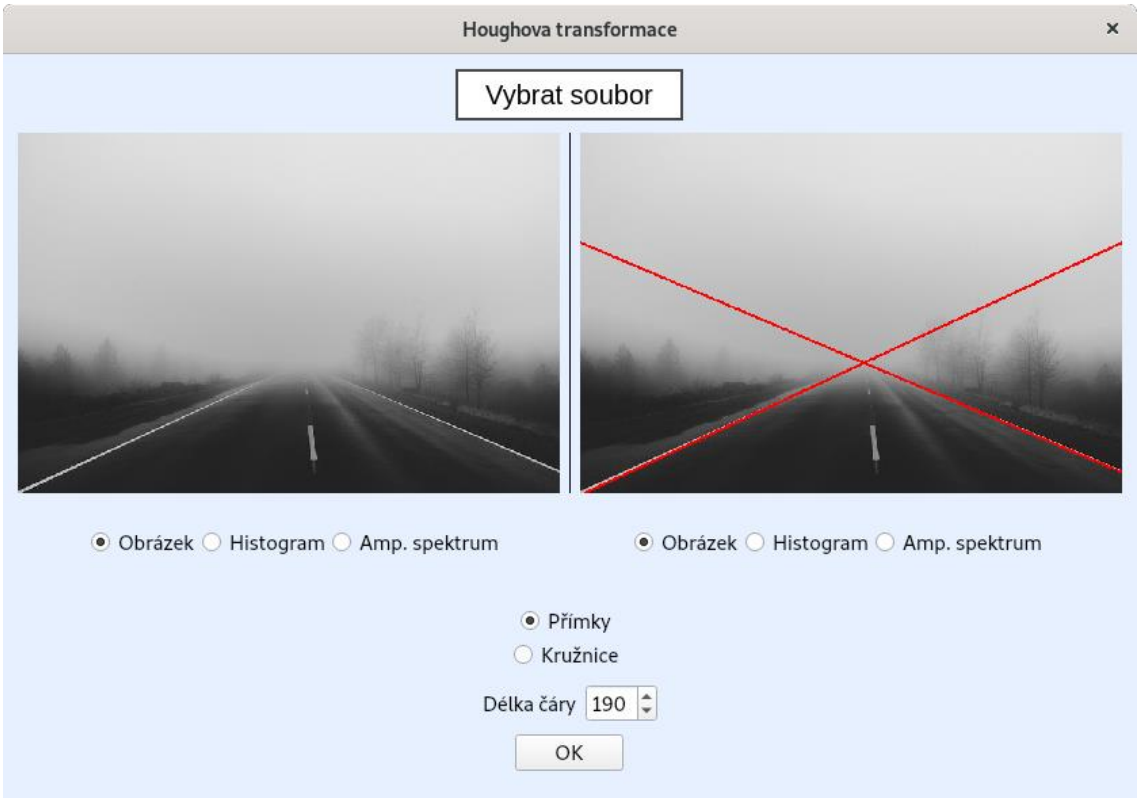
Pro algoritmus je ale nutné, aby byly parametry křivky konečné, což neplatí pro parametr k v Rovnici 9.1 (vertikální přímky mají nekonečný gradient). Proto je vhodné rovnici přímky přepsat do tvaru

$$\rho = x\cos\theta + y\cos\theta, \tag{9.2}$$

kde ρ představuje vzdálenost přímky od počátku souřadného systému a θ úhel, který přímka svírá s osou x.

Prvním krokem je vytvoření dvourozměrného pole A s nulovými prvky, ve kterém řádky představují parametr ρ a sloupce parametr θ. Tomuto poli se také říká akumulátor. Velikost akumulátoru závisí na tom, jakým způsobem se rozhodneme diskretizovat prostor příznaků tak, aby byl konečný.

V dalším kroku projdeme obraz f pixel po pixelu. Pro každý pixel $f(x, y)$ zvýšíme hodnotu ve všech prvcích akumulátoru $A(\rho, \theta)$ o jedničku, pro které dané x a y vyhovuje Rovnici 9.1. Lokální maxima v akumulátoru představují přímky, které jsou v původním obrazu [1].



Obrázek 9.1: Okno Houghova transformace

V okně s Houghovými transformacemi (Obrázek 9.1) jsou implementovány transformace pro detekci přímek a kružnic. V případě detekce přímek lze nastavit minimální délku čáry v pixelech a v případě kružnic minimální vzdálenost kružnic.

10 Hledání vzorů

Hledání vzorů je nejjednodušším postupem při hledání známých objektů v obraze. Principem je nalezení oblasti v obraze, která odpovídá vzoru, tzn. hledáme oblast, jejíž hodnoty pixelů jsou stejné (velmi podobné) jako hodnoty pixelů vzoru. Obraz vzoru by se tedy neměl lišit otočením ani jiným měřítkem oproti hledanému vzoru.

Mějme obraz I o rozměrech $W \times H$ a obraz vzoru T o rozměrech $w \times h$. Hledání vzoru probíhá umístěním obrazu vzoru na všechny možné pozice v obraze I a spočítáním zvolené metriky v každé pozici [5]. Výsledek výpočtu je poté uložen v matici R s rozměry $(W - w + 1) \times (H - h + 1)$. Jako metriku můžeme zvolit například součet rozdílů umocněných na druhou, podle

$$R(x, y) = \sum_{x',y'} [T(x', y') - I(x + x', y + y')]^2, \qquad (10.1)$$

kde $x' = 0 \ldots w - 1$ a $y' = 0 \ldots h - 1$. Minimum v matici R odpovídá nejpravděpodobnější pozici levého horního rohu hledaného vzoru v obraze. Dalším příkladem metriky může být korelační koeficient

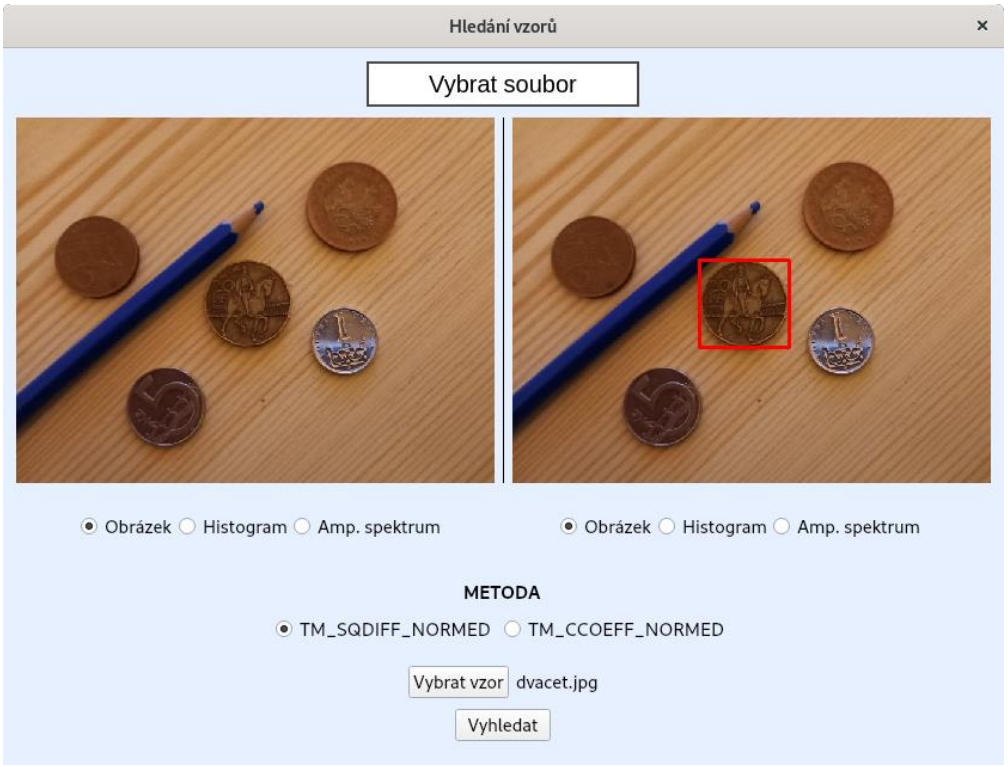
$$R(x, y) = \sum_{x',y'} T'(x', y') \cdot I'(x + x', y + y'), \qquad (10.2)$$

kde

$$T'(x', y') = T(x', y') - \frac{1}{w \cdot h} \cdot \sum_{x'',y''} T(x'', y'') \qquad (10.3)$$

$$I'(x + x', y + y') = I(x + x', y + y') - \frac{1}{w \cdot h} \cdot \sum_{x'',y''} I(x + x'', y + y''). \qquad (10.4)$$

V tomto případě pak v matici R hledáme maximum.



Obrázek 10.1: Okno Hledání vzorů

V systému jsou k výběru metody TM_SQDIFF_NORMED a TM_CCOEEF_NORMED implementované v knihovně OpenCV. Jedná se o metody hledání vzorů používající metriky uvedené v rovnicích 10.1, resp. 10.2 navíc doplněné o normalizační člen. Na Obrázku 10.1 je náhled okna Hledání vzorů. Po otevření obrázku a zvolení vzoru je možné vyhledat daný vzor s použitím zaškrtnuté metody.

11 Parametrizace objektů

Cílem parametrizace objektů je nalézt příznaky v obraze, díky kterým budeme schopni lépe rozlišit či rozpoznat objekty v obraze nebo popsat celý obraz. Těmito příznaky rozumíme významné části obrazu, které je vhodné použít k popisu daného objektu a ideálně které jsou invariantní vůči změně měřítka, posunu, rotaci a osvětlení.

11.1 Harrisův detektor rohů

Jedním z nejpoužívanějších příznaků jsou rohy. Roh můžeme definovat jako bod v němž se protínají dvě hrany. K jejich detekci lze použít Harrisův detektor rohů [7].

Harrisův detektor rohů hledá body, které se výrazně liší od bodů ve svém sousedství. Tato odlišnost je vyjádřena následující rovnicí:

$$E(x,y) = \sum_{u,v} w(u,v)[I(x+u, y+v) - I(x,y)]^2, \tag{11.1}$$

kde w představuje váhy a u a v jsou posuny ve směru osy *x*, resp. osy *y*. Maximalizací této funkce dostaneme

$$E(x,y) = (x,y)\, M\, (x,y)^T, \tag{11.2}$$

kde

$$M = \sum_{u,v} w(u,v) \begin{pmatrix} X^2 & XY \\ XY & Y^2 \end{pmatrix} \tag{11.3}$$

X je derivace obrazu ve směru osy x a Y je derivace ve směru osy *y*. K jejich nalezení lze použít například postup uvedený v kapitole 6.1.

Odezva detektoru vypadá následovně:

$$R = \alpha\beta - k(\alpha + \beta)^2 \tag{11.4}$$

α a β představují vlastní čísla M a k je konstanta určující citlivost detektoru. Jako rohy zvolíme body, které mají odezvu maximální nebo větší než námi zvolený práh.

11.2 Shi-Tomasův detektor rohů

Shi-Tomasův detektor rohů používá k určení rohů stejné kritérium jako Harrisův detektor (Rovnice 11.1). Od Harrisova detektoru se liší pouze výpočtem odezvy detektoru. Odezva Shi-Tomasova detektoru se spočítá následovně:

$$R = \min(\alpha, \beta), \tag{11.5}$$

kde α a β jsou vlastní čísla matice M z Rovnice 11.3. Jako rohy jsou opět vybrány body s maximální odezvou nebo odezvou větší než zvolený práh.

11.3 SIFT příznaky

Detektory uvedené v předchozích podkapitolách 11.1 a 11.2 jsou invariantní vůči rotaci, ale ne vůči změně měřítka. Detekce rohů u těchto detektorů závisí na zvolené velikosti okénka, kterým se obraz prohledává.

Z tohoto důvodu byl vymyšlen nový algoritmus s názvem SIFT (Scale-Invariant Feature Transform) [8], který nalezne klíčové body v obraze a je invariantní i vůči změně měřítka.

Celý algoritmus se skládá z několik kroků. Prvním krokem je Gaussovské rozostření obrazu v různé míře a v různých měřítkách. Tyto různě rozostřené obrazy jsou od sebe odečteny. Získáme tedy několik obrazů rozdílů různě rozostřených verzí obrazu. V těchto rozdílových obrazech nalezneme lokální extrémy, ty budou představovat kandidáty na klíčové body.

V dalším kroku jsou vyřazeny neužitečné příznaky, jako jsou body ležící na hranách nebo na plochách s téměř konstantním jasem. Každý potencionální příznak je spolu se svým 3x3 okolím proložen trojdimenzionální kvadratickou funkcí. Podle tvaru této funkce se z kandidátů odstraní body, které leží na hraně nebo jsou málo kontrastní.

Dalším krokem je spočítání gradientu a jeho velikosti pro každý z dosud nalezených bodů. Na základě těchto údajů je vytvořen histogram s 36 třídami odpovídající 360 stupňům rozdělených po 10. V histogramu je nalezena třída s nejvyšší hodnotou a u bodů s touto orientací jsou přidány další příznaky s orientací odpovídající třídám histogramu, které mají velikost do 80 % nejvyšší třídy. Tím je zvýšena váha těchto velmi význačných příznaků.

Posledním krokem je vytvoření deskriptoru pro každý příznak. U každého příznaku je jeho 16x16 okolí rozděleno do menších 4x4 oblastí. Pro každou z těchto oblastí je vytvořen histogram s 8 třídami, opět podle gradientů jednotlivých pixelů. Spojením těchto histogramů získáme 4x4x8 = 128hodnotový příznakový vektor. Tento vektor je dále normalizován na jednotkovou velikost, poté jsou hodnoty vyšší než 0,2 změněny na 0,2 a nakonec je vektor znovu normalizován na jednotkovou velikost. Tímto způsobem je příznak odolnější vůči změnám osvětlení.

11.4 SURF příznaky

Dalším z algoritmů pro hledání klíčových bodů v obraze je SURF (Speeded-Up Robust Features) [9]. Jedná se o zrychlenou verzi algoritmu SIFT. Algoritmus SURF se

skládá ze stejných kroků jako SIFT (nalezení zajímavých bodů, výpočet deskriptoru), ale liší se v provedení samotných kroků.

SURF pracuje nad tzv. integrálním obrazem I_Σ. Jedná se o obraz, kde hodnota každého pixelu (x, y) představuje součet pixelů původního obrazu I nahoru a doleva od pixelu (x, y) včetně, tedy:

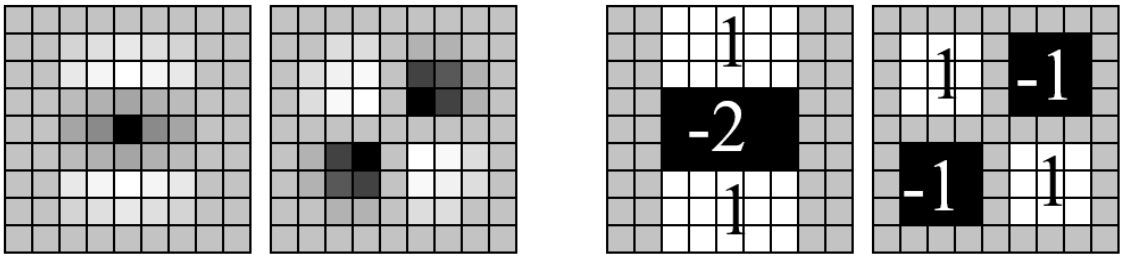
$$I_\Sigma = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(i,j) \tag{11.6}$$

Pomocí integrální obrazu lze spočítat součet intenzit v jakékoliv obdélníkové oblasti pomocí pouze čtyř součtů (krajní body oblasti).

Pro určení klíčových bodů se používá aproximace determinantu Hessovy matice. Hessova matice $\mathcal{H}(x, y, \sigma)$ v bodě (x, y) v měřítku σ je definována následovně:

$$\mathcal{H}(x,y,\sigma) = \begin{pmatrix} L_{xx}(x,y,\sigma) & L_{xy}(x,y,\sigma) \\ L_{xy}(x,y,\sigma) & L_{yy}(x,y,\sigma) \end{pmatrix} \tag{11.7}$$

Zde $L_{xx}(x, y, \sigma)$ představuje konvoluci druhé derivace Gaussovy funkce $\frac{\partial^2}{\partial x^2} g(\sigma)$ a vstupního obrazu I v bodě (x, y), podobně pro $L_{xy}(x, y, \sigma)$ a $L_{yy}(x, y, \sigma)$. Druhé derivace Gaussovy funkce jsou aproximovány tzv. box filtry (Obrázek 11.1). Odezvy těchto filtrů se nad integrálním obrazem dají spočítat velice efektivně

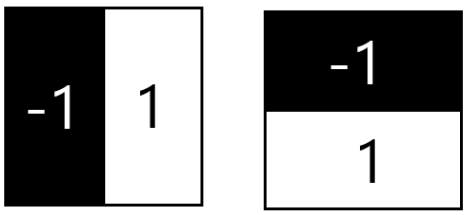


Obrázek 11.1: Zleva doprava: Diskrétní druhé derivace Gaussovy funkce ve směru *y* a *xy* a jejich aproximace pomocí box filtrů

Aproximace druhé derivace Gaussovy funkce box filtry o rozměrech 9x9 odpovídá měřítku σ = 1,2. Aproximace jednotlivých prvků Hessovy matice označíme D_{xx}, D_{yy} a D_{xy}. Jedná se tedy o konvoluce box filtrů se vstupním obrazem. Determinant je poté spočítán podle:

$$\det(\mathcal{H}_{approx}) = D_{xx}D_{yy} - (0{,}9D_{xy})^2 \tag{11.8}$$

Aby byly příznaky invariantní vůči změně měřítka jsou aplikovány box filtry o různých rozměrech, což přináší další urychlení (např. oproti SIFT, u kterého je toto řešeno změnou měřítka samotného obrazu).



Obrázek 11.2: Haarovy filtry

Aby byl SURF invariatní vůči rotaci, je okolo každého klíčového bodu spočítána Haarova vlnková transformace ve směru x a y filtrem uvedeným na Obrázku 11.2 v kruhovém okolí s poloměrem $6s$, kde s je měřítko, ve kterém byl klíčový bod nalezen. Odezvy transformace poté tvoří vektor ve 2D prostoru. Převažující orientace je spočítána jako součet všech odezev v okénku pokrývající úhel 60°.

Posledním krokem je spočítání deskriptoru pro každý bod. Okolo každého nalezeného bodu je vytvořeno čtvercové okolí o rozměru $20s$ s orientací vypočtenou v předchozím kroku. V tomto okolí se znovu vypočte Haarova vlnková transformace dx ve směru x a dy ve směru y. Toto okolí se ještě jednou rozdělí na 4 menší 4x4 okolí a v každém z nich se vypočte

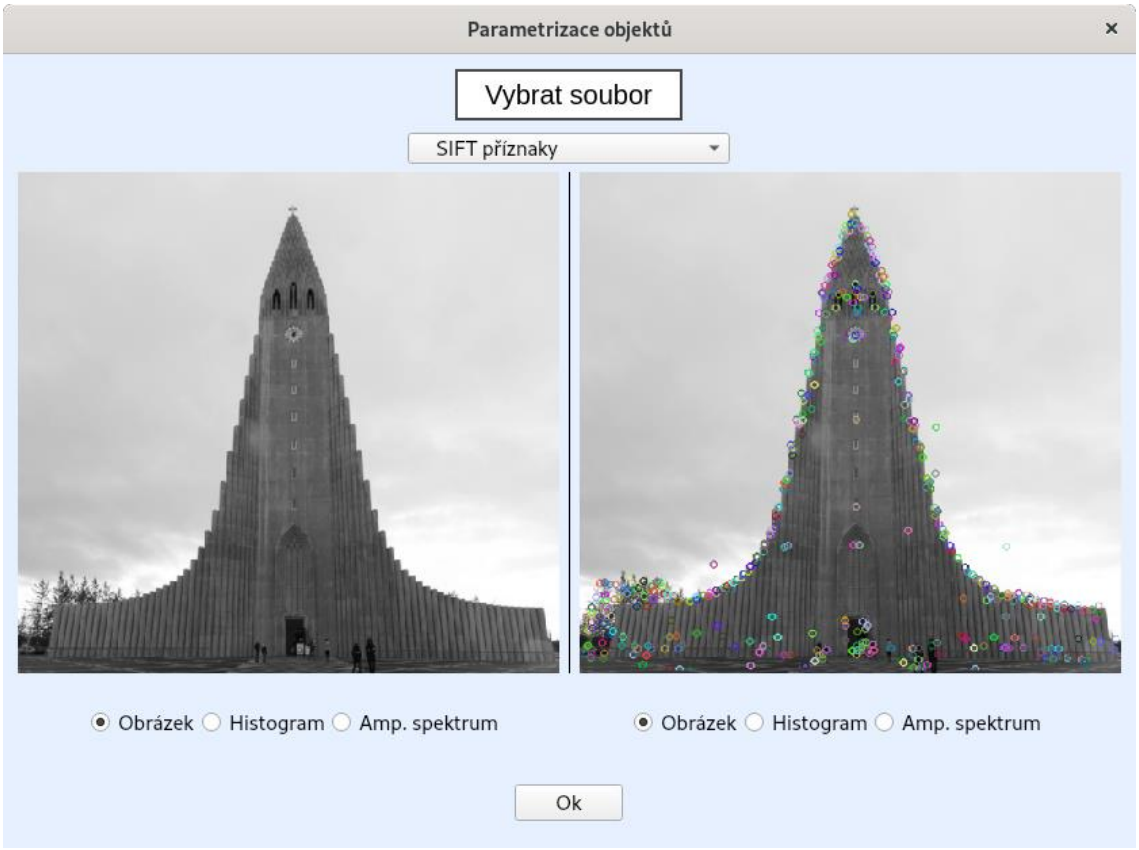
$$v = \left(\sum dx, \sum dy, \sum |dx|, \sum |dy|\right) \tag{11.9}$$

Spojením všech těchto vektorů tedy získáme 4x4x4 = 64hodnotový příznakový vektor pro každý klíčový bod.

11.5 Okno Parametrizace objektů

Všechny popsané metody v této kapitole je možné vizualizovat v okně Parametrizace objektů (Obrázek 11.3). Požadovaná metoda je vybrána pod tlačítkem pro výběr obrázku. Po načtení obrázku a potvrzení tlačítkem jsou klíčové body zakresleny na obrázek. U Shi-Tomasova detektoru je navíc možné zvolit maximální počet rohů, které má detektor vrátit.

Dále je v okně také implementováno rozpoznávání objektů na základě SIFT příznaků. V tomto případě je kromě hlavního obrázku načten i obraz vzoru, který chceme v hlavním obrazu nalézt, a navíc je zadán minimální počet shod mezi obrázky. Po načtení obou obrázků a potvrzení jsou v případě nalezení zadaného počtu shod zobrazeny oba obrazy vedle sebe a čárami jsou propojeny shodné klíčové body obrázků.



Obrázek 11.3: Okno Parametrizace objektů

12 Detekce a identifikace obličeje

Detekce a identifikace obličeje je jednou z nejvíce zkoumaných oblastí počítačového vidění, a proto existuje velké množství algoritmů pro detekci a identifikaci osob na základě obrazu obličeje. Spousta těchto algoritmů je součástí známých knihoven pro počítačové vidění, včetně OpenCV.

12.1 Viola-Jones detektor

Viola-Jones detektor [10] je klasifikátor pro detekci obličeje v obrazu složený z několika velmi jednoduchých slabých klasifikátorů, těmi může být například SVM nebo obyčejný perceptron. Každý ze slabých klasifikátorů je natrénovaný na jiném z Haarových příznaků, které jsou počítané nad integrálními obrazy (Rovnice 11.6) pomocí různých Haarových filtrů (např. Obrázek 11.2). Odezvy těchto filtrů lze nad integrálními obrazy spočítat velmi rychle.

Slabé klasifikátory jsou vybírány pomocí algoritmu AdaBoost (Adaptive Boosting). Vstupem algoritmu je trénovací množina obrázků $(x_1, y_1), \dots, (x_n, y_n)$, kde x_i je obrázek a y_i třída obrázku ($y_i = 1$ pokud je obrázek obličej, $y_i = 0$ v opačném případě) a vybrané příznaky. Samotný algoritmus AdaBoost pak vypadá následovně:

1. Inicializuj váhy $w_{1,i} = \frac{1}{2m}$ pro $y_i = 0$ resp. $w_{1,i} = \frac{1}{2l}$, pro $y_i = 1$. Zde m je počet obrázků neobsahující obličej a l počet obrázků obličejů.
2. Pro $t = 1 \dots T$ opakuj:
   a. Normalizuj vektor vah w_t na jednotkovou velikost
   b. Pro každý příznak j natrénuj klasifikátor h_j, pouze na tomto jednom příznaku. Spočítej chybu ϵ_j

$$\epsilon_j = \sum_i w_{t,i} \left| h_j(x_i) - y_i \right| \tag{12.1}$$

   c. Vyber klasifikátor h_t s nemenší chybou ϵ_t.
   d. Aktualizuj váhy

$$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}, \tag{12.2}$$

   kde $e_i = 0$, pokud byl obrázek x_i klasifikován správně, v opačném případě $e_i = 1$ a $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$
3. Výsledný silný klasifikátor je

$$h(x) = \begin{cases} 1, & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t, \\ 0, & v\ opačném\ případě \end{cases} \tag{12.3}$$

kde $\alpha_t = \log \frac{1}{\beta_t}$.

Samotná detekce je prováděna tak, že se prochází různě velké oblasti obrazu a jsou vyhodnocovány kaskádou klasifikátorů. Protože by vyhodnocování každé oblasti všemi klasifikátory bylo výpočetně náročné, tak v případě neshody u několika prvních klasifikátorů je klasifikace přerušena a oblasti je rovnou přiřazena třída 0.

12.2 Detektor z knihovny dlib

Dlib je další známá knihovna, ve které jsou implementovány algoritmy z oblasti strojového učení a počítačového vidění. Detektor obličejů implementovaný v této knihovně je založený na suport-vector machine natrénovaném na histogramech orientovaných gradientů (HOG). Histogram orientovaných gradientů obrazu I je vytvořen takto:

- Pro každý pixel je spočítána velikost g a směr gradientu k
- Obraz je rozdělen do několika menších oblastí a pro každou je spočítán histogram velikostí gradientů s třídami rozdělenými například po 20 stupních od 0 do 180 stupňů
- Histogramy jsou poté spojeny do jednoho vektoru a normalizovány na jednotkovou velikost. Tento vektor představuje deskriptor používaný k trénování různých algoritmů.

Support vector machine (SVM) je klasifikační metoda strojového učení. Cílem této metody je nalézt přímku, plochu nebo obecně n-rozměrnou nadrovinu, která optimálně, s maximální vzdáleností, odděluje prvky jednotlivých kategorií. Naučený předpis této nadroviny se poté používá ke klasifikaci neznámých dat. Při klasifikaci nás tedy zajímá, kde leží testovaný prvek vzhledem k nadrovině (například nad/pod přímkou). Obecnou nadrovinu lze popsat rovnicí

$$\vec{w} \cdot \vec{x} + b = 0. \tag{12.4}$$

Mějme vektor trénovacích dat $(x_1, y_1), \ldots, (x_N, y_N)$, kde x_i je příznakový vektor (HOG) a y_i třída vzorku (+1 obličej, resp. -1 není obličej). Kriteriální funkce, kterou chceme vzhledem k parametrům $\vec{w}$ a b u SVM minimalizovat pak je

$$H(\vec{w}, b) = \frac{1}{2}\|\vec{w}, b\|^2 + C\sum_{i=1}^{N} max\left(0, 1 - y_i(x_i^T\vec{w} + b)\right), \quad (12.5)$$

kde C je volitelný člen penalizující špatné klasifikace.

12.3 Identifikace pomocí knihovny face_recognition

Knihovna face_recognition pro Python obsahuje natrénovaný model konvoluční neuronové sítě, který pro každý obraz obličeje vygeneruje 128hodnotový vektor, který danou osobu dobře popisuje.

Trénování tohoto modelu probíhalo tak, že byly síti vždy předány zároveň dva obrázky stejné osoby a jeden obrázek jiné osoby. Síť v každém takovém kroku upraví své parametry tak, aby si vygenerované vektory pro stejné osoby byly blíže, zatímco vektor jiné osoby se od nich vzdálil. Výsledná síť natrénovaná na velkém množství obrázků je pak schopná generovat velmi dobře popisující příznakové vektory.

Výsledné vektory vygenerované touto sítí lze pak porovnávat s vektory reprezentující známé osoby a se zadanou tolerancí určit, zda se jedná o jednu z osob z databáze.

12.4 Okno Detekce a identifikace obličeje

V systému jsou implementovány všechny metody popsané výše v této kapitole. Po zvolení obrázku je možné zobrazit lokaci obličeje pomocí Viola-Jones detektoru z OpenCV nebo detektoru z knihovny dlib.

Dále je pak možná identifikace osob pomocí knihovny face_recognition. Po načtení hlavního obrazu je nutné načíst i obraz vzoru. Po potvrzení jsou v hlavním obraze vyznačeny pouze osoby, které jsou zároveň i v obrazu vzoru (Obrázek 12.1). Pro detekci obličeje u identifikace osob je použit detektor z knihovny dlib.

Obrázek 12.1: Okno Detekce a identifikace obličeje

Závěr

V rámci této práce byl vytvořen komplexní systém pro vizualizaci metod a algoritmů implementovaných zejména knihovnou OpenCV. Tento systém je rozdělen do 11 celků, z nichž každý obsahuje metody a algoritmy, které si jsou navzájem příbuzné. Ovládání systému probíhá přes grafické uživatelské rozhraní a každý z těchto 11 celků je zastoupen vlastním oknem. V každém okně je možné vizualizovat výsledky vybraných metod a algoritmů.

V jednotlivých kapitolách této práce bylo představeno několik metod a algoritmů z oblasti zpracování obrazu a počítačového vidění. Mezi tyto metody byl zařazen rozklad obrazu v různých barevných prostorech, geometrické a jasové transformace obrazu, filtrování šumu nejběžnějšími metodami a detekce hran pomocí tří rozdílných detektorů. Dále byly prozkoumány různé algoritmy pro segmentaci obrazu, některé z binárních morfologických transformací a Houghova transformace. Poslední kapitoly jsou věnovány hledání vzorů, detekci klíčových bodů používaných k rozpoznávání objektů v obraze a na závěr je zařazena detekce a identifikace osob.

Systém zdaleka nepokrývá všechny metody a algoritmy z knihovny OpenCV, neboť se jedná o velmi rozsáhlou knihovnu, a proto by bylo možné ho dále rozšířit o další z dostupných metod. Využití by mohl nalézt například u rychlého návrhu systémů pro zpracování obrazu nebo jako učební pomůcka.

Literatura

[1] ŠONKA, Milan, Václav HLAVÁČ a Roger BOYLE. *Image processing, analysis, and machine vision*. 4th edition. United States of America: Cengage Learning, 2015. International Edition. ISBN 978-1-133-59369-0.

[2] GONZALEZ, Rafael C. a Richard E. WOODS. Digital Image Processing, Global Edition. 4th edition. United States of America: Pearson, 2018. ISBN 9781292223070.

[3] HLAVÁČ, Václav a Miloš SEDLÁČEK. Zpracování signálů a obrazů. 2. přeprac. vyd. Praha: ČVUT, 2007. ISBN 978-80-01-03110-0.

[4] PARIS, Sylvain, Pierre KORNPROBST, Jack TUMBLIN a Fredo DURAND. Bilateral Filtering: Theory and Applications. Foundations and Trends® in Computer Graphics and Vision [online]. 2008, 4(1), 1-73 [cit. 2020-11-30]. ISSN 1572-2740. Dostupné z: http://dx.doi.org/10.1561/0600000020

[5] Template Matching. *OpenCV* [online]. [cit. 2021-02-14]. Dostupné z: https://docs.opencv.org/3.4/de/da9/tutorial_template_matching.html

[6] BARNES, Richard, Clarence LEHMAN a David MULLA. Priority-flood: An optimal depression-filling and watershed-labeling algorithm for digital elevation models. Computers & Geosciences [online]. 2014, 62, 117-127 [cit. 2021-03-14]. ISSN 00983004. Dostupné z: doi:10.1016/j.cageo.2013.04.024

[7] HARRIS, C. a M. STEPHENS. A Combined Corner and Edge Detector. In: *Procedings of the Alvey Vision Conference 1988* [online]. Alvey Vision Club, 1988, 1988, 23.1-23.6 [cit. 2021-04-06]. Dostupné z: doi:10.5244/C.2.23

[8] LOWE, David G. Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision [online]. 2004, 60(2), 91-110 [cit. 2021-04-11]. ISSN 0920-5691. Dostupné z: doi:10.1023/B:VISI.0000029664.99615.94

[9] BAY, Herbert, Tinne TUYTELAARS a Luc VAN GOOL. SURF: Speeded Up Robust Features. LEONARDIS, Aleš, Horst BISCHOF a Axel PINZ, ed. Computer Vision – ECCV 2006 [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, 2006, s. 404-417 [cit. 2021-04-12]. Lecture Notes in Computer Science. ISBN 978-3-540-33832-1. Dostupné z: doi:10.1007/11744023_32

[10] VIOLA, P. a M. JONES. Rapid object detection using a boosted cascade of simple features. In: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001 [online]. IEEE Comput. Soc, 2001, I-511-I-518 [cit. 2021-04-13]. ISBN 0-7695-1272-0. Dostupné z: doi:10.1109/CVPR.2001.990517