UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

ALGORITMY NA DETEKCIU HRÁN V OBRÁZKOCH

DIPLOMOVÁ PRÁCA

2018 Bc. Katarína BENEŠOVÁ

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

ALGORITMY NA DETEKCIU HRÁN V OBRÁZKOCH

DIPLOMOVÁ PRÁCA

Študijný program:	Ekonomicko-finančná matematika a modelovanie
Študijný odbor:	1114 Aplikovaná matematika
Školiace pracovisko:	Katedra aplikovanej matematiky a štatistiky
Vedúci práce:	Mgr. Soňa Kilianová, PhD.

Bratislava 2018 **Bc. Katarína BENEŠOVÁ**

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Bc. Katarína Benešová
Študijný program: ekonomicko-finančná matematika a modelovanie (Jednoodborové štúdium, magisterský II. st., denná forma)
Študijný odbor: aplikovaná matematika
Typ záverečnej práce: diplomová
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Algoritmy na detekciu hrán v obrázkoch.
Algorithms for edge detection in images.

Cieľ: Obsahom a cieľom práce bude prehľadne spracovať rôzne algoritmy detekcie hrán v obrázkoch, využívajúc metódy z rôznych oblastí matematiky (okrem iného napr. aj diferenciálne rovnice). Niektoré zo študovaných algoritmov budú numericky implementované, preto bude potrebná aj programovacia a numerická zručnosť.

Vedúci: Mgr. Soňa Kilianová, PhD.
Katedra: FMFI.KAMŠ - Katedra aplikovanej matematiky a štatistiky
Vedúci katedry: prof. RNDr. Daniel Ševčovič, CSc.

Dátum zadania: 25.01.2017

Dátum schválenia: 27.01.2017

prof. RNDr. Daniel Ševčovič, CSc.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie

Moja úprimná vďaka patrí mojej školiteľke, Mgr. Soni Kilianovej, PhD., za nesmiernu ochotu pri konzultovaní a cenné pripomienky pri tvorbe a finalizácii práce. Ďalej by som sa chcela poďakovať Jakubovi Krchňavému za poskytnutie fotografií. V neposlednom rade patrí moja vďaka rodine a priateľom za pochopenie a podporu.

Abstrakt v štátnom jazyku

BENEŠOVÁ, Katarína: Algoritmy na detekciu hrán v obrázkoch [diplomová práca], Univerzita Komenského v Bratislave, Fakulta matematiky, fyziky a informatiky, Katedra aplikovanej matematiky a štatistiky; školiteľ: Mgr. Soňa Kilianová, PhD., Bratislava, 2018, 80 s.

V práci sa venujeme metódam detekcie hrán so zameraním na Cannyho detektor. V súvislosti so zvýšením kvality modifikujeme Cannyho metódu a za tým účelom skúmame ďalšie možnosti na predspracovanie a post-spracovanie obrázkov, ako napr. bilaterálny filter, ekvalizáciu histogramu, ohraničenie s hysterézou a pod. Cieľom práce je preskúmať známe metódy a vhodne ich modifikovať, aby sme dosiahli lepšie výsledky. Cieľom je tiež navrhnúť automatickú voľbu parametrov pre takmer všetky fázy spracovania a vytvoriť algoritmus, ktorý zo vstupnej fotografie vytvorí obrázok s hranami so zameraním na esteticky pekný výsledok.

Kľúčové slová: detekcia hrán, Cannyho metóda, filtrácia obrázkov, Kirschov detektor, ekvalizácia histogramu, ohraničenie s hysterézou, post-spracovanie hrán

Abstract

BENEŠOVÁ, Katarína: Algorithms for edge detection in images [master thesis], Comenius University in Bratislava, Faculty of Mathematics, Physics and Informatics, Department of Applied Mathematics and Statistics; Supervisor: Mgr. Soňa Kilianová, PhD., Bratislava, 2018, 80 p.

In this thesis we examine methods of edge detection, particularly the Canny edge detector. In order to improve the quality of the output image we modify Canny's method by researching and developing methods for image pre-processing and post-processing, such as bilateral filtering, histogram equalization, hysteresis threshold and others. The goal of this thesis is to research known methods of edge detection in general and modify them in a suitable way, so that we can obtain better results. The goal is also to propose an autonomous choice of parameters for almost all phases of the process and to design an algorithm that transforms the input image to an image of detected edges with the focus on aesthetically neat output.

Keywords: edge detection, Canny edge detector, image filtering, Kirsch detector, histogram equalization, hystereris threshold, edge post-processing

Obsah

Úvod

V tejto práci sa zaoberáme detekciou hrán v čiernobielych obrázkoch so zameraním na bežné fotografie postáv, zvierat a budov. Práca je prakticky zameraná, s dôrazom na programovanie a esteticky pekné výsledné obrázky.

Detekcia hrán v obrázkoch je jednou zo základných oblastí spracovania obrazu a v súčasnosti sú známe mnohé prístupy na jej realizáciu. Typicky sú metódy založené na aproximácii veľkosti gradientu intenzít obrázka, pretože prudká zmena intenzity indikuje hranu, kým malá zmena indikuje plynulý farebný prechod. Aproximáciu magnitúdy gradientu je preto v princípe možné použiť ako intenzity nového obrázku s hranami.

Detekciu hrán treba chápať ako celý proces spracovania obrázku. Samotná aproximácia gradientu nie je postačujúca, pretože jej priamou aplikáciou dôjde k detekcii prebytočný hrán (šumu) alebo sa môže stať, že niektoré hrany detekované nebudú. Pre kvalitný výsledok je potrebné obrázok najprv vhodne transformovať a následne selektovať iba niektoré zo získaných hrany.

V našej práci sa zameriame najmä na Cannyho metódu, ktorá bola navrhnutá v roku 1986 Johnom Cannym [3], [4]. Táto viac-kroková metóda pozostáva okrem samotnej detekcie hrán z predspracovania obrázku a dodatočného post-spracovania získaných hrán. Odkedy bola metóda publikovaná boli vyvinuté ďalšie možnosti realizácie jednotlivých krokov, napr. [16], [9]. V našej práci sa pozrieme na princíp jednotlivých fáz Cannyho metódy, popíšeme iné spôsoby, ako sa dajú realizovať a modifikujeme ich. Tiež prezentujeme návrh pridania ďalšieho kroku predspracovania a invertovania s preškálovaním, ktoré je potrebné po aproximácii gradientu intenzít.

V teoretických prácach na tému detekcie hrán sa autori často nezameriavajú na praktické otázky realizácie popísaných metód. Napríklad, po detekcii hrán aproximáciou gradientu často dostaneme hodnoty, ktoré sú mimo bežného rozsahu intenzít $[0, 255]$. Otázkou je, ako ich vhodným spôsobom spätne preškálovať. V závislosti od použitého programovacieho jazyku môžu byť tiež invertované voči predošlému kódovaniu čiernej a bielej. Ďalšou otázkou je voľba vhodných parametrov, na ktorej môže zlyhať aj funkčná metóda. V našej práci navrhneme aj riešenia týchto praktických otázok.

V prvej kapitole popíšeme Cannyho metódu. Uvedieme akým spôsobom sa v nej obrá-

zok predspracováva Gaussovským filtrom, ako sa aproximuje gradient intenzity Sobelovým operátorom a ako sa selektujú finálne hrany podľa [3], [4], [16]. Uvedieme tiež, aké parametre sú potrebné k realizácii Cannyho metódy. Na konci prvej kapitoly zobrazujeme ukážku použitia, na ktorej si môžeme všimnúť, že hrán bolo v niektorých oblastiach detekovaných príliš veľa a niektoré podstatné hrany chýbajú.

V druhej kapitole uvádzame iné možnosti realizácie jednotlivých krokov Cannyho metódy a ukážky ich použitia na reálnych fotografiách. Tiež navrhujeme pridať krok s ekvalizáciou histogramu [17] s cieľom zvýšiť kontrast na obrázku. Tento krok sa v Cannyho ani inej štandardnej metóde nevyskytuje, ale myslíme si, že pre obrázky so slabým kontrastom je vhodné ho pridať. V tejto kapitole formalizujeme myšlienky z [17] a navrhneme alternatívnu realizáciu posledného kroku tejto transformácie oproti [17]. Tiež prezentujeme našu modifikáciu určenia tzv. dynamického rozsahu intenzít, s ktorým sa v tejto metóde pracuje. V rámci predspracovania navrhujeme použiť vyhladzovací filter, ktorý z obrázku odstráni šum miernym rozmazaním, ale ktorý, na rozdiel od Gaussovského filtra z Cannyho metódy, zachová hrany nerozmazané [16]. Ďalej uvádzame rôzne operátory pre aproximáciu gradientu ako alternatívu k Sobelovmu, napr. Prewittov a Kirschov [9], [6]. Následne navrhujeme pozmeniť spôsob stenčovania hrán v post-spracovaní oproti Cannyho metóde [3], [4]. V rámci druhej kapitoly tiež spomenieme detekciu hrán spojitými metódami pomocou diferenciálnych rovníc [8], [9].

V tretej kapitole sumarizujeme návrh nášho modifikovaného algoritmu a najmä ako navrhujeme voliť parametre.

Celú našu modifikovanú metódu, popísanú druhej a tretej kapitole, sme naprogramovali v Python 3. Zdrojový kód ku dynamickej ekvalizácii histogramu spôsobom z [17], aj našu modifikáciu, prikladáme v prílohe A. Na záver uvádzame v štvrtej kapitole Galéria výsledkov obrázky spracované naším kódom pre rôzne typy obrázkov a varianty algoritmu.

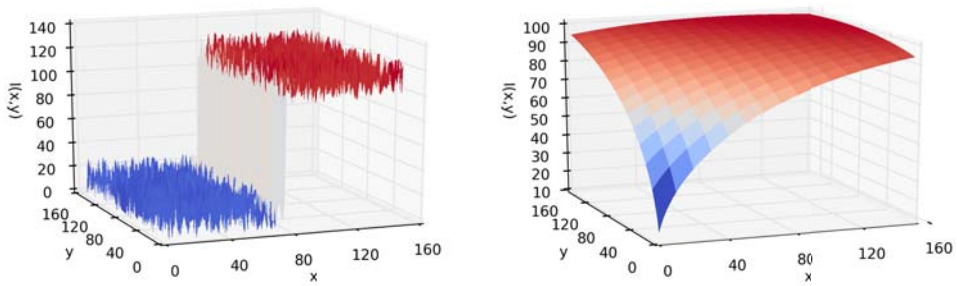
1 Základné myšlienky detekcie hrán a Cannyho metóda

Ako sme naznačili v úvode, okrem samotnej detekcie hrán ako prudkých zmien intenzity treba proces detekcie hrán chápať ako niekoľkofázovú transformáciu obrázka pozostávajúcu z predspracovania, detekcie zmien intenzity a post-spracovania získaných hrán. V tejto kapitole uvádzame známe myšlienky a zaužívané metódy detekcie hrán diskrétnym prístupom. Popíšeme najmä Cannyho detektor hrán, ktorý bude východiskom pre túto prácu.

1.1 Gradienty v obrázku

Budeme uvažovať čiernobiely obrázok ako mriežku $m \times n$ pixlov. Obrázok je daný funkciou $I(x, y) : \mathbb{R}^2 \to \mathbb{R}$, ktorá každému pixlu s polohou $[x, y]$ priradí intenzitu čiernej farby $I(x, y)$. V takomto kontexte je prirodzené predstaviť si intenzitu $I(x, y)$ ako tretí rozmer, teda skôr ako výšku, než farbu [5]. V súlade s použitým programovacím jazykom Python 3 budeme rozsah intenzít kódovať celými číslami od 0 po 255 tak, že 0 predstavuje čiernu a 255 bielu.

V bode $[x, y]$ sa nachádza hrana, ak je v jeho okolí výrazná zmena intenzity čiernej a nenachádza, ak sa intenzita mení plynulo. Oba prípady sú znázornené na Obr. 1.



Obr. 1: Zmena intenzity čiernej je výrazná na okolí bodu s hranou (vľavo) a plynulá na okolí bez hrany (vpravo)

Ak chceme detekovať hrany, je prirodzené pozrieť sa na zmenu funkcie $I(x, y)$ na okolí $[x, y]$, teda na veľkosť gradientu $I(x, y)$. Táto myšlienka je základom diskrétnych metód. Formálne môžeme takúto detekciu hrán definovať naledovne:

Definícia 1.1 (Detekcia hrán [14])**.**
Detekcia hrán ako výraznej zmeny intenzity čiernej je priradenie $g : \mathbb{R}^2 \to \mathbb{R}$

$$g(x, y) = |\nabla I(x, y)|. \tag{1}$$

Intenzity nového obrázku s hranami sú potom dané hodnotami $g(x, y)$. Keďže funkcia $I(x, y)$ v realite nie je hladká, jej gradient môžeme vypočítať iba približne.

1.2 Aproximácia gradientu intenzity

Gradient $I(x, y)$ by sme chceli aproximovať pomocou intenzít v okolitých bodoch. Funkciu $I(x, y)$ má zmysel uvažovať len v $m \times n$ diskrétnych bodoch na definičnom obore danom veľkosťou obrázku: $\mathcal{D} = [0, 1, \ldots, m-1] \times [0, 1, \ldots, n-1]$. Za počiatok súradníc budeme v teórii považovať ľavý dolný roh $[0, 0]$. Nech premenné x, y reprezentujú horizontálny, resp. vertikálny smer rastúci smerom doprava, resp. hore. Označme odhad zmeny intenzity v horizontálnom smere ako

$$\frac{\partial I(x, y)}{\partial x} \approx G_x, \tag{2}$$

a zmenu vo vertikálnom smere ako

$$\frac{\partial I(x, y)}{\partial y} \approx G_y. \tag{3}$$

Ďalej označme $\hat{g}(x, y)$ výslednú aproximáciu veľkosti gradientu $I(x, y)$ pri vhodne zvolenej norme G_x a G_y:

$$\hat{g}(x, y) = \|(G_x, G_y)\|, \tag{4}$$

napríklad v euklidovskej [9]:

$$\hat{g}(x, y) = \sqrt{G_x^2 + G_y^2}. \tag{5}$$

Aproximáciu smeru gradientu môžeme vypočítať ako [9]:

$$\angle\hat{g}(x, y) = \tan^{-1}\left(\frac{G_y}{G_x}\right). \tag{6}$$

Pomocou (4) získame nový obrázok, kde má každý bod $[x, y]$ novú intenzitu $\hat{g}(x, y)$, ktorá je odhadom $g(x, y)$ z (1).

V ďalšom sa budeme venovať celému procesu detekcie hrán vrátane počiatočných a dodatočných transformácií. Tento proces dobre reprezentuje známy Cannyho detektor hrán, ktorého štruktúra bude slúžiť ako základ pre vybudovanie nášho algoritmu.

1.3 Cannyho metóda

Myšlienky detektorov hrán sú založené na aproximácii gradientu, ako sme popísali v časti 1.2. Priamou aplikáciou aproximácie gradientu na obrázok dôjde aj k detekcii šumu, čo vedie k prebytočným hranám. Preto samotná aproximácia gradientu $I(x, y)$ nedáva postačujúci výsledok a obrázok treba predspracovať, napríklad miernym rozmazaním, a po aproximácii gradientu post-spracovať selekciou vhodných hrán.

Pri procese predspracovania, detekcie hrán a post-spracovania budeme v našom algoritme vychádzať zo známej Cannyho metódy. Jeho myšlienka sa dá podľa [3], [4] zhrnúť do takýchto krokov:

1. Predspracovanie: rozmazanie obrázku Gaussovským filtrom
2. Detekcia hrán aproximáciou gradientu použitím Sobelovho detektora
3. Post-spracovanie: stenčenie hrán získaných po druhom kroku
4. Post-spracovanie: selekcia finálnych hrán metódou tzv. ohraničenia s hysterézou.

Naším cieľom bude zachovať myšlienku Cannyho metódy, vylepšiť jednotlivé kroky, pridať ďalšie predspracovanie a automatizovať výber väčšiny parametrov potrebných k ich realizácii.

Ďalej detailnejšie rozoberieme kroky Cannyho metódy a popíšeme ich výhody a nevýhody.

1.3.1 Predspracovanie Gaussovským priestorovým filtrom

V prvom kroku je vhodné obrázok simplifikovať s cieľom odstrániť šum, ktorý by bol detekovaný ako hrany. Základnou používanou metódou je aplikácia priestorového filtra, ktorý obrázok čiastočne rozmaže, čím dôjde k vyhladeniu funkcie $I(x, y)$.

Priestorový filter v danom bode $u = [x_0, y_0] \in \mathcal{D}$ je spojitou verziou váženého priemeru intenzít v okolitých bodoch ξ. Čím ďalej je bod ξ od u, tým menšiu váhu dostane intenzita v bode ξ. Priestorový filter je definovaný nasledovne:

Definícia 1.2 (Priestorový filter intenzity čiernej [16])**.**
Výsledkom aplikácie priestorového filtra v danom bode $u = [x_0, y_0] \in \mathcal{D}$ pomocou intenzít

v okolitých bodoch $\xi = [x, y]$ je nová intenzita $W(u)$ daná nasledovne:

$$W(u) = z^{-1}(u) \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} I(\xi) f_d(\xi, u) d\xi, \tag{7}$$

kde $f_d(\xi, u)$ je vhodne zvolená váhovacia funkcia reprezentujúca vzdialenosť u od ξ (d - distance) a $z(u)$ je normalizačná konštanta

$$z(u) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f_d(\xi, u) d\xi. \tag{8}$$

Jedným z najpoužívanejších je Gaussovský priestorový filter, ktorý je použitý aj v Cannyho metóde. V tomto filtri sa funkcia $f_d(\xi, u)$ volí ako funkcia hustoty dvojrozmerného normálneho rozdelenia so stredom v bode u a varianciou σ_d^2 [16]:

$$f_d(\xi, u) = e^{-\frac{1}{2}\left(\frac{d(\xi,u)}{\sigma_d}\right)^2}, \tag{9}$$

kde $d(\xi, u)$ je euklidovská vzdialenosť ξ od u [16]:

$$d(\xi, u) = \|\xi - u\|_2, \tag{10}$$

a normalizačná konštanta $z^{-1}(u)$ je rovnaká pre všetky u a má tvar

$$z^{-1} = \frac{1}{2\pi\sigma_d}. \tag{11}$$

Z charakteru Gaussovskej funkcie je zrejmé, že do novej intenzity $W(u)$ vstupujú prakticky iba intenzity bodov ξ z blízkeho okolia u, pretože intenzity ďaleko od u dostanú veľmi malé váhy. Rozsah tohto okolia je závislý od parametra σ_d.

Ďalej sa pozrieme ako Gaussovský filter prakticky aplikovať.

1.3.2 Aplikácia Gaussovského priestorového filtra na obrázok

Gaussovský filter prakticky predstavuje vážený priemer intenzít na štvorci $s \times s$ bodov $\xi_1, \xi_2, \ldots, \xi_{s \times s}$ so stredom v bode u s váhami danými $f_d(\xi, u)$ [16].

Pozn. *Pre body u na krajoch obrázku neexistujú intenzity $I(\xi)$ pre celé $s \times s$ okolie, preto sa chýbajúce intenzity nahradia vhodným spôsobom. V jazyku Python je štandardne predvolená tzv. reflektujúca hranica, kde sa chýbajúce intenzity nahradia zrkadlovým obrátením niekoľkých intenzít pri okraji obrázka [11]. Napríklad, ak označíme hranicu obrázka*

symbolicky ako | a ak sú intenzity pri okraji a, b, c, d, e, f, g, h, dodefinovanie šiestich bodov pomocou reflektujúcej hranice môžeme schematicky reprezentovať nasledovne [11]:

$$gfedcb|abcdefgh|gfedcba.$$

V našom algoritme použijeme toto riešenie.

Keďže $f_d(\xi, u)$ závisí iba od relatívnej vzdialenosti ξ od u, jej hodnoty nezávisia od konkrétnej polohy u na obrázku. Po voľbe σ_d a veľkosti okolia s sú všetky potrebné hodnoty $f_d(\xi, u)$ známe a rovnaké pre všetky body u.

Aplikáciu Gaussovského filtra môžeme zhrnúť do troch krokov:

1. Zvolíme σ_d, nepárne číslo s a bez ujmy na všeobecnosti zvolíme $u_0 = [0, 0]$.

2. Funkciu $f_d(\xi, u_0)$ vyčíslime v $s \times s$ pixloch $\overline{\xi}_1, \overline{\xi}_2, \ldots, \overline{\xi}_{s\times s}$ na sústredných štvorcoch so stredom v u_0 so stranami dĺžky $1, 3, 5, \ldots, k$ pixlov, t.j. na štvorci s vrcholmi $[-\frac{s-1}{2}, -\frac{s-1}{2}], [-\frac{s-1}{2}, \frac{s-1}{2}], [\frac{s-1}{2}, \frac{s-1}{2}], [\frac{s-1}{2}, -\frac{s-1}{2}]$ a $s \times s$ mrežovými bodmi na celých číslach. Tým získame $s \times s$ váh w_i pre intenzity bodov $\xi_1, \xi_2, \ldots \xi_{s\times s}$ z okolia každého bodu u, závislé na vzdialenosti od príslušného bodu u:

$$w_i = w_i(d(\xi_i, u)) = f_d(\overline{\xi}_i, u_0), \qquad i = 1, 2, \ldots, s \times s, \tag{12}$$

   kde ξ_i zodpovedá bodu $\overline{\xi}_i$, pre ktorý $d(\overline{\xi}_i, u_0) = d(\xi_i, u)$.

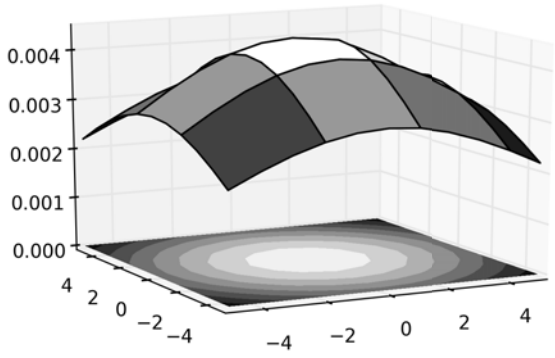
   Normalizačná konštanta z má tvar:

$$z = \sum_{i=1}^{s^2} w_i. \tag{13}$$

3. Pre každý bod u vypočítame novú intenzitu $W(u)$, ktorá je váženým priemerom intenzít $I(\xi_1), I(\xi_2), \ldots, I(\xi_{s\times s})$ v $s \times s$ okolitých bodoch s váhami a normalizačnou konštantou ako sme popísali v kroku 2:

$$W(u) = \frac{1}{z} \sum_{i=1}^{s^2} w_i I(\xi_i). \tag{14}$$

Príklad funkcie $f_d(\xi, u)$ pre konkrétnu voľbu σ_d a s môžeme vidieť na Obr. 2.



Obr. 2: Funkcia $f_d(\xi, u_0)$ s parametrom $\sigma_d = 6$ na 11×11 okolí $u_0 = [0, 0]$

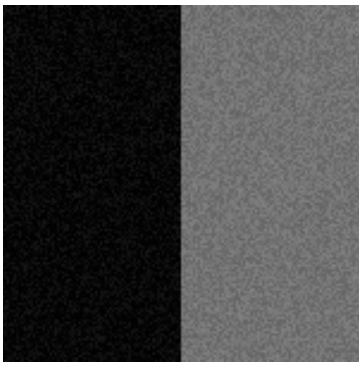
Keďže váhy sme vyčísľovali na sústredných štvorcoch, môžeme ich po voľbe σ_d a s prirodzene reprezentovať $s \times s$ maticou, kde pozícia váhy pre bod ξ_i voči stredu matice je daná relatívnou pozíciou ξ_i voči príslušnému u. V tabuľke 1 môžeme vidieť $s \times s$ maticu váh, ktorá vznikla z hodnôt $f_d(\xi, u)$ z Obr. 2 po vydelení normalizačnou konštantou z (13). Váhy sme zaokrúhlili na tri desatinné miesta a vyňali $\frac{1}{1000}$ kvôli lepšej prehľadnosti. V matici si môžeme všimnúť zachovanú rotačnú symetriu Gaussovskej funkcie.

$\frac{1}{1000}$

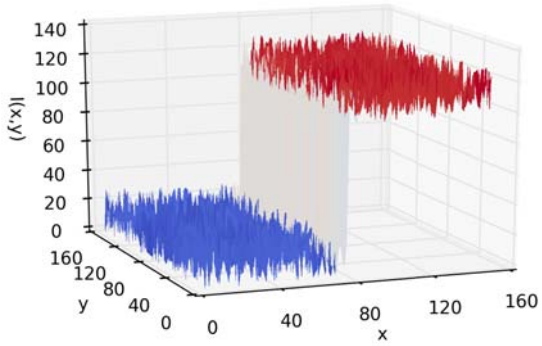
5	6	7	7	7	8	7	7	7	6	5
6	7	8	8	8	9	8	8	8	7	6
7	8	8	9	9	9	9	9	8	8	7
7	8	9	10	10	10	10	10	9	8	7
7	8	9	10	10	11	10	10	9	8	7
8	9	9	10	11	11	11	10	9	9	8
7	8	9	10	10	11	10	10	9	8	7
7	8	9	10	10	10	10	10	9	8	7
7	8	8	9	9	9	9	9	8	8	7
6	7	8	8	8	9	8	8	8	7	6
5	6	7	7	7	8	7	7	7	6	5

Tabuľka 1: Matica váh Gaussovského filtra s parametrami $\sigma_d = 6, s = 11$, ktorá vznikla z hodnôt $f_d(\xi, u)$ z Obr. 2.

Aplikáciou Gaussovského filtra vyhladíme obrázok a odstránime prebytočné hrany (šum), ale tiež nutne rozmažeme aj skutočné hrany, keďže v okolí hrany sa typicky vyskytujú intenzity rôznej veľkosti. Po filtrácii Gaussovským filtrom sa skutočná hrana rozšíri o oblasť s priemernou intenzitou jej okolia, teda sa rozmaže a bude hrubšia. Príklad použitia Gaussovského filtra z Tabuľky 1 na oblasť s hranou uvádzame na Obr. 3.

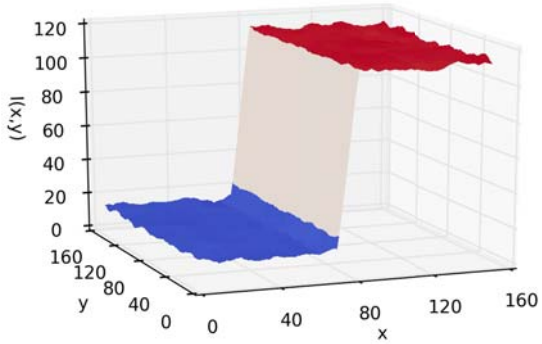


(a) Pôvodný obrázok



(b) $I(x, y)$ pôvodného obrázka

(c) Po aplikácii Gaussovského filtra



(d) $I(x, y)$ filtrovaného obrázka

Obr. 3: Ukážka použitia Gaussovského filtra z Tabuľky 1 vpravo. Funkcia $I(x, y)$ je vyhladená (d) a pôvodná hrana je rozmazaná (c)

1.3.3 Detekcia hrán Sobelovým operátorom

Po vyhladení obrázka Gaussovským priestorovým filtrom môžeme prejsť k samotnej detekcii hrán. Sobelov operátor je z triedy diskrétnych metód, ktoré v princípe používajú vážený súčet intenzít v okolitých pixloch s cieľom aproximovať gradient. Metódy sa líšia voľbou váh a veľkosti uvažovaného okolia.

Sobelov operátor je jedným z najpoužívanejších filtrov, ktorý váhuje 3×3 okolie bodu $[x, y]$ a detekuje horizontálne a vertikálne zmeny intenzity. Aplikáciu tohto filtra v bode $[x, y]$ explicitným spôsobom uvádzame v nasledovnej definícii 1.3 a alternatívnu definíciu pomocou konvolučných matíc v definícii 1.4.

Definícia 1.3 (Sobelov detektor [9])**.**

Nová intenzita $\hat{g}(x, y)$ pre bod $[x, y] \in \mathcal{D}$ pri detekcii hrán Sobelovým detetorom je daná nasledovne:

$$\begin{aligned} G_x = {} & I(x+1, y+1) - I(x-1, y+1) + 2I(x+1, y), \\ & - 2I(x-1, y) + I(x+1, y-1) - I(x-1, y-1), \end{aligned} \tag{15}$$

$$\begin{aligned} G_y = {} & - I(x-1, y+1) + I(x-1, y-1) - 2I(x, y+1) \\ & + 2I(x, y-1) - I(x+1, y+1) + I(x+1, y-1), \end{aligned} \tag{16}$$

$$\hat{g}(x, y) = |G_x| + |G_y|. \tag{17}$$

Vo vzorcoch (15) a (16) ide o aproximáciu derivácie symetrickou diferenciou, takže pravé strany by mali byť vydelené číslom 8. Pri Sobelovom detektore však ide iba o koeficienty, ktorými sa násobia intenzity okolitých bodov, preto sa normalizácia vynecháva. Z dôvodov výpočtovej efektivity je vo výpočte $\hat{g}(x, y)$ v (17) v súlade s Python 3 zvolená L_1 norma namiesto štandardnej euklidovskej [10].

Koeficienty z (15) a (16) môžeme schematicky zapísať do konvolučných matíc tak, že pozícia v matici relatívne k jej stredu udáva, ktorý bod s relatívnou polohou k $[x, y]$ dostane príslušnú váhu, napr. koeficient v pravom hornom rohu násobí hodnotu $I(x+1, y+1)$. Sobelov detektor definovaný pomocou konvolučných matíc aplikujeme ako Hadamardov súčin s príslušným 3×3 okolím centrálneho bodu $[x, y]$ pre všetky $[x, y] \in \mathcal{D}$.

Za účelom aplikácie Sobelovho detektora opäť dodefinujeme chýbajúce intenzity $I(x, y)$ pre okrajové body pomocou reflektujúcej hranice ako sme popísali v časti 1.3.2. Polohu centra $[x, y]$ budeme v konvolučných maticiach značiť tučným písmom.

Definícia 1.4 (Konvolučné matice Sobelovho detektora [9])**.**

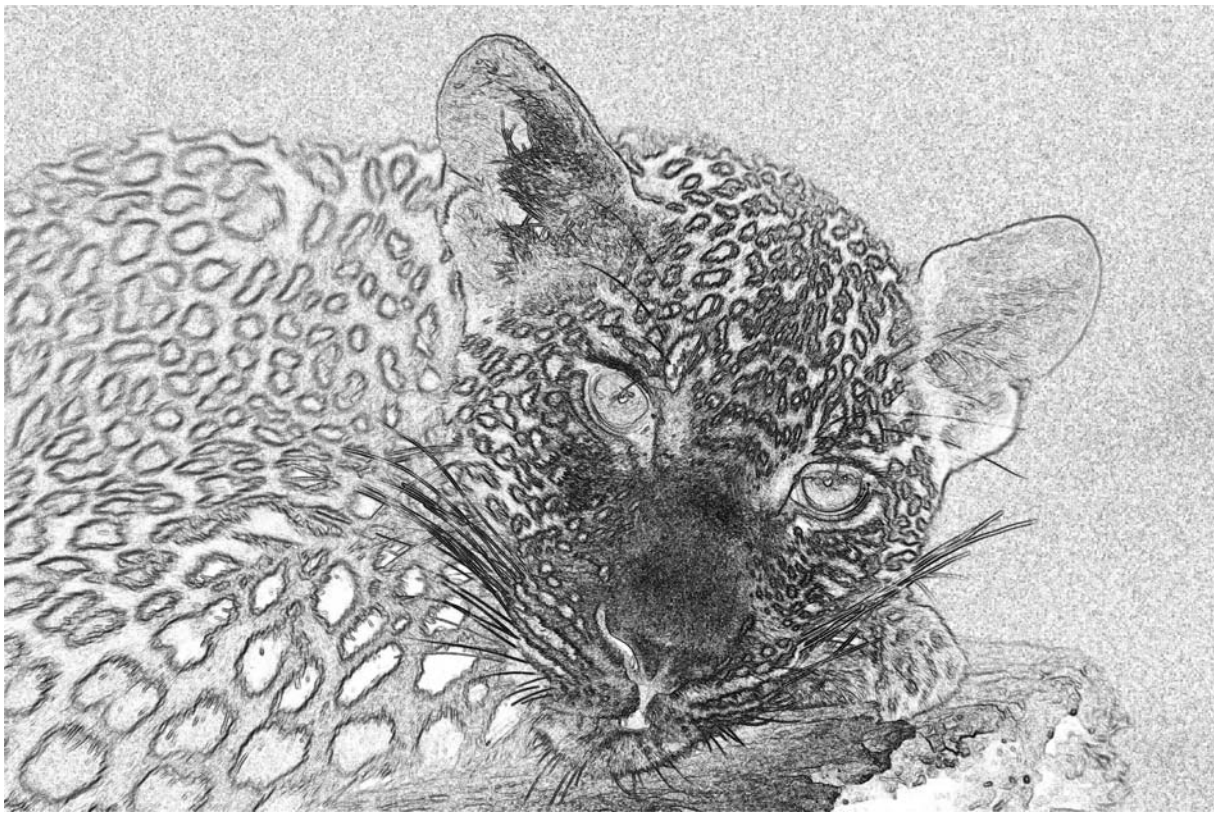
Sobelov detektor možno reprezentovať nasledovnými konvolučnými maticami:

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & \mathbf{0} & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & \mathbf{0} & 0 \\ 1 & 2 & 1 \end{bmatrix}. \tag{18}$$

Matica S_x zodpovedá výpočtu G_x z (15) a S_y zodpovedá výpočtu G_y z (16).

Príklad detekcie hrán Sobelovým detektorom uvádzame na Obr. 4.

(a)

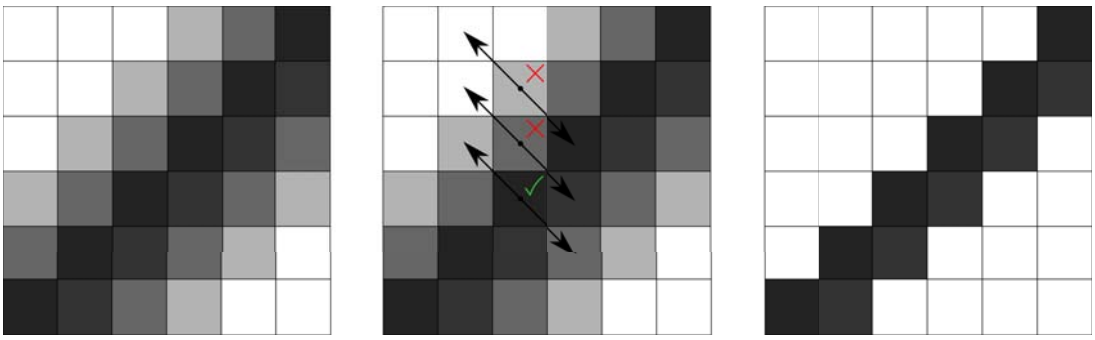


(b)

Obr. 4: Ukážka detekcie Sobelovým filtrom: pôvodný obrázok (a), po detekcii (b), invertované a preškálované na $[0, 255]$ pomocou vzťahu (98)

1.3.4 Post-spracovanie: stenčenie hrán v smere gradientu

V druhom kroku sme získali aproximáciu veľkosti gradientu $\hat{g}(x, y)$ v každom bode $u = [x, y]$ z obrázka. Ak je táto hodnota v bode u tmavšia, t.j. u nás menšia, ako v jeho dvoch susedných pixloch v smere gradientu (resp. v smere gradientu a v smere opačnom), označíme bod u ako pixel s hranou. Inak ho označíme ako pozadie a intenzitu zmeníme na bielu. Za smer gradientu sa považuje smer, v ktorom dochádza k najväčšej zmene intenzity [4]. Ukážku tohoto princípu môžeme vidieť na Obr. 5



Obr. 5: Stenčenie hrán: 6x6 obrázok (vľavo), porovnanie intenzity susedných bodov v smere gradientu pre tri vybrané body (stred), obrázok po selekcii (vpravo)

Týmto krokom sa snažíme spätne stenčiť hrubé hrany, ktoré sme dostali v dôsledku aplikácie Gaussovského priestorového filtra. Hrany, ktoré prešli touto prvou selekciou, t.j. neboli označené ako pozadie, spracovávame ďalej v kroku 4.

1.3.5 Post-spracovanie: ohraničenie s hysterézou

V poslednom kroku sa snažíme odstrániť príliš slabé hrany, ale zároveň vo výsledku zachovať súvislé hrany, metódou ohraničenia s hysterézou (z angl. *hysteresis thresholding*).

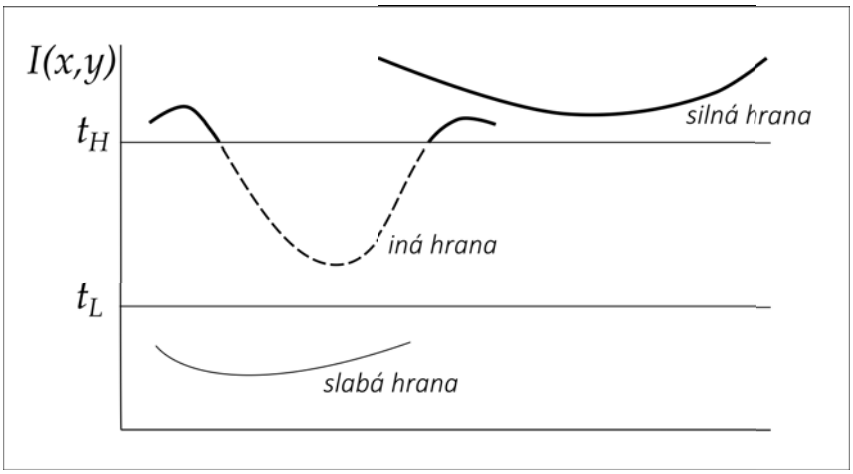
Uvažujeme všetky súvislé krivky získané krokmi 1-3 a zvolíme dve hranice pre intenzity čiernej: t_H a t_L. Ďalej postupujeme nasledovne [4]:

1. Krivky obsahujúce iba body, ktorých intenzity ležia pod hranicou t_L, nazveme *slabé hrany* a celé ich odstránime.

2. Krivky obsahujúce iba body, ktorých intenzity ležia nad hranicou t_H, nazveme *silné hrany*. Tieto krivky prešli selekciou a označíme ich ako *finálne hrany*.

3. O ostatných krivkách, ktoré nie sú *slabé* ani *silné*, sa rozhodujeme podľa toho, či obsahujú časť, ktorá je *silnou hranou*. Ak áno, potom je celá krivka označená ako *finálna hrana*, inak je odstránená.

Myšlienku tohto kroku ilustrujeme na Obr. 6.



Obr. 6: Myšlienka ohraničenia s hysterézou. Hrana prerušovanou čiarou by bola vybraná medzi finálne hrany, pretože obsahuje časť so silnou hranou.

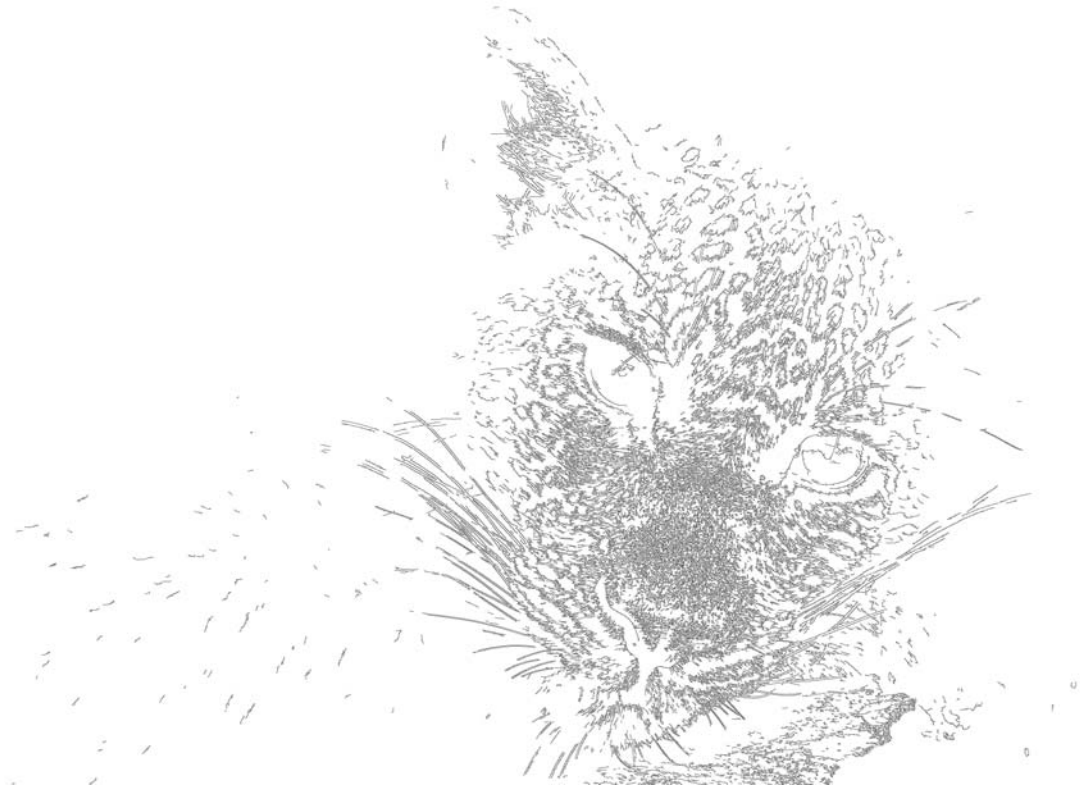
Týmto spôsobom odstránime iba také hrany, ktoré nedosiahnu ani t_L, alebo ju síce dosiahnu, ale nie sú spojené s niektorou zo silných hrán.

Pozn. *V [3], [4] uvažujú opačné kódovanie čiernej a bielej farby, teda čím vyššia intenzita, tým bližšie k čiernej. V našom prípade v obrázku z kroku 3 invertujeme každú intenzitu k jednoduchou transformáciou $k \longrightarrow 255 - k$, urobíme ohraničenie s hysterézou a invertujeme rovnakým spôsobom naspäť.*

1.4 Ukážka použitia Cannyho detektora

Cannyho detektor je zabudovaný v Python 3 v knižnici OpenCV, avšak parametre t_L a t_H musí zadať používateľ, od čoho značne závisí výsledok. Na Obr. 7 uvádzame príklad detekcie Cannyho metódou s voľbou parametrov $t_L = 100$ a $t_H = 200$ uvedenou v dokumentácii OpenCV [11]. Keďže Cannyho metóda zabudovaná v OpenCV vráti obrázok v opačnom kódovaní čierna = 255, biela = 0, invertujeme ho jednoduchou transformáciou $k \longrightarrow 255 - k$.

(a)



(b) $t_L = 100, t_H = 200$

Obr. 7: Ukážka transformácie obrázka *(a)* Cannyho metódou *(b), invertované.* Zdroj obrázku:

`https://www.elephantplains.co.za`

2 Možnosti realizácie procesu detekcie hrán a návrh modifikovaného algoritmu

Schéma Cannyho detektora sa dá vyjadriť v troch všeobecných fázach:

1. Predspracovanie: príprava obrázku
2. Detekcia hrán aproximáciou gradientu
3. Post-spracovanie: selekcia finálnych hrán

V navrhovanej modifikácii algoritmu budeme vychádzať z tejto všeobecnej schémy, pričom jednotlivé kroky zmeníme tak, aby sme dostali krajší výsledok. V ďalšom uvedieme iné možnosti realizácie jednotlivých krokov Cannyho metódy a vyberieme najvhodnejšie. Niektoré metódy modifikujeme vlastným spôsobom. V rámci detekcie hrán aproximáciou gradientu tiež spomenieme základy spojitého prístupu založenom na riešení parciálnych diferenciálnych rovníc.

2.1 Zvýšenie kontrastu ekvalizáciou histogramu

V rámci predspracovania navrhujeme zvýšiť kontrast obrázku, čím dôjde k zvýrazneniu prechodov medzi hranami. Môže sa stať, že vstupný obrázok je v príliš bledej alebo príliš tmavej škále. Tým pádom sú hrany ťažšie rozoznateľné, pretože všetky intenzity sú vtesnané do užšieho rozsahu ako intenzity v obrázku s celou škálou $[0, 255]$.

Kontrast môžeme zvýšiť ekvalizáciou histogramu obrázku, t.j. preškálovaním intenzít, ktoré ležia v užšom intervale, na plnú škálu $[0, 255]$. Týmto krokom tiež čiastočne normalizujeme vstupné obrázky v zmysle rozsahu a zastúpenia intenzít, čo nám uľahčí výber parametrov potrebných pri ďalšom spracovaní.

Ďalej rozoberieme a formalizujeme myšlienku globálnej a dynamickej ekvalizácie popísanú v [17]. V rámci dynamickej ekvalizácie navrhneme našu modifikáciu určenia tzv. dynamického rozsahu intenzít a alternatívnu verziu tretieho kroku.

2.1.1 Globálna ekvalizácia histogramu

Idea zvýšenia kontrastu spočíva v čiastočnom vyrovnaní histogramu obrázku, čím sa presunie plocha pod histogramom zo stredu škály, zodpovedajúcej stredne šedým odtieňom, ku 0 (čierna) a 255 (biela). Malo by dôjsť k zvýrazneniu farebných prechodov medzi plochami s odlišnou intenzitou a medzi hranami a pozadím, príp. ich výplňou.

Definujme histogram obrázku ako početnosť pixlov s intenzitou k v rozmedzí od čiernej po bielu:

Definícia 2.1 (Histogram obrázku)**.**
Globálny histogram obrázku s intenzitami $I(x, y)$ je daný vzťahom:

$$H(k) = \sum_{I(x,y)=k} 1, \qquad k = 0, 1, \ldots, 255 \tag{19}$$

Globálna ekvalizácia histogramu je podľa [17] mapovanie pixlov s intenzitou k na nové intenzity pomocou transformačnej funkcie z (20):

Definícia 2.2 (Globálna ekvalizácia histogramu [17])**.**
Globálna ekvalizácia histogramu (GEH) je definovaná vzťahom:

$$s_k = \sum_{i=0}^{k} \frac{H(k)}{N}, \qquad k = 0, 1 \ldots, 255, \qquad s_k \in [0, 1], \tag{20}$$

$$N = \sum_{k=0}^{255} H(k) \tag{21}$$

kde N je celková plocha pod histogramom, t.j. počet pixlov na obrázku, s_k predstavuje odhad distribučnej funkcie intenzít v bode k. Hodnoty s_k prevedieme spätne na škálu $[0, 255]$ faktorom 255 a nové intenzity $\bar{I}(x, y)$ mapujeme spôsobom

$$k \longrightarrow 255 s_k,$$

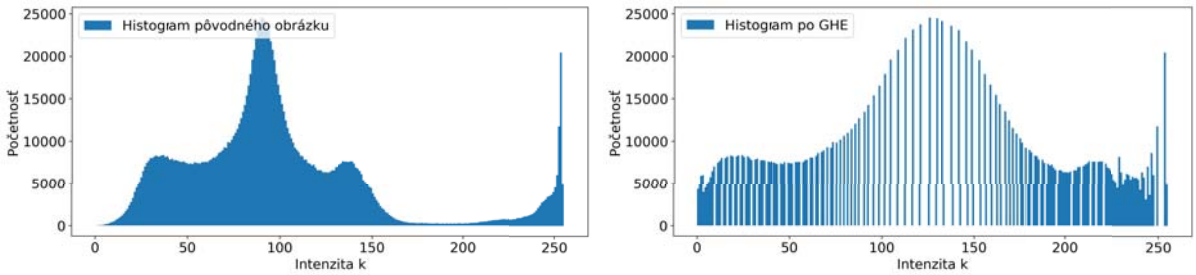
t.j. vzťahom:

$$\bar{I}(x, y)\Big|_{[x,y]:I(x,y)=k} = 255 s_k \,^*, \qquad [x, y] \in \mathcal{D} \tag{22}$$

** zaokrúhlené na celé čísla*

Pre každú intenzitu k napočítame vzťahom (20) podiel pixlov s intenzitou nanajvýš k. Voľne povedané to znamená, že ak je málo pixlov tmavších ako k, nová hodnota pre pixle

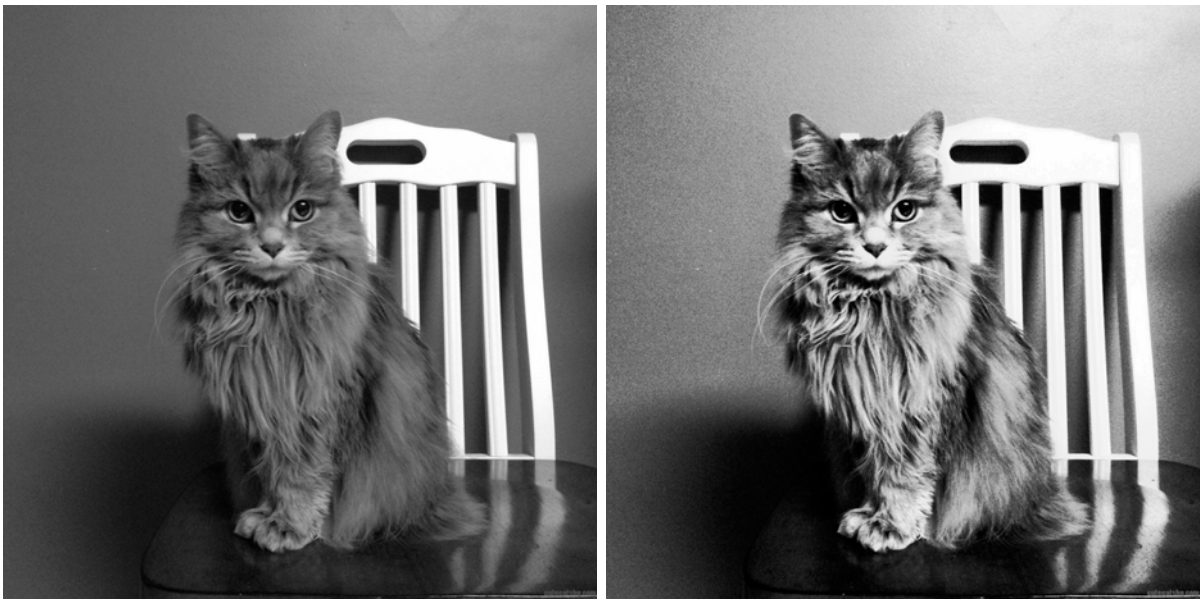
s intenzitou k bude menšia, t.j. budú tmavšie, a ak je málo pixlov bledších ako k, budú bledšie. Ukážku transformácie histogramu pomocou GEH uvádzame na Obr. 8.



Obr. 8: Histogram obrázku pred GEH (vľavo) a po GEH (vpravo)

Hoci použitím GEH dosiahneme zvýšenie kontrastu, metóda má neželané vedľajšie efekty, predovšetkým príliš silné zvýšenie kontrastu. Navyše, ak je n_k veľké, spôsobí väčší nárast s_k oproti s_{k-1}. V tom prípade sa môže stať, že na niekoľko po sebe idúcich intenzít nasledujúcich po intenzite $255s_{k-1}$, na ktorú bola namapovaná intenzita $k-1$, sa nemusí namapovať žiadna pôvodná intenzita. Zároveň to spôsobí, že intenzity s menším n_k budú naopak namapované na veľmi malý rozsah, príp. na rovnakú intenzitu. Tieto vedľajšie efekty sú hlavným zdrojom straty významných čŕt na obrázku [17].

Ukážku použitia GEH môžeme vidieť na Obr.9. Použitím GEH došlo ku zvýšeniu kontrastu, ale došlo aj k zvýrazneniu nechcených hrán (šumu).



Obr. 9: Ukážka transformácie obrázka (vľavo) s histogramom na Obr. 8 hore globálnou ekvalizáciou histogramu (vpravo). Zdroj obrázku: `www.pinterest.com`

Prílišným vyrovnaním histogramu sa teda ľahko môže stať, že sa aj hrany, ktoré by sme chceli zachovať, stanú ťažko rozoznateľné, alebo budú zvýraznené prechody, ktoré by sme nechceli detekovať ako hrany. Ako riešenie tohoto problému bola v [17] navrhnutá dynamická ekvalizácia.

2.1.2 Dynamická ekvalizácia histogramu

Pri dynamickej ekvalizácii histogramu (DEH) v snahe eliminovať dominanciu niektorých intenzít na priveľkej škále v novom histograme aplikujeme ekvalizáciu osobitne po častiach histogramu (subhistogramoch). Princíp DEH a približný návod na realizáciu bol opísaný v [17]. Ďalej uvádzame našu interpretáciu, formalizáciu a návrh na implementáciu.

Zaveďme nasledovnú pomocnú definíciu:

Definícia 2.3 (Subhistogram)**.**
Subhistogram $h_{L,U}(k)$ je výsek $U-L$ hodnôt globálneho histogramu $H(k)$ alokovaný škálou intenzít medzi dolnou hranicou L (vrátane) a hornou hranicou U (nevrátane):

$$h_{L,U}(k) = H(k), \qquad k \in L, L+1, \ldots, U-1 \tag{23}$$

Pri DEH má každý subhistogram $h_{L,U}(k)$ vopred pridelenú škálu odtieňov šedej, na ktorú budú mapované jeho intenzity $k \in [L, U)$. Týmto zabránime prílišnej expanzii dominujúcich intenzít [17].

Uvažujme obrázok s intenzitami $I(x, y)$ s histogramom $H(k)$, ktorého hraničné hodnoty sú:

$$\begin{aligned} M_{min} &= \min_{H(k)>0} k, \\ M_{max} &= \max_{H(k)>0} k+1. \end{aligned} \tag{24}$$

Celkovým dynamickým rozsahom intenzít šedej budeme nazývať šírku $D \leq 255$ danú, ako:

$$D = M_{max} - M_{min}. \tag{25}$$

V [17] bolo navrhnuté považovať za hraničné hodnoty prvú a poslednú nenulovú hodnotu ako uvádzame v (24). Pri použití na reálne fotografie sme sa stretli s tým, že intenzitu 0 a 255 bude mať takmer určite niekoľko málo pixlov, čo vedie k $D = 255$, hoci intenzít blízko 0 a 255 nie je veľa. Plná škála $D = 255$ pri DEH bráni expanzii histogramu

smerom ku krajným intenzitám bielej a čiernej, čím strácame časť želaného efektu. Preto navrhujeme pozmeniť toleranciu 0 pixlov v (24) na

$$\begin{aligned} M_{min} &= \min_{H(k)>a} k, \\ M_{max} &= \max_{H(k)>a} k+1, \end{aligned} \tag{26}$$

a zo skúseností z experimentov navrhujeme položiť $a = 100$.

Cieľom DEH je zväčšiť v obrázku celkový dynamický rozsah D na plnú škálu $\bar{D} = 255$, avšak nedovoliť dominujúcim intenzitám zabrať príliš široké spektrum intenzít ako v prípade GEH. Myšlienku tejto metódy ilustrujeme na Obr. 11.

Dynamická ekvalizácia histogramu pozostáva z nasledovných krokov:

1. Segmentácia histogramu

Na histogram najprv aplikujeme vyhladzovací filter. V [17] bolo navhrnuté použiť bližšie nešpecifikovaný 1×3 filter. Zvolili sme Gaussovský 1D filter, ktorý je jednorozmernou analógiou filtra, ktorý sme popísali v kapitole 1.3.1. Experimentálne sme zistili, že filter veľkosti 3 má na histogram príliš slabý efekt, preto navrhujeme použiť 1×7 filter so $\sigma = 3$.

Následne detekujeme lokálne minimá, ktoré použijeme na segmentáciu histogramu na subhistogramy. Predpokladáme, že použitím vyhladzovacieho filtra sme zabránili detekcii nesignifikantných miním.

Nech $k_1, k_2, \ldots, k_l$ sú intenzity, v ktorých sa dosahujú lokálne minimá histogramu $H(k)$ a označme

$$k_0 \coloneqq M_{min}, \qquad k_{l+1} \coloneqq M_{max}. \tag{27}$$

Dynamický rozsah D rozdelíme na nasledovné intervaly:

$$R_i = [k_i, k_{i+1}), \qquad i = 0, 1, \ldots, l \tag{28}$$

a histogram pôvodného obrázka rozdelíme na subhistogramy intenzít z intervalov (28):

$$
\begin{aligned}
h_0(k) &\coloneqq h_{M_{min},k_1}(k) = h_{k_0,k_1}(k),\\
h_1(k) &\coloneqq h_{k_1,k_2}(k),\\
h_2(k) &\coloneqq h_{k_2,k_3}(k),\\
&\dots\\
h_{l-1}(k) &\coloneqq h_{k_{l-1},k_l}(k),\\
h_l(k) &\coloneqq h_{k_l,M_{max}}(k) = h_{k_l,k_{l+1}}(k).
\end{aligned}
\tag{29}
$$

Ak je medzi dvomi susednými minimami veľký rozsah intenzít, príslušný subhistogram by mohol zabrať príliš širokú časť škály v novom histograme, čím by nastal neželaný efekt ako pri GEH. Preto otestujeme, aká veľká variancia v porovnaní s normálnym rozdelením sa dosahuje na každom subhistograme [17].

Označme celkový počet pixlov v subhistograme h_i ako N_i, výberovú strednú hodnotu ako μ_i [12]:

$$
\mu_i = \frac{\sum_{k=k_i}^{k_{i+1}-1} k h_i(k)}{N_i} \qquad i = 0, 1, \dots, l, \tag{30}
$$

$$
N_i = \sum_{k=k_i}^{k_{i+1}-1} h_i(k) \qquad i = 0, 1, \dots, l, \tag{31}
$$

a výberovú varianciu ako σ_i^2 [12]:

$$
\sigma_i^2 = \frac{\sum_{k=k_i}^{k_{i+1}-1} (k-\mu_i)^2 h_i(k)}{N_i} \qquad i = 0, 1, \dots, l \tag{32}
$$

V histograme normálne rozdelených intenzít zaberá plocha medzi $\mu_i + \sigma_i$ a $\mu_i - \sigma_i$ približne 68.3% celkovej plochy pod histogramom. Ak je podiel plochy vymedzenej $\mu_i + \sigma_i$, $\mu_i - \sigma_i$ v subhistograme menší ako 68.3%, rozdelíme ho na tri subhistogramy oddelené v $\mu_i + \sigma_i$ a $\mu_i - \sigma_i$ (viď Obr. 11). Kontrolu rozsahu opakujeme, kým nie je podmienka 68.3% splnená pre všetky, aj novo vzniknuté subhistogramy [17].

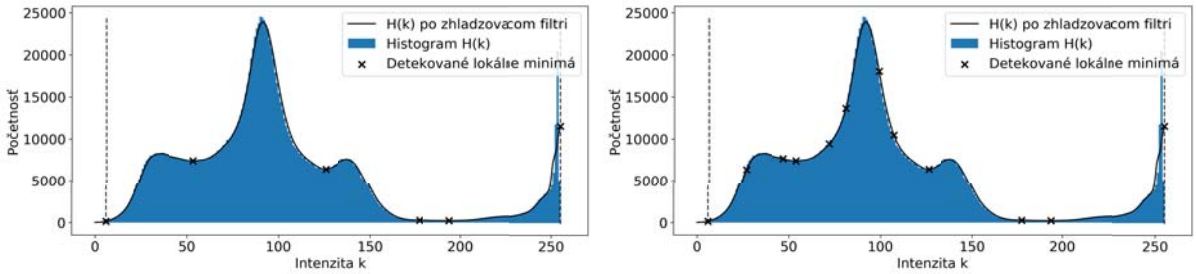
Napokon získavame segmentáciu celkového dynamického rozsahu D na intervaly R_i vrátane novo získaných lokálnych miním (l označuje v ďalšom už tento nový počet):

$$
R_i = [k_i, k_{i+1}), \qquad i = 0, 1, \dots, l \tag{33}
$$

a segmentáciu pôvodného histogramu na subhistogramy h_i:

$$h_i(k) := h_{k_i,k_{i+1}}(k), \qquad i = 0, 1, \ldots, l \tag{34}$$

Príklad histogramu s detekovanými lokálnymi minimami a finálnu segmentáciu po dodatočnom rozdelení úsekov s priveľkou varianciou uvádzame na Obr.10.



Obr. 10: Lokálne minimá histogramu obrázku z Obr. 9 (vľavo) a dodatočná segmentácia po kontrole variancie (vpravo)

2. Alokácia novej škály

Následne určíme škálu intenzít, ktorú budú zaberať jednotlivé subhistogramy v novom histograme. Nech D_i je dynamický rozsah subhistogramu h_i:

$$D_i = k_{i+1} - k_i \qquad i = 0, 1, \ldots, l, \tag{35}$$

$$\sum_{i=0}^{l} D_i = M_{max} - M_{min}, \tag{36}$$

kde k_i, k_{i+} sú lokálne minimá z kroku 1.

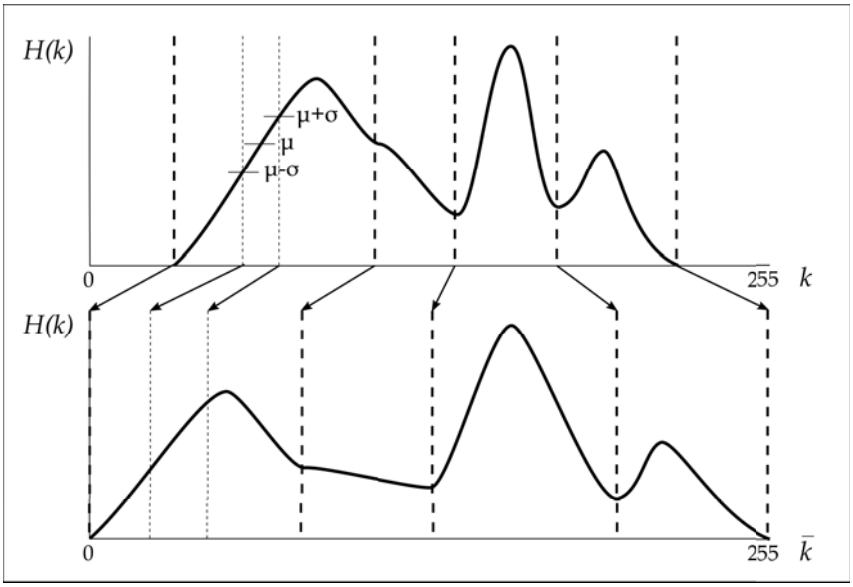
Každému subhistogramu h_i s dynamickým rozsahom D_i priradíme nový dynamický rozsah $\bar{D}_i$ tak, aby

$$\sum_{i=0}^{l} \bar{D}_i = 256,$$

pričom ich poradie na škále intenzít $[0, 255]$ ostáva zachované (viď Obr.11).

Veľkosť $\bar{D}_i$ z (35) vypočítame ako násobok pôvodného rozsahu D_i úmerný podielu plochy pod h_i s celkovej plochy pod $H(k)$. Nech F_i je plocha pod $h_i(k)$, t.j. počet pixlov s intenzitou v intervale $[k_i, k_{i+1})$:

$$F_i = \sum_{k=k_i}^{k_{i+1}-1} h_i(k), \qquad i = 0, 1, \ldots, l. \tag{37}$$



Obr. 11: Myšlienka DEH. Pôvodný histogram $H(k)$ (hore) je rozdelený na subhistogramy, ktoré sú rozšírené na plnú škálu v rámci povoleného rozsahu. Subhistogram h_1 bolo potrebné rozdeliť kvôli príliš veľkej variancii intenzít.

Označme $factor_i$ podiel z novej škály $\bar{D} = 256$ alokovaný pre subhistogram h_i. Tento podiel a nový dynamický rozsah $\bar{D}_i$ vypočítame nasledovne [17]:

$$factor_i = D_i(\ln F_i)^x \qquad x \in [0, 1, \ldots, 5], \qquad i = 0, 1, \ldots, l, \tag{38}$$

$$\bar{D}_i = \frac{factor_i}{\sum_{j=0}^{l+1} factor_j} 256 \qquad i = 0, 1, \ldots, l. \tag{39}$$

Napokon, hranice intervalov, v ktorých ležia nové histogramy, dostaneme ako:

$$\begin{aligned} \bar{k}_0 &= 0 \\ \bar{k}_1 &= \bar{k}_0 + \bar{D}_0 \\ &\ldots \\ \bar{k}_i &= \bar{k}_{i-1} + \bar{D}_{i-1} \\ &\ldots \\ \bar{k}_{l+1} &= \bar{k}_l + \bar{D}_l = 256, \end{aligned} \tag{40}$$

a intervaly, na ktoré treba po zaokrúhlení $\bar{k}_i$ rozdeliť škálu $[0, 255]$, sú:

$$\bar{R}_i := [\bar{k}_i, \bar{k}_{i+1}), \qquad i = 0, 1, \ldots, l. \tag{41}$$

Parameter x budeme nazývať koeficient zvýraznenia a kontrolujeme ním, aký silný efekt chceme dosiahnuť. Pre $x = 0$ do výpočtu (38) plocha pod príslušným subhistogramom nevstupuje, takže celkový dynamický rozsah v novom obrázku bude rozdelený medzi subhistogramy v rovnakom pomere ako v pôvodnom obrázku. Toto jednoduché tvrdenie zhrnieme do vety.

Veta 2.1. *Ak $x = 0$, potom pre pôvodné dynamické rozsahy D_i s celkovým súčtom D a nové dynamické rozsahy $\bar{D}_i$ s celkovým súčtom $\bar{D}_i$ platí*

$$\frac{D_i}{D} = \frac{\bar{D}_i}{\bar{D}}, \qquad i = 0, 1, \ldots, l, \qquad ak\ x = 0. \tag{42}$$

Dôkaz: *Pre $x = 0$ platí v (38)*

$$factor_i = D_i \qquad i = 0, 1, \ldots, l.$$

Potom nové dynamické rozsahy D_i sú:

$$\bar{D}_i = \frac{D_i}{\sum_{j=0}^{l+1} D_j} 256 = \frac{D_i}{D} 256 = \frac{D_i}{D} \bar{D}, \tag{43}$$

z čoho dostávame (42).

Čím väčší zvolíme koeficient x, tým viac zaváži veľkosť plochy pod subhistogramom. Pre obrázky, kde je pôvodný dynamický rozsah D oveľa menší ako 256, postačí zvoliť $x = 0$. Ak $D \doteq 256$, potom z vety 2.1 vyplýva, že DEH by nemala z hľadiska redistribúcie dynamických rozsahov takmer žiadny efekt. V takom prípade je vhodné zvoliť väčšie x. Parameter x je jediný, ktorý treba pri DEH určiť.

3. Ekvalizácia na subhistogramoch

V poslednom kroku vykonáme ekvalizáciu na každom subhistograme osobitne podobne ako v Definícii 2.2, ale získané hodnoty s_k z (20) potrebujeme previesť na škálu určenú v predošlom kroku vzťahom (41) [17]. Je teda potrebné namapovať intenzity zo škály $R_i = [k_i, k_{i+1})$ na $\bar{R}_i = [\bar{k}_i, \bar{k}_{i+1})$, teda D_i intenzít na nový počet $\bar{D}_i$ intenzít. V [17] nešpecifikujú, ako presne sa má táto transformácia zrealizovať. Navrhujeme dva spôsoby, prvý je analógiou Definície 2.2 z [17] a druhý je náš návrh na úpravu. Ukážku realizácie oboch prístupov uvádzame v Príklade 2.1.

3a. Jednoduchá ekvalizácia

Pri dynamickej ekvalizácii môžeme podobne ako v Definícii 2.2 vypočítať podiel aproximácie integrálu kumulatívnej distribučnej funkcie intenzít postupne pre všetky $k \in R_i$. Tieto čísla $s_k \in (0, 1]$ môžeme previesť na škálu $[U, L)$ ako $(U - L - 1)s_k + L$ a zaokrúhliť na celé čísla, čím získame intenzity z intervalu $\bar{R}_i$. Keďže spravidla $D_i \neq \bar{D}_i$ a výsledné intenzity je nutné zaokrúhliť, viaceré intenzity budú napamované na rovnakú výslednú intenzitu, ak $D_i > \bar{D}_i$ a niektoré intenzity budú v novom rozsahu $\bar{R}_i$ vynechané, ak $D_i < \bar{D}_i$. Dostávame nasledovný algoritmus:

Algoritmus 2.1 (Jednoduchá dynamická ekvalizácia histogramu)**.**
Nech $H(k)$ je pôvodný histogram obrázku segmentovaný na škály intenzít R_i s histogramom h_i, $i = 0, 1, \ldots, l$, získané v prvom kroku. Nech $\bar{R}_i = [\bar{k}_i, \bar{k}_{i+1})$ je nová škála šírky $\bar{D}_i$ alokovaná pre R_i.
Dynamickú ekvalizáciu aplikujeme osobitne pre každý subhistogram h_i vypočítaním:

$$s_K^i = \frac{\sum_{j=1}^{K} h_i(k_j^i)}{N_i}, \qquad K = 1, 2, \ldots D_i, \tag{44}$$

$$N_i = \sum_{j=1}^{D_i} h_i(k_j^i), \tag{45}$$

a pôvodné intenzity mapujeme na nové ako:

$$\begin{aligned} k_K^i &\longrightarrow (\bar{D}_i - 1)s_K^i + \bar{k}_i{}^*, \qquad \textit{t.j. vzťahom:} \\ \bar{I}(x,y)\Big|_{[x,y]:I(x,y)=k_K^i} &= (\bar{D}_i - 1)s_K^i + \bar{k}_i{}^*, \qquad K = 1, 2, \ldots \bar{D}_i, \qquad [x, y] \in \mathcal{D} \end{aligned} \tag{46}$$

** zaokrúhlené na celé čísla*

Kedže prvá zložka s_K^k je ostro väčšia ako nula, prvých niekoľko hodnôt $\bar{R}_i$ bude často vynechaných, najmä ak je $\bar{D}_i \gg D_i$, ako môžeme vidieť v Príklade 2.1. Preto navrhujeme aj nasledovnú alternatívu.

3b. Ekvalizácia s interpoláciou

V snahe využiť celú šírku výstupného rozsahu $\bar{R}_i$ budeme vyžadovať priradenie krajných intenzít

$$\begin{aligned} k_i &\longrightarrow \bar{k}_i, \qquad i = 0, 1, \ldots, l \\ k_{i+1} - 1 &\longrightarrow \bar{k}_{i+1} - 1, \qquad i = 0, 1, \ldots, l \end{aligned}$$

a zostávajúce intenzity vnútri intervalu R_i rozdelíme ďalej uvedeným spôsobom.

Navrhujeme rozdeliť každý interval R_i rovnomerne na $\bar{D}_i$ hodnôt a početnosti pixlov s intenzitami, ktoré nie sú celé čísla, dopočítať ako konvexnú kombináciu dvoch najbližších známych hodnôt (viď Obr.12). Nech $K_j^i \in \mathbb{R}, j = 0, 1, \ldots \bar{D}_i, \bar{D}_i + 1$ je rovnomerné delenie intervalu R_i také, že

$$\begin{aligned} k_0^i &= k_i \\ k_{\bar{D}_i+1}^i &= k_{i+1} - 1. \end{aligned} \tag{47}$$

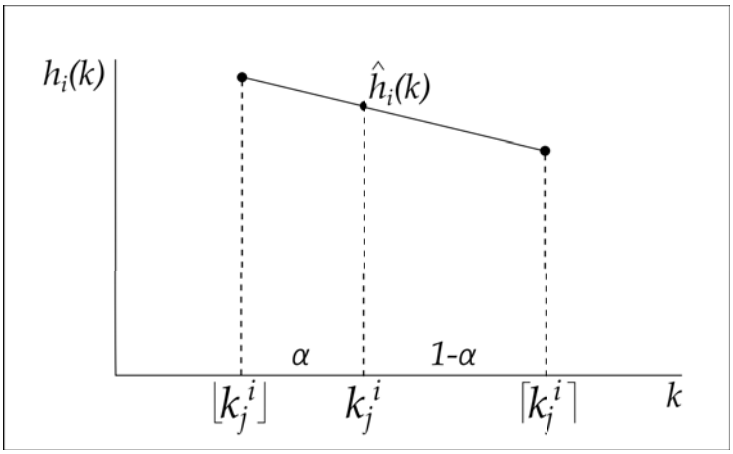
Aproximáciu $h_i(k_j^i)$ navrhujeme vypočítať nasledovne:

$$\alpha_j = k_j^i - \lfloor k_j^i \rfloor, \qquad j = 1, 2, \ldots, \bar{D}_i \tag{48}$$

$$\hat{h}_i(k_j^i) = \alpha_j h_i(\lfloor k_j^i \rfloor) + (1 - \alpha_j) h_i(\lceil k_j^i \rceil), \qquad j = 1, 2, \ldots, \bar{D}_i \tag{49}$$

$$\hat{h}_i(k_0^i) \coloneqq h_i(k_i) \tag{50}$$

kde $\lfloor k_j^i \rfloor$, resp. $\lceil k_j^i \rceil$ je dolná, resp. horná celá časť k_j^i, ako ilustrujeme na Obr.12.



Obr. 12: Ilustrácia aproximácie počtu pixlov s intenzitou $k_j^i \in \mathbb{R}$ ako konvexnej kombinácie známych hodnôt v $\lfloor k_j^i \rfloor, \lceil k_j^i \rceil \in N$

Získali sme subhistogram šírky $\bar{D}_i$ intenzít v pôvodnom intervale R_i, pomocou ktorého môžeme vypočítať hodnoty s_j^i ako:

$$s_j^i = \frac{\sum_{t=0}^{j} \hat{h}_i(k_t^i)}{N_i}, \qquad j = 1, 2, \ldots \bar{D}_i, \tag{51}$$

$$N_i = \sum_{t=0}^{\bar{D}_i} \hat{h}_i(k_t^i) + h_i(k_{\bar{D}_i+1}^i), \tag{52}$$

kde v (51) počítame odhad distribučnej funkcie v bodoch $k_1^i, k_3^i, \ldots, k_{\bar{D}_i}^i$, v ktorých sme interpolovali, a v (52) počítame normalizáciu, t.j. celkový počet pixlov v novom subhistograme $\hat{h}_i$ vrátane ľavého kraja $h_i(k_0^i)$ a pravého kraja $h_i(k_{\bar{D}_i+1}^i)$. Hodnoty s_j^i v hraničných bodoch sú:

$$s_0^i := 0, \tag{53}$$

$$s_{\bar{D}_i+1}^i := 1, \tag{54}$$

teda máme spolu $\bar{D}_i + 2$ hodnôt. Zostáva namapovať intenzity zo škály $R_i = [k_i, k_{i+1})$ na $\bar{R}_i = [\bar{k}_i, \bar{k}_{i+1})$.

Hodnoty s_j^i prevedieme na pôvodnú škálu R_i ako:

$$B_j^i := s_j^i(D_i - 1) + k_i, \qquad j = 0, 1, \ldots, \bar{D}_i + 1, \tag{55}$$

čím dostávame nové delenie škály R_i na $\bar{D}_i + 2$ bodov.

Pre každú intenzitu k_K^i z pôvodného delenia pôvodného intervalu R_i zistíme, ku ktorému v poradí z bodov B_j^i z (55) je najbližšie, t.j. hľadáme index j. Napokon jej priradíme $j - tu$ intenzitu z intervalu $\bar{R}_i$:

$$k_K^i \longrightarrow \bar{k}_i + \operatorname*{argmin}_j \left| k_K^i - B_j^i \right|. \tag{56}$$

Tento postup zhrnieme do nasledovného algoritmu:

Algoritmus 2.2 (Dynamická ekvalizácia histogramu s interpoláciou)**.**
Nech $H(k)$ je pôvodný histogram obrázku segmentovaný na škály intenzít R_i s histogramom h_i, $i = 0, 1, \ldots, l$, získané v prvom kroku. Nech $\bar{R}_i = [\bar{k}_i, \bar{k}_{i+1})$ je nová škála šírky $\bar{D}_i$ alokovaná pre R_i, ktoré sme získali v druhom kroku.
Najprv urobíme rovnomerné delenie intervalu R_i v bodoch $k_0^i, k_1^i, \ldots, k_{\bar{D}_i+1}^i$, v ktorých lineárne interpolujeme histogram h_i:

$$\alpha_j = k_j^i - \lfloor k_j^i \rfloor, \qquad j = 1, 2, \ldots, \bar{D}_i \tag{57}$$

$$\hat{h}_i(k_j^i) = \alpha_j h_i(\lfloor k_j^i \rfloor) + (1 - \alpha_j) h_i(\lceil k_j^i \rceil), \qquad j = 1, 2, \ldots, \bar{D}_i \tag{58}$$

$$\hat{h}_i(k_0^i) := h_i(k_i) \tag{59}$$

Dynamickú ekvalizáciu aplikujeme osobitne pre každý subhistogram h_i výpočtom:

$$s_0^i := 0, \tag{60}$$

$$s_{\bar{D}_i+1}^i := 1, \tag{61}$$

$$s_j^i = \frac{\sum_{t=0}^{j} \hat{h}_i(k_t^i)}{N_i}, \qquad j = 1, 2, \ldots \bar{D}_i, \tag{62}$$

$$N_i = \sum_{t=0}^{\bar{D}_i} \hat{h}_i(k_t^i) + h_i(k_{\bar{D}_i+1}^i), \tag{63}$$

pomocou ktorých rozdelíme interval R_i v bodoch:

$$B_j^i := s_j^i(D_i - 1) + k_i, \qquad j = 0, 1, \ldots, \bar{D}_i + 1 \tag{64}$$

a pôvodné intenzity mapujeme na nové ako:

$$k_K^i \longrightarrow \bar{k}_i + \operatorname*{argmin}_j \left| k_K^i - B_j^i \right| \qquad \textit{t.j. vzťahom:} \tag{65}$$

$$\bar{I}(x,y)\Big|_{[x,y]:I(x,y)=k_K^i} = \bar{k}_i + \operatorname*{argmin}_j \left| k_K^i - B_j^i \right| \qquad K = 1, 2, \ldots D_i, \qquad [x,y] \in \mathcal{D}, \tag{66}$$

v prípade rovnosti na menšie j.

Použitie jednoduchej DEH aj DEH s interpoláciou z tretieho kroku na jeden subhistogram uvádzame v nasledovnom príklade.

Príklad 2.1 (Ukážka DEH na jednom subhistograme)**.**
Nech druhý subhistogram zľava (t.j. h_1) je na pôvodnom rozsahu R_1:

$$h_1 = [540, 550, 320, 313, 295],$$
$$R_1 = [10, 15) = [10, 11, 12, 13, 14],$$

t.j. $k_1 = 10, k_2 = 15$ a $D_1 = k_2 - k_1 = 5$.
Nech bol tomuto subhistogramu alokovaný nový dynamický rozsah $\bar{R}_1$:

$$\bar{R}_1 = [3, 14) = [3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13],$$

t.j. $\bar{k}_1 = 3, \bar{k}_2 = 14$ a $\bar{D}_1 = \bar{k}_2 - \bar{k}_1 = 9$.

a) Jednoduchá DEH

Celkový počet pixlov s intenzitami na tomto histograme je $540 + 550 + \ldots + 295 = 2018$. Hodnoty s_K^i vypočítame podľa (44) ako

$$s_1^1 = \frac{540}{2018}, s_2^1 = \frac{550}{2018}, \ldots, s_5^1 = \frac{295}{2018}$$

a po preškálovaní na

$$\bar{k}_K^1 = (\bar{D}_1 - 1)s_K^1 + \bar{k}_1 = 8s_K^1 + 3, \qquad K = 1, 2, \ldots, 5$$

z (46) a po zaokrúhlení dostávame nové priradenie $k_K^1 \longrightarrow \bar{k}_K^1$ (v tabuľke je s_K^1 zaokrúhlené na tri desatinné miesta):

K	1	2	3	4	5
k_K^1	10	11	12	13	14
s_K^1	0.268	0.540	0.699	0.854	1
$\bar{k}_K^1$	6	8	10	12	13

Vidíme, že prvá intenzita z intervalu $[3, 14)$, na ktorú sa namapovala nejaká pôvodná intenzita, je až šesť.

b) DEH s interpoláciou

V druhom prístupe vypočítame rovnomerné delenie intervalu R_1 s $\bar{D}_1 = 9$ novými bodmi, v ktorých aproximujeme hodnoty $\hat{h}_1$ podľa (57) - (59):

j	1	2	3	4	5	6	7	8	9
k_j^1	10.4	10.8	11.2	11.6	12.	12.4	12.8	13.2	13.6
$\hat{h}_1(k_j^1)$	544	548	504	412	320	317.2	314.4	309.4	302.2
α_j	0.4	0.4	0.8	0.2	0.6	0	0.4	0.8	0.6

Z týchto hodnôt a $\hat{h}_0^1 = 540, h_{10}^1 = 295$ vypočítame podľa (60)-(63) hodnoty s_j^i a podľa (64) nové delenie intervalu R_i (v tabuľke zaokrúhlené na tri desatinné miesta):

j	0	1	2	3	4	5	6	7	8	9	10
s_j^1	0	0.246	0.37	0.485	0.578	0.651	0.723	0.794	0.864	0.933	1
B_j^1	10	10.984	11.482	11.939	12.313	12.604	12.892	13.177	13.458	13.732	14

a podľa (65) nájdeme pre každé k_K^1 najbližšie B_j^1 (druhý riadok), resp. jeho index j (tretí riadok) a dostávame priradenie pre pôvodné intenzity $k_K^1 \longrightarrow \bar{k}_K^1$ (štvrtý riadok):

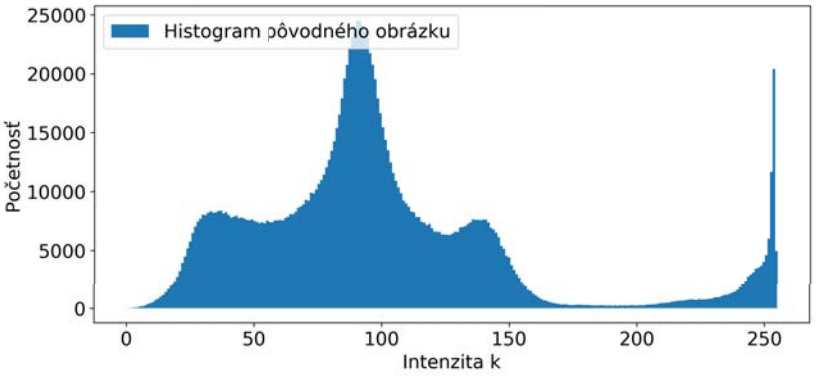
K	1	2	3	4	5
k_K^1	10	11	12	13	14
$\min_j \left\lvert k_K^1 - B_j^1 \right\rvert$	10	10.984	11.939	12.892	14
$\operatorname{argmin}_j \left\lvert k_K^1 - B_j^1 \right\rvert$	0	1	3	6	10
$\bar{k}_K^1$	3	4	6	9	13

Vidíme, že v tomto prípade je využitá celá šírka novej škály $\bar{R}_i$.

Kým pri jednoduchej DEH je tendencia koncentrovať intenzity viac k pravému kraju intervalu R_i, pri DEH s interpoláciou sú nové intenzity koncentrované tam, kde bolo pôvodne veľa plochy a presun intenzít nie je až tak výrazný (tvar histogramu sa viac podobá na pôvodný histogram), ako môžeme vidieť na Obr.13.

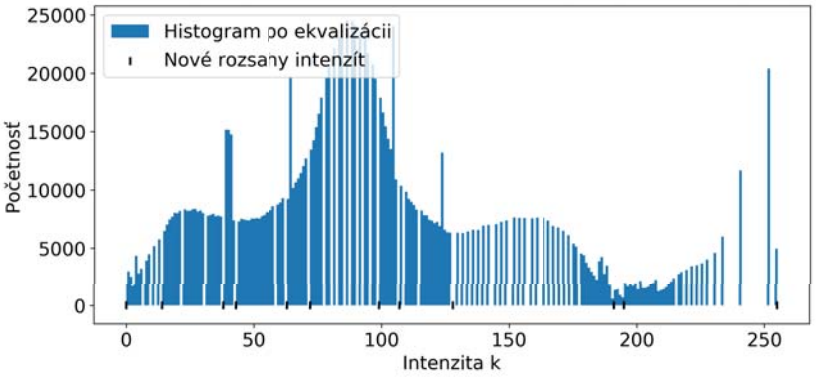
Obidve verzie algoritmu DEH sme naprogramovali v Python 3 a príslušný zdrojový kód je k dispozícii v prílohe A. Výsledky môžeme vidieť na Obr.13, kde sme aplikovali rôzne spôsoby DEH na pôvodný obrázok *(a)* s histogramom *(b)*. Metódou jednoduchej DEH sme dostali obrázok *(c)* s histogramom *(d)*, metódou DEH s interpoláciou obrázok *(e)* s histogramom *(f)* a metódou GEH obrázok *(g)* s histogramom *(h)*.

(a) Originál



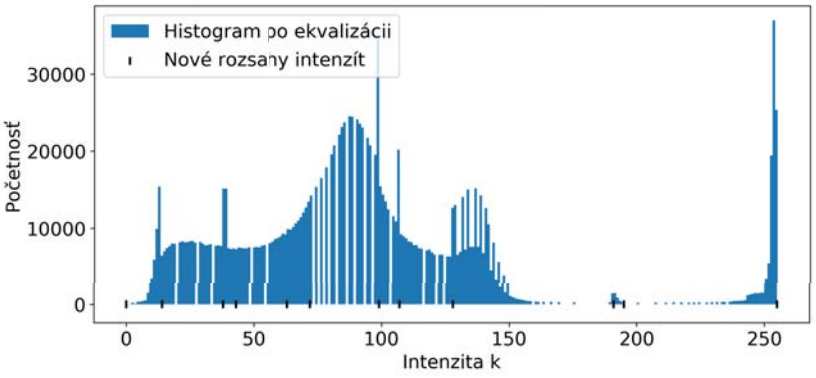
(b)

(c) Jednoduchá DEH, $x = 4$



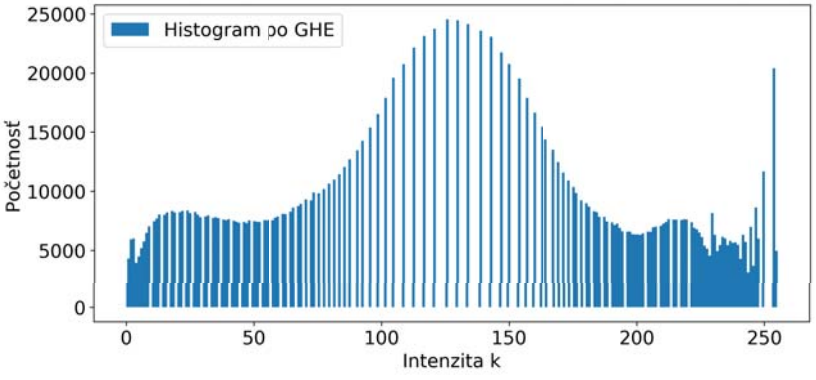
(d) $x = 4$

(e) DEH s interpol., $x = 4$



(f) $x = 4$

(g) GEH



(h)

Obr. 13: Rôzne spôsoby ekvalizácie histogramu

2.2 Vyhladenie bilaterálnym filtrom intenzity

Po ekvalizácii histogramu môžeme prejsť k simplifikácii obrázka. Nedostatok Gaussovského filtra v podobe rozmazaných hrán rieši bilaterálny filter, do ktorého okrem váhy reprezentujúcej vzdialenosť dvoch bodov vstupuje aj váha podobnosti ich intenzít [16].

Bilaterálny filter je rozšírením priestorového filtra o druhú váhovaciu funkciu $f_c(I(\xi), I(u))$, ktorá váhuje podobnosť intenzít v bodoch ξ a u. Čím väčší je rozdiel intenzít $|I(\xi) - I(u)|$, tým menšiu váhu dostane intenzita v bode ξ. Cieľom tohto filtra je zahrnúť do váženého priemeru intenzít okolitých pixlov iba body, ktoré majú podobnú intenzitu ako centrálny bod u, čím zabránime efektu rozmazania hrán ako pri Gaussovskom filtri na Obr. 3c. Bilaterálny filter je definovaný nasledovne:

Definícia 2.4 (Bilaterálny filter intenzity čiernej [16])**.**
Výsledkom aplikácie priestorového filtra v danom bode $u = [x_0, y_0]$ pomocou intenzít v okolitých bodoch $\xi = [x, y]$ je nová intenzita $W(u)$ daná nasledovne:

$$W(u) = z^{-1}(u) \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} I(\xi) f_d(\xi, u) f_c(I(\xi), I(u)) d\xi \tag{67}$$

kde $f_d(\xi, u)$ a $f_c(I(\xi), I(u))$ sú vhodne zvolené váhovacie funkcie reprezentujúce vzdialenosť u od ξ (d - distance), resp. rozdiel intenzít $I(\xi)$, $I(u)$ (c - colour) a $z(u)$ je normalizačná konštanta

$$z(u) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f_d(\xi, u) f_c(I(\xi), I(u)) d\xi. \tag{68}$$

Častou voľbou funkcií $f_d(\xi, u)$ a $f_c(I(\xi), I(u))$ je tak ako pri priestorovom filtri Gaussovská funkcia vzdialenosti bodov, resp. ich intenzít [16]:

$$f_d(\xi, u) := e^{-\frac{1}{2}\left(\frac{d(\xi,u)}{\sigma_d}\right)^2}, \tag{69}$$

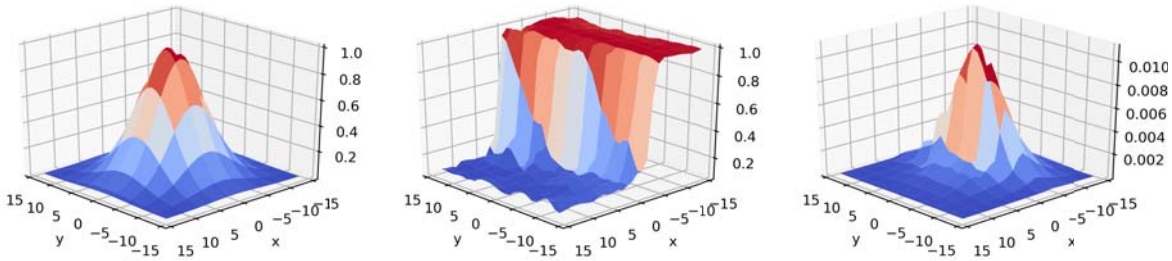
$$f_c(\xi, u) := e^{-\frac{1}{2}\left(\frac{d(I(\xi),I(u))}{\sigma_c}\right)^2}, \tag{70}$$

kde $d(a, b)$ je euklidovská vzdialenosť jej vstupov:

$$d(a, b) := \|a - b\|_2 . \tag{71}$$

Keďže vo filtri vystupuje súčin funkcií $f_d(\xi, u)$ a $f_c(I(\xi), I(u))$, máme zabezpečené, že intenzita $I(\xi)$ dostane veľkú váhu, len ak je bod ξ blízko bodu u a zároveň je intenzita $I(\xi)$ podobná ako $I(u)$. Príklad súčinu $f_d(\xi, u)$ a $f_c(I(\xi), I(u))$ ilustrujeme na Obr. 14 pre

bod tesne napravo od hrany. Vidíme, že body na opačnej strane hrany dostali malé váhy, pretože $d(I(\xi), I(u))$ je veľké, a do priemeru prakticky nevstupujú.



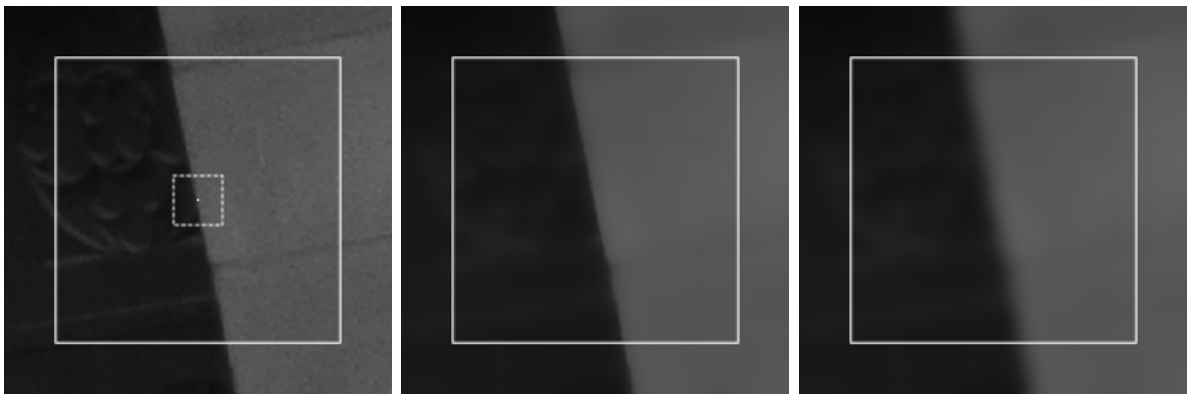
Obr. 14: Váhy $f_d(\xi, u)$ (vľavo), $f_c(I(\xi), I(u))$ (v strede), ich súčin (vpravo) pre bod u v strede Obr. 15a pri $\sigma_d = 5, \sigma_c = 30$ a $s = 29$.

Prakticky sa takýto bilaterálny filter aplikuje analogicky ako Gaussovský priestorový filter, ktorý sme popísali v časti 1.3.2, s rozdielom, že váhu w_i v (12) počítame ako:

$$w_i = w_i(d(\xi_i, u), d(I(\xi_i), I(u))) = f_d(\overline{\xi}_i, u_0) f_c(I(\overline{\xi}_i), I(u_0)), \qquad i = 1, 2, \ldots, s^2, \qquad (72)$$

kde ξ_i prislúcha bod $\overline{\xi}_i$, pre ktorý $d(\overline{\xi}_i, u_0) = d(\xi_i, u)$.

Ukážku použitia filtra z Obr. 14 uvádzame na Obr. 15. Po filtrácii (v strede) ostáva hrana ostrá na rozdiel od efektu Gaussovského filtra (vpravo).



(a) *(b)* $\sigma_d = 5, \sigma_c = 30, s = 29$ *(c)* $\sigma_d = 5, s = 29$

Obr. 15: Na obrázku vľavo sú všetky body ξ, ktoré vstupujú do filtra, výsek s bodmi u (plná čiara) a 29×29 okolie centrálneho pixlu pri hrane (prerušovaná čiara); obrázok po použití bilaterálneho filtra (v strede) a Gaussovského priestorového filtra (vpravo).

Voľbou σ_d (d - *distance*), resp. σ_c (c - *colour*) kontrolujeme váhovanie vzdialenosti od centrálneho bodu u, resp. toleranciu nepodobnosti intenzít. Pre veľké σ_c sú všetky

hodnoty $f_c(I(\xi), I(u))$ približne rovnaké a táto zložka filtra stráca efekt, teda ide o jednoduchý priestorový filter. Voľbe parametrov σ_d, σ_c a veľkosti okolia s sa budeme venovať v kapitole 3.

2.3 Diskrétne operátory pre aproximáciu gradientu

Po predpríprave obrázka ekvalizáciou histogramu a bilaterálnym filtrom môžeme prejsť k samotnej detekcii hrán aproximáciou gradientu funkcie intenzít $I(x, y), [x, y] \in \mathcal{D}$.

Dvomi jednoduchými schémami aproximácie gradientu sú dopredná diferencia

$$\begin{aligned} &I(x+1) - I(x), \\ &I(y+1) - I(y), \end{aligned} \tag{73}$$

a centrálna diferencia

$$\begin{aligned} &\frac{1}{2}[I(x+1) - I(x-1)], \\ &\frac{1}{2}[I(y+1) - I(y-1)], \end{aligned} \tag{74}$$

kde škálový faktor $\frac{1}{2}$ je spôsobený vzdialenosťou 2 pixle [9]. Škálový faktor sa v definíciách operátorov aj pri ich aplikácii často vynecháva, keďže výsledný obrázok môžeme ľahko preškálovať. Uvedené schémy zodpovedajú nasledovným 2D konvolučným maticiam [9]:

$$\text{Dopredná diferencia:} \qquad h_1(x,y) = \begin{bmatrix} 0 & 0 \\ \mathbf{-1} & 1 \end{bmatrix}, h_2(x,y) = \begin{bmatrix} 1 & 0 \\ \mathbf{-1} & 0 \end{bmatrix},$$

$$\text{Centrálna diferencia:} \qquad h_1(x,y) = \begin{bmatrix} 0 & 0 & 0 \\ -1 & \mathbf{0} & 1 \\ 0 & 0 & 0 \end{bmatrix}, h_1(x,y) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & \mathbf{0} & 0 \\ 0 & -1 & 0 \end{bmatrix},$$

kde pozícia centra $[x, y]$ je vyznačená tučným. Tieto filtre slúžia ako základ pre odvodenie množstva detektorov, ktoré sa líšia konkrétnou voľbou koeficientov v konvolučných maticiach. Ďalej uvádzame niekoľko najznámejších.

Podobne ako v predošlom, pre body na okrajoch obrázka neexistuje celé okolie, ktoré vstupuje do súčinu s konvolučným operátorom. Chýbajúce intenzity opäť dodefinujeme pomocou reflektujúcej hranice, ako sme popísali v časti 1.3.2.

Prirodzenou požiadavkou je vyžadovať nulovú odozvu pri aplikácii na oblasť s konštantnou intenzitou. Preto musí byť súčet koeficientov v konvolučných maticiach rovný nule [9].

2.3.1 Detektory odvodené od doprednej diferencie

Symbolickou rotáciou filtrov pre doprednú diferenciu o 45° môžeme odvodiť Robertov detektor diagonálnych hrán.

Definícia 2.5 (Robertov detektor [9])**.**
Nová intenzita $\hat{g}(x, y)$ pre bod $[x, y]$ pri detekcii hrán Robertovym detektorom (z angl. Robert's Cross operator) je daná nasledovne:

$$G_x = I(x, y+1) - I(x+1, y), \tag{75}$$

$$G_y = -I(x, y) + I(x+1, y+1), \tag{76}$$

$$\hat{g}(x, y) = \|(G_x, G_y)\|, \tag{77}$$

kde $\|.\|$ je vhodne zvolená norma. Zodpovedajúce konvolučné matice sú:

$$R_x = \begin{bmatrix} 1 & 0 \\ \mathbf{0} & -1 \end{bmatrix} \quad R_y = \begin{bmatrix} 0 & 1 \\ -\mathbf{1} & 0 \end{bmatrix}, \tag{78}$$

kde matica R_x zodpovedá výpočtu G_x z (75) a R_y zodpovedá výpočtu G_y z (76).

Robertov detektor je kvôli malej konvolučnej matici citlivý na šum. Keďže uvažujeme iba intenzity v bezprostrednom okolí, ľahko sa môže stať, že v bode $[x, y]$ bude detekovaná hrana, aj keď v skutočnosti ide iba o šum.

2.3.2 Detektory odvodené od symetrickej diferencie

Hoci sa počíta s použitím Gaussovského alebo iného vyhladzovacieho filtra, v snahe zmenšiť citlivosť na šum sa do konvolučných matíc zahŕňa aj jednoduchý vyhladzovací filter v ortogonálnom smere. Takýmto filtrom môže byť napríklad priemer intenzít v troch susedných pixloch. Uvažujme impulznú odozvu [15] jednorozmerného vyhladzovacieho filtra $h_s(x)$ a jednorozmernú symetrickú diferenciu $h_c(y)$ [9]:

$$h_s(x) = \begin{bmatrix} 1 & \mathbf{1} & 1 \end{bmatrix} \quad h_c(y) = \begin{bmatrix} -1 & \mathbf{0} & 1 \end{bmatrix}, \tag{79}$$

Súčinom $h_s(x)$ a $h_c(y)$ získame nasledovný 2D filter [9]:

$$\begin{bmatrix} 1 \\ \mathbf{1} \\ 1 \end{bmatrix} \begin{bmatrix} -1 & \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & \mathbf{0} & 1 \\ -1 & 0 & 1 \end{bmatrix} \tag{80}$$

Rovnakým spôsobom vznikne v ortogonálnom smere z $h_s(y)$ a $h_c(x)$ matica otočená o $90°$. Spolu s maticou z (80) definujú známy Prewittov detektor. Ďalej budeme detektory definovať iba pomocou konvolučných matíc, ktorých aplikáciu sme popísali v definícii Sobelovho detektora 1.4 a v definícii Robertovho detektora 2.5.

Definícia 2.6 (Prewittov detektor [9])**.**
Konvolučné matice Prewittovho detektora sú:

$$P_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & \mathbf{0} & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad P_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & \mathbf{0} & 0 \\ -1 & -1 & -1 \end{bmatrix} \tag{81}$$

Ďalšie filtre tohoto druhu vzniknú inou voľbou vyhladzovacej zložky $h_s(x)$. Napríklad Sobelovmu detektoru z Cannyho metódy, ktorý sme popísali v 1.3.3, zodpovedá

$$h_s(x) = \begin{bmatrix} 1 & \mathbf{2} & 1 \end{bmatrix}. \tag{82}$$

Sobelov detektor je voči Prewittowmu preferovaný práve kvôli tejto voľbe $h_s(x)$, keďže vedie k hladšiemu výsledku ako $h_s(x)$ z (79) [9].

2.3.3 Kirschov detektor

Tento detektor odhadne gradient v horizontálnom, vertikálnom aj diagonálnom smere a vyberie ten, v ktorom má gradient najväčšiu magnitúdu. Kvôli prehľadávaniu v ôsmich smeroch sa nazýva aj kompasový [6] a jeho konvolučné matice môžeme symbolicky značiť ako svetové strany.

Definícia 2.7 (Kirschov detektor 3×3 [6])**.**
Konvolučné matice Kirschovho 3×3 *detektora sú:*

$$K_N = \begin{bmatrix} 5 & 5 & 5 \\ -3 & \mathbf{0} & -3 \\ -3 & -3 & -3 \end{bmatrix} \quad K_{NW} = \begin{bmatrix} 5 & 5 & -3 \\ 5 & \mathbf{0} & -3 \\ -3 & -3 & -3 \end{bmatrix}$$

$$K_W = \begin{bmatrix} 5 & -3 & -3 \\ 5 & \mathbf{0} & -3 \\ 5 & -3 & -3 \end{bmatrix} \quad K_{SW} = \begin{bmatrix} -3 & -3 & -3 \\ 5 & \mathbf{0} & -3 \\ 5 & 5 & -3 \end{bmatrix}$$

$$K_S = \begin{bmatrix} -3 & -3 & -3 \\ -3 & \mathbf{0} & -3 \\ 5 & 5 & 5 \end{bmatrix} \quad K_{SE} = \begin{bmatrix} -3 & -3 & -3 \\ -3 & \mathbf{0} & 5 \\ -3 & 5 & 5 \end{bmatrix}$$

$$K_E = \begin{bmatrix} -3 & -3 & 5 \\ -3 & \mathbf{0} & 5 \\ -3 & -3 & 5 \end{bmatrix} \quad K_{NE} = \begin{bmatrix} -3 & 5 & 5 \\ -3 & \mathbf{0} & 5 \\ -3 & -3 & -3 \end{bmatrix}$$

a výsledná aproximácia $g(x, y)$ *je*

$$\hat{g}(x, y) = \max_i |K_i \circ [x, y]_{3\times 3}| \qquad i = N, NW, ..., NE, \tag{83}$$

kde $[x, y]_{3\times 3}$ *je* 3×3 *okolie bodu* $[x, y]$ *a* $\circ$ *je Hadamardov súčin.*

Hoci je Kirschov detektor výpočtovo náročnejší ako Sobelov a Prewittov, vďaka selekcii smeru s najväčšou odozvou sú hrany výraznejšie. Navyše, jeho aplikáciou priamo získavame informáciu o smere najsilnejšieho gradientu, ktorú využijeme neskôr pri stenčovaní hrán.

V [6] bol použitý Kirschov detektor 5×5 z dôvodu detekovania príliš jemných detailov jeho 3×3 verziou. Rozšírený Kirschov operátor uvádzame v definícii 2.8. V našom algoritme budeme uvažovať možnosť použiť aj túto verziu Kirschovho detektora a zvoľbe veľkosti konvolučnej matice sa budeme venovať v kapitole 3.

Definícia 2.8 (Kirschov 5×5 detektor [6])**.**
Konvolučné matice Kirschovho 5×5 *detektora sú:*

$$K_N = \begin{bmatrix} 9 & 9 & 9 & 9 & 9 \\ 9 & 5 & 5 & 5 & 9 \\ -7 & -3 & \mathbf{0} & -3 & 7 \\ -7 & -3 & -3 & -3 & -7 \\ -7 & -7 & -7 & -7 & -7 \end{bmatrix} \quad K_{NW} = \begin{bmatrix} 9 & 9 & 9 & 9 & -7 \\ 9 & 5 & 5 & -3 & -7 \\ 9 & 5 & \mathbf{0} & -3 & -7 \\ 9 & -3 & -3 & -3 & -7 \\ -7 & -7 & -7 & -7 & -7 \end{bmatrix}$$

$$K_W = \begin{bmatrix} 9 & 9 & -7 & -7 & -7 \\ 9 & 5 & -3 & -3 & -7 \\ 9 & 5 & \mathbf{0} & -3 & -7 \\ 9 & 5 & -3 & -3 & -7 \\ 9 & 9 & -7 & -7 & -7 \end{bmatrix} \quad K_{SW} = \begin{bmatrix} -7 & -7 & -7 & -7 & -7 \\ 9 & -3 & -3 & -3 & -7 \\ 9 & 5 & \mathbf{0} & -3 & -7 \\ 9 & 5 & 5 & -3 & -7 \\ 9 & 9 & 9 & 9 & -7 \end{bmatrix}$$

$$K_S = \begin{bmatrix} -7 & -7 & -7 & -7 & -7 \\ -7 & -3 & -3 & -3 & -7 \\ -7 & -3 & \mathbf{0} & -3 & -7 \\ 9 & 5 & 5 & 5 & 9 \\ 9 & 9 & 9 & 9 & 9 \end{bmatrix} \quad K_{SE} = \begin{bmatrix} -7 & -7 & -7 & -7 & -7 \\ -7 & -3 & -3 & -3 & 9 \\ -7 & -3 & \mathbf{0} & 5 & 9 \\ -7 & -3 & 5 & 5 & 9 \\ -7 & 9 & 9 & 9 & 9 \end{bmatrix}$$

$$K_E = \begin{bmatrix} -7 & -7 & -7 & 9 & 9 \\ -7 & -3 & -3 & 5 & 9 \\ -7 & -3 & \mathbf{0} & 5 & 9 \\ -7 & -3 & -3 & 5 & 9 \\ -7 & -7 & -7 & 9 & 9 \end{bmatrix} \quad K_{NE} = \begin{bmatrix} -7 & 9 & 9 & 9 & 9 \\ -7 & -3 & 5 & 5 & 9 \\ -7 & -3 & \mathbf{0} & 5 & 9 \\ -7 & -3 & -3 & -3 & 9 \\ -7 & -7 & -7 & -7 & -7 \end{bmatrix}$$

a výsledná aproximácia $g(x, y)$ je

$$\hat{g}(x, y) = \max_i |K_i \circ [x, y]_{5\times5}| \qquad i = N, NW, ..., NE, \tag{84}$$

kde $[x, y]_{5\times5}$ je 5×5 okolie bodu $[x, y]$ a $\circ$ je Hadamardov súčin.

2.4 Spojité metódy detekcie hrán pomocou diferenciálnych rovníc

V tejto časti popíšeme základy detekcie hrán spojitým prístupom pomocou diferenciálnych rovníc (PDE).

Označme $I_t(u), u = [x, y]$ intenzity obrázku v čase t. PDE pre vývoj intenzít obrázku

môžeme odvodiť pomocou Gaussovského filtra:

$$I_t = G_\sigma I_0, \tag{85}$$

kde G_σ je Gaussova funkcia s varianciou σ a $I_0 = I(x, y)$ je pôvodný obrázok [9]. Ak položíme

$$\sigma = \sqrt{t}, \tag{86}$$

potom efekt Gaussovského filtra môžeme dosiahnuť izotropnou (t.j. rozpínajúcou sa rovnako v oboch smeroch) difúznou rovnicou [9]:

$$\frac{\partial I_t(u)}{\partial t} = \nabla^2 I_t(u), \tag{87}$$

kde $\nabla^2 I_t$ je Laplacián I_t.

V prípade anizotropnej difúzie sa rovnica (87) zmení o difúzny koeficient $c(u)$ [9]:

$$\frac{\partial I_t(u)}{\partial t} = \mathrm{div}(c(u)\nabla I_t(u)), \tag{88}$$

$$I_0 = I(x, y). \tag{89}$$

Pomocou anizotrópnej difúzie môžeme docieliť podobnú transformáciu obrázka ako bilaterálnym filtrom, ktorý sme popísali v kapitole 2.2 [9].

Difúzny koeficient $c(u)$ je typicky funkciou $|\nabla I_t(u)|$ a je klesajúcou funkciou tohoto výrazu. Pre malé hodnoty $|\nabla I_t(u)|$ sa $c(u)$ blíži k 1 a s rastúcim $|\nabla I_t(u)|$ sa blíži k 0. Difúzny koeficient $c(u)$ by mal spĺňať tieto podmienky [9]:

1. $\lim_{|\nabla I_t(u)|\to 0} c(u) = C$, kde $0 < C < \infty$,
2. $\lim_{|\nabla I_t(u)|\to \infty} c(u) = 0$,
3. $c(u)$ je klesajúca funkcia od $|\nabla I_t(u)|$.

Prvá vlastnosť zaručuje izotropné zhladzovanie v oblasti s podobnými intenzitami, druhá vlastnosť zachováva hrany a tretia zaručuje numerickú stabilitu [9].

Prístupy rôznych autorov sa líšia v použití iného difúzneho koeficientu $c(u)$, ktorého voľba je najpodstatnejším prvkom pri detekcii hrán difúziou. Autori Perona, Malik [13]

použili nasledovné difúzne koeficienty:

$$c(u) = \exp\left\{-\left(\frac{\|\nabla I_t(u)\|}{k}\right)^2\right\}, \qquad k \in \mathbb{R} \tag{90}$$

$$c(u) = \frac{1}{1+\left(\frac{\|\nabla I_t(u)\|}{k}\right)^2}, \qquad k \in \mathbb{R}, \tag{91}$$

ktoré sú však veľmi citlivé na šum [9]. Autori Catte a kol. [2], Alvarez a kol. [1] tento problém vyriešili zakomponovaním Gassovského filtra:

$$c(u) = \exp\left\{-\left(\frac{\|\nabla S(u)\|}{k}\right)^2\right\}, \qquad k \in \mathbb{R}, \tag{92}$$

$$S = I_t * G_\sigma, \tag{93}$$

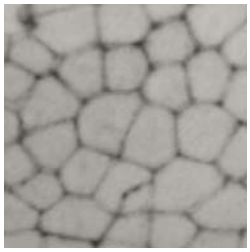
kde S je konvolúcia I_t a Gaussovského filtra so štandardnou odchýlkou σ [9]. Príklady iných, zložitejších difúznych koeficientov a ich vlastnosti možno nájsť v [9].

Ďalšie možné rozšírenie spočíva v úprave samotnej PDE. Rovnicu (87) možno rozšíriť o nehomogénnu pravú stranu $I_0(u) - I_t(u)$:

$$\frac{\partial I_t(u)}{\partial t} - \nabla^2 I_t(u) = I_0(u) - I_t(u), \tag{94}$$

aby sme zachovali čiastočnú podobnosť pôvodnému obrázku, vyhli sa nutnosti zvoliť čas zastavenia vývoja PDE a zabránili výsledku v podobe triviálneho riešenia, t.j. obrázku s konštantnou intenzitou [9].

Ďalej uvádzame príklad detekcie hrán ako kriviek, ktoré ohraničujú objekty. Tento prístup je vhodný na obrázky s homogénnou štruktúrou, ktorú typicky tvoria uzavreté krivky, ako je štruktúra včelých plástov alebo buniek, napr. embrya ryby danio pruhované [8] ako môžeme vidieť na Obr. 16.



Obr. 16: Bunková štruktúra ryby danio pruhované. Zdroj: [8]

V článku [8] používajú na detekciu hraníc medzi bunkami ryby danio pruhované tzv.

GSUBSURF (Generalized Subjective Surfaces) metódu, ktorá je daná nasledovnou PDE:

$$v_t - w_a \nabla q \cdot \nabla v - w_c q |\nabla v| \nabla \cdot \left(\frac{\nabla v}{|\nabla v|} \right) = 0, \tag{95}$$

kde w_a je váha pre advekčný člen a w_c je váha pre člen zakrivenia [8]. Funkcia $q(r)$ je detektor hrán, ktorý zvolili ako

$$q(r) = \frac{1}{1 + Mr^2}, \qquad M > 0, \tag{96}$$

kde M je parameter citlivosti a $r = |\nabla I(x, y)|$ je magnitúda gradientu vstupného obrázka [8].

Pri detekcii hrán pomocou PDE je potrebné inicializovať počiatočnú segmentáciu objektov v_0, ktorá sa bude rozpínať smerom k hraniciam objektu znútra alebo zmršťovať k objektu zvonku. V [8] najprv identifikovali pozície jednotlivých buniek pomocou tzv. LSCD (level-set center detection) algoritmu a následne inicializovali v_0 ako:

$$v_0(u) = \frac{1}{1 + d(u)}, \tag{97}$$

kde $d(u)$ je euklidovská vzdialenosť u od identifikovaných pozícií buniek.

V našej práci sa zoberáme fotografiami, na ktorých sú objekty ako tváre, budovy, zvieratá a pod. Na týchto obrázkoch sa nachádzajú hrany nielen ako hranice objektov, ktoré detekujeme, ako v prípade bunkovej štruktúry, ale aj ako neuzavreté krivky dotvárajúce objekty na obrázku, napr. tiene, vlasy, črty tváre a pod. Naším cieľom nie je detekovať štruktúru hraníc objektov ako v prípade [8], preto tento typ detekcie hrán nie je vhodný pre naše obrázky. Tiež by bolo nesmierne komplikované a pri niektorých obrázkoch možno nerealizovateľné inicializovať počiatočné v_0. Zo spojitého prístupu však využijeme funkciu $q(r)$ na invertovanie hrán získaných aproximáciou gradientu diskrétnym spôsobom.

2.5 Invertovanie a preškálovanie

Po zvýšení kontrastu, rozmazaní vhodným filtrom a detekcii hrán nasleduje post-spracovanie. V závislosti od použitej metódy aproximácie gradientu môžu hodnoty $\hat{g}(x, y)$ ležať mimo intervalu $[0, 255]$, takže je ešte potrebné ich preškálovať. Keďže v Python 3 je biela kódovaná ako 255, získané intenzity treba invertovať, pretože čím výraznejšia hrana, tým bledšiu dostaneme novú intenzitu $\hat{g}(x, y)$.

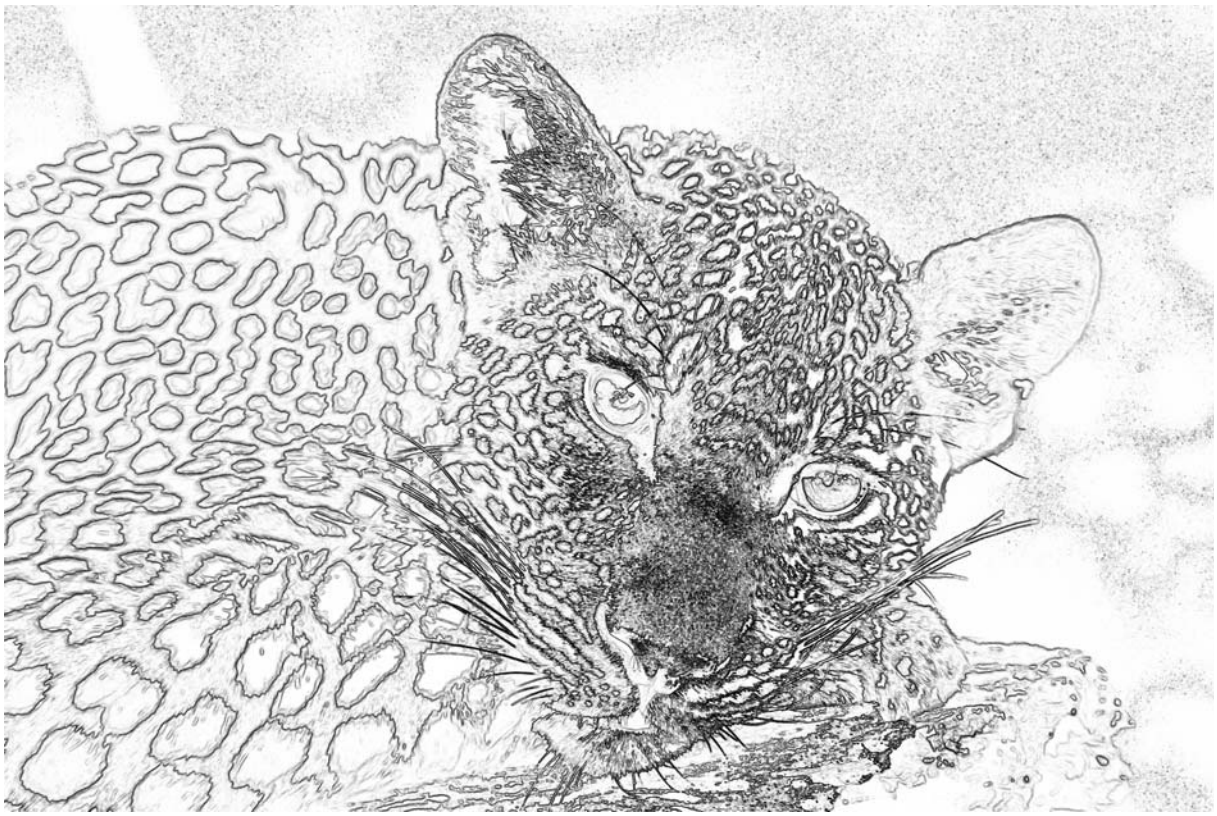
Za účelom invertovania navrhujeme použiť transformáciu $q(r)$ zo spojitých metód z (96) a preškálovanie na interval $[0, 255]$ zrealizujeme jednoducho vynásobením 255 a zaokrúhlením:

$$\hat{g}_{inv}(x, y) = 255 \times q(\hat{g}(x, y))\ ^*, \tag{98}$$

$$q(r) = \frac{1}{1 + Mr^2}, \qquad M > 0, \tag{99}$$

** zaokrúhlené na celé čísla*

Parametrom M kontrolujeme, nakoľko dôjde k stmaveniu získaných hrán. Čím väčšie M, tým tmavší je výstup. Voľbe parametra M sa budeme venovať v kapitole 3. Obrázok invertovaný pomocou (98) ilustrujeme na Obr. 17.

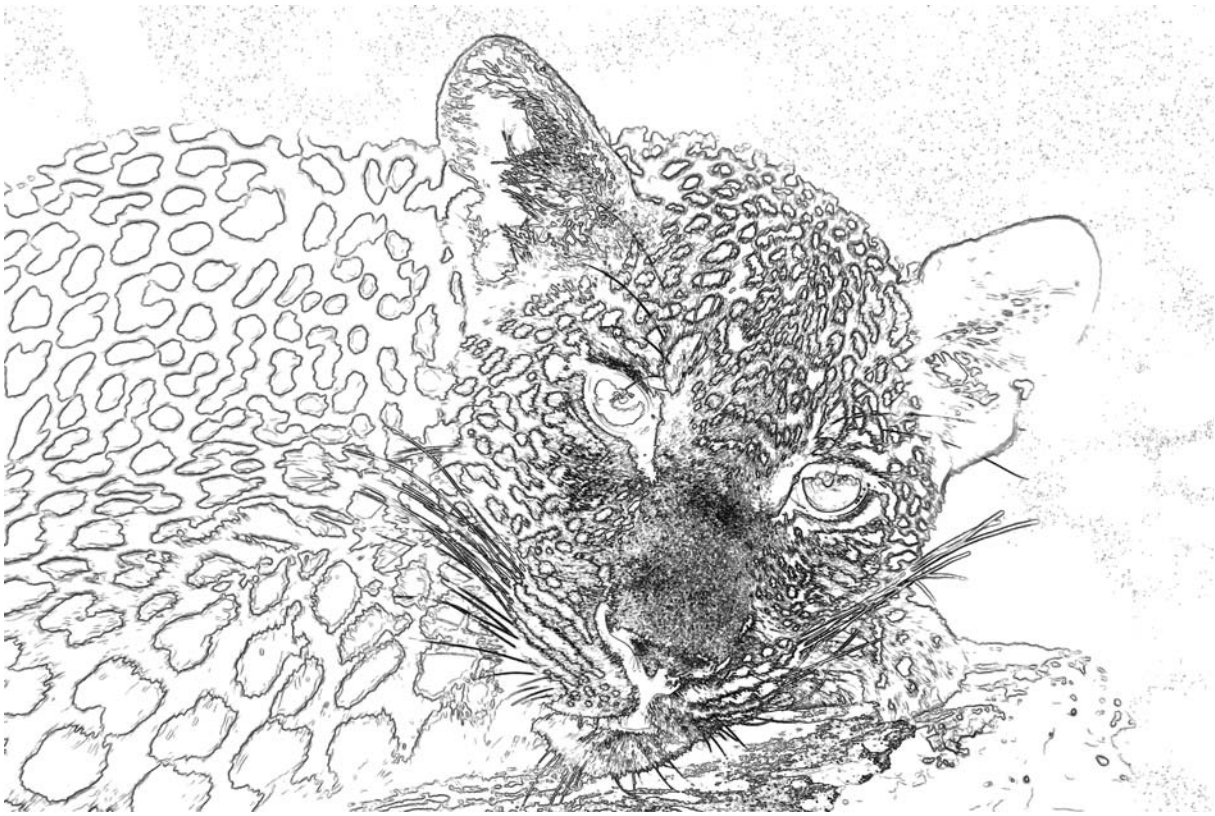


Obr. 17: Na Obr. 7*(a)* bola aplikovaná jednoduchá DEH s $x = 0$, bilaterálny filter so $\sigma_d = 3, \sigma_c = 20, s = 7$, Kirschov operátor 3×3, následne invertovanie s $M = 2.5 \times 10^{-5}$ a preškálovanie na $[0, 255]$

2.6 Ohraničenie s hysterézou

Po invertovaní obrázku prejdeme k selekcii finálnych hrán metódou ohraničenia s hysterézou z Cannyho metódy, ako sme popísali v kapitole 1.3.5. Voľbu t_L a t_H navrhneme v

kapitole 3. Ukážku ohraničenia z hysterézou na obrázku môžeme vidieť na Obr. 18.



Obr. 18: Ukážka aplikácie ohraničenia s hysterézou pri $T_L = 207, T_H = 158$ na Obr. 17

2.7 Čiastočné stenčenie hrán

S cieľom dosiahnuť esteticky krajší výstupný obrázok navrhujeme pozmeniť stenčenie hrán z Cannyho metódy ako sme popísali v kapitole 1.3.4. V Cannyho metóde prejdú selekciou iba tie body, v ktoré sú najtmavšie (t.j. majú najmenšiu hodnotu) spomedzi ich dvoch susedov v smere gradientu. Inak zmeníme ich intenzitu na bielu. Týmto spôsobom efektívne získame obrysy objektov, ale hrany sú na pohľad príliš tenké a často nesúvislé.

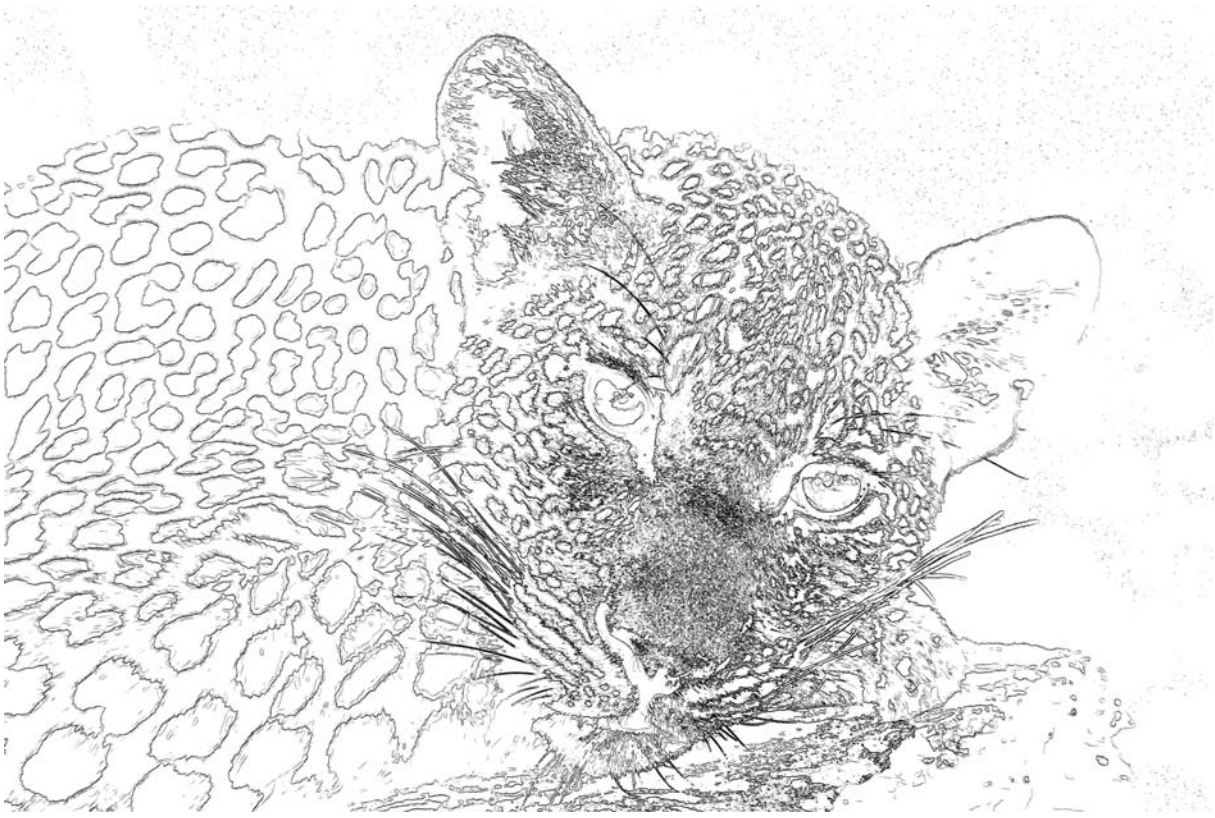
Ako prvú zmenu navrhujeme nevyžadovať ostrú rovnosť v podmienke minima, ale dovoliť prejsť selekciou aj v prípade rovnosti. Ďalej navrhujeme zachovanie viac ako jedného z troch susedných bodov.

Uvažujme body, v ktorých sa nadobúda neostré minimum v smere gradientu. Máme tri možnosti:

1. Minimálne stenčenie hrán: zachováme bod v minime aj oba jeho susedné body
2. Čiastočné stenčenie hrán: zachováme bod v minime a tmavší z jeho susedných bodov

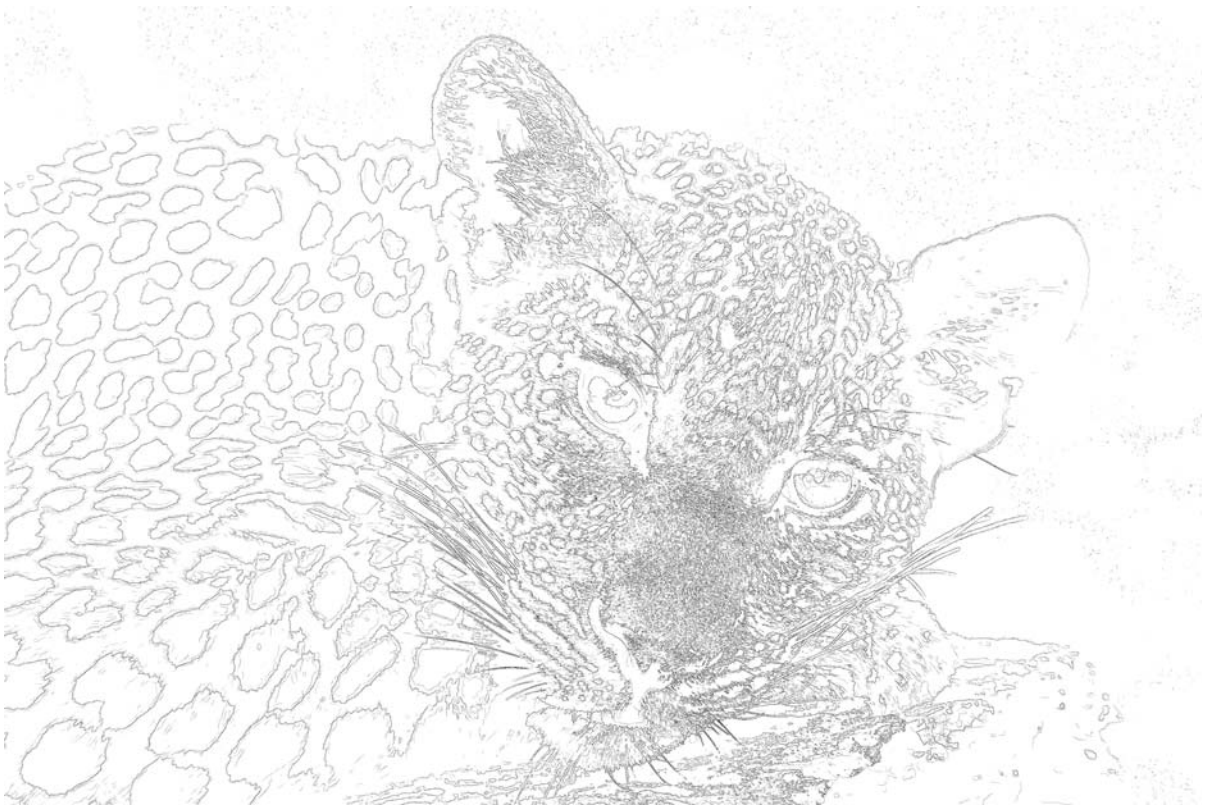
3. Úplné stenčenie hrán: zachováme iba bod v minime (Cannyho metóda)

Ostatné body, ktoré nie sú minimom ani s ním nesusedia, budú zmenené na bielu. Porovnanie uvedených troch spôsobov môžeme vidieť na Obr. 19-21. Z experimentovania s niekoľkými obrázkami sa nám javí, že najrozumnejšia bilancia hrúbky hrán nastáva pri druhej možnosti. Voľba závisí od cieleného efektu: ak potrebujeme výraznejšie a hrubšie hrany, vhodný je prvý spôsob. Ak potrebujeme iba hranice objektov, postačí tretí spôsob. V našej modifikácii algoritmu budeme používať druhý spôsob.



Obr. 19: Minimálne stenčenie hrán - 1. spôsob

Obr. 20: Čiastočné stenčenie hrán - 2. spôsob



Obr. 21: Úplné stenčenie hrán - 3. spôsob

3 Návrh nášho algoritmu a voľby parametrov

V tejto kapitole sumarizujeme návrh nášho algoritmu a navrhneme voľbu parametrov v jednotlivých krokoch.

Náš algoritmus pozostáva z nasledovných krokov:

1. Dynamická ekvalizácia histogramu jednoduchou metódou, alebo metódou s interpoláciou
2. Bilaterálny filter
3. Detekcia hrán Kirschovým operátorom
4. Invertovanie a preškálovanie
5. Ohraničenie s hysterézou
6. Stenčenie hrán

Kroky $1, 2, 4$ a 5 obsahujú parametre, ktoré je potrebné vybrať v závislosti od charakteru obrázku a pri nesprávnej voľbe môže aj funkčná metóda dávať slabé výsledky. Keďže v literatúre sa tento výber často ponecháva na čitateľa, skúmali sme, ako ich vhodne voliť a ďalej prezentujeme naše pozorovania a návrhy.

Experimentovali sme s poradím $1. - 2.$ kroku a $5. - 6.$ kroku. Osvedčilo sa nám vykonať najprv DEH a následne bilaterálny filter, pretože týmto spôsobom je z obrázku odstráneného viac šumu. Pri zmene poradia $5. - 6.$ kroku neboli výsledky zásadne odlišné. Rozhodli sme sa urobiť ohraničenie s hysterézou pred stenčovaním hrán z dôvodu menšej výpočtovej zložitosti. Ohraničením bude totiž veľká časť intenzít zmenená na bielu a biele pixle pri stenčovaní hrán netreba uvažovať, keďže nie sú hranami.

Najskôr uvádzame riešenie voľby parametrov k piatemu kroku, pretože obsahuje charakteristiky obrázka, ktoré využijeme aj pri iných krokoch.

3.1 Voľba hraničných hodnôt pri ohraničení s hysterézou

V tomto kroku potrebujeme zvoliť dva parametre: t_H pre ohraničenie silných hrán a t_L pre ohraničenie slabých hrán, ako sme popísali v časti 1.3.5. Problém voľby parametra t_H

sme identifikovali ako problém oddelenia objektov od pozadia, na riešenie ktorého bola v [12] navrhnutá tzv. Otsouva hranica (z angl. Otsu threshold).

Označme T maximálnu hodnotu intenzity vyskutujúcej sa na obrázku. Myšlienka metódy Otsuovej hranice spočíva v oddelení histogramu obrázku v hodnote k^* na dve časti C_0 a C_1 [12]:

$$C_0 = [0, 1, \ldots, k^*] \tag{100}$$

$$C_1 = [k^* + 1, k^* + 2, \ldots, T]. \tag{101}$$

Interval C_0 predstavuje v našom prípade intenzity objektov a C_1 intenzity pozadia. Ako uvádzame nižšie, hraničná intenzita k^* je optimalizovaná tak, aby variancia intenzít v rámci C_i bola čo najväčšia. Preto tento prístup funguje najlepšie, ak je obrázok tzv. bimodálny, teda obsahuje dva výrazné subhistogramy [12].

Označme podobne ako pri ekvalizácii histogramu odhad hustoty intenzít ako [12]:

$$p_i = \frac{H(i)}{N}, \qquad p_i \geq 0, \qquad \sum p_i = 1 \tag{102}$$

$$N = \sum_{i=0}^{T} H(k). \tag{103}$$

Pravedodobnosť výskytu škály intentzít C_0, resp. C_1 označme ako [12]:

$$\omega_0(k) \coloneqq P(C_0) = \sum_{i=0}^{k} p_i, \tag{104}$$

$$\omega_1(k) \coloneqq P(C_0) = \sum_{i=k+1}^{T} p_i = 1 - \omega_0 \tag{105}$$

a výberovú strednú hodnotu na každej z oblastí ako [12]:

$$\mu_0(k) \coloneqq \sum_{i=0}^{k} iP(i|C_0) = \frac{\sum_{i=0}^{k} ip_i}{\omega_0}, \tag{106}$$

$$\mu_1(k) \coloneqq \sum_{i=k+1}^{T} iP(i|C_1) = \frac{\sum_{i=k+1}^{T} ip_i}{\omega_1} = \frac{\mu_T - \mu_0}{1 - \omega_0}, \tag{107}$$

kde μ_T je celkový priemer intenzít na obrázku:

$$\mu_T = \sum_{i=0}^{T} ip_i. \tag{108}$$

Výberové variancie intenzít na oblastiach C_0, C_1 a celková variancia intenzít obrázka σ_T^2 sú potom [12]:

$$\sigma_0^2 = \sum_{i=0}^{k} (i - \mu_0)^2 P(i|C_0) = \sum_{i=0}^{k} (i - \mu_0)^2 \frac{p_i}{\omega_0} \tag{109}$$

$$\sigma_1^2 = \sum_{i=k+1}^{T} (i - \mu_1)^2 P(i|C_1) = \sum_{i=k+1}^{T} (i - \mu_1)^2 \frac{p_i}{\omega_1} \tag{110}$$

$$\sigma_T^2 = \sum_{i=0}^{T} (i - \mu_T)^2 p_i \tag{111}$$

Optimálna hranica k^* je taká, ktorá maximalizuje medzi-súborovú varianciu (z angl. between class variance), ktorá je definovaná nasledovne [12]:

$$\sigma_B^2(k) = \omega_0(k)(\mu_0(k) - \mu_T)^2 + \omega_1(k)(\mu_1(k) - \mu_T)^2, \tag{112}$$

a teda optimálne k hľadáme ako [12]:

$$k^* = \max_{0 \leq k < T-1} \sigma_B^2(k). \tag{113}$$

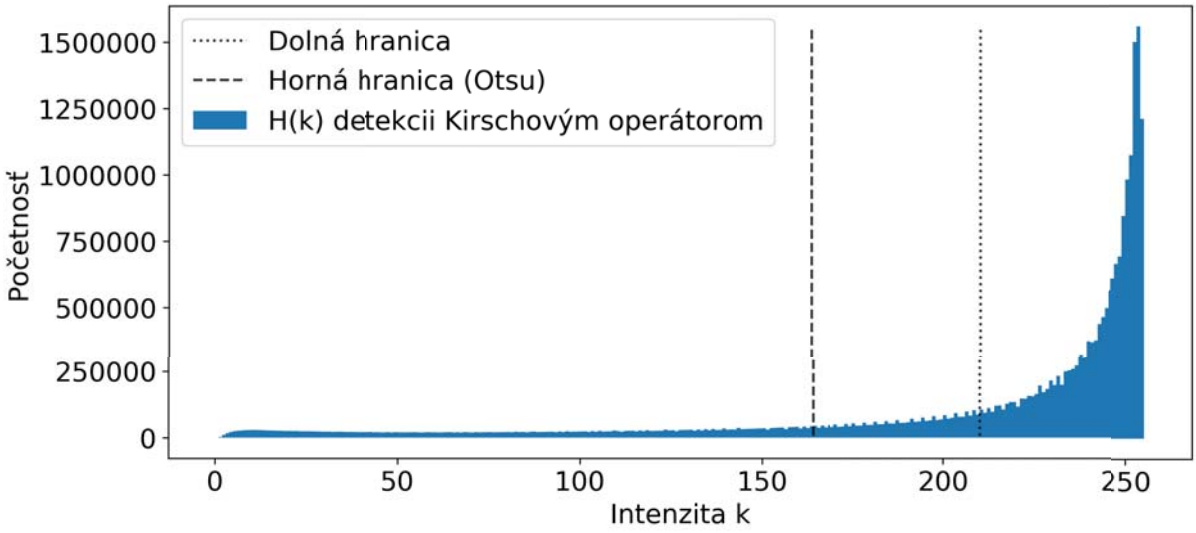
Po nájdení optimálnej hodnoty k^* budeme pri ohraničení s hysterézou voliť hodnoty t_H, t_L ako bolo navrhnuté v [7]:

$$t_H = k^*, \tag{114}$$

$$t_L = \frac{t_H}{2}. \tag{115}$$

V našom značení a v Python 3 indexujeme čiernu ako 0 a bielu ako 255, čo je opačný koncept ako v prípade [3], [4], [7], [12]. Preto intenzity k z obrázku z kroku 4 invertujeme jednoduchou transformáciou $k \longrightarrow 255 - k$, nájdeme k^*, aplikujeme ohraničenie s hysterézou a invertujeme rovnakým spôsobom naspäť.

Ukážku histogramu s nájdenou hranicou k^*, ktorú použijeme ako t_H, môžeme vidieť na Obr. 22. Pre invertovaný obrázok sme našli hranice $t_H = k^* = 91$ a $t_L = \frac{t_H}{2} \doteq 45$, takže po spätnom invertovaní dostávame $t_H = 255 - 91 = 164$ a $t_L = 255 - 45 = 210$.



Obr. 22: Príklad histogramu jeho a separácia v bodoch $t_H = k^* = 164$ a $t_L = 210$ metódou Otsu threshold

3.2 Voľba koeficientu zvýraznenia a metódy v DEH

V kapitole 2.1 sme popísali dva spôsoby dynamickej ekvalizácie histogramu: jednoduchú DEH a DEH s interpoláciou. Jednoduchá DEH popísaná v [17] je vhodná, ak je žiadame výraznú ekvalizáciu histogramu. Ak má táto metóda príliš silný efekt aj s voľbou $x = 0$, je vhodné použiť našu modifikáciu (DEH s interpoláciou), pri ktorej nedochádza k tak radikálnej zmene intenzít.

V druhom kroku DEH vystupuje v (38) parameter x, ktorým kontrolujeme, aký silný efekt zvýšenia kontrastu chceme dosiahnuť. V [17] navrhujú voliť x z intervalu $[0, 1, \ldots, 5]$ v závislosti od dynamického rozsahu pôvodného obrázka. Ak histogram siaha až ku krajným hodnotám 0 a 255, odporúčajú zvoliť vyššie x a ak je rozsah menší, dostatočný efekt bude mať aj $x = 0$.

Vlastnosť rozptylu intenzít na pôvodnom obrázku navrhujeme kvantifikovať prirodzene pomocou celkovej variancie, resp. štandardnej odchýlky obrázku σ_T z (111). Skúmali sme, akú varianciu dosahujú obrázky na vzorke bežných fotografií a hľadali sme najvhodnejšiu voľbu x vzhľadom na výsledný obrázok po šiestom kroku (pri voľbe ostatných parametrov tak, ako uvádzame v tejto kapitole). Hodnota 5 bola príliš vysoká pre všetky obrázky, preto sme použili najviac $x = 4$.

Výsledky pozorovania uvádzame v Tabuľke 2. Pre obrázky s varianciou vyššou ako 82

navrhujeme krok 1 vynechať, alebo použiť DEH s interpoláciou. Keďže rozloženie intenzít vrámci histogramu je veľmi individuálne, navrhujeme podobne ako v [17] vyskúšať viaceré hodnoty x a vybrať najvhodnejšiu. Keďže x volíme z maximálne 6 hodnôt, vyskúšať aj všetky možnosti nie tak náročné [17].

Variancia intenzít	$\sigma_T \leq 50$	$50 < \sigma_T \leq 60$	$60 < \sigma_T \leq 67$	$67 < \sigma_T \leq 73$	$73 < \sigma_T \leq 82$
x	0	1	2	3	4

Tabuľka 2: Návrh voľby parametra x pri jednoduchej DEH v závislosti od variancie intenzít pôvodného obrázku.

3.3 Voľba veľkosti bilaterálneho filtra a parametrov variancie

K realizácii druhého kroku potrebujeme zvoliť tri parametre: veľkosť filtra s z (72), varianciu σ_d v Gaussovskej funkcii váhujúcej vzdialenosti z (69):

$$f_d(\xi, u) = e^{-\frac{1}{2}\left(\frac{d(\xi,u)}{\sigma_d}\right)^2}, \tag{116}$$

$$d(\xi, u) = \|\xi - u\|_2 \tag{117}$$

a varianciu σ_c v Gaussovskej funkcii váhujúcej podobnosti intenzít z (70)

$$f_c(\xi, u) = e^{-\frac{1}{2}\left(\frac{d(I(\xi),I(u))}{\sigma_c}\right)^2}, \tag{118}$$

$$d(I(\xi), I(u)) = |I(\xi) - I(u)|. \tag{119}$$

Veľkosť filtra s navrhujeme voliť v závislosti od rozmerov obrázku. Podľa dokumentácie k Python 3 sú z hľadiska výpočtovej náročnosti na real-time aplikácie odporúčané filtre s veľkosťou $s = 5$ a na offline aplikácie s maximálnou veľkosťou $s = 9$ [11]. Keďže pracujeme s fotografiami, rozdelíme obrázky na tri skupiny podľa toho, akým kvalitným fotoaparátom boli odfotené a v našom algoritme bude pri výbere s rozhodujúca dĺžka uhlopriečky. Tento návrh voľby s uvádzame v tabuľke 3.

s	Dĺžka uhlopriečky D (px)	Typický rozmer (px)	Parameter fotoaparátu
5	$D \leq 2560$	2048×1536	3.1 MP
7	$2560 < D \leq 4560$	2580×2048	5 MP
9	$D > 4560$	3648×2736	10 MP

Tabuľka 3: Voľba veľkosti filtra s podľa veľkosti uhlopriečky obrázku (v pixloch). Zdroj údajov zo stĺpcov 3 a 4: `https://www.umass.edu/it/sites/it/files/2012/04/03/image-dimensions.pdf`

Pri parametroch σ_d a σ_c musíme zohľadniť, že do funkcie $f_d(\xi, u)$ vstupujú oveľa menšie hodnoty ako do $f_c(\xi, u)$. Vzdialenosti pixlov dosahujú maximálne hodnotu $\frac{s-1}{\sqrt{2}}$:

$$\|\xi - u\|_2 \leq \sqrt{2\left(\frac{s-1}{2}\right)^2} \doteq \begin{cases} 2.8, & s = 5 \\ 4.2, & s = 7 \\ 5.7, & s = 9, \end{cases} \tag{120}$$

kým rozdiely intenzít dosahujú hodnoty až do 255. Umocnením čitateľa exponentu v (116) a (118) sa rozdiel ešte rádovo zväčší. Keďže obe funkcie $f_d(\xi, u)$ a $f_c(\xi, u)$ uvažujeme na rovnakej $s \times s$ oblasti, parameter σ_c budeme musieť voliť väčší.

Ak by sme zvolili príliš veľkú σ_c pri malej σ_d, hodnoty f_d by boli plného rozsahu, kým hodnoty f_c by mala velkú varianciu a hodnoty na uvažovanej $s \times s$ oblasti by boli takmer identické. Tým by sa stratil efekt f_c v súčine $f_d f_c$, filter mal na obrázok efekt ako Gaussovský priestorový filter a hrany by boli rozmazané. Ak by sme naopak zvolili príliš veľkú σ_d, intenzity by sa násobili prakticky iba hodnotami f_c, čo by na obrázok nemalo efekt [16].

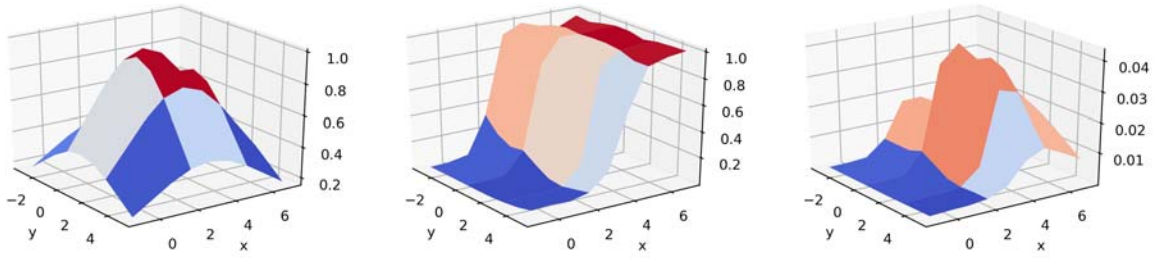
Pri voľbe σ_c a σ_d navrhujeme obmedziť sa na určitý rozsah oboch paramterov a vybrať takú kombináciu, pri ktorej sa váhy na okolí s hranou čo najviac približujú tvaru váh z Obr. 14. Na základe odporúčaní a ukážok použitia bilaterálneho filtra v [16] sme sa rozhodli pozrieť rozsah $\sigma_d \in [1, 8]$ a $\sigma_c \in [10, 30]$ a vybrať takú kombináciu, ktorej váhy budú mať v okolí hrany najstrmší prechod medzi vysokými a nízkymi váhami. Týmto spôsobom by malo dôjsť k najmenšiemu rozmazaniu hrán.

Na základe našej analýzy a validovaní na reálnych obrázkoch navrhujeme voliť

$$\sigma_c = 20, \tag{121}$$

$$\sigma_d = 3. \tag{122}$$

Váhy bilaterálneho filtra pri takejto voľbe pre bod hrane a jeho 9×9 okolie môžeme vidieť na Obr. 23.



Obr. 23: Váhy bilaterálneho filtra pre bod na hrane pri $\sigma_c = 20$, $\sigma_d = 3$, $s = 9$

3.4 Voľba operátora pre aproximáciu gradientu a jeho veľkosti

Experimentovali sme s operátormi pre aproximáciu gradientu diskrétnym prístupom uvedenými v kapitole 2.3. Rozdiel medzi jednotlivými operátormi nebol zásadný, avšak hrany detekované Kirschovým operátorom boli najvýraznejšie, keďže v tomto prístupe vyberáme maximum zo všetkých ôsmich smerov. Použitím Kirschovho operátora navyše priamo získame aj smery, v ktorých dochádza k najväčšej zmene intenzity, čo využijeme pri stenčovaní v kroku 6.

Čo sa týka voľby veľkosti konvolučnej matice, experimentálne sme porovnávali finálne výsledky po kroku 6 pre rozmery 3×3 a 5×5 pri rovnakých zvyšných parametroch. Pri použití menšej konvolučnej matice sa zachováva viac detailov. V prípade detekcie hrán na obrázku s ľudskými postavami sa to ukázalo ako výhoda, pretože boli zachované aj niektoré tiene a črty tváre pôsobili prirodzenejšie. Tiež bol vo finálnom obrázku väčší rozsah intenzít, vďaka čomu si obrázok zachoval určitú hĺbku (objekty na obrázku pôsobili trojdimenzionálne). Pri veľkosti 5×5 boli detekované všetky podstatné hrany, napr. obrysy postáv, ale bolo zachovaných menej detailov. Výsledný obrázok mal menší rozsah intenzít, kvôli čomu pôsobil viac plocho (dvojdimenzionálne). Porovnanie výsledných obrázkov pri použití 3×3 a 5×5 operátora môžeme vidieť v kapitole 4.

3.5 Voľba koeficientu stmavenia pri invertovaní

Intertovanie a škálovanie realizujeme funkciou $q(r)$ ako sme popísali v časti 2.5:

$$\hat{g}_{inv}(x,y) = 255 \times q(\hat{g}(x,y))\ ^*, \tag{123}$$

$$q(r) = \frac{1}{1+Mr^2}, \qquad M > 0, \tag{124}$$

** zaokrúhlené na celé čísla*

kde r sú získané hrany aproximáciou gradientu. Navrhujeme voľbu abstraktného parametra M previesť na parameter s ľahšie predstaviteľnou interpretáciou. Dopočítajme parameter M v závislosti od toho, aké percento zo získaných intenzít chceme previesť na čiernu, resp. takmer čiernu.

Najprv zvolíme hodnotu intenzity, od ktorej nižšie intenzity už budeme považovať za čierne. Po analýze čierno-bieleho spektra sme sa rozhodli zvoliť intenzitu $B = 13$. Nech zvolený $p-$percentný kvantil je r_p, napr. $p = 99$ pre Obr. 17 vpravo. Aby bolo $100 - p$ percent intenzít prevedených na takmer čiernu, vyžadujeme platnosť nerovnosti:

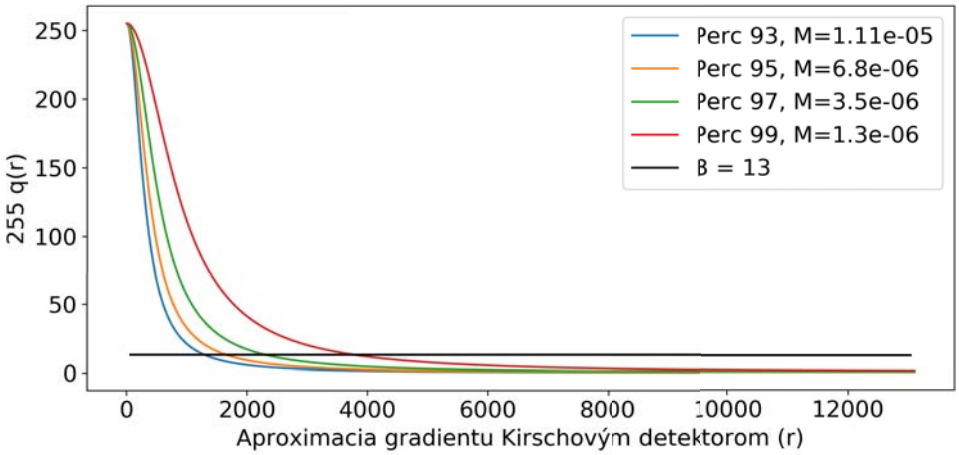
$$255\frac{1}{1+Mr^2} \leq B, \qquad \text{ak } r > r_p, \tag{125}$$

ktorú ekvivalentnými úpravami a dosadením $B = 13$ prevedieme na hraničnú podmienku:

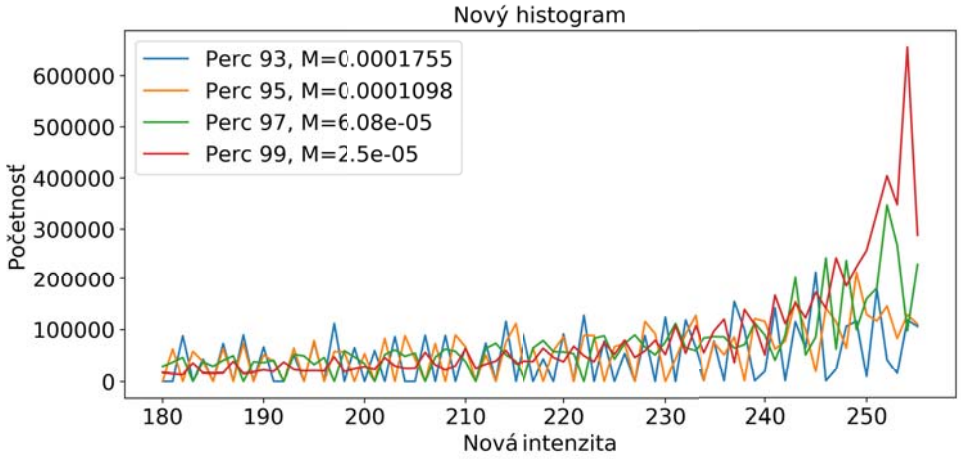
$$M = \frac{242}{13r_p^2} \doteq \frac{19}{r_p^2}. \tag{126}$$

Experimentálne sme zistili, že percento p je vhodné voliť z intervalu $[92, 93, \ldots, 99]$, vo väčšine prípadov 99. Túto hodnotu budeme používať ako predvolenú. Väčšinou sme sa stretli s tým, že na výslednom obrázku bolo priveľa hrán (šumu). Ak je na výslednom obrázku príliš málo hrán, je potrebné zvoliť menšie percento.

Graf funkcie $q(r)$ pre rôzne hodnoty zvolených percent, resp. parametra M pre rozsah intenzít Obr. 17 uvádzame na Obr. 24. Môžeme vidieť ako s rastúcim kvantilom stúpa hraničná intenzita, od ktorej vyššie hodnoty sa prevedú na čiernu, t.j. priesečník $q(r)$ s priamkou $B = 13$. Na Obr. 25 môžeme vidieť výsek nového histogramu pre intenzity bledšie ako 180, kde je pre voľbu percenta 99 najviac pixlov bielej farby, čo značí, že je na obrázku najmenej šumu.



Obr. 24: Transformačná funkcia $q(r)$ pre rôzne hodnoty zvoleného percenta, resp. M



Obr. 25: Výsek histogramu po invertovaní a preškálovaní pre rôzne hodnoty zvoleného percenta, resp. M

4 Galéria výsledkov

Na všetky obrázky v tejto sekcii sme použili náš algoritmus s krokmi:

1. Ekvalizácia histogramu metódou s voľbou koeficientu zvýraznenia x podľa variance, ako sme popísali v časti 3.2. Uvádzame porovnanie jednoduchej DEH a DEH s interpoláciou.

2. Bilaterálny filter s $\sigma_c = 20$, $\sigma_d = 3$ a veľkosťou s podľa dĺžky diagonály obrázku, ako sme popísali v časti 3.3.

3. Detekcia hrán Kirschovým operátorom z časti 2.3.3. Uvádzame porovnanie rozmerov 3×3 a 5×5.

4. Invertovanie funkciou $q(r)$ a preškálovanie na $[0, 255]$ z časti 2.5. Uvádzame porovnanie rôznej voľby percenta p.

5. Ohraničenie s hysterézou s voľbou t_L, t_H pomocou Otsuovej hranice ako sme popísali v 3.1.

6. Stenčenie hrán z časti 2.7 druhým spôsobom, t.j. čiastočné stenčenie.

Pre diverzitu obrázkov dopĺňame k obrázkom zvierat z predošlých kapitol obrázky postáv a budov. Originály uvádzame na prvých dvoch obrázkoch č. 26 a 27. Na nasledujúcich dvoch obrázkoch č. 28 a 29 s postavami môžeme vidieť porovnanie použitia jednoduchej DEH a DEH s interpoláciou. Na ďalších dvoch obrázkoch budovy č. 30 a 31 je porovnanie použitia 3×3 a 5×5 Kirschovho detektora. Na obrázkoch č. 31, 32 a 33 ilustrujeme porovnanie voľby percenta pri invertovaní, od čoho závisí množstvo výsledných hrán.

Obr. 26: Fotografia postáv. Zdroj: Jakub Krchňavý

Obr. 27: Fotografia budovy. Zdroj: vlastná fotografia

Obr. 28: $x = 4, \sigma_d = 3, \sigma_c = 20, s = 9, p = 99\%, M = 6.79 \times 10^{-5}, t_L = 210, t_H = 165$, jednoduchá DEH

Obr. 29: $x = 4, \sigma_d = 3, \sigma_c = 20, s = 9, p = 99\%, M = 6.92 \times 10^{-5}, t_L = 210, t_H = 165$, DEH s interpoláciou

Obr. 30: $\sigma_d = 3, \sigma_c = 20, s = 9, p = 99\%, M = 1.5 \times 10^{-5}, t_L = 209, t_H = 163$, jednoduchá DEH, Kirschov operátor 3×3

Obr. 31: $\sigma_d = 3, \sigma_c = 20, s = 9, p = 99\%, M = 4 \times 10^{-7}, t_L = 210, t_H = 164$, jednoduchá DEH, Kirschov operátor 5×5

Obr. 32: $\sigma_d = 3, \sigma_c = 20, s = 9, p = 98\%, M = 7 \times 10^{-7}, t_L = 207, t_H = 158$, jednoduchá DEH, Kirschov operátor 5×5

Obr. 33: $\sigma_d = 3, \sigma_c = 20, s = 9, p = 97\%, M = 1 \times 10^{-6}, t_L = 205, t_H = 154$, jednoduchá DEH, Kirschov operátor 5×5

Záver

Cieľom práce bolo prehľadne spracovať metódy na detekciu hrán v obrázkoch a niektoré z nich naprogramovať. Popísali sme známy Cannyho detektor, ktorý slúžil ako základ pre túto prácu. Ďalej sme popísali širšiu škálu známych metód, ktorými sa dajú substituovať niektoré kroky Cannyho metódy.

Z metód popísaných v prvých dvoch kapitolách sme vybrali pre nás najvhodnejšie, navrhli ich modifikácie a pridali sme predspracovanie obrázka dynamickou ekvalizáciou histogramu s návrhom vlastnej modifikácie. Tiež sme navrhli spôsob invertovania a preškálovania po detekcii hrán aproximáciou gradientu.

Naprogramovali sme kompletný algoritmus na detekciu hrán vo fotografiách využitím známych, modifikovaných a nami navrhnutých metód. Skúmali sme voľbu parametrov k tomuto algoritmu a navrhli sme ich automatickú voľbu, alebo aspoň zúžili rozsah na niekoľko hodnôt, z ktorých možno vybrať v závislosti od cieleného výzoru. Uviedli sme ukážky, na ktorých možno vidieť efekt rôznej voľby hodnôt niektorých parametrov.

Možným nedostatkom navrhnutého algoritmu je citlivosť na parametre z dôvodu individuality obrázkov. Môže sa stať, že na obrázku sa v princípe ťažko detekujú hrany, pretože napríklad obsahuje iba plynulé prechody. Taktiež je možné, že kvoli anomáliám na obrázku, príp. jeho histograme, ktoré sa nenachádzali na nami skúmaných obrázkoch, bude efekt ekvalizácie histogramu alebo bilaterálneho filtra príliš silný. To môže viesť k detekcii prebytočných hrán.

Ďalším pokračovaní práce by mohlo byť porovnanie použitia diferenciálnych rovníc voči diskrétnemu bilaterálnemu filtru, prípadne náhrada použitých metód inou alternatívou. Tiež je priestor na hlbšie skúmanie voľby parametrov na širšom spektre obrázkov. Keďže algoritmus sa kvôli mnohým krokom stal výpočtovo relatívne náročný, predovšetkým na veľkých obrázkoch, mohol by sa nahradiť umelou neurónovou sieťou. Sieť by mohla byť natrénovaná na výstupoch nášho algoritmu a pokiaľ by sa zabránilo tzv. overfittingu, mohla by dávať dokonca lepšie výsledky.

Zoznam použitej literatúry

[1] ALVAREZ, L., LION, P.-L., MOREL J.-M., Image Selective Smoothing and Edge Detection by Nonlinear Diffusion II. *SIAM Journal on Numerical Analysis*, 1992, vol. 29, s. 845-866.

[2] CATTE, F. et al., Image Selective Smoothing and Edge Detection by Nonlinear Diffusion. *SIAM Journal on Numerical Analysis*, 1992, vol. 29., s .182-193

[3] CANNY, J. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1986, vol. PAMI-8, s. 679-714.

[4] DING, L., GOSHTASBY, A. On the Canny edge detector. *Pattern Recognition*, 2001, vol. 34, s. 721-725.

[5] JACOBS, D. Image gradients, Class notes for CMSC 426. Department of Computer Science. University of Maryland, Washington, 2005.

[6] KEKRE, B., GHARGE, S. Image Segmentation using Extended Edge Operator for Mammographic Images. *International Journal on Computer Science and Engineering*, 2010, vol. 02, no. 04, s. 1086-1091.

[7] MEI, F., YUE, G., YU, Q. The Study on An Application of Otsu Method in Canny Operator. *Proceedings of the 2009 International Symposium on Information Processing*, 2009, s. 109-112.

[8] MIKULA, K., SMÍŠEK, M., ŠPIR, R. Parallel Algorithms for Segmentation of Cellular Structures in 2D+time and 3D Morphogenesis Data. *Proceedings of ALGORITMY 2012*, 2012, s. 416-426.

[9] MLSNA, P., RODRÍGUEZ J. The Essential Guide to Image Processing (Second Edition), Chapter 19 – Gradient and Laplacian Edge Detection. *Academic press*, 2009

[10] OpenCV 2.4.13.6 documentation. Image Filtering. Dostupné na internete (6.6.2017): `https://docs.opencv.org/2.4/modules/imgproc/doc/imgproc.html`

[11] OpenCV 3.1.0 documentation. OpenCV modules. Dostupné na internete (6.6.2017): `https://docs.opencv.org/3.1.0/`

[12] OTSU, N. A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 1979, vol. SMC-9, no. 1., s. 62-66.

[13] PERONA, P., MALIK, J. Scale-space and Edge Detection Using Anisotropic Diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1990, vol. PAMI-12, p. 629-639.

[14] SU, Q. Color Image Watermarking: Algorithms and Technologies. *Walter de Gruyter GmbH & Co KG*, 2017, s. 57.

[15] TAN, L., JIANG, J. Digital Signal Processing. *Academic Press*, 2013.

[16] TOMASI, C. MANDUCHI, R. Bilateral Filtering for Gray and Color Images. *Proceedings of the 1998 IEEE International Conference on Computer Vision*, India, 1998, No.98CH36271, s. 839-846.

[17] WADUD, A. a col., A Dynamic Histogram Equalization for Image Contrast Enhancement. *IEEE Transactions on Consumer Electronics*, 2007, vol. 53, no. 2, s. 593-600.

Príloha A

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from scipy import ndimage as ndi
from scipy.signal import argrelmin

def HE(sub, old_L, old_U, rmin=0, rmax=256, which_method='interpolation'):
    ''' Vrati nove mapovanie intenzit z [old_L, old_U) na range [rmin, rmax)

    :param sub: subhistogram, right end excluded
    :param old_L: lower bound of original range, incl. [
    :param old_U: upper bound of original range, excl. )
    :param rmin: lower bound of new range, incl. [
    :param rmax: upper bound of new range, exc. )
    :param which_method: simple: ako v clanku DEH, interpolation: interpolovat,
    rovnomernejsi histogram
    :return: dataframe s mapovanim stara i -> nova i '''

    # POVODNY RANGE
    x_old = np.arange(old_L, old_U, 1).astype(int)
    #print('Stary rozsah [{}, {})'.format(old_L, old_U))
    df = pd.DataFrame({'h': sub}, index=x_old)
    df.index.name = 'Stara_intenzita'

    # NOVY RANGE
    knew = np.arange(rmin, rmax, 1)
    # print('Novy rozsah [{},{})'.format(rmin, rmax))
    # print('Nove intenzity {}'.format(knew))

    # Matica priradeni povodne x -> nove x
    mapping = pd.DataFrame({'Nova_intenzita': np.zeros_like(x_old) - 1}, index=x_old)
    mapping.index.name = 'Stara_intenzita'

    # ak je novy range sirky 1, vsetky pixle na tu hodnotu
    if len(knew) == 1:
        for k in x_old:
            mapping.loc[k] = knew[0]
        return mapping

    if which_method == 'interpolation':

        # Interpolacia stareho intervalu: nove x
        xnew = np.linspace(old_L, old_U - 1, rmax - rmin)
        #print('Nove x {}'.format(xnew))
```

```
44          xnew = xnew[1:-1]  # prva a posledna intenzita su kraje (vratane), tie neinterp
45          xhorne = np.ceil(xnew).astype(int)
46          xdolne = np.floor(xnew).astype(int)
47
48          # Konvexna kombinacia h
49          hhat = []
50          for x, xd, xh in zip(xnew, xdolne, xhorne):
51              h_dolne = df.loc[xd].values
52              h_horne = df.loc[xh].values
53              alpha = x - xd
54              hhat.append(alpha * h_horne + (1 - alpha) * h_dolne)
55
56          # h v lavom kraji pridat na zaciatok
57          hhat.insert(0, sub[0])
58          hhat = np.ravel(hhat)  # na single array
59          #print('Odhad h v novych bodoch: \n {} \n {} \n'.format(xnew, hhat[1:]))
60
61          # s_k
62          N = sum(hhat) + sub[-1]       # celkovy pocet pixlov
63          s = np.cumsum(hhat)[1:] / N
64          s = np.insert(s, 0, 0)        # prva hodnota s je 0
65          s = np.insert(s, len(s), 1)  # posledna hodnota s je 1
66
67          # Stary interval rozdelit na novy pocet
68          bound_new = s * (len(sub) - 1) + old_L
69          #print('s, nove delenie \n {} \n {} \n'.format(np.round(s,3), np.round(bound_new
    ,3)))
70
71          # Naplnenie matice mapping
72          for k in x_old:
73              # najst v ktorom intervale lezi x
74              for idx, b in enumerate(bound_new[:-1]):
75                  if (k >= b) & (k <= bound_new[idx + 1]):
76                      # ku ktoremu kraju je blizsie, tu intenzitu dostane
77                      if np.abs(k - b) < np.abs(k - bound_new[idx + 1]):
78                          mapping.loc[k] = knew[idx]
79                      else:
80                          mapping.loc[k] = knew[idx + 1]
81
82          return mapping
83
84      else:
85
86          s = np.cumsum(sub) / sum(sub)
87          #print('s {}'.format(s))
88
89          # nove intenzity
90          knew = s * (rmax - rmin - 1) + rmin
```

```
91          knew = np.round(knew).astype(int)
92
93          # mapping df
94          mapping['Nova_intenzita'] = knew
95          return mapping
96
97  def find_range(hist, min_num):
98      ''' Najst krajne useky ktore su skoro 0 (aby kraje neboli vzdy 0, 255)
99
100     :param hist: original histogram
101     :param min_num: <=min_num povazujeme za nulovu pocetnost
102     :return: [Mmin, Mmax]: Prva nenulova intenzita pri 0, prva uz nulova pri 255 '''
103
104     almost_zero = np.where(hist <= min_num)[0]    # returns tuple -> [0]
105     print('Tieto intenzity ma menej ako {} pixlov: {}'.format(min_num, almost_zero))
106
107     count_near_0 = 0
108     # prejst intenzity 0 az 255 kym je suvisly usek skoro nulovych, zapamat kde skoncime
109     for i in range(1, 256):
110         idx = 0 + i
111         if np.any(idx in almost_zero):
112             count_near_0 += 1
113         else:
114             break
115
116     count_near_255 = 0
117     # prejst intenzity 255 az po 0 kym je suvisly usek skoro nulovych, zapamatam kde
        skoncime
118     for idx in range(255, -1, -1):
119         if np.any(idx in almost_zero):
120             count_near_255 += 1
121         else:
122             break
123
124     return count_near_0+1, 255 - count_near_255
125
126 def graph_minima(hist, hist_smooth, argmins, Mmin, Mmax, savefig=False, label = 'Graph',
        fname = 'Img', fsize=15):
127     ''' Vykresli histogram, vyhladeny histogram a detekovane lokalne minima
128
129     :param: hist: povodny histogram
130     :param: hist_smooth: vyhladeny histogram
131     :param: argmins: lokalne minima
132     :param: Mmin: minimalna intenzita na obrazku (ma ju >= 100 pixlov)
133     :param: Mmax: maximalna intenzita na obrazku (ma ju >= 100 pixlov)
134     :param: savefig: bool, ak True graf je ulozeny ako png s DPI=300. Default False
135     :param: label: nazov grafu
136     :param: fsize: font size for graph labels. Default 13
```

```
137     :return: None '''
138
139     mins = hist_smooth[argmins]
140
141     plt.figure(figsize=(11, 5))
142     plt.fill_between(x=np.linspace(0, 255, 256), y1=hist, step='pre', zorder=1, label='
        Histogram H(k)')
143     plt.plot(hist_smooth, zorder=2, c='k', label='H(k) po zhladzovacom filtri')
144     plt.scatter(argmins, mins, c='k', s=60, marker='x', linewidth=2, zorder=3, label='
        Detekované lokálne minimá')
145     plt.tick_params(axis='both', which='major', labelsize=fsize)
146
147     # hranice Mmin, Mmax
148     plt.axvline(x=Mmin, linestyle='--', ymin=0.05, ymax=0.95, c='k', alpha=0.8)
149     plt.axvline(x=Mmax, linestyle='--', ymin=0.05, ymax=0.95, c='k', alpha=0.8)
150
151     plt.xlabel('Intenzita k', fontsize=fsize); plt.ylabel('Početnosť', fontsize=fsize)
152     plt.legend(fontsize=fsize)
153     if savefig == True:
154         plt.savefig('{} {}.png'.format(fname, label), bbox_inches='tight', dpi=300)
155
156 def DHE_main(img, hist, x = 0, method = 'interpolation', savehist=False, img_name = 'Img'
        , fsize = 15):
157     ''' Hlavna funkcia pre ekvalizaciu histogramu. Vola find_range(), grahp_minima(), HE
        ()
158
159     :param img: povodny obrazok
160     :param hist: histogram povodneho obrazku
161     :param method: simple: ako v clanku DEH, interpolation: interpolovat, rovnomernejsi
        histogram
162     :param x: level zvysenia kontrastu
163     :return: obrazok so zvysenym kontrastom
164     '''
165     # ---------- 1) SEGMENTACIA HISTOGRAMU ---------- #
166
167     # a) zhladenie histogramu povodneho obrazku 1x7 filtrom
168     gaussf = cv2.getGaussianKernel(ksize=7, sigma=3).flatten()
169     hist_smooth = ndi.convolve(hist, gaussf)
170
171     # Mmin a Mmax: hranicne intenzity na obrazku
172     # intenzity ktorych je <= min_num pixlov povazujeme za nulovy pocet
173     Mmin, Mmax = find_range(hist, min_num=100)  # ak je ich do 100 -> akoby ich bolo 0
174     print('Prva nenulova intenzita pri 0: Mmin = {}'.format(Mmin))
175     print('Prva uz nulova intenzita pri 255: Mmax = {}'.format(Mmax))
176
177     # b) lokalne minima
178     search_range = hist_smooth[Mmin:Mmax]    # detekcia minim medzi [X_min, X_max)
179     argmins = argrelmin(search_range, axis=0, order=1)[0]    # order = 2
```

```
180     argmins = argmins+Mmin          # posunutie na interval [Mmin+1:Mmax)
181
182     # pridat M_min, M_max
183     argmins = np.insert(argmins, 0, Mmin)
184     argmins = np.insert(argmins, argmins.shape[0], Mmax)
185
186     print('Minima su v bodoch (vratane Mmin, Mmax) {}'.format(argmins))
187     graph_minima(hist, hist_smooth, argmins, Mmin, Mmax, savefig=savehist, label='Povodny
        histogram', fname=img_name+' 1', fsize=fsize)
188
189     # c) kontrola variancie na subhistogramoch a dodatocne delenie
190     iter = 0
191
192     allgood = False
193     while allgood == False:
194
195         iter += 1
196         new_argmins = []
197
198         for i in range(len(argmins) - 1):
199
200             X = np.arange(argmins[i], argmins[i + 1])  # grey levels v subhistograme
201             subhist = hist[X]                          # subhistogram medzi dvomi
    minimami
202             N = np.sum(subhist)                        # pocet pixlov ktore maju tieto
    gray levels
203
204             # priemer a stand. odchylka
205             mi = np.sum(subhist*X)/N
206             std = np.sqrt(np.sum(subhist*np.square(X-mi))/N)
207             muplus = np.ceil(mi+std).astype(int)
208             muminus = np.floor(mi-std).astype(int)
209
210             # napocitanie pixlov s intenzitami v [mi-std, mi+std], X posunute aby to boli
        indexy subhist
211             n_medzi = np.sum(hist[muminus:muplus+1])
212
213             perc = n_medzi/N    # podiel pixlov medzi mi-std, mi+std
214             # intervaly kratsie ako 5 nedelime
215             if perc < 0.683:
216                 print('Interval {}: medzi mu-std, mu+std: {}% < 68.3%'.format(i, np.round
    (perc * 100, 1)))
217                 print('Interval {} mu-std, mu+std {} {}'.format(i, muminus, muplus))
218                 if muminus >= argmins[i]:
219                     new_argmins.append(muminus)
220                 if muplus < argmins[i+1]:
221                     new_argmins.append(muplus)
222
```

```
223         print('Dodatocne delenie najdene v iteracii {}: {}'.format(iter, new_argmins))
224         argmins = np.append(argmins, new_argmins)
225         argmins = np.unique(argmins).astype(int)
226 
227         if len(new_argmins) == 0:
228             allgood = True
229 
230     print('Finalne delenia: {}'.format(argmins))
231     graph_minima(hist, hist_smooth, argmins, Mmin, Mmax, savefig=savehist, label='Finalne
        delenie', fname=img_name+' 2',fsize=fsize)
232 
233     # ---------- 2) LOKACIA NOVEJ SKALY ---------- #
234 
235     # povodne D_i (spans)
236     spans = np.diff(argmins)
237     print('Povodne D_i {}'.format(spans))
238 
239     # stare hranice subhistogramov
240     argmins[-1] += 1  # aby bol posledny interval [,Mmax)
241     print('Hranice starych subhistogramov (lavy kraj patri, pravy nepatri) {} \n'.format(
        argmins))
242 
243     # a) CDF na histogramoch, intervaly [,)
244     F = [np.sum(hist[argmins[i]:argmins[i + 1]]) for i in range(len(argmins)-1)]
245     F = np.asarray(F)
246 
247     # factors a nove D_i (ranges)
248     factor = spans * np.power(np.log(F), x)
249     ranges = factor / np.sum(factor) * 256
250     print('Nove D_i (zaokruhlene) {}'.format(np.round(ranges).astype(int)))
251 
252     # b) nove hranice R_i
253     bord_new = [sum(ranges[:i]) for i in range(len(ranges) + 1)]
254     bord_new = np.round(bord_new).astype(int)  # brat ako intervaly [,)
255     print('Hranice novych subhistogramov (lavy kraj patri, pravy nepatri) {} \n'.format(
        bord_new))
256 
257 
258     # ---------- 3) EKVALIZACIA NA SUBHISTOGRAMOCH ---------- #
259     # kazdemu subhistogramu dovolime rozsirit/zuzit len na nove R_i
260 
261     DHE = img.copy()  # aby to neprepisalo povodny
262     mapping_all = []       # nove intenzity
263 
264     if method == 'simple':
265         print('Jednoducha metoda')
266     elif method == 'interpolation':
267         print('Metoda s interpolaciou')
```

```
268     else:
269         print('Zadaj method=\'simple\' alebo method=\'interpolation\'')
270         return None
271 
272     # HE na kazdom subhistograme
273     for i in range(len(argmins)-1):
274         sub = hist[argmins[i]:argmins[i + 1]]
275         if bord_new[i+1] > bord_new[i]:    # ak bol priradeny usek dlzky 0 -> pass
276             mapping_i = HE(sub, argmins[i], argmins[i+1], rmin=bord_new[i], rmax=bord_new
    [i + 1], which_method=method)
277             mapping_all.append(mapping_i)
278 
279     #print('\n Mapovanie posledneho intervalu: \n {}'.format(mapping_all[-1]))
280 
281     # Mapovanie na novu intenzitu
282     for mappi in mapping_all:
283         for k in mappi.index:
284             mask = (img == k)
285             DHE[mask] = mappi.loc[k]
286 
287     # Novy histogram
288     hist_new = cv2.calcHist([DHE], [0], None, [256], [0, 256]).flatten().astype(int)
289     bord_new[-1] -= 1
290     plt.figure(figsize=(11, 5))
291     plt.fill_between(x=np.linspace(0, 255, 256), y1=hist_new, step='pre', label='
    Histogram po ekvalizácii')
292     plt.scatter(bord_new, np.ones_like(bord_new)+2, c='#000000', s=45, marker='|',
    linewidth=2, label='Nové rozsahy intenzít')
293     plt.tick_params(axis='both', which='major', labelsize=fsize)
294     plt.xlabel('Intenzita k', fontsize=fsize); plt.ylabel('Početnosť', fontsize=fsize)
295     plt.legend(fontsize=fsize, loc=2)
296     if savehist == True:
297         plt.savefig('{} 3 New histogram, method {}.png'.format(img_name, method),
    bbox_inches='tight', dpi=300)
298 
299     return DHE
```

Listing 1: Súbor *functions.py*. Funkcie sa importujú a volajú hlavným súborom *main.py*

```
1 # Import kniznic a funkcii
2 import cv2
3 from my_functions import DHE_main
4 
5 # Import obrazku
6 fname = 'cat.jpg' #'Img_name.jpg'   # sem treba dat meno vstupneho obrazku, musi byt v
      rovnakom priecinku ako main.py
7 img = cv2.imread(fname, 0)      # 0 = nacitat ciernobielo
8 
```

```
9  # Ulozit original ako ciernobiely
10 cv2.imwrite(fname + '_origGS.jpg', img)
11
12 # Histogram povodneho obrazku
13 hist = cv2.calcHist([img], [0], None, [256], [0, 256]).flatten().astype(int)
14
15 # Volba parametra x
16 x = 3
17
18 # Jednoducha ekvalizacia
19 imgDHE_simple = DHE_main(img, hist, x, method = 'simple', savehist=True)
20 cv2.imwrite(fname + ' DHE simple, x = {}.jpg'.format(x),imgDHE_simple)  # ukladanie
       vystupneho obrazku
21
22 # Ekvalizacia s interpolaciou
23 imgDHE_interp = DHE_main(img, hist, x, method = 'interpolation', savehist = True)
24 cv2.imwrite(fname + ' DHE interp, x = {}.jpg'.format(x),imgDHE_interp)  # ukladanie
       vystupneho obrazku
```

Listing 2: Súbor *main.py*