VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MATEMATIKY
INSTITUTE OF MATHEMATICS

SROVNÁNÍ HRANOVÝCH DETEKTORŮ
THE COMPARISON OF EDGE DETECTORS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

Marek Dula

VEDOUCÍ PRÁCE
SUPERVISOR

Mgr. Jana Procházková, Ph.D.

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadaní bakalářské práce

Ústav:	Ústav matematiky
Student:	**Marek Dula**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Matematické inženýrství
Vedoucí práce:	**Mgr. Jana Procházková, Ph.D.**
Akademický rok:	2020/21

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Srovnání hranových detektorů

Stručná charakteristika problematiky úkolu:

Hledání hran v obraze patří do oblasti počítačového vidění. Jedná se o důležitý krok pro identifikaci objektů v obraze nebo určování hlavních směrů pro určení geometrie obrazu. Existuje více druhů detektorů (Sobel, Prewitt, Canny), které využívají různé typy změn v obraze, nejčastěji popsané pomocí první nebo druhé derivace.
Úkolem v bakalářské práci bude programové zpracování několika vybraných hranových detektorů a následné porovnání výsledků na množině obrazů. Student se také seznámí se šumem a algoritmy k jeho odstranění z obrazu. V práci vyhodnotí výsledky detektorů pro různě silně zašumělé obrazy.

Cíle bakalářské práce:

1. Studium problematiky hranových detektorů.
2. Studium filtrů pro odstranění šumu v obrazech.
3. Programové zpracování různých filtrů s nebo bez použití odstranění šumu.
4. Srovnání výsledné detekce na sadě různých obrazů.

Seznam doporučené literatury:

GONZALEZ, Rafael C. a Richard E. WOODS. Digital image processing. 3rd ed. Upper Saddle River, N.J.: Prentice Hall, c2008. ISBN 978-0-13-168728-8.

ZIOU, Djemel a Salvatore TABBONE. Edge detection techniques - an overview. Pattern Recognition and Image Analysis: Advances in Mathematical Theory and Applications. 8(4), 537-559.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2020/21

V Brně, dne

L. S.

prof. RNDr. Josef Šlapal, CSc. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

Abstrakt
V této bakalářské práci se zaměřujeme na porovnání různých metod pro nalezení hran v obraze a filtraci šumu v obraze. V úvodu se věnujeme zavedení základních pojmů souvisejících s danou problematikou. Následně popisujeme jednotlivé metody detekování hran a filtrování šumu v obrazu. Další část obsahuje programové zpracování jednotlivých metod v softwaru Matlab. Na závěr srovnáváme jednotlivé filtry šumu a dále i hranové detektory.

Summary
In this bachelor thesis we focus on the comparison of different methods for finding edges and filtering noise in the image. In the introduction, we focus on basic concepts related to the issue. Subsequently, we describe the individual methods of edge detection and noise filtering in the picture. The next part contains software processing of individual methods in Matlab software. Finally, we compare the individual noise filters and edge detectors.

Klíčová slova
Hranové detektory, Filtry šumu, Sobelův operátor, Cannyho Hranový detektor, Otsu metoda, operátor Prewittové, Robertsův operátor a metoda, Gaussův filtr, Filtrace šumu Fourierovou transformací

Keywords
Edge detectors, Noise filters, Sobel operator, Canny Edge detector, Otsu method, Prewitt operator, Roberts operator and method, Gaussian filter, Fourier transform noise filtering

DULA, M.*Srovnání hranových detektorů.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2021. 34 s. Vedoucí Mgr. Jana Procházková, Ph.D.

Prohlašuji, že jsem tuto bakalářskou práci na téma *Srovnání hranových detektorů* vypracoval samostatně, a to z uvedených zdrojů.

Marek Dula

Děkuji mé vedoucí Mgr. Janě Procházkové, Ph.D. za pomoc a rady, bez ní by tato práce nemohla vzniknout. Zároveň bych chtěl poděkovat i Bc. Martinovi Buriánkovi za pomoc. Ale i všem ostatním, kteří mně jakkoliv pomohli.

Marek Dula

Obsah

Úvod

V první kapitole bakalářské práce se zaměříme na zavedení pojmů, které jsou pro problematiku hranových detektorů důležité, a budou se dále v práci vyskytovat. Ve druhé kapitole se podíváme na vybrané způsoby detekování hran v obraze. Jedná se konkrétně o metodu Robertsova operátoru, Sobelova operátoru, operátoru Prewittové, Cannyho hranový detektor a Otsu metodu. Vysvětlíme si jejich princip a ukážeme aplikaci těchto detektorů. Následně se ve třetí kapitole zaměříme i na jednotlivé metody pro odstranění šumu, ukážeme jejich aplikaci na obraze a jak fungují. Ve čtvrté části programově zpracujeme dříve popisované metody.

V páté kapitole se blíže podíváme na filtrace šumu Fourierovou transformací a určíme si optimální nastavení tohoto filtru. Následně porovnáme nalezené optimální nastavení filtru s Gaussovým filtrem a průměrováním. Filtraci šumu budeme porovnávat s ohledem na následné zpracovávání obrazu hranovým detektorem.

V sedmé kapitole porovnáme hranové detektory pro různé obrazy. Budeme sledovat především přesnost jejich detekce a čas výpočtu. Společně s šestou kapitolou se jedná o hlavní části této práce a snažíme se v nich objektivně zhodnotit a porovnat jednotlivé metody.

1. Základní pojmy

1.1. Základní pojmy

V této kapitole si zadefinujeme základní pojmy, které budeme dále využívat. Toto téma je zpracované například v literatuře [1], [2] a [10].

Obraz je diskrétní funkce $f(x, y)$, kde x a y představuje umístění vůči zvolenému referenčnímu bodu a $f(x, y)$ jas v daném místě obrazu. Funkci $f(x, y)$ nazýváme obrazovou funkcí.

Barevný obraz má funkci $f(x, y)$ definovanou pomocí tří hodnot, kde jednotlivé hodnoty představují intenzitu základních barev, tj. červené, zelené a modré. Barevný obraz lze převést na černobílý například zprůměrováním těchto hodnot, nebo pomocí vzorce využívajícího různé váhy daných hodnot.

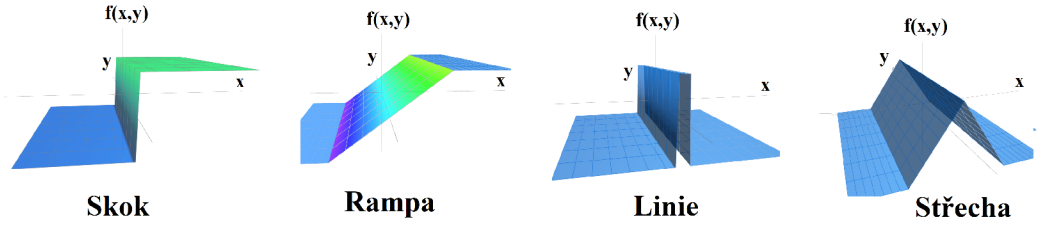
Černobílý obraz má funkci $f(x, y)$ definovanou jednou hodnotou. Tato hodnota nám udává intenzitu.

Pixel je prvek obrazové funkce. Jeho pozice je pevně určena hodnotami x a y.

Okolní pixely jsou pixely, jejichž pozice x a y není odlišná o více než jedna, pokud neuvádíme jinak, od daného pixelu. Jednotkové okolí pixelu s funkční hodnotou $f(x, y)$ můžeme vidět na obrázku 2.1.

Hraniční pixel je pixel, v jehož okolí dochází k velké změně hodnot obrazové funkce, neboli intenzity.

Hrana je množinou sousedních hraničních pixelů, hrana nám rozděluje oblasti v obrazu s podstatně rozdílnou intenzitou, neboli hodnotou obrazové funkce.
Základní profily obrazové funkce v okolí hrany dle [8] lze vidět na obrázku 1.1. Typ hrany Skok odpovídá okamžitému přechodu mezi různými stupni intenzity jasu. Typ hrany Rampa odpovídá postupné změně intenzity mezi různými stupni intenzity jasu. Typ hrany Linie je skoková změna na jinou intenzitu jasu následovaná skokovou změnou zpátky na původní intenzitu jasu. Typ hrany Střecha odpovídá postupné změně intenzity a následně postupné změně intenzity zpět na původní hodnotu.



Obrázek 1.1: Základní typy hran

Hranový detektor je výpočetní postup určený k detekci hranových pixelů, a případně i dále upravuje tuto množinu dle daných požadavků. Výstupem z hranového detektoru je pozice hran v obraze.

Definice 1.1 (Konvoluce)**.** *Konvoluce je obecně operace dvou funkcí, označujeme ji* $*$ *a je definována dle [1] jako:*

$$f(t) * h(t) = \int_{-\infty}^{\infty} f(\tau)h(t-\tau)d\tau \tag{1.1}$$

V této práci budeme uvažovat dvoudimenzionální konvoluci diskrétní obrazové funkce $f(x, y)$ a konvoluční masky $h(x, y)$, definovanou jako:

$$f(x, y) * h(x, y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} f(x-i, y-j)h(i, j) \tag{1.2}$$

Padding je rozšíření obrazové funkce $f(x, y)$, obvykle nulami. Padding se používá při konvoluci, abychom mohli zachovat stejnou velikost oblasti obrazové funkce před a po konvoluci. Oblast obrazové funkce $f(x, y)$ rozšíříme v každém směru o k hodnot z konvoluční masky. Funkční hodnotu $f(x, y)$ v rozšířené oblasti položíme rovnu nule.

Definice 1.2 (Gradient)**.** *Gradient je vektor směrových derivací obrazové funkce. Je vázán na příslušný pixel a je definován vztahem:*

$$G[f(x, y)] = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \\ \dfrac{\partial f}{\partial y} \end{bmatrix} \tag{1.3}$$

Dále u gradientu definujeme jeho velikost, a to jako:

$$G = \sqrt{G_x^2 + G_y^2} \tag{1.4}$$

Práh je mezní hodnota gradientu. Rozděluje nám hodnoty na dvě části. Slouží k určení hodnot gradientu, které můžeme považovat za hranu. Metody využívající práh jsou: Robertsův operátor, Sobelův operátor, operátor Prewittové a Cannyho hranový detektor.

2. Základní hranové detektory

V této části se podíváme na základní metody, které se používají k nalezení hraničních pixelů a hran. První z nich jsou založeny na výpočtech využívajících okolní pixely a derivace. Konkrétně se jedná o metody: Robertsův operátor, Sobelův operátor, operátor Prewittové a Cannyho hranový detektor. Následně se podíváme na Otsu metodu, která využívá jiný princip, a to statistickou analýzu četnosti v obraze.

2.1. Robertsův operátor a metoda

Robertův operátor byl publikovaný Lawrence Robertsnem v knize Machine Perception of Three-Dimensional Solids v roce 1963 [3] (Někdy bývá uvedeno 1965).

Robertsova metoda využívá k detekci hran transformovanou hranovou funkci $F(x, y)$ definovanou vztahem:

$$F(x, y) = \sqrt{f(x, y)} \tag{2.1}$$

Na transformovanou hranovou funkci následně použijeme Robertsův operátor, který je zaveden jako:

$$M(x, y) = \sqrt{(F(x, y) - F(x + 1, y + 1))^2 + (F(x + 1, y) - F(x, y + 1))^2} \tag{2.2}$$

Robertsův operátor slouží pro výpočet velikosti gradientu. Můžeme jej převést dle obrázku 2.1 na konvoluci, kde nám bude definovat dvě následující konvoluční masky:

$$\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \tag{2.3}$$

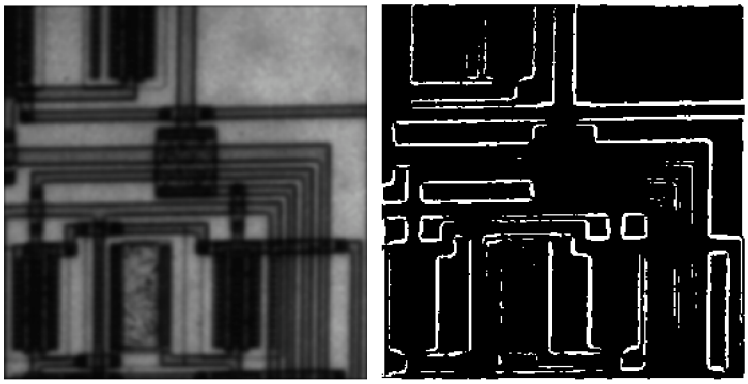
f(x-1,y+1)	f(x,y+1)	f(x+1,y+1)
f(x,y-1)	f(x,y)	f(x+1,y)
f(x-1,y-1)	f(x,y-1)	f(x+1,y-1)

Obrázek 2.1: Matice indexů

Tyto dvě masky můžeme považovat za směrové derivace G_x a G_y a následně z nich určit i směr gradientu, ovšem v této metodě směr gradientu nepotřebujeme. Roberts následně popisuje ve své knize Machine Perception of Three-Dimensional Solids [3] postup, jak pokračovat při hledání hran.

Rozdělíme si obrazovou funkci do menších čtverců 4x4 sousedních pixelů. Následně si zavedeme práh jako 1/10 největší hodnoty gradientu. V každém z menších celků nalezneme největší hodnotu gradientu a v případě, že je nalezená hodnota větší, nežli zavedený práh, tak s ní dále počítáme. Těmito nalezenými pixely následně proložíme čtyři linie pěti různých délek se sklony 0, 1, ∞, -1 a spočítáme korelaci s okolními body. Pokud je korelace větší než druhý práh, jehož velikost volíme okolo 3, je daný pixel a nejlepší směr zapsán. Hodnota 3 vychází z empirických vztahů. Zapsané pixely následně propojujeme tak, že pokud jsou pixely ze sousedních čtverců a směr propojení se neliší od zapsaného směru alespoň jednoho z pixelů o více než 23°, tak jsou propojeny. Pixely, které tímto způsobem nepropojíme s žádným jiným pixelem, odstraníme. V případě, že máme 3 pixely navzájem propojené do trojúhelníku, nejdelší stranu odstraníme. Dále pokud máme 4 pixely spolu propojeny do smyčky, tak nejbližší nesousedící pixely sloučíme do jednoho pixelu. Na závěr odstraníme všechny pixely, které jsou na jedné straně nepropojeny a na druhé propojeny k více než jedné hraně.

Na obrázku 2.2 je ukázka výstupu této metody pro obrázek zvětšeného elektrického obvodu.



(a) Původní obraz *[Knihovna softwaru Matlab]* (b) Detekované hrany

Obrázek 2.2: Aplikace Robertsova operátoru

2.2. Sobelův operátor

Sobelův operátor byl poprvé prezentovaný Irwinem Sobelem a Gary Feldmanem na Stanford Artificial Intelligence Laboratory v roce 1968. [9]

Sobelův operátor využívá velikost první derivaci obrazové funkce, neboli gradientu, kde jednotlivé směrové derivace definujeme jako:

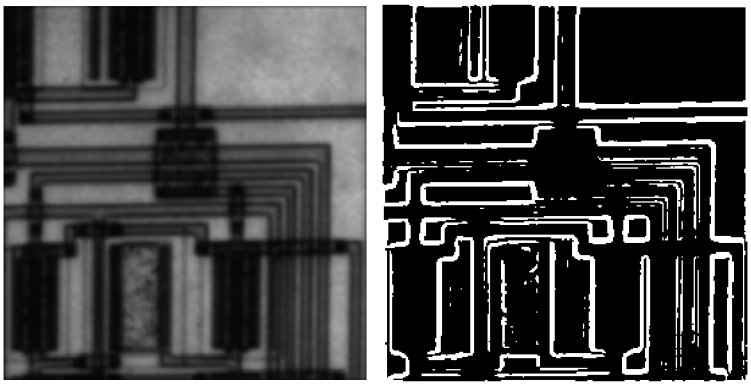
$$G_x = f(x+1,y-1)+2f(x+1,y)+f(x+1,y+1)-(f(x-1,y-1)+2f(x-1,y)+f(x-1,y+1)) \tag{2.4}$$

$$G_y = f(x+1,y-1)+2f(x,y-1)+f(x-1,y-1)-(f(x+1,y+1)+2f(x,y+1)+f(x-1,y+1)) \tag{2.5}$$

Operátory pro výpočet gradientu můžeme počítat jako konvoluci s použitím konvolučních masek, které získáme dle schématu na obrázku 2.1 jako:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{2.6}$$

Za hranu následně považujeme všechny hraniční pixely, jejichž velikost gradientu je větší, nežli předepsaný práh. Výstup této metody můžeme sledovat na obrázku 2.3.



(a) Původní obraz (b) Detekované hrany

Obrázek 2.3: Hrany dle Sobelova operátoru

2.3. Operátor Prewittové

Operátor Prewittové byl prvně publikovaný J. M. S. Prewitt v knize Picture Processing and Psychopictorics v roce 1970. [6]

Funguje na podobném principu jako Sobelův operátor, avšak neklade důraz na střed. Využívá velikost první derivaci obrazové funkce, neboli gradientu, kde jednotlivé směrové derivace definujeme jako:

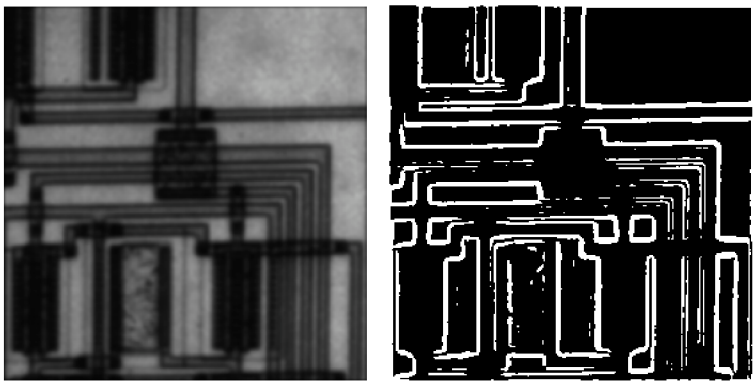
$$G_x = (f(x+1, y-1)+f(x+1, y)+f(x+1, y+1))-(f(x-1, y-1)+f(x-1, y)+f(x-1, y+1)) \tag{2.7}$$

$$G_y = (f(x+1, y-1)+f(x, y-1)+f(x-1, y-1))-(f(x+1, y+1)+f(x, y+1)+f(x-1, y+1)) \tag{2.8}$$

Operátory pro výpočet gradientu můžeme počítat jako konvoluci s použitím konvolučních masek:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad G_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \tag{2.9}$$

Za hranu následně považujeme všechny hraniční pixely, jejichž velikost gradientu je větší, nežli předepsaný práh. Výstup této metody můžeme pozorovat na obrázku 2.4.



(a) Původní obraz (b) Detekované hrany

Obrázek 2.4: Hrany dle operátoru Prewittové

2.4. Cannyho hranový detektor

Cannyho hranový detektor poprvé publikoval John Canny v publikaci Finding Edges and Lines in Images z roku 1983. Podrobněji ji následně popisuje v knize A Computational Approach To Edge Detection [4]

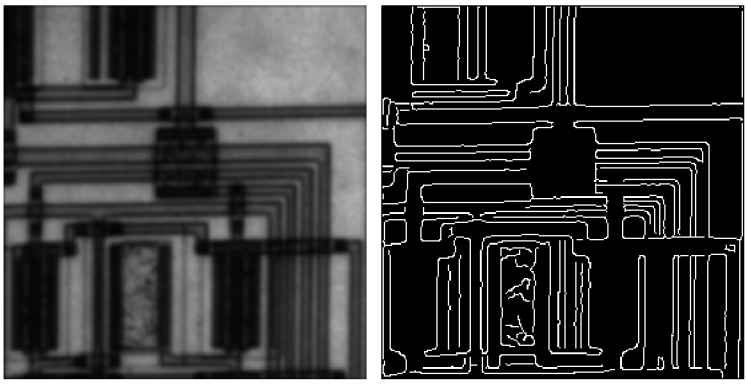
Tato metoda klade důraz na to, aby měla vysokou úspěšnost v hledání všech hran, aby se všechny nalezené hrany co nejvíce shodovaly se skutečnou pozicí hran a aby výstupem k jedné skutečné hraně byla jenom jedna hrana.

Tato metoda využívá pro odstranění šumu z obrazu Gaussův filtr, který je blíže popsaný v části 3.1 tohoto textu. Poté z upraveného obrazu určí gradient. Pro další postup určíme směr gradientu určený vztahem (2.10).

$$\theta = tan^{-1}\left[\frac{G_y}{G_x}\right] \tag{2.10}$$

Po vypočítání směru gradientu budou potlačeny ty hodnoty, které nejsou vůči okolním bodům v normálovém směru gradientu největší. Směr gradientu dělíme do 4 skupin, kde skupiny reprezentují dvojici protilehlých sousedních pixelů.

Pro určení hodnot, které zůstaly dosud neurčené, využíváme tzv. prahování. Nastavíme velikosti horního a dolního prahu σ_1 a σ_2, které jsou obvykle ve vztahu 1:2 až 1:3. Všechny hodnoty nad vyšším prahem jsou přijaty, a všechny hodnoty pod menším prahem jsou zamítnuty. O zbylých hodnotách se rozhoduje v závislosti na tom, jestli množina sousedních bodů sousedí s již přijatou hranu. Pokud alespoň jeden z množiny sousedních bodů sousedí s již přijatou hodnotou, je celá množina přijata za hranu. V opačném případě zamítáme celou množinu.
Aplikaci metody můžeme vidět na obrázku 2.5.



(a) Původní obraz (b) Detekované hrany

Obrázek 2.5: Aplikace Cannyho detektoru

2.5. Otsu metoda

Otsu metoda byla poprvé publikovaná Nobuyuki Otsu v knize "A threshold selection method from gray-level histograms"z roku 1983. [5]

Tato metoda používá pro nalezení hran statistiku, nikoliv derivaci obrazu. Vychází z normalizovaného histogramu obrazu, který definujeme jako:

$$p_i = \frac{n_i}{N} \tag{2.11}$$

Počet výskytů n_i pixelu dané intenzity i dělíme celkovým počtem pixelů v obraze N. V uvedených výpočtech dále uvažujeme, že hodnota jasu nabývá hodnot od 1 do L.

Následně se snažíme najít mezní hodnotu jasu k, která nám rozdělí obraz na dvě množiny. Množina C_0 nám bude reprezentovat pozadí a množina C_1 objekt. Rozdělení probíhá tak, abychom maximalizovali rozptyl mezi jednotlivými třídami.
Pravděpodobnost, že se pixel nachází v jednotlivých množinách, definujeme dle [5] jako:

$$\omega_0 = \sum_{i=1}^{k} p_i, \tag{2.12}$$

$$\omega_1 = \sum_{i=k+1}^{L} p_i = 1 - \omega_0. \tag{2.13}$$

Střední hodnotu z jednotlivých množin a celého obrazu dále definujeme jako:

$$\mu_0 = \sum_{i=1}^{k} \frac{ip_i}{\omega_0} \tag{2.14}$$

$$\mu_1 = \sum_{i=k+1}^{L} \frac{ip_i}{\omega_1} \tag{2.15}$$

$$\mu_T = \sum_{i=1}^{L} ip_i \tag{2.16}$$

Rozptyl v jednotlivých množinách je definovaný jako:

$$\sigma_0^2 = \sum_{i=1}^{k} \frac{(i-\mu_0)p_i}{\omega_0} \tag{2.17}$$

$$\sigma_1^2 = \sum_{i=k+1}^{255} \frac{(i-\mu_1)p_i}{\omega_1} \tag{2.18}$$

Za pomoci statistické analýzy můžeme určit kritéria pro vhodnost daného k. Nalezení optimálního k je potom optimalizační úloha na nalezení maxima těchto kritérií:

$$\lambda = \frac{\sigma_B^2}{\sigma_W^2} \qquad \kappa = \frac{\sigma_T^2}{\sigma_W^2} \qquad \eta = \frac{\sigma_B^2}{\sigma_T^2} \tag{2.19}$$

kde σ_W^2, σ_B^2 a σ_T^2 definujeme vztahy:

$$\sigma_W^2 = \omega_0\sigma_0^2 + \omega_1\sigma_1^2 \tag{2.20}$$

$$\sigma_B^2 = \omega_0\omega_1(\mu_1 - \mu_0) \tag{2.21}$$

$$\sigma_T^2 = \sum_{i=1}^{L}(i-\mu_T)^2 p_i \tag{2.22}$$

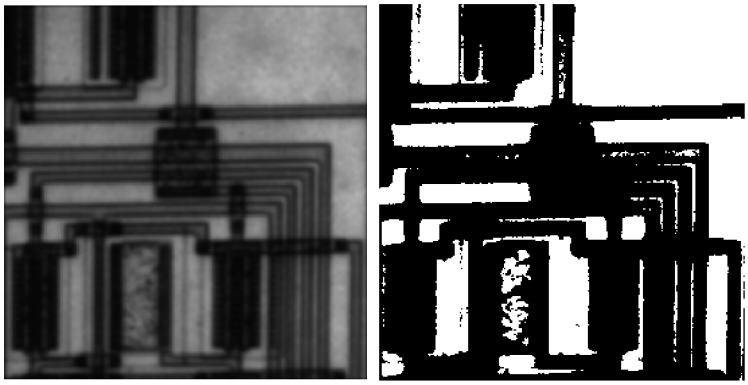
Jednotlivé kritéria jsou vzájemně ekvivalentní a σ_T^2 je nezávislá na zvoleném k. Z toho důvodu k nalezení optimálního k volíme maximalizaci σ_B^2, což je ekvivalentní s maximalizací η.

$$\sigma_B^2(k) = \frac{[\mu v_T \omega(k) - \mu(k)]^2}{\mu(k)[1-\omega(k)]} \tag{2.23}$$

A optimální $k*$ poté nalezneme jako

$$\sigma_B^2(k*) = max(\sigma_B^2(k)) \tag{2.24}$$

Ukázku použití Otsu metody můžeme sledovat na obrázku 2.6.



(a) Původní obraz (b) Detekované hrany

Obrázek 2.6: Aplikace Otsu metody

3. Filtry pro odstranění šumu

V této části se zaměříme na filtry používané k odstranění šumu v obrazu za účelem lepší detekce hran v daném obrazu.

Šum je vada obrazu, ke které mohlo dojít při pořizování obrazu, při jeho přenosu, či případném ukládání. Šum lze modelovat za pomoci pravděpodobnostní funkce. Rozdělení šumu lze nalézt např. v [1]. Máme dva hlavní typy šumu, kterými se budeme dále zabývat, a to Gaussův šum a impulzní šum.

Gaussův šum je šum, jehož vlivem dochází ke změně intenzity všech pixelů a lze jej charakterizovat dle Gaussovy funkce, definované rovnicí:

$$p(z) = \frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-(z-\mu)}{2\sigma^2}} \tag{3.1}$$

kde z reprezentuje intenzitu, μ značí střední hodnotu intenzity a σ směrodatnou odchylku. Druhou mocninu směrodatné odchylky σ^2 nazýváme rozptyl.

Impulzní šum, neboli "šum sůl a pepř"je šum, kde dochází ke změně intenzity nezávisle na původní hodnotě intenzity u náhodných pixelů. Tímto šumem bývají postihnuty pouze jednotlivé pixely, nikoli však jejich okolí.

3.1. Gaussův filtr

Gaussův filtr [1] vychází z normálního (neboli Gaussova) rozdělení. Za pomoci konvoluční masky B definované jako:

$$B(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2 y^2}{2\sigma^2}} \tag{3.2}$$

kde se σ^2 udává rozptyl normálního rozdělení a volí se dle požadované síly filtru. S větším σ^2 Gaussův filtr filtruje šum více, ale dochází k většímu rozmazání obrazu. Proměnné x, y udávají vzdálenost od středu konvoluční masky v daných osách. Pro vytvoření konvoluční masky filtru vytvoříme čtvercovou jednotkovou síť, následně za x a y postupně dosazujeme souřadnice pixelu v okolí středu, a dosazujeme na jednotlivé místa v konvoluční masce odpovídající hodnoty. Následně musíme konvoluční masku normalizovat, tak aby součet všech prvků v matici byl roven jedné.

Příklady masek pro různě zvolené σ^2:

$$B(\sigma^2 = 1) = \frac{1}{273} \begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

$$B(\sigma^2 = 1.35) = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

$$B(\sigma^2 = 3) = \frac{1}{450}\begin{bmatrix} 14 & 17 & 18 & 17 & 14 \\ 17 & 20 & 21 & 20 & 17 \\ 18 & 21 & 22 & 21 & 18 \\ 17 & 20 & 21 & 20 & 17 \\ 14 & 17 & 18 & 17 & 14 \end{bmatrix}$$

Následnou konvolucí obrazu s maskou Gaussova filtru dochází k odstranění šumu.

Ukázka filtru pro σ^2 zvolené jako 1, 1.35 a 3 lze vidět na obrázku 3.1.

(a) Testovací Obraz *[Knihovna softwaru Matlab]*

(b) $\sigma^2 = 1$

(c) $\sigma^2 = 1.35$

(d) $\sigma^2 = 3$

Obrázek 3.1: Porovnání Gaussova filtru pro různé σ^2

3.2. Filtrace šumu Fourierovou transformací

Tato metoda je založena na Fourierově transformaci. Pracujeme s diskrétním obrazem a převedeme jej do frekvenčního spektra. Šum se nám po této transformaci projeví ve vysokých frekvencích, zatímco hrany v nízkých frekvencích. Odstraníme vysoké frekvence, které jsou typické pro šum, a provedeme zpětnou transformaci. S více odebranými vysokými frekvencemi je filtrace šumu větší, ale dochází i k většímu odstranění detailů v obrazu. Diskrétní Fourierova transformace $\mathcal{F}$, na které je tato metoda založena, je blíže popsána například v knize [7], kde je i podrobně matematicky probrána. Definována je jako:

$$\mathcal{F}(l_1, l_2) = \frac{1}{XY} \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} f(x, y) e^{(-2\pi i(\frac{xl_1}{X} + \frac{yl_2}{Y}))}, \tag{3.3}$$

kde hodnoty X a Y nám udávají velikost obrazu.

Z obrazu získaného za pomoci $\mathcal{F}$ následně odstraníme část obrazu odpovídající vysokým frekvencím, které jsou charakteristické pro šum. Odstraníme je tím, že dané hodnoty nastavíme jako nulové. Následně je potřeba převést obraz zpět. Toho docílíme za pomoci zpětné diskrétní Fourierovy transformace.

$$f(x, y) = \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} \mathcal{F}(l_1, l_2) e^{(2\pi i(\frac{xl_1}{X} + \frac{yl_2}{Y}))} \tag{3.4}$$

Takto získáváme původní obraz s odstraněným šumem.

Ukázku filtru pro různá množství ponechaných hodnot $\mathcal{F}$ můžeme vidět na obraze 3.2. Hodnota v udává, jak velká část Fourierovy transformace $\mathcal{F}$ bude ponechána. Přesněji můžeme proměnnou v vidět v části 4.2.

(a) Testovací Obraz (b) v = 0,3

(c) v = 0,5 (d) v = 0,7

Obrázek 3.2: Porovnání Fourierovy filtrace pro různé v

3.3. Průměrování

V případě průměrování vezmeme novou hodnotu obrazové funkce $f(x, y)$ jako průměr hodnot $f(x, y)$ a okolních hodnot. Jedná se tedy o konvoluci obrazové funkce s maticí

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix} \tag{3.5}$$

Průměrování je zvláštním případem Gaussova filtru, kdy jej uvažujeme na oblasti 3x3 a $\sigma^2 = \infty$. Podrobné zpracování této metody nalezneme například v [1].

4. Programové zpracování

V této části se zabýváme programovým zpracováním jednotlivých metod v softwaru Matlab. Většina uvedených metod má i přímou implementaci v Matlabu, avšak my si ukážeme naprogramování těchto metod bez jejich použití. Nejdříve se zaměříme na programové zpracování filtrů šumu a následně hranových detektorů. Výsledky těchto algoritmů budou ukázány a porovnány v kapitolách 5 a 6.

4.1. Gaussův filtr šumu

Ukázka programového zpracování Gaussova filtru šumu do funkce *Gauss*, která pro černobílý obraz odstraní šum Gaussovím filtrem, pro $\sigma^2 = 1.35$: V případě jinak zvolené masky Gaussova filtru, nebo jiného σ^2 je nutné změnit hodnoty proměnné *Maska* na hodnoty pro zvolené σ^2.

```
function [Gauss] = Gauss(X)
      Maska = 1/159*[2  , 4   ,5   ,4   ,2  ;
                     4  ,9   ,12  ,9   ,4  ;
                     5  ,12  ,15  ,12  ,5  ;
                     4  ,9   ,12  ,9   ,4  ;
                     2  ,4   ,5   ,4   ,2  ]; %Maska filtru
      Gauss = uint8(conv2(X,Maska,'same'));
end
```

Funkce *conv2* provede konvoluci matice X, ve které máme uložené hodnoty obrazové funkce, a matice $Maska$. Parametrem *'same'* u volání funkce udáváme, že velikost vstupní matice bude stejná jako výstupní. Aby toho funkce dosáhla, provádí padding. Vlivem toho můžeme při okraji sledovat přechod barev u okraje směrem k černé. Funkce *uint8* udává formát výstupních dat jako celé čísla od 0 po 255.

4.2. Filtrace šumu Fourierovou transformací

Ukázka programového zpracování filtrace šumu Fourierovou transformací do funkce *fourier*, která převede obraz na černobílý, pokud již není, a následně odstraní šum. Hodnota v udává, jak velkou část Fourierovy transformace zachováváme a nabývá hodnoty od 0 po 1. Nejedná se o poměr co do počtu pixelů, avšak o velikost stran matice ponechaných hodnot.

```
function [F] = fourier(X, v)
[m,n,l] = size(X);
v = 0.5*v;
    if l > 1
        X = rgb2gray(X);
    end
    %kolik pixelu od stredu nechám
    mp = mod(m,2)/2+uint64(m*v);
```

```
    np = mod(n,2)/2+uint64(n*v);

    F = fft2(X); %Fourierova transformace
    F = (fftshift(F)); %prohozeni kvadranu
    Fp = zeros(m,n); %nulova matice a prirazenim
    Fp(m/2-mp+1:m/2+mp,n/2-np+1:n/2+np) =...
    F(m/2-mp+1:m/2+mp,n/2-np+1:n/2+np);
    F = uint8(real(ifft2(ifftshift(Fp)))); %zpetna transformace
end
```

Funkce *size* nám vrátí velikost matice X, která obsahuje obraz. Funkce *rgb2gray* převání obraz na obraz ve stupních šedi. Funkce *fft2* trasformuje matici X Fourierovou transformací. Funkce *fftshift* převrací kvadranty matice, zpětným převrácením je pak funkce *ifftshift*. Funkce *ifft2* provádí zpětnou Furierovou transformaci. Funkce *real* vezme pouze reálnou část hodnot.

4.3. Operátorové detektory hran

V této části si ukážeme funkci na detekci hran v černobílém obraze za pomoci Robertsnova operátoru, Sobelova operátoru a operátoru Prewitové. Do funkce *Hrana* jako vstupní parametr X dosazujeme matici s daty obrazu, za parametr *typ* příjmení autora operátoru a za *citlivost* práh citlivosti.

```
function [Hran] = Hrana(X,typ,citlivost)
[m,n] =size(X);
Hran = false(m,n);
 %% Prewit
 if typ == "Sobel"
    DX=[1 0 -1; 2 0 -2; 1 0 -1]./8;
    DY=[1 2 1; 0 0 0; -1 -2 -1]./8;
    Der=sqrt(conv2(X,DX,'same').^2+abs(conv2(X,DY,'same').^2));
    Hran = Der > citlivost;
 end
 %% Prewit
 if typ == "Prewitt"
    DX=[1 0 -1; 1 0 -1; 1 0 -1]./6;
    DY=[1 1 1; 0 0 0; -1 -1 -1]./6;
    Der=sqrt(conv2(X,DX,'same').^2+abs(conv2(X,DY,'same').^2));
    Hran = Der > citlivost;
 end
  %% Roberts
 if typ == "Roberts"
    DX=[-1 0; 0 1]./2;
    DY=[0 -1; 1 0]./2;
    Der=sqrt(conv2(X,DX,'same').^2+ abs(conv2(X,DY,'same').^2));
    Hran = Der > citlivost;
 end
```

```
end
```

4.4. Cannyho hranový detektor

V této části si ukážeme rozšíření pro předchozí funkci *Hrana*, uvedenou v části 4.3, pro Cannyho hranový detektor. Funkce využívá i funkci pro Gaussův filtr šumu naprogramovanou v části 4.1. Následující část ukazuje pouze části výpočtu hran pomocí Cannyho detektoru. Celý kód naleznete v Příloze v souboru *Hrana*.

```
if typ == "Canny"

    X = int16(Gauss(X));     %Odstranuji šum

        %% Zvyraznuji hrany
    Maska = 1/8*[-1 -1 -1;-1 8 -1;-1 -1 -1];
    zvyrazhrany = int16(conv2(X, Maska,'same'));
    X = uint8(zvyrazhrany + int16(X));

    %%Pocítám DX a DY dle Sobela
    DX=1/8*[1 0 -1; 2 0 -2; 1 0 -1];
    DY=1/8*[1 2 1; 0 0 0; -1 -2 -1];
    DerX = single(conv2(X,DX,'same'));
    DerY = single(conv2(X,DY,'same'));


    %%Výpocet Smeru a velikosti derivace
    derS = uint8(sqrt(DerX.*DerX + DerY.*DerY));
    dersmer = atan(DerX./DerY);

    %%maxima z hran
  derSS = uint8(zeros(m,n));
  for i = 2:m-1
      for j = 2:n-1
          %0
          if ((-0.3927 < dersmer(i,j)) &&  ...
              if max(derS(i-1,j),derS(i+1,j)) <= derS(i,j)
                  derSS(i,j) = derS(i,j);
              end
```

Obdobně provedeme ověření, že je *derS* v kolmém směru na *dersmer* největší pro ostatní směry. Po doplnění ostatních směrů ukončíme oba cykly *for*.

```
        %% Prahování
  [indx,indy] = find(derSS > citlivost);
  if isempty(indx)
      return
  end
```

```
    for i=1:m*n
            if (citlivost > derSS(indx(i)-1,indy(i)-1)) && atd.
                derSS(indx(i)-1,indy(i)-1) = 255;
                indx=[indx; indx(i)-1];
                indy=[indy; indy(i)-1];
            end
```

Obdobně projdeme i zbylé pixely z okolí již přijatých hranových pixelů, abychom zjistili, jestli se v nich nenachází pixel, jehož gradient je větší nežli menší práh. Pokud ano, považujeme i příslušný okolní pixel jako hraniční pixel. Po ověření okolí všech hraničních pixelů následuje konec cyklu a zápis výsledku do matice *Hran*.

```
        if i == length(indx)
        break
        end
    end
    Hran = (derSS>citlivost);
end
```

4.5. Otsu metoda

Tato část obsahuje programové zpracování funkce detekování hran Otsu metodou pro černobílý obraz. Výstup metody je matice s hodnotami ve formátu bool. Pozice hodnot matice odpovídají příslušnému pixelu. Hodnota *true* značí, že příslušný pixel spadá do množiny pozadí, naopak hodnota *false* značí, že příslušný pixel spadá do množiny popředí. Hrana se pak nachází na okraji těchto množin.

```
function [Hran] = HranaOtsu(X)

        [m,n] =size(X);
        Hran = false(m,n);
        bins= 0:1:256;
        hist = histcounts(X , bins)./(m*n)
        Rozptyl=zeros(1, 256);
        parfor k =1:255
            w1=sum(hist(1:k));
            w2=sum(hist(k:255));
            m1=sum((0:k-1).*hist(1:k)./w1);
            m2=sum((k:255).*hist(k+1:256)./w2);
                Rozptyl(k) = w1*w2*((m1-m2)^2);
        end
    [maxRozptyl, idx] = max(Rozptyl);
     X=(X>idx);
     Hran = X;
end
```

Funkce *histcounts* vytvoří histogram obrazu, parametr bins udává dělení množiny. Funkce *parfor* je jako metoda *for*, která ovšem provádí paralelní výpočty, a je tedy rychlejší.

5. Srovnání a optimalizace filtrů šumu

V této části porovnáme výsledky jednotlivých metod pro odstranění šumu. Nejdříve určíme optimální nastavení a následně porovnáme metody pro odstranění šumu.

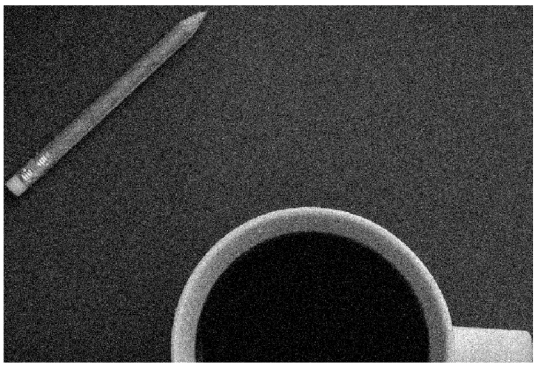
5.1. Způsob srovnávání

Testovat budeme odstranění šumu se zaměřením na následnou detekci hran v daném obraze. Test provedeme na sérii uměle zašumělých obrazů a výsledky pro různé metody a nastavení budeme porovnávat s obrazem bez šumu. Při jejich porovnávání budeme využívat Sobelův operátor. Testovací obraz je naprosto čistý obraz bez šumu.

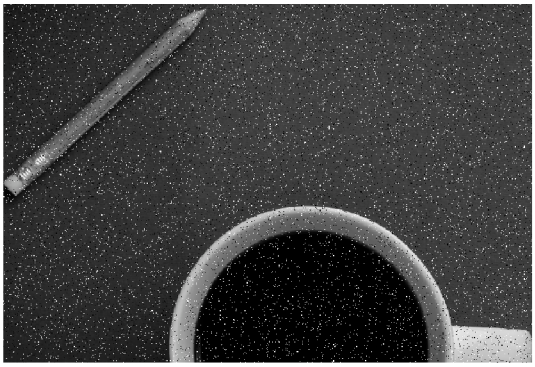
Testovací obraz použitý při testování můžeme vidět na obrázku 5.1.

Obrázek 5.1: Testovací obraz *[Thebrave Alex]*

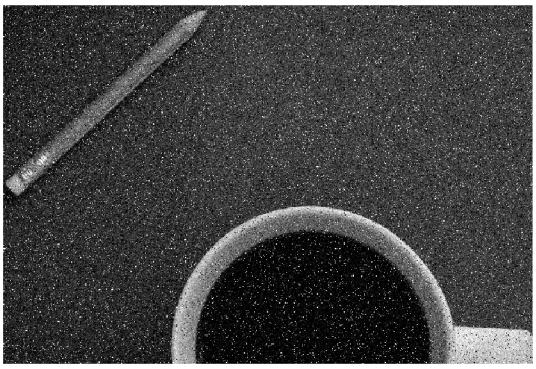
Vzhledem k velikosti obrazu budu pro větší názornost ukazovat pouze část ze zašuměného obrazu. Budeme uvažovat šumy popsané v úvodu kapitoly 3. Můžeme je sledovat na obrázcích 5.2.



(a) S Gaussovým šumem



(b) S Impulzním šumem

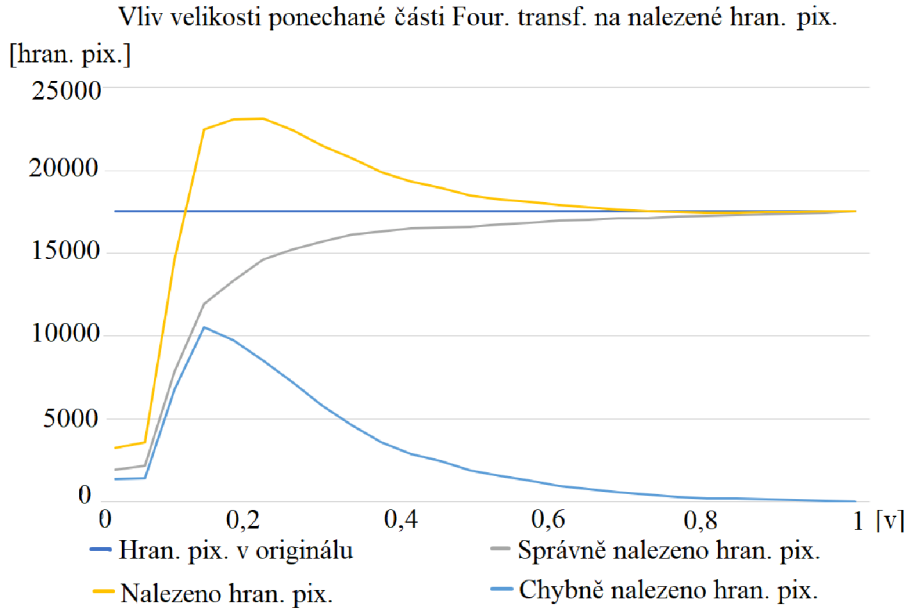


(c) Přidány oba šumy

Obrázek 5.2: Obrazy s uměle přidaným šumem

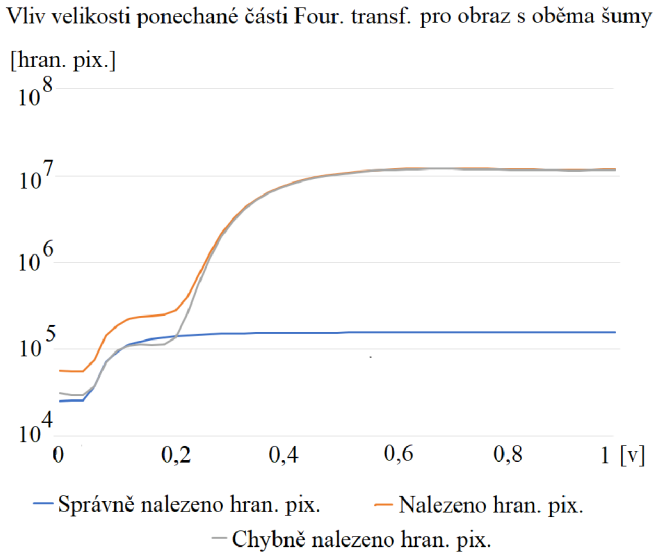
5.2. Optimalizace filtrace šumu Fourierovou transformací

Zaměříme se na optimální volbu v z kódu 4.2, udávající jak velkou část Fourierovy transformace zachováváme. Jako optimální volbu budeme volit tu, kde je nalezena velká část hranových pixelů z nalezených hranových pixelů na původním obrazu, a zároveň je nalezeno co nejmenší množství odlišných hranových pixelů. Vliv velikosti v z kódu na původní obraz můžeme sledovat na grafu na obrázku 5.3. V grafu můžeme pozorovat znatelný pokles v počtu nalezených hranových pixelů pro $v = 0,13$, jedná se o mezní hodnotu, pro menší hodnoty již dochází k velké ztrátě dat v obrazu.



Obrázek 5.3: Vliv v na originální obraz

Obdobně si můžeme vytvořit i graf pro jednotlivé zašumělé obrazy. Vzhledem k tomu že grafy vypadají obdobně, uvedeme zde pouze graf pro oba šumy, a to na obrázku 5.4.



Obrázek 5.4: Vliv v na obraz, kde jsou přidány oba šumy

V grafu 5.4 můžeme pozorovat velké potlačení šumu do $v = 0,23$, avšak i znatelný pokles v počtu nalezených hranových pixelů do $v = 0,13$, obdobně jako v grafu 5.3. Spojení obou grafů omezuje optimální hodnotu v na interval 0,15 až 0,23. V našem dalším postupu budeme uvažovat hodnotu v jako 0,19.

5.3. Srovnání filtrů šumu

V této části se zaměříme na srovnání různých Gaussových filtrů šumu, včetně průměrování a filtrace šumu Fourierovou transformací. Citlivost hranového detektoru uvažujeme

konstantní. Pro obraz bez šumu (ideální případ) je Sobelovým operátorem nalezeno 17550 hranových pixelů. Hranové pixely nalezené na stejných souřadnicích po filtraci šumu považujeme za správné. V opačném případě je považujeme za chybné.

Tabulka ukazuje počet nalezených hranových pixelů po filtraci originálního obrazu jednotlivými metodami filtrace šumu.

Metoda	Správně	Chybně	Celkem	Chybně/správně
Gaussův filtr pro $\sigma^2 = 1$	15215	2530	17745	0,166
Gaussův filtr pro $\sigma^2 = 1{,}35$	14606	3185	17791	0,218
Gaussův filtr pro $\sigma^2 = 3$	13724	4163	17887	0,303
Průměrování	15709	2226	17935	0,142
Four. transformace pro $v = 0{,}19$	14013	7180	23272	0,661

Za pomoci dat z uvedené tabulky můžeme určit, který z uvedených filtrů šumu nejméně znehodnocuje výchozí obraz. Jako nejhorší metoda se jeví filtrace Fourierovou transformací, při které je více než třetina nalezeních hranových pixelů nesprávná.

Nyní provedeme stejnou tabulku pro jednotlivé typy šumu, abychom mohli porovnat vhodnost jednotlivých filtrů na daný typ šumu.

Tabulka pro Gaussův šum:

Metoda	Správně	Chybně	Celkem	Chybně/správně
Gaussův filtr pro $\sigma^2 = 1$	14941	11244	26185	0,753
Gaussův filtr pro $\sigma^2 = 1{,}35$	14502	5765	20267	0,398
Gaussův filtr pro $\sigma^2 = 3$	13727	5434	19161	0,396
Průměrování	15194	33679	48873	2,217
Four. transformace pro $v = 0{,}19$	13842	10613	24455	0,767

Tabulka pro Impulzní šum:

Metoda	Správně	Chybně	Celkem	Chybně/správně
Gaussův filtr pro $\sigma^2 = 1$	14832	193048	207880	13,016
Gaussův filtr pro $\sigma^2 = 1{,}35$	14284	51423	65707	3,6
Gaussův filtr pro $\sigma^2 = 3$	13582	13029	26611	0,959
Průměrování	15244	345736	360980	22,68
Four. transformace pro $v = 0{,}19$	13475	10432	23907	0,774

Tabulka pro Gaussův i impulzní šum zároveň:

Metoda	Správně	Chybně	Celkem	Chybně/správně
Gaussův filtr pro $\sigma^2 = 1$	14685	272012	286697	18,523
Gaussův filtr pro $\sigma^2 = 1{,}35$	14255	83459	97714	5,855
Gaussův filtr pro $\sigma^2 = 3$	13584	21979	35563	1,618
Průměrování	14974	444016	458990	29,652
Four. transformace pro $v = 0{,}19$	13487	11402	24889	0,845

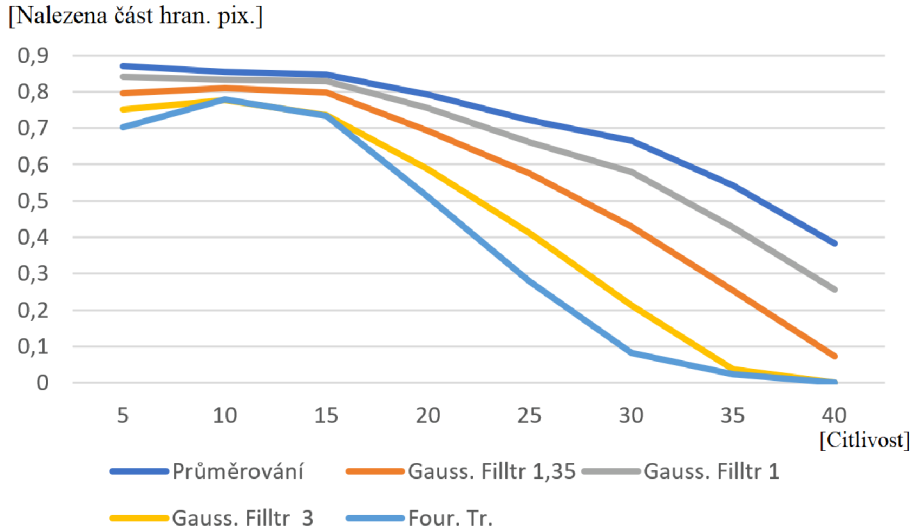
Můžeme vidět, že pro Gaussův šum se vyplatí využívat Gaussův filtr, a za optimální můžeme označit $\sigma^2 = 1{,}35$. Případně i $\sigma^2 = 3$, kde již ovšem není nalezen větší počet hranových pixelů.

V případě impulzního šumu se jako nejlepší metoda jeví filtrace šumu Fourierovou transformací. Lze využít i Gaussův filtr s $\sigma^2 = 3$, avšak dostáváme z ní více špatných hraničních pixelů.

Pro kombinaci obou šumů jednoznačně nejlépe vychází filtrace šumu Fourierovou transformací. Jako naprosto nevhodná se pak jeví metoda průměrování.

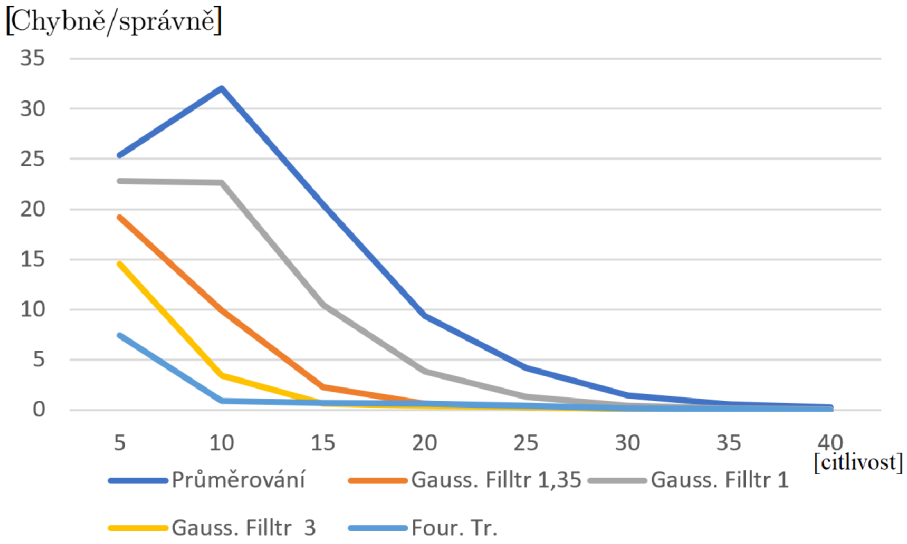
Na závěr sestavíme graf závislosti nalezených hranových pixelů na citlivosti hranového detektoru. Budeme tak moci porovnat, jak jednotlivé filtry ovlivňují různě silné hrany.

V grafu na obrázku 5.5 vidíme, jak velká část z hranových pixelů nalezených na nezašumělém obrazu pro danou citlivost byla nalezena po filtraci šumu z obrazu, do kterého jsou přidány oba šumy.



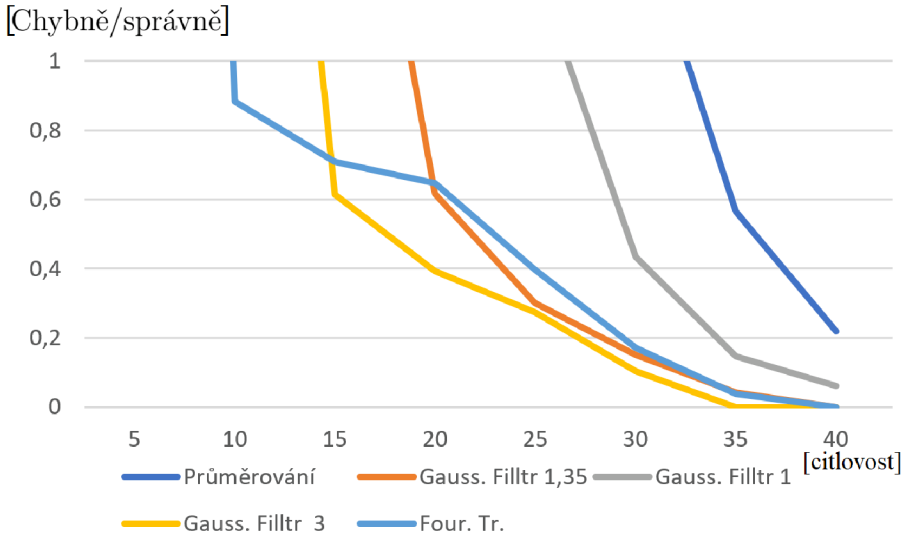
Obrázek 5.5: Podíl správně nalezených hran. pixelů, v závislosti na citlivosti.

Z grafu je patrné, která z uvedených metod má tendenci zmenšovat derivaci pro dané hranové pixely, a tím snižovat počet nalezených hranových pixelů. Ke grafu 5.5 je ovšem nutné uvažovat i následující graf na obrázku 5.6 ukazující poměr správně a chybně nalezených hran. pixelů v závislosti na citlivosti. Případně jeho část na obrázku 5.7.



Obrázek 5.6: Poměr správně a chybně nalezených hran. pixelů v závislosti na citlivosti

Na následujícím obrázku 5.7 si přiblížíme část grafu 5.6 pro hodnoty poměru správně a chybně nalezených hranových pixelů menších než jedna.



Obrázek 5.7: Detail grafu poměru správně a chybně nalezených hran. pixelů v závislosti na citlivosti

Můžeme vidět, že po filtraci průměrováním i při vyšší citlivosti nalezne hranový detektor značnou část hranových pixelů, avšak má při dané citlivosti i největší podíl chybně nalezených hranových pixelů. Zároveň však můžeme vidět, že metody filtrace šumu zmenšují výslednou sílu hrany. Proto při zvyšujícím se prahu citlivosti dochází ke zmenšování počtu nalezených hranových pixelů. Tento jev můžeme sledovat například u filtrace Fourierovou transformací, po které pro citlivost 40 již nebyly nalezené žádné hranové pixely.

6. Srovnání hranových detektorů

V této části se zaměříme na srovnání jednotlivých hranových detektorů. Nejdříve si přiblížíme časovou náročnost jednotlivých metod. Následně se podíváme na jejich přesnost a na vliv šumu na jejich přesnost. Uvedeme i srovnání dalších rozdílů mezi jednotlivými metodami.

6.1. Časová náročnost hranových detektorů

V této části se zaměříme na čas výpočtu jednotlivých metod pro různě velké obrazy. Při testování budeme využívat tři obrazy znázorněné na obrázcích 6.1.

(a) Moon (196 Mpix.) *[Bartosz Wojczyński]*

(b) Domy (18 Mpix.)

(c) Kočka (8 Mpix.)

Obrázek 6.1: Testovací obrázky pro měření časové náročnosti

V následující tabulce uvádíme průměrné časy výpočtu jednotlivých metod v sekundách. Výpočet byl prováděn na notebooku Lenovo legion 5 s procesorem AMD Ryzen 7 4800H. Časová náročnost závisí i na výkonu výpočetního zařízení. Proto je z následující tabulky hodnotnější porovnání jednotlivých časů mezi sebou, nežli hodnoty jako takové. Jednotlivé metody jsou v tabulce označeny dle příjmení autora metody.

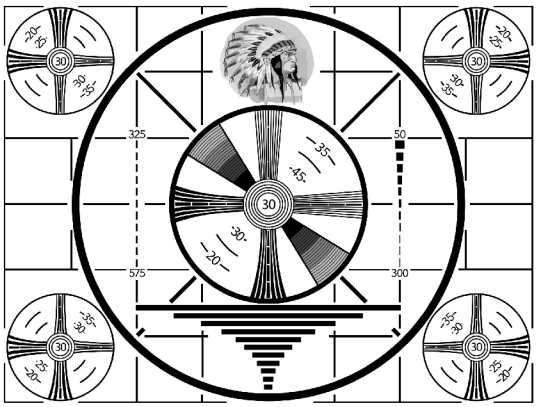
Tabulka průměrného času metod v sekundách pro jednotlivé obrazy:

Obraz	Roberts	Sobel	Prewittová	Canny	Otsu
Moon	2,05	2,17	2,17	30,5	0,13
Domy	0,17	0,17	0,17	3	0,02
Kočka	0,08	0,08	0,08	0,52	0,01

Můžeme vidět, že nejrychlejší metodou pro nalezení hran je Otsu metoda. Zároveň se časová náročnost Otsu metody i nejméně zvětšuje při zvětšování velikosti obrazu. Velkou nevýhodou Otsu metody je ovšem její omezené použití. Naopak zdaleka nejpomalejší metodou pro nalezení hran je Cannyho hranový detektor. Časy jednotlivých operátorů jsou skoro totožné. Jedná se o očekávaný výsledek, neboť jejich výpočet se liší pouze zvolenou maticí konvoluce.

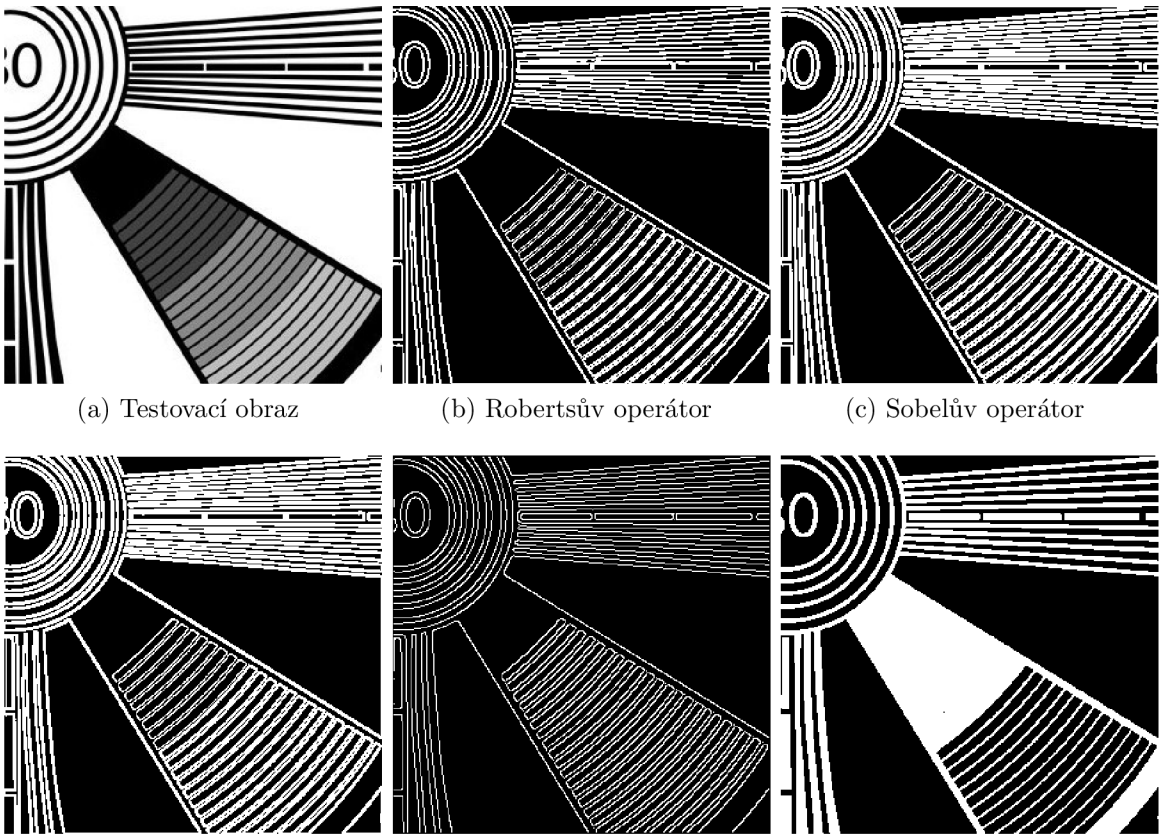
6.2. Přesnost nalezených hran

V této části se blíže podíváme na přesnost a detailnost nalezených hran. Za tímto účelem budeme využívat starý testovací pattern pro televize zobrazený na obrázku 6.2, konkrétně jeho část zobrazenou na obraze 6.3(a). Budeme sledovat přesnost jednotlivých hranových detektorů pro obraz bez šumu a v další části 6.3 se zaměříme i na obraz se šumem.



Obrázek 6.2: Starý testovací TV pattern

Aplikaci hranových detektorů na obraz bez šumu můžeme vidět na obraze 6.3. Pro všechny metody, kde se uvažuje citlivost metody, volíme stejnou hodnotu citlivosti.



(a) Testovací obraz (b) Robertsův operátor (c) Sobelův operátor

(d) operátor Prewittové (e) Cannyho hranový detektor (f) Otsu metoda

Obrázek 6.3: Přesnost nalezení hran pro obraz bez šumu

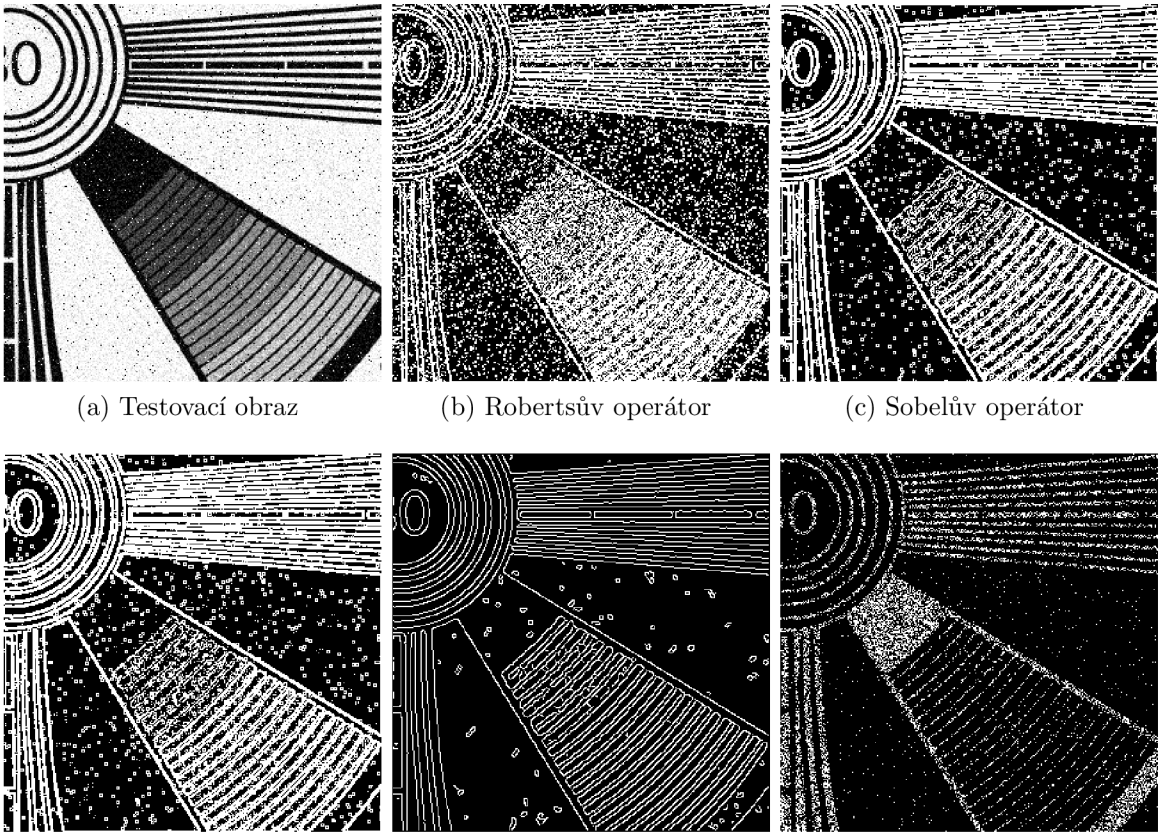
Nejlepší přesnost z uvedených metod předvedl Cannyho hranový detektor. Nalezená hrana má jednotkovou šířku a udává tedy přesnou pozici v obraze. Zároveň došlo k přesnému nalezení hran i v případě těsné blízkosti více hran. Jedinou nepřesnost, kterou můžeme v případě Cannyho hranového detektoru pozorovat, je zaoblení nalezených hran při ostré změně směru hrany. K této nepřesnosti dochází vlivem Gaussova filtru šumu, který tato metova využívá.

V případě aplikace Otsu metody nedošlo k dokonalému rozdělení obrazu na objekt a pozadí. Hrany nalezené pomoci Otsu metody zachovávají veškeré detaily, a to i v případě, že se nachází blízko. U Robertsova operátoru, Sobelova operátoru a operátoru Prewittové dochází v případě blízkosti více hran ke splynutí nalezených hran, neboť nalezená hrana má větší šířku. V případě Robertsova operátoru je šířka nalezené hrany menší, jelikož využívá menší matici konvoluce. Díky domu je z operátorových metod nejpřesnější.

6.3. Vliv šumu na přesnost hranových detektorů

V této části budeme aplikovat hranové detektory na testovací obrázek z části 6.2, který opatříme Gaussovým i impulzním šumem. Nebudeme předem filtrovat šum ze zašumělého obrazu, budeme tak moci porovnat vliv šumu na jednotlivé detektory. Ukážeme si i vliv velikosti šumu na vybrané metody.

Aplikaci hranových detektorů na zašumělý obraz můžeme sledovat na obraze 6.4. Citlivost je volena stejně pro všechny metody.



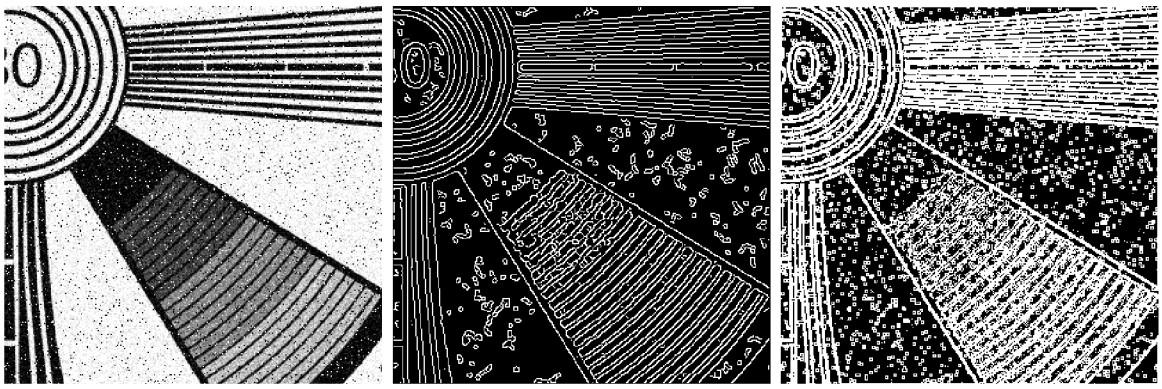
(a) Testovací obraz (b) Robertsův operátor (c) Sobelův operátor

(d) operátor Prewittové (e) Cannyho hranový detektor (f) Otsu metoda

Obrázek 6.4: Přesnost nalezení hran pro obraz se šumem

Z uvedených metod si se šumem nejlépe poradil Cannyho hranový detektor. V případě hrany, kde dochází k velké změně intenzity, ji byl schopen i přes šum přesně najít, pokud je ovšem změna intenzity menší, může být nalezená hrana trochu nepřesná. Gaussův šum této metodě nedělá problém, avšak v případě skupinky impulzního šumu jej může chybně označit jako hranu. Naopak nejhůře si se šumem poradil Robertsův operátor, který využívá k určení hrany pouze rozdíl intenzity dvou pixelů, kde došlo k označení většiny šumu jako hrany. Velký vliv má šum i na Otsu metodu, kde množina popředí představuje spíš body, nežli plochu v obraze. Šum ovlivňuje i mezní hodnotu intenzity Otsu metody. Impulzní šum se bez předchozí filtrace šumu projeví na výstupu Otsu metody vždy. V případě Sobelova operátoru a operátoru Prewitové je vidět velký vliv impulzního šumu. Vliv Gaussova šumu na tyto metody není příliš patrný.

Vliv dvojnásobného množství šumu oproti 6.4 na Cannyho hranový detektor a Sobelův operátor můžeme pozorovat na obraze 6.5. Citlivost je volena stejná pro všechny metody.



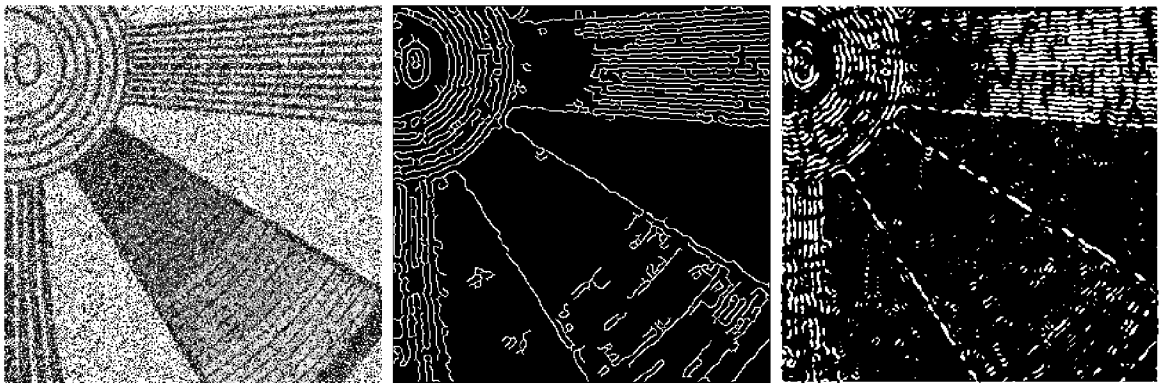
(a) Testovací obraz s více šumu (b) Cannyho hranový detektor (c) Sobelův operátor

Obrázek 6.5: Vliv zvětšeného šumu na vybrané hranové detektory

Ve výsledných nalezených hranách můžeme vidět větší množství chybných hran. Za zmínku stojí, že v případě blízkosti hran s velkou změnou intenzity dokáže Cannyho hranový detektor dobře vyfiltrovat šum v jejich okolí, jak můžeme vidět v horní části obrazu. Dochází k tomu díky potlačení nedominantních hodnot gradientu při této metodě.

Vliv šumu můžeme omezit použitím filtrace šumu. Pro metody využívající práh jej můžeme omezit i zvýšením prahu, kdy ovšem dochází i k nenalezení slabších hran v obraze. V případě impulzního šumu může být potřebná velikost navýšení prahu i velmi velká, a proto je lepší v jeho případě využít i filtrace šumu.

Možnost eliminace šumu si ukážeme na následujícím obraze 6.6. Základní obraz jsme opatřily Gaussovým šumem a polovinu pixelů obrazu jsme změnili impulzním šumem. Pro eliminaci šumu jsme využili Fourierovu transformaci a zvýšení prahu detektorů.



(a) Testovací obraz (b) Cannyho hranový detektor (c) Sobelův operátor

Obrázek 6.6: Možnost eliminace vlivu šumu v obraze

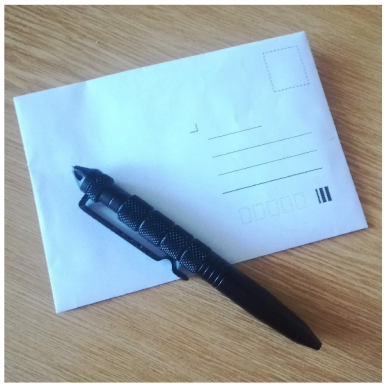
Můžeme vidět, že i při takto velkém šumu jsou hranové detektory po filtraci šumu schopné hrany detekovat. Přesnost nalezených hran již ovšem není nikterak velká, a v případě hrany s malou změnou intenzity již nedochází k její detekci.

6.4. Omezení použití hranových detektorů

V této části se zaměříme na možnosti použití jednotlivých detektorů. Zaměříme se hlavně na různá omezení, která pro používání jednotlivých hranových detektorů vznikají. Největší omezení v tomto směru můžeme očekávat od Otsu metody.

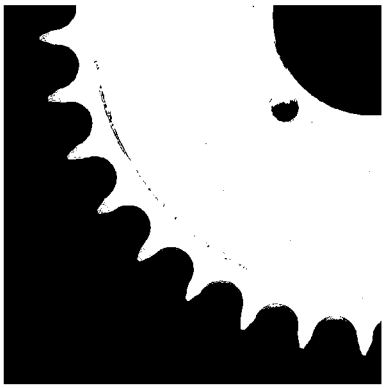
V případě Otsu metody ji lze použít pouze v případě znatelného rozdílu jasu mezi objektem, jehož hrany hledáme, a jeho okolím. Pro Otsu metodu může představovat značný problém změna osvětlení objektu na obraze. Další problém může nastat v případě více objektů na obraze, jejichž hrany chceme najít, jelikož Otsu metoda není schopna rozdělit obraz na více než dvě části a rozeznat tak jednotlivé objekty. Ukázku správného obrazu můžeme vidět na 6.7(a). Nesprávné obrazy pro tuto metodu můžeme pozorovat na obraze 6.7(b) a (c).

(a) Ozubené kolo *[Jan Koníček]*



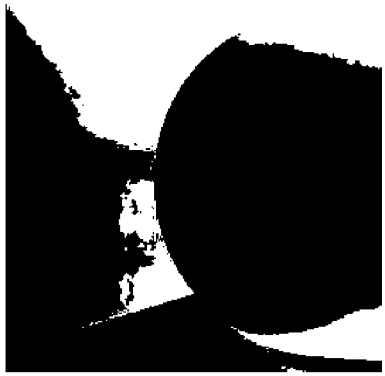
(b) Dva objekty

(c) Koule *[Mahdis Mousavi]*



(d) Aplikace Otsu metody na obraz (a)

(e) Aplikace Otsu metody na obraz (b)



(f) Aplikace Otsu metody na obraz (c)

Obrázek 6.7: Omezení aplikace Otsu metody

Obraz b) je nevhodný z důvodu více objektů na obraze, nedochází tedy k nalezení hran obou objektů, a z důvodu jejich rozdílné intenzity i k špatnému odstranění pozadí. Obraz c) je nevhodný z důvodu malé změny intenzity a kvůli změně osvětlení v obraze.

V případě Cannyho hranového detektoru, případně i operátorových detektorů, může být omezení pro aplikaci jejich časová náročnost. Například pokud potřebujeme najít hrany pro obrazy z aktuálního záběru kamery.

7. Závěr

V této bakalářské práci jsme se zaměřili na problematiku hranových detektorů. Popsali jsme metody k detekování hran a jejich princip. Obdobně jsme se v této práci věnovali i popisu metod k filtraci šumu v obraze a popsali jejich princip.

Všechny metody, které byly uvedeny, jsme i programově zpracovali v programu Matlab. Pro filtraci šumu byl konkrétně naprogramován Gaussův filtr šumu a filtr šumu Fourierovou transformací. Programy pro tyto metody můžeme nalézt v příloze *Filtry*. Z hranových detektorů byl naprogramován Robertsův operátor, Sobelův operátor, operátor Prewitové, Otsu metoda a Cannyho hranový detektor. Programové zpracování těchto metod je k vidění v příloze *Hrana*.

Zabývali jsme se i optimálním nastavením filtrů šumu a jejich vzájemným srovnáním z hlediska kvality filtrace různých typů šumu. Obdobně byly srovnány i jednotlivé hranové detektory. V jejich srovnání jsme se zaměřili hlavně na jejich přesnost, časovou náročnost a možnosti použití.

V případě hledání hran v obraze bez šumu je vhodná Otsu metoda, pokud ji lze na daný obraz využít, případně metoda Robertsova operátoru. V případě, že se v obraze nachází Gaussův šum, je lepší využít Cannyho hranový detektor, případně Gaussův filtr šumu a jiný detektor hran. Pokud se v obraze nachází impulzní šum, ať již samostatně nebo i společně s Gaussovým šumem, je pro dobré výsledky při detekci hran nutná filtrace šumu v obraze. Z uvedených metod filtrace šumu můžeme pro tento případ doporučit filtrace šumu Fourierovou transformací.

Volba konkrétní metody je dosti závislá na požadované aplikaci, dovolujeme si proto volbu metody ponechat na čtenáři.

Literatura

[1] GONZALEZ, Rafael C. a Richard E. WOODS. Digital image processing. 3rd ed. Upper Saddle River,N.J.: Prentice Hall, c2008. ISBN 978-0-13-168728-8

[2] JAIN, Ramesh a Rangachar KASTURI. Machine vision. Boston: McGraw-Hill, 1995. ISBN 0-07-032018-7.

[3] ROBERTS, Lawrence Gilman. Machine Perception of Three-Dimensional Solids. Massachusetts, 1963. Massachusetts institute of technology.

[4] J. Canny, "A Computational Approach to Edge Detection," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 8, pp. 679-698, 1986.

[5] Nobuyuki Otsu, "A threshold selection method from gray-level histograms". IEEE Trans. Sys., Man., Cyber. 9 (1): 62–66. 1979.

[6] J. M. S. Prewitt, "Object enhancement and extraction," Picture Processing and Psychopictorics, B. Lipkin and A. Rosenfeld, Eds., New York: Academic Press, 1970, pp. 75-149

[7] Václav Bezvoda, Josef Ježek, Stanislav Saic, Karel Segeth:Dvojrozměrná diskrétní-Fourierova transformace a její použití 1, SPN (1988)

[8] N. Senthilkumaran, Rajesh Reghunadhan, "Edge Detection Techniques for Image Segmentation - A Survey of Soft Computing Approaches", INFORMATION PAPER International Journal of Recent Trends in Engineering. 2007.

[9] Sobel, Irwin. "An Isotropic 3x3 Image Gradient Operator". Presentation at Stanford A.I. Project 1968.

[10] ZIOU, Djemel a Salvatore TABBONE. Edge detection techniques - an overview. Pattern Recognition and Image Analysis: Advances in Mathematical Theory and Applications. 8(4)

8. Seznam příloh

Hrana.m Obsahuje programové zpracování zde uvedených hranových detektorů v programu Matlab.

Filtr.m Obsahuje programové zpracování zde uvedených filtrů šumu v programu Matlab.