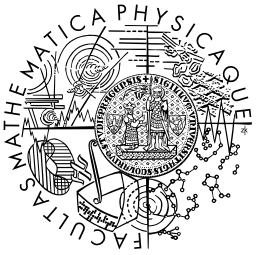


MATEMATICKO-FYZIKÁLNÍ FAKULTA
Univerzita Karlova

DIPLOMOVÁ PRÁCE

Matěj Lébl

Aplikace výpočetních metod v třídění skleněných kamenů

Katedra numerické matematiky

Vedoucí diplomové práce:	RNDr. Iveta Hnětynková, Ph.D.
Konzultant:	Ing. Jaroslav Vlach, Ph.D.
Studijní program:	Matematika
Studijní obor:	Numerická a výpočtová matematika

Praha 2017

Prohlašuji, že jsem tuto práci vypracoval(a) samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ..... Podpis autora

Rád bych poděkoval doktorce Ivetě Hnětynkové za ochotu, trpělivost a čas, který mi v průběhu zpracování této práce věnovala. Mé poděkování patří i rodině a blízkým za podporu a docentu Václavu Kučerovi za cenné poznámky.

Název práce: Aplikace výpočetních metod v třídění skleněných kamenů

Autor: Matěj Lébl

Katedra: Katedra numerické matematiky

Vedoucí diplomové práce: RNDr. Iveta Hnětynková, Ph.D., Katedra numerické matematiky

Konzultant: Ing. Jaroslav Vlach, Ph.D.

Abstrakt: Cílem předložené práce je využít matematických metod zpracování obrazu k návrhu automatické výstupní kontroly kvality skleněných bižuterních kamenů. Hlavním matematickým objektem je zde matice specifických vlastností, reprezentující digitální snímek zkoumaných výrobků. Práce shrnuje matematický popis digitálního obrazu a některé standardní metody zpracování obrazu. Dále je navrženo kompletní řešení zadané úlohy složené z lokalizace kamene na snímku a následné analýzy lokalizované oblasti. Pro účel lokalizace jsou představena dvě vlastní řešení. První je založeno na konvoluci matic a optimalizováno pomocí Fourierovy transformace. Druhé využívá matematických metod prahování a mediánové filtrace a projekce dat do jedné dimenze. Lokalizovaná oblast je analyzována s využitím statistického rozložení celkové světlosti kamenů. Metody jsou implementovány v prostředí MATLAB.

Klíčová slova: zpracování digitálního obrazu, detekce objektů, konvoluce, prahování

Title: Application of computational methods in classification of glass stones

Author: Matěj Lébl

Department: Department of Numerical Mathematics

Supervisor: RNDr. Iveta Hnětynková, Ph.D., Department of Numerical Mathematics

Consultant: Ing. Jaroslav Vlach, Ph.D.

Abstract: The goal of this thesis is to employ mathematical image processing methods in automatic quality control of glass jewellery stones. The main mathematical subject is a matrix of specific attributes representing digital image of the studied products. First, the thesis summarizes mathematical definition of digital image and some standard image processing methods. Then, a complete solution to the considered problem is presented. The solution consists of stone localization within the image followed by analysis of the localized area. Two localization approaches are presented. The first is based on the matrix convolution and optimized through the Fourier transform. The second uses mathematical methods of thresholding and median filtering, and data projection into one dimension. The localized area is analyzed based on statistical distribution of the stone brightness. All methods are implemented in the MATLAB environment.

Keywords: digital image processing, object detection, convolution, thresholding

Obsah

Úvod

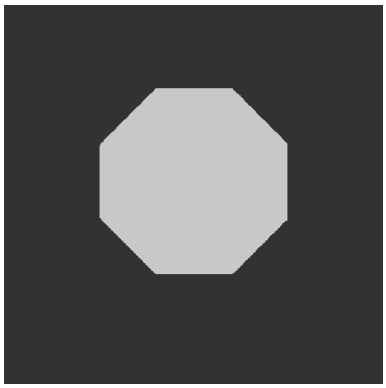
Matematické metody zpracování obrazu poskytují silné nástroje pro práci s obrazovými daty reprezentovanými například digitálními obrazy, záznamy 2D a 3D kamer či snímky z termokamer, výpočetní tomografie, záznamy z ultrazvuku nebo magnetické rezonance. Tyto nástroje slouží jak k vylepšení obrazových vlastností snímků (odstranění šumu, zaostření), tak k analýze snímaných dat a práci s nimi. Matematické zpracování obrazu má široké spektrum uplatnění, využívá se v počítačových programech na úpravu fotografií, při zpracování dat ze satelitních snímků a elektronových mikroskopů nebo v medicínské diagnostice. S jejich pomocí dokážeme z obrazů získat důležité informace o nasnímaných objektech jako je jejich tvar nebo podobnost s referenčními objekty v databázích. Forenzní analýza využívá zpracování obrazu například k vylepšení záznamů z bezpečnostních kamer, identifikaci objektů nebo při analýze otisků prstů. Zpracování obrazu je také nezbytné pro strojové vidění a automatickou navigaci robotů. Zapojením výpočetních technologií dokážeme automatizovat mnohé procesy, například kontrolu kvality produktů. Matematické zpracování obrazu využívá znalosti ostatních matematických oborů, zejména pak výpočetní matematiky, statistiky, matematické analýzy a optimalizace. Základním matematickým objektem je zde typicky matice specifických vlastností

Cílem této práce je využít poznatky z oblasti matematického zpracování obrazu a navrhnout a implementovat vhodný postup pro řešení zadané úlohy: automatického třídění skleněných bižuterních kamenů podle kvality na základě analýzy jejich snímků získaných řádkovou kamerou. Kameny mají tvar šestnáctiúhelníku a velmi malé rozměry (1mm - 1cm). Jsou vyráběny ve velkých počtech čítajících tisíce kusů. Ruční výstupní kontrola je složitá a časově náročná. Náš návrh automatické kontroly se skládá ze tří částí. Nejprve proběhne nasnímání destičky, ve které je zasazeno větší množství kamenů, pomocí řádkové kamery. Data jsou uložena ve formě matice digitálního obrazu velké dimenze. Následuje lokalizace jednotlivých objektů (kamenů) v obraze. Nakonec je lokalizovaná oblast analyzována a je rozhodnuto o kvalitě příslušného kamene. Tvar kamenů je na snímcích velmi blízký kruhu, pro jejich lokalizaci tedy navrhneme dvě vlastní metody založené na identifikaci kruhu. Konkrétně jde o metodu projekce dat do jedné dimenze využívající matematické metody prahování a mediánové filtrace a metodu založenou na konvoluci matic realizovanou za pomoci Fourierovy transformace. Oba přístupy porovnáme se standardní metodou lokalizace kruhů – Houghovou transformací. Dále navrhneme metodu automatické výstupní kontroly kvality kamenů, založenou na statistickém rozložení jejich celkové světlosti. Mimo požadavku na přesnost je kladen značný důraz na rychlost celého výpočtu.

V Kapitole 1 je popsán matematický úvod do zpracování obrazu. Ukážeme, jak matematicky reprezentovat obraz pomocí matice, definujeme maticové verze konvoluce a Fourierovy transformace. Zabývat se budeme také přítomností šumu v obraze. V Kapitole 2 uvedeme některé metody zpracování obrazu. Definujeme histogram obrazu, popíšeme metody odstranění šumu, prahovaní obrazu a zmíníme základní hranové detektory. V Kapitole 3 uvedeme standardní metodu pro lokalizaci kruhu a představíme dva vlastní návrhy přístupů k úloze lokalizace kruhu využívající některé standardní metody uvedené v Kapitolách 1

a 2. Dále diskutujeme možnosti jejich použití na zadanou úlohu a možné zobecnění umožňující lokalizaci dalších tvarů. V Kapitole 4 pak otestujeme metody popsané v Kapitole 3 na sadě reálných testovacích snímků, vyhodnotíme jejich přesnost a časovou náročnost. Nakonec navrhneme a otestujeme kompletní řešení problému separace kamenů podle jejich kvality.

Testovací data použitá v Kapitole 4 byla poskytnuta firmou Preciosa, a.s. Veškeré numerické experimenty byly provedeny za pomoci softwaru MATLAB verze R2015a a jeho rozšíření Image processinng toolbox. Uvedené výpočetní časy odpovídají počítači s následující konfigurací: Intel core i5-3210M 2.50Ghz, 8GB RAM, NVIDIA GeForce GT350M, Windows 10 64bit. V práci budeme pro demonstraci některých uvedených postupů a operací využívat referenční obraz představující zjednodušený idealizovaný model skutečného kamene viz Obr. 1.



Obrázek 1: Model kamene. Digitální obraz s celočíselnými pixelovými hodnotami. Rozměry 583×583 pixelů. Pixelové hodnoty kamene 200, pixelové hodnoty pozadí 50, kámen zaujímá 20% plochy obrazu.

Seznam značení

$\mathbb{R}$	obor reálných čísel
$\mathbb{R}^+$	obor nezáporných reálných čísel
$\mathbb{R}^n$	n-dimenzionální vektorový prostor nad $\mathbb{R}$
$\mathbb{R}^{n\times m}$	prostor matic typu $n \times m$ nad $\mathbb{R}$
$\mathbb{Z}$	obor celých čísel
$\mathbb{N}$	obor přirozených čísel
$\mathbb{N}_0$	obor přirozených čísel sjednocený s nulou
$L_1(\mathbb{R})$	prostor Lebesgueovsky integrovatelných funkcí na $\mathbb{R}$
$\mathfrak{f} : \mathbb{Z} \to \mathbb{R}$	reálná funkce jedné celočíselné proměnné
$f : \mathbb{R} \to \mathbb{R}$	reálná funkce jedné reálné proměnné
$f : \mathbb{R}^n \to \mathbb{R}$	reálná funkce n reálných proměnných
$f : \mathbb{R}^n \to \mathbb{R}^m$	reálná vektorová funkce n reálných proměnných se složkami $f_1, \ldots, f_m : \mathbb{R}^n \to \mathbb{R}$
$spt(f)$	nosič funkce f, t.j. množina, na níž funkce f nabývá nenulových nodnot
$A = (a_{i,j})_{i,j=1}^{n,m}$	matice z prostoru $\mathbb{R}^{n\times m}$ a její prvky
$\mathcal{I}$	matice odpovídající digitálnímu obrazu
$\mathcal{I}(i,j)$	prvek matice $\mathcal{I}$ na pozici i,j
$\star$	operace konvoluce
i	imaginární jednotka
e	Eulerovo číslo
$[\alpha]$	Celá část čísla $\alpha \in \mathbb{R}$

1. Matematický základ

Tato kapitola slouží k seznámení čtenáře s matematickou definicí obrazu. Ukážeme, jak popsat obraz pomocí matice a představíme některé důležité operace, které se hojně využívají při analýze a zpracování obrazu. Níže uvedené poznatky byly čerpány z odborných publikací [5], [6], [11], [15], [18], [23].

1.1 Definice obrazu

V této části zavedeme pojem obraz dle [15], ukážeme, jak jej matematicky popsat a uvedeme několik základních typů obrazu.

Definice 1. ***2D Obrazem*** *nazveme funkci dvou proměnných* $f : \mathbb{R}^2 \to \mathbb{R}$*, kde*

1. f *má kompaktní nosič,*
2. $0 \leq f(x,y) < \infty$ *pro všechna* $(x,y) \in \mathbb{R}^2$*,*
3. $\int_{\mathbb{R}^2} f(x,y)\mathrm{d}x\mathrm{d}y$ *je konečný.*

Pojem 2D obraz zde označuje funkci určující **světelnou intenzitu (světlost)** na oblasti v $\mathbb{R}^2$ dané jejím nosičem. Odtud požadavek na nezápornost funkce f – nemůžeme mít negativní světelnou intenzitu. Hodnota $f(x,y)$ odpovídá světlosti obrazu na (prostorových) souřadnicích (x,y). Bod 1. je omezující, ale přirozený požadavek daný strukturou okolního světa. Lidské vnímání je omezeno, stejně tak jednotlivé objekty, které pozorujeme. I v případě nekonečné scény se lidské oko dokáže zaměřit pouze na omezenou část. Požadavek 3. je motivován fyzikálním chováním energie světla.

Poznámka. Definici 1 (odpovídá například fotografii nebo malbě) lze rozšířit i na funkce $f : \mathbb{R}^n \to \mathbb{R}, \quad n \in \mathbb{N}$. Běžně se setkáváme s případy $n = 3$, kde přidáním další proměnné můžeme získat například informace o dalším rozměru (3D model budovy), nebo o vývoji v čase (video) a $n = 4$, kombinace obou předchozích možností (3D video-model srdce).

2D obraz z Definice 1 lze chápat jako spojitý, černobílý (šedotónový) obraz. Běžná snímací zařízení (fotoaparáty, kamery) však nejsou schopna zachytit spojitý obraz. Je tedy přirozené definovat diskretizovaný (digitální) obraz jako dvoudimenzionální funkci na konečném počtu bodů. Struktura diskretizovaného obrazu je dána architekturou snímacího čipu. Tradičně uvažujeme pravidelnou obdélníkovou síť bodů, příslušnou diskretizovanou funkci reprezentujeme jako matici hodnot původní funkce v bodech sítě.

Definice 2. ***Digitálním obrazem*** *rozumíme matici* $\mathcal{I} \in \mathbb{R}^{n\times m}$ *jejíž prvky splňují* $0 \leq \mathcal{I}(i,j) < \infty$*,* $i = 1, \ldots n$*,* $j = 1, \ldots m$*.*

Všechny požadavky z Definice 1 jsou pro digitální obraz splněny. Pozice pixelů jsou také někdy označovány písmeny x a y podle souřadného systému s osami x a y. Jednotlivé prvky $\mathcal{I}(i,j)$ digitálního obrazu nazýváme **pixely**, hodnota

$\mathcal{I}(i,j)$ určuje světlost pixelu na pozici (i,j), někdy označovanou jako **pixelovou hodnotu**. Pro šedotónový obraz se standardně uvažuje

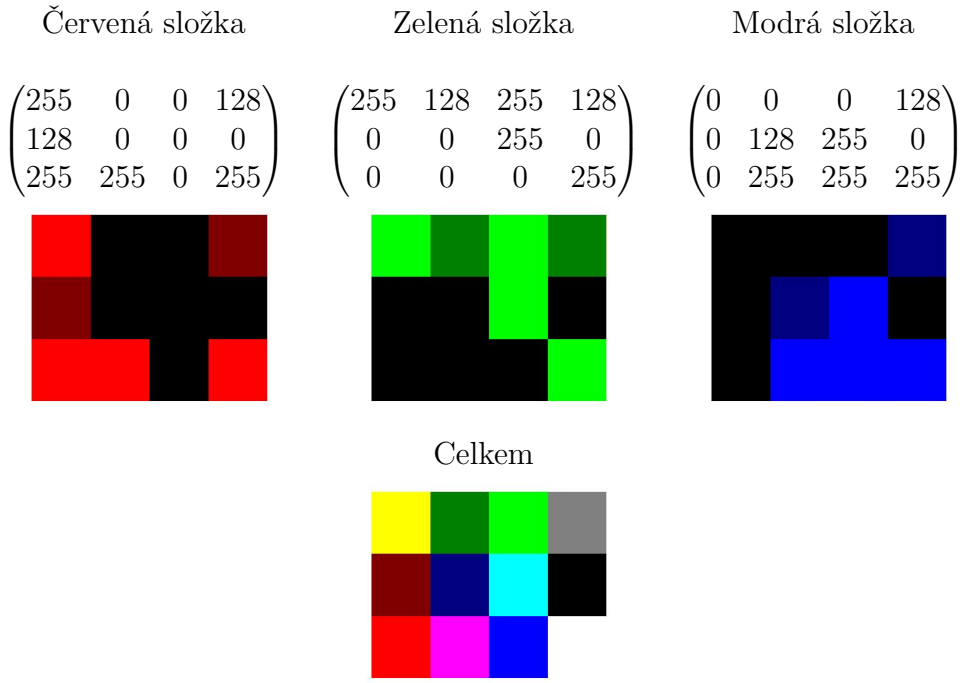
$$\mathcal{I}(i,j) \in \mathbb{N}, \qquad 0 \leq \mathcal{I}(i,j) \leq 255.$$

Nulová hodnota odpovídá černému pixelu, 255 bílému. Rozdělení na 256 světelných intenzit odpovídá maximálnímu rozlišení popsatelnému 8 bity.

Chceme-li pracovat s **barevným obrazem**, můžeme jej reprezentovat jako funkci

$$c : \mathbb{R}^2 \to \mathbb{R}^q, \qquad c = (c^1, \ldots, c^q),$$

kde c^i je 2D obraz a q je počet kanálů – barev. Nejčastěji uvažujeme $q = 3$, jednotlivé složky c odpovídají po řadě červené, zelené a modré složce obrazu (RGB standard). Barevný digitální obraz je pak reprezentován trojicí digitálních obrazů $\mathcal{I}_r, \mathcal{I}_g, \mathcal{I}_b \in \mathbb{R}^{n \times m}$ splňujících Definici 2. Každý obraz zastupuje jednu barevnou složku: červená – zelená – modrá. Příklad jednotlivých složek a složeného barevného obrazu je na Obr. 1.1.



Obrázek 1.1: Vizualizace jednotlivých složek $\mathcal{I}_r, \mathcal{I}_g, \mathcal{I}_b \in \mathbb{R}^{3\times4}$ a výsledný barevný digitální obraz $\mathcal{I}$ složený podle modelu RGB.

Poznámka. Existují i barevné modely nesplňující Definici 2, například HSV (Hue, Saturation, Value), viz [21]. Tento model odpovídá způsobu vnímání barev lidským okem. Jednotlivé složky jsou:

Hue Převládající barevný odstín, měřený na barevném kole (0° – 360°). Obecně se odstín označuje názvem barvy.

Saturation Sytost barvy. Určuje množství šedi v poměru k odstínu. Měří se v procentech: 0 % (šedá) až 100 % (plně sytá barva).

Value Hodnota jasu. Relativní světlost nebo tmavost barvy.

Zavedeme ještě jeden speciální případ digitálního obrazu, který využijeme později.

Definice 3. ***Binárním obrazem*** *nazýváme digitální obraz, jehož prvky $\mathcal{I}(i,j)$ nabývají pouze dvou pixelových hodnot: 0 (odpovídá černé) a 1 (odpovídá bíle barvě).*

Uložení binárního obrazu je méně náročné na paměť, stejně tak výpočetní operace na něm je možné provádět rychleji než na šedotónových, tím spíše barevných, obrazech. V případech, kdy je důležitá pouze struktura snímaných dat, je vhodné volit binární reprezentaci. Příkladem z praxe může být úloha rozpoznávání stíhacích letounů podle siluety. Nyní se zaměříme na některé důležité pojmy a operace, které se používají při zpracování obrazu.

1.2 Konvoluce

Konvoluce je operace na dvou funkcích, která vrací třetí funkci, na kterou lze v některých případech nahlížet jako na modifikovanou verzi jedné ze dvou vstupních funkcí. Uplatňuje se mimo jiné například ve statistice, funkcionální analýze, při analýze diferenciálních rovnic a ve zpracování obrazu. Mimo spojité konvoluce zavedeme také konvoluci dvou matic, kterou budeme využívat v následujících kapitolách.

Definice 4. *Nechť f a g jsou obrazy podle Definice 1.* ***Konvolucí*** *funkcí f a g nazveme funkci $f \star g$ definovanou vztahem*

$$(f \star g)(x,y) := \int_{\mathbb{R}^2} f(r,s)\, g(x-r, y-s)\mathrm{d}r\mathrm{d}s,$$

pro ty $(x,y) \in \mathbb{R}^2$, pro které integrál existuje.

Takto definovaná funkce splňuje požadavky Definice 1 (viz [24, věta 1.5.19]), a je tedy opět obrazem. Dále zavedeme diskrétní konvoluci funkcí dvou proměnných.

Definice 5. *Nechť $\mathfrak{f}, \mathfrak{g} : \mathbb{Z}^2 \to \mathbb{R}$.* ***Diskrétní konvolucí*** *funkcí $\mathfrak{f}(x,y)$ a $\mathfrak{g}(x,y)$ nazveme funkci*

$$(\mathfrak{f} \star \mathfrak{g})(x,y) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \mathfrak{f}(n,m)\mathfrak{g}(x-n, y-m), \qquad (x,y) \in \mathbb{Z}^2.$$

Jestliže je nějaká z funkcí $\mathfrak{f}$, $\mathfrak{g}$ definovaná pouze na podmnožině $\mathbb{Z}^2$, dodefinujeme ji v ostatních bodech nulou.

Konečně bychom chtěli definovat konvoluci dvou matic – digitálních obrazů. Pro naše potřeby budeme v následujícím textu využívat konvoluci obrazu reprezentovaného maticí A s maticí M s menšími rozměry, pro jednoduchost tedy uvedeme pouze tuto variantu. Nechť

$$A \in \mathbb{R}^{n\times m},\ M \in \mathbb{R}^{k\times l}, \qquad \text{kde } k \le n,\ l \le m.$$

Označme $A(x,y) = a_{x,y}$ prvek matice obrazu A na pozici (x,y) a definujme funkce $\mathfrak{f}_A, \mathfrak{f}_M : \mathbb{Z}^2 \to \mathbb{R}$ předpisem:

$$\mathfrak{f}_A(x,y) = \begin{cases} a_{x,y} & \text{pokud } x \in [1,n] \wedge y \in [1,m] \\ 0 & \text{jinak} \end{cases}$$

a analogicky pro $\mathfrak{f}_M$. Konvoluci matic tak lze převést na diskrétní konvoluci funkcí

$$A \star M = \mathfrak{f}_A \star \mathfrak{f}_M : \mathbb{Z}^2 \to \mathbb{R},$$

a využít Definici 5. Jelikož jsme ale původně chtěli provést konvoluci dvou matic, je přirozené požadovat, aby výsledkem byla opět matice. Uvedeme si tři z možných způsobů, jak tohoto dosáhnout (viz [23]).

Poznámka. Dále bude symbol $A \star M$ označovat matici z $\mathbb{R}^{r \times s}$ reprezentující výsledek konvoluce dvou matic. Rozměry matice r a s jsou závislé na rozměrech matic A, M a na zvolené interpretaci konvoluce.

Termín 6. [1] *Plná konvoluce (typu „full“). $A \star M = C \in \mathbb{R}^{(n+k-1)\times(m+l-1)}$, kde*

$$C(x,y) = \sum_{i=1}^{k} \sum_{j=1}^{l} A(x+1-i, y+1-j) M(i,j),$$

pro $x = 1,\ldots,n+k-1$, $y = 1,\ldots,m+l-1$. Hodnoty $A(i,j)$, kde $i < 1, j < 1, i > n$ nebo $j > m$, položíme formálně rovny nule (matici A rozšíříme nulami).

Pro ilustraci uvažujme

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 5 \end{bmatrix} \qquad \text{a} \qquad M = 1/9 * \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}.$$

Vyčíslení prvku matice $A \star M$ na pozici (1,1) pro konvoluci typu „full“ lze pak znázornit následovně:

$$\begin{array}{ccccc} 1/9*0 & 1/9*0 & 1/9*0 & 0 & 0 \\ 1/9*0 & 1/9*0 & 1/9*0 & 0 & 0 \\ 1/9*0 & 1/9*0 & 1/9*1 & 2 & 3 \\ 0 & 0 & 2 & 3 & 4 \\ 0 & 0 & 3 & 4 & 5 \end{array}$$

Nyní posčítáme všechny součiny ($8 * 1/9 * 0 + 1/9 * 1 = 1/9$) a výsledek uložíme na pozici (1,1) matice $A \star M = C$. Výsledná matice C po provedení všech kroků bude tvaru:

$$C = \begin{bmatrix} 1/9 & 1/3 & 2/3 & 5/9 & 1/3 \\ 1/3 & 8/9 & 5/3 & 4/3 & 7/9 \\ 2/3 & 5/3 & 3 & 7/3 & 4/3 \\ 5/9 & 4/3 & 7/3 & 16/9 & 1 \\ 1/3 & 7/3 & 4/3 & 1 & 5/9 \end{bmatrix}. \tag{1.1}$$

[1] Termín nahrazuje Definici v případech, kdy by korektnost zavedení definice byla na úkor srozumitelnosti výkladu.

Termín 7. *Konvoluce zachovávající velikost (typu „same“). Pro jednoduchost uvažujme k, l lichá. $A \star M = S \in \mathbb{R}^{n\times m}$, tedy matice S má stejnou velikost jako A a tvoří podmatici matice C z Termínu 6 tvaru*

$$S(x,y) = C\left(x + \frac{k-1}{2}, y + \frac{l-1}{2}\right),$$

pro $x = 1,\dots,n$, $y = 1,\dots,m$. Zjednodušeně řečeno matice S vznikne z matice C vynecháním prvních a posledních $\frac{k-1}{2}$ řádků a $\frac{l-1}{2}$ sloupců.

Poznámka. Pro k nebo l sudé není jednoznačně dáno, jaké sloupce (řádky) bychom měli vynechat. Například MATLAB vynechá prvních $\frac{k}{2}$ a posledních $\frac{k}{2}-1$ sloupců (stejně tak $\frac{k}{2}$ prvních a $\frac{l}{2}-1$ posledních řádků). V reálných výpočtech obvykle volíme k a l liché.

Výpočet prvku $A \star M$ na pozici (1,1) pro konvoluci typu „*same*“ znázorníme stejně jako v předchozím případě:

$$\begin{array}{cccc} 1/9*0 & 1/9*0 & 1/9*0 & 0 \\ 1/9*0 & 1/9*1 & 1/9*2 & 3 \\ 1/9*0 & 1/9*2 & 1/9*3 & 4 \\ 0 & 3 & 4 & 5 \end{array}$$

Tedy

$$A \star M(1,1) = 5*1/9*0 + 1/9*1 + 2*1/9*2 + 1/9*3 = 8/9.$$

Celá matice $A \star M = S$ je pak

$$S = \begin{bmatrix} 8/9 & 5/3 & 4/3 \\ 5/3 & 3 & 7/3 \\ 4/3 & 7/3 & 16/9 \end{bmatrix},$$

což přesně odpovídá matici C z (1.1) bez prvního a posledního řádku a sloupce.

Termín 8. *Platná konvoluce (typu „valid“). $A \star M = V \in \mathbb{R}^{(n-k+1)\times(m-l+1)}$, přičemž matice V odpovídá těm hodnotám C z Termínu 6, kde nebylo nutné dodefinovávat A nulami, to jest*

$$V(x,y) = C(x+k-1, y+l-1),$$

pro $x = 1,\dots,n-k$, $y = 1,\dots,m-l$.

Výpočet jednoho prvku matice $A \star M = V$ lze znázornit analogicky jako v předchozích příkladech

$$\begin{aligned} V(1,1) &= A \star M(1,1) = 1/9*(1+2+2+3+3+3+4+4+5) = 3 \\ &= C(3,3) = S(2,2). \end{aligned}$$

V MATLABu slouží k počítání konvoluce matic `A` a `M` příkaz `conv2(A,M,'type')`, kde `type` je možné volit jako `full`, `same` nebo `valid`.

1.3 Fourierova transformace

Zatím jsme na obraz nahlíželi jako na body v prostoru s danou intenzitou. Pomocí tohoto intuitivního přístupu si snadno dokážeme představit základní operace s obrazem. Pro určité úlohy je však výhodné obraz chápat jako složení různých frekvencí, podobně jako lze některé reálné funkce rozvinout do báze trigonometrických polynomů. Základům takzvané **analýzy ve frekvenční oblasti** se věnuje například [11], [18] nebo [20]. Dále si popíšeme hlavní nástroj analýzy ve frekvenční oblasti – Fourierovu transformaci.

Definice 9. *Nechť $f : \mathbb{R} \to \mathbb{R}$, $f \in L_1(\mathbb{R})$. **Fourierovou transformací** funkce $f(x)$ nazveme funkci $F(\xi)$ definovanou předpisem*

$$F(\xi) = \int_{-\infty}^{\infty} f(x)\mathrm{e}^{-2\pi\mathrm{i}x\xi}\mathrm{d}x.$$

Operátor Fourierovy transformace přiřazující $F(\xi)$ k $f(x)$ značíme $\mathcal{F}$.

***Diskrétní Fourierovou transformací** N hodnot f_k, $k = 0, \ldots, N-1$ nazveme N hodnot F_n, $n = 0, \ldots, N-1$ definovaných předpisem*

$$F_n = \sum_{k=0}^{N-1} f_k \mathrm{e}^{\frac{-2\pi ink}{N}}.$$

***Inverzní Fourierovou transformací** rozumíme funkci přiřazující transformované funkci F původní funkci f.*

Následující věta popisuje důležitou vlastnost $\mathcal{F}$-transformace a její vztah ke konvoluci. Její důkaz lze nalézt například v [4].

Věta 1 (Hlavní věta konvoluce)**.** *Nechť f,$g : \mathbb{R} \to \mathbb{R}$, f,$g \in L_1(\mathbb{R})$ a nechť F a G jsou jejich odpovídající Fourierovy transformace. Potom*

$$\mathcal{F}(f \star g) = F * G.$$

Díky této větě jsme schopni nahradit výpočetně náročnou konvoluci násobením za cenu jedné $\mathcal{F}$-transformace a jedné inverzní $\mathcal{F}$-transformace. Toho využijeme později při optimalizaci výpočetní náročnosti algoritmu lokalizace kružnice v obraze. K tomu ovšem potřebujeme zavést $\mathcal{F}$-transformaci pro matice a využít maticové varianty Věty 1.

Definice 10. *Nechť $A \in \mathbb{R}^{N\times M}$. Pro $n = 1, \ldots, N$ a $m = 1, \ldots, M$ položme*

$$\mathcal{F}(A)(n,m) = \sum_{k=1}^{N}\sum_{l=1}^{M} a_{k,l}\mathrm{e}^{\frac{-2\pi ink}{N}}\mathrm{e}^{\frac{-2\pi iml}{M}}.$$

Potom matici $\mathcal{F}(A) \in \mathbb{R}^{N\times M}$ s prvky $\mathcal{F}(A)(n,m)$ nazýváme $\mathcal{F}$-transformací matice A.

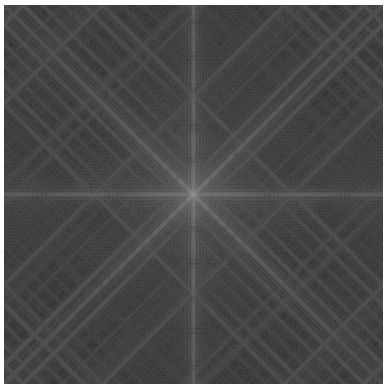
Věta 2. *Nechť A,$B \in \mathbb{R}^{N\times M}$ a nechť $\mathcal{F}(A)$, $\mathcal{F}(B)$ jsou jejich Fourierovy transformace. Potom*

$$\mathcal{F}(A \star B)(n,m) = \mathcal{F}(A)(n,m) * \mathcal{F}(B)(n,m), \tag{1.2}$$

kde $n = 1, \ldots, N$, $m = 1, \ldots, M$ a konvoluce je typu „same“.

Důkaz tohoto tvrzení lze nalézt například v [10]. Stejně jako Věta 1 poskytuje Věta 2 alternativní způsob, jak počítat konvoluci dvou matic. Je důležité si uvědomit, že dle (1.2) se $\mathcal{F}(A \star B)$ nepočítá jako součin matic $\mathcal{F}(A)$ a $\mathcal{F}(B)$. $\mathcal{F}(A \star B)$ dostaneme z $\mathcal{F}(A)$ a $\mathcal{F}(B)$ jejich vynásobením po prvcích.

Příklad použití maticové $\mathcal{F}$-transformace na obraz z Obr. 1 je uveden na Obr. 1.2. Díky symetriím v původním obraze je možné pozorovat symetrie i v jeho $\mathcal{F}$-transformaci. V MATLABu získáme Fourierovu transformaci digitálního obrazu reprezentovaného maticí `A` příkazem `fft2(A)`. Využití $\mathcal{F}$-transformace při zpracování obrazů se budeme více věnovat v dalších kapitolách.

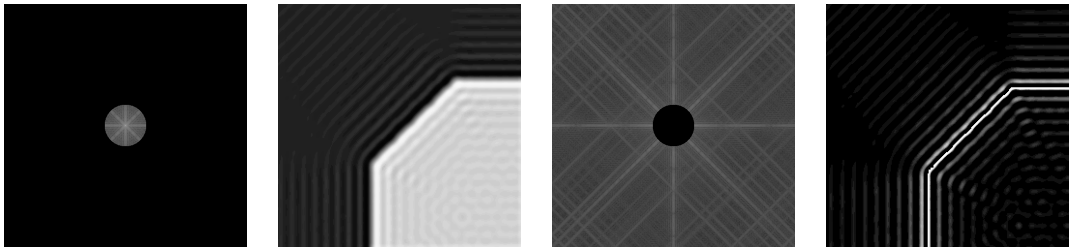


Obrázek 1.2: Výsledek použití maticové verze Fourierovy transformace (Definice 10) na modelový obraz z Obr. 1.

Poznámka. Fourierovu transformaci funkce f můžeme chápat jako její projekci do báze trigonometrických polynomů. Proměnná ξ pak odpovídá frekvenci. Plochy a hrubé obrysy v obraze budou odpovídat nízkým frekvencím, naopak hrany a jemné detaily jsou po $\mathcal{F}$-transformaci reprezentovány vysokými frekvencemi. Tuto skutečnost lze dobře nahlédnout například zkoumáním inverzní diskrétní $\mathcal{F}-$transformace pro různé jednoduché množiny $F_0, \ldots, F_{N-1}$. Detailnější popis je možné najít například v [20]. Na Obr. 1.3 je demonstrováno, jak se změní Obr. 1 upravíme-li jeho $\mathcal{F}$-transformaci (Obr. 1.2) a následně provedeme inverzní transformaci. Nejprve ponecháme pouze nízké frekvence (Obr. 1.3a). Výsledek (Obr. 1.3b) je velmi blízký Obr.1 – plochy jsou dobře zachovány. Poté naopak z Obr. 1.2 odstraníme nízké frekvence a ponecháme vysoké (Obr. 1.3c). Výsledný obraz (Obr. 1.3d) neobsahuje plochy, zachovány jsou pouze obrysy – hrany.

1.4 Přítomnost šumu v obraze

Podstatnou část reálných dat, kterými se oblast zpracování obrazu zabývá, tvoří fotografie a další snímky získané různými snímacími zařízeními. Při pořizování takových snímků vždy dochází ke zkreslenému zachycení reality například vlivem pohybu fotoaparátu (třesoucí se ruce), chybou přímo na snímači (citlivost na teplo, nedostatečné osvětlení), při ukládání a přenosu dat a podobně. K dispozici tak máme pouze obraz $\mathcal{G}$, který je více či méně poškozenou variantou snímané scény – ideálního obrazu $\mathcal{I}$. Nyní zavedeme zjednodušený, ale pro náš případ dostatečný model chyb v digitálním obraze.



(a) (b) (c) (d)

Obrázek 1.3: a) oříznutí vysokých frekvencí v Obr. 1.2 a b) detail výsledku použití inverzní $\mathcal{F}$-transformace na a). Informace o plochách v obraze je zachována, došlo pouze k rozmazání hran. c) oříznutí nízkých frekvencí v Obr. 1.2 a d) detail výsledku inverzní $\mathcal{F}$-transformace použité na c). Informace o plochách v obraze chybí, zachovány jsou pouze hrany.

Poškození obrazu $\mathcal{G}$ můžeme neformálně rozdělit na **chyby způsobené zařízením**, například kvůli poškozené čočce nebo přehřátí snímacího čipu, a na **chyby způsobené okolnostmi**, jenž vznikají vlivem špatné manipulace se zařízením, například pohyb fotoaparátu při stisknutí spouště. V následující definici myslíme chybami ty, které nejsou způsobené nesprávnou manipulací se snímacím zařízením.

Definice 11. *Nechť $\mathcal{I} \in \mathbb{R}^{n\times m}$ je digitální obraz a $\mathcal{G}$ je tentýž obraz zatížený chybami.* ***Aditivním šumem*** *rozumíme matici $\mathcal{N} \in \mathbb{R}^{n\times m}$ definovanou jako*

$$\mathcal{N} = \mathcal{G} - \mathcal{I}\,.$$

Aditivní šum není digitálním obrazem, neboť může nabývat záporných hodnot. Nulová hodnota šumu odpovídá nepoškozenému pixelu, kladná nebo záporná hodnota šumu značí zesvětlení, respektive ztmavení pixelu. Při vizualizaci šumu upravíme hodnoty tak, aby byly všechny nezáporné, takto modifikovaná matice je již digitálním obrazem. Uvedeme si tři základní modely šumu, detailnější informace jsou uvedeny například v [3].

Gaussovský (bílý) šum Pixelové hodnoty šumu jsou náhodné veličiny s normálním (Gaussovským) rozdělením a jsou navzájem nekorelované – nezávisí na okolních pixelech. Slovem bílý pokládáme střední hodnotu rovnu nule. Rozptyl udává, jak moc může být konkrétní pixel poškozen. Gaussovský šum lze popsat pomocí hustoty příslušné náhodné veličiny x:

$$\rho(x) = \frac{1}{\sqrt{2\pi\sigma}} e^{-(x-\overline{x})^2/2\sigma^2},$$

kde $\overline{x}$ je střední hodnota (tedy pro bílý šum $\overline{x} = 0$) a σ^2 je rozptyl. Hodnoty $\rho(x)$ jsou v určitém smyslu blízké (v závislosti na σ) střední hodnotě $\bar{x}$. Pravděpodobnost, že pro dané x platí $\bar{x} - 2\sigma < \rho(x) < \bar{x} + 2\sigma$, je přibližně 95%. Na Obr. 1.4a je Obr. 1 poškozený Gaussovským šumem, samotný šum a $\mathcal{F}$-transformace zašuměného obrazu - porovnej s Obr. 1.2.

Šum sůl a pepř Tento šum je určen dvěma pravděpodobnostními hodnotami p, $q \in [0,1]$, kde $p + q \leq 1$. Pixelové hodnoty šumu pak nabývají pouze 3 intenzit:

1. MAX s pravděpodobností p – pixel $\mathcal{G}$ je bílý.
2. MIN s pravděpodobností q – pixel $\mathcal{G}$ je černý.
3. Nuly s pravděpodobností $1-p-q$ – pixel $\mathcal{G}$ je nepoškozen.

MAX a MIN jsou takové hodnoty, aby poškozené pixely obrazu $\mathcal{G}$ nabývaly maximální, respektive minimální (nulové) hodnoty. Při vizualizaci šumu sůl a pepř zobrazíme pixely s hodnotou MAX jako bílé, pixely s hodnotou MIN jako černé a pixely s hodnotou nula jako šedé. Příklad takto zašuměného obrazu, příslušného šumu a $\mathcal{F}$-transformace poškozeného obrazu je na Obr. 1.4b.

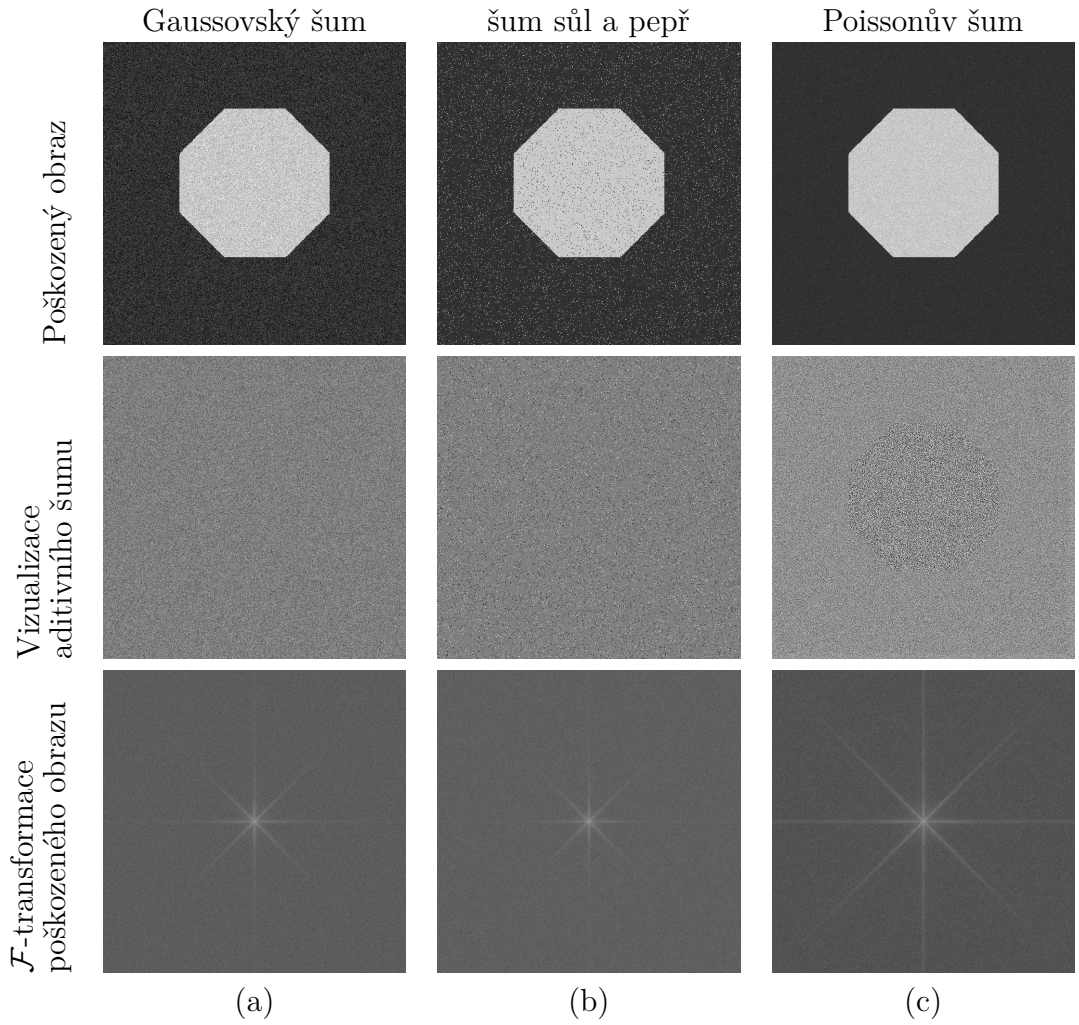
Poissonův šum Pixelové hodnoty šumu jsou náhodné veličiny s Poissonovým rozdělením. Pro tento šum platí, že střední hodnota je rovna rozptylu. Poissonův šum je korelovaný s daty. Hustota náhodné veličiny x popisující Poissonův šum je dána vztahem:

$$\rho(x,\lambda) = \frac{\lambda^x \mathrm{e}^{(-\lambda)}}{x!}, \tag{1.3}$$

kde λ je střední hodnota (a zároveň rozptyl). Modelujeme-li Poissonův šum pro konkrétní obraz $\mathcal{I} \in \mathbb{R}^{n\times m}$, odpovídá x ve vztahu (1.3) hodnotě šumu $\mathcal{N}$ na pozici (i,j) a λ hodnotě pixelu nepoškozeného obrazu na pozici (i,j), $i = 1,\ldots,n, j = 1,\ldots,m$ – odtud přímo plyne korelovanost s daty. Tu je možné pozorovat na Obr. 1.4c.

Poznámka. Mezi další často využívané modely šumu patří takzvané **barevné šumy**, jež lze modelovat za využití Fourierovy transformace Gaussovského šumu. $\mathcal{F}$-transformace bílého šumu má všechny hodnoty přibližně stejných řádů, to znamená, že v šumu jsou zhruba rovnoměrně zastoupeny všechny frekvence. Potlačíme-li vysoké (nízké) frekvence a naopak zesílíme nízké (vysoké), získáme šum, který označujeme například jako červený, růžový, modrý, fialový a podobně, viz například [7]. Názvy barevných šumů jsou odvozeny z viditelného spektra - bílé světlo má zastoupeny všechny frekvence, nízké frekvence odpovídají například červenému a vysoké fialovému světlu.

Poissonův šum nejlépe odpovídá chybám vznikajícím na snímacích senzorech, jako takový je ovšem náročnější na odstranění. Použitím vhodné transformace je možné jej převést na Gaussovský bílý šum, viz [1]. Ten je (podobně jako normální rozdělení ve statistice) často dostatečnou aproximací skutečného šumu, vzniklého například vlivem přehřátí senzoru nebo nedostatečného osvětlení. Šum typu sůl a pepř odpovídá extrémním případům poškození. Buď je pixel nepoškozen, nebo je jeho poškození maximální – pixelová hodnota nabývá maxima nebo minima. Informace o dalších modelech šumu a jeho odstraňování lze nalézt například v [25] a v [20]. V MATLABu je pro testovací účely možné přidat do obrazu `A` šum příkazem `imnoise(A,'type')`, kde `type` je druh šumu: `gaussian`, `salt & pepper` a `poisson` pro výše uvedené šumy.



Obrázek 1.4: Ilustrace modelů aditivního šumu pro Obr. 1. U Gaussovského šumu jsou všechny pixely poškozeny rovnoměrně. Šum sůl a pepř zanesl do obrazu černé a bílé pixely, zbytek pixelů je nepoškozen. U Poissonova šumu je dobře patrná korelovanost s daty.

2. Numerické metody zpracování obrazu

V této kapitole představíme základní matematické metody zpracování obrazu. Zaměříme se zejména na předzpracování, konkrétně uvedeme standardní přístupy k úloze odstranění šumu z obrazu, řešení problému prahování a zmíníme, jak v obraze nalézt hrany.

Vedle modelového obrazu Obr. 1 využijeme k demonstraci popisovaných metod barevnou fotografii [19] a její černobílou variantu získanou pomocí MATLABovské funkce `rgb2gray` viz Obr. 2.1.

(a) (b)

Obrázek 2.1: Testovací obraz reprezentovaný fotografií v a) barevné a b) šedotónové variantě. Rozměry: 2048×1343 pixelů, barevný model: RGB, pixelové intenzity v rozmezí $[0, 255]$.

2.1 Histogram a transformace intenzit

Histogram představuje užitečný nástroj k charakterizaci rozložení barev v obrazu. Jeho znalost nám může pomoct například při identifikaci šumu typu sůl a pepř nebo později při úloze prahování. Podrobnější informace je možno nalézt například v [5].

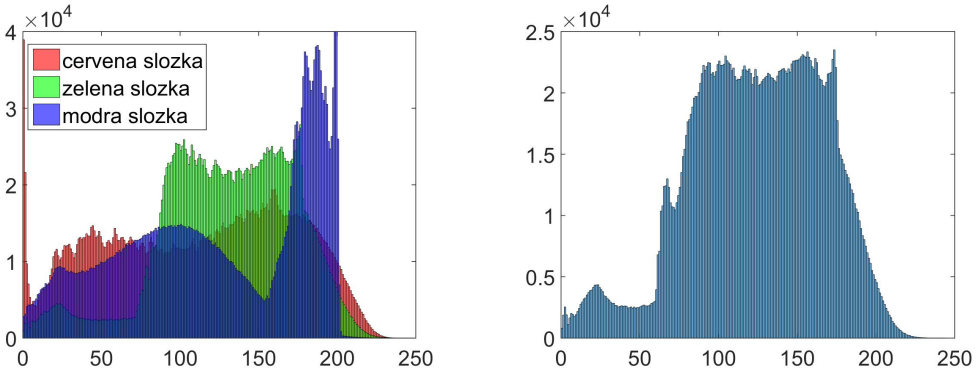
Definice 12. ***Histogramem*** *digitálního obrazu $\mathcal{I} \in \mathbb{R}^{n\times m}$ s celočíselnými pixelovými hodnotami v intervalu $[0, L]$ je diskrétní funkce $h : [0, L] \to \mathbb{N}_0$, kde*

$$h(k) = n_k, \qquad k = 0, \ldots, L,$$

a n_k je počet pixelů obrazu $\mathcal{I}$ se světelnou intenzitou k. ***Normalizací histogramu*** *rozumíme podělení hodnot $h(k)$ celkovým počtem pixelů $n * m$.*

V MATLABu lze histogram obrazu $\mathcal{I}$ získat příkazem `imhist(I)`. Na Obr. 2.2a je vykreslen histogram jednotlivých barevných složek referenčního Obr.2.1a a histogram jeho černobílé varianty Obr. 2.1b.

Často je žádoucí pixely obrazu zařadit do skupin podle jejich intenzity. K tomu zavedeme následující dva pojmy.



(a) (b)

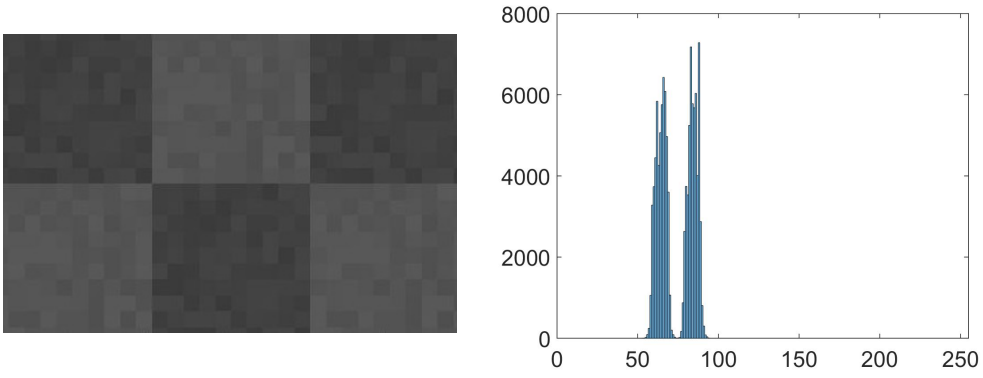
Obrázek 2.2: Histogram a) jednotlivých RGB složek Obr. 2.1a a b) černobílého obrazu Obr. 2.1b.

Termín 13. ***Hladinou intenzit*** (α,β) *nazveme pixely digitálního obrazu* $\mathcal{I} \in \mathbb{R}^{n\times m}$ *splňující* $\alpha \leq \mathcal{I}(i,j) \leq \beta$*, kde* α*,* $\beta \in \mathbb{N}_0$*,* $\alpha < \beta$ *jsou konstanty vymezující konkrétní hladinu.*

Dominantní skupinou *nazveme takovou hladinu intenzit* (α,β)*, že počet pixelů hladiny intenzit* (γ,δ) *s* $\gamma < \alpha$*,* $\delta > \beta$ *není „výrazně větší“ než v případě hladiny intenzit* (α,β)*.*

Na Obr. 2.3 je ukázán příklad obrazu s dvěma dominantními skupinami intenzit a jeho histogram.

Poznámka. Termíny 13 jsou zavedeny velmi volně. Hladinu intenzit si lze představit jako ty pixely obrazu, které mají „podobnou“ barvu. Dominantní skupiny vznikají například v obrazech s velkými plochami pixelů se stejnou (podobnou) intenzitou. Pro další popis histogramu viz například [23].



(a) (b)

Obrázek 2.3: Obraz $\mathcal{I} \in \mathbb{R}^{n\times m}$ s dvěma dominantními skupinami světelných intenzit a jeho histogram.

Na základě analýzy a následné úpravy histogramu je založeno několik metod pro úpravu obrazu. Některé z nich si popíšeme.

Definice 14. *Nechť $\mathcal{I} \in \mathbb{R}^{n\times m}$ je digitální obraz,* $\mathfrak{t}$ *je reálná funkce,* $\mathfrak{t} : \mathbb{R} \to \mathbb{R}^+$. *Dále nechť $\mathcal{J} \in \mathbb{R}^{n\times m}$ je digitální obraz, který vznikne z $\mathcal{I}$ pomocí funkce* $\mathfrak{t}$ *následujícím způsobem:*

$$\mathcal{J}(i,j) = \mathfrak{t}(\mathcal{I}(i,j)),$$

$i = 1, \ldots, n$, $j = 1, \ldots, m$. *Pak $\mathcal{J}$ nazýváme* ***transformovaným obrazem*** *a* $\mathfrak{t}$ *nazýváme* ***transformací intenzit****.*

Pomocí transformace intenzit můžeme popsat některé operace měnící pixelové hodnoty obrazu. Například zesvětlení obrazu odpovídá transformace

$$\mathfrak{t}(r) = r + a, \qquad a \in \mathbb{R}^+,$$

kde a určuje, o kolik zvýšíme hodnotu (světlost) každého pixelu. Podobně negativ obrazu s celočíselnými intenzitami v intervalu $[0, L]$, viz Obr. 2.4, lze získat použitím transformace $\mathfrak{t}$ definované předpisem:

$$\mathfrak{t}(r) = L - r, \qquad r \in \mathbb{N}_0 \cup [0, L]\,.$$

Obrázek 2.4: Negativ testovací fotografie z Obr. 2.1b získaný pomocí transformace intenzit $\mathfrak{t}(r) = 255 - r,\ r \in [0, 255]$.

Další používané transformace jsou například:

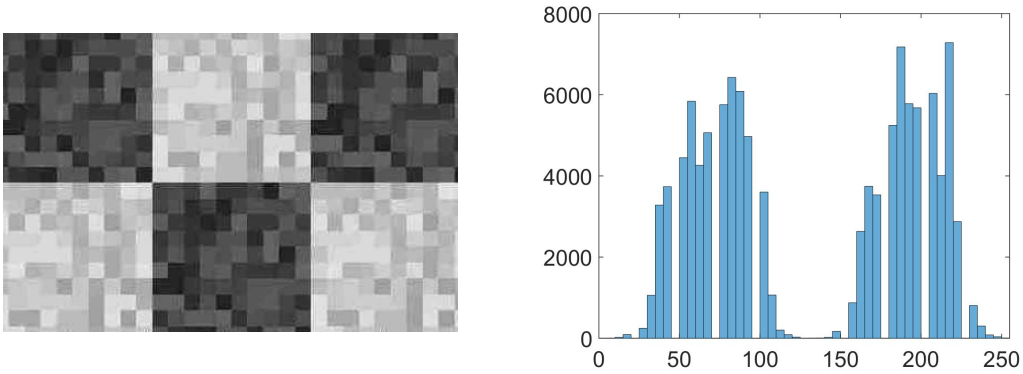
Logaritmická transformace $\mathfrak{t}(r) = c{*}log(1{+}r)$, $c \in \mathbb{R}^+$. Zlepšuje rozlišitelnost nízkých intenzit. Například pro $c = 106$ a obraz s pixelovými hodnotami v intervalu $[0, 255]$ budou pixely s hodnotami $0, 1, 2, 3, 4$ a 5 v transformovaném obraze hodnoty po řadě $0, 32, 51, 64, 74$ a 82.

Exponenciální transformace $\mathfrak{t}(r) = \mathrm{e}^{b*r} - 1$, $b \in \mathbb{R}^+$. Opak logaritmické transformace. Použijeme ji, chceme-li zlepšit rozlišitelnost světlých pixelů.

Lineární transformace $\mathfrak{t}(r) = \left[\frac{L}{b-a} * r - \frac{a*L}{b-a}\right]$, $a,b \in [0, L]$, $r \in [a, b]$. Lineárně transformuje pixelové hodnoty obrazu z intervalu $[a,b]$ na celočíselné hodnoty v intervalu $[0, L]$. Pro $\mathcal{I} \in \mathbb{R}^{n\times m}$ lze například zvolit

$$a = \min_{i=1,\ldots,n, j=1,\ldots,m} \mathcal{I}(i,j), \qquad b = \max_{i=1,\ldots,n, j=1,\ldots,m} \mathcal{I}(i,j).$$

Po transformaci dostaneme obraz s celočíselnými pixelovými hodnotami v intervalu $[0, L]$.



(a) (b)

Obrázek 2.5: Použití lineární transformace intenzit $\mathfrak{t}(r) = \left[\frac{255}{50} * r - \frac{50*255}{50}\right]$ na Obr. 2.3a a histogram transformovaného obrazu. Porovnej s histogramem na Obr. 2.3b.

Výše popsané transformace samozřejmě ovlivní histogram obrazu. Na Obr. 2.5 vidíme použití lineární transformace na Obr.2.3a a příslušný histogram. Ten je modifikován stejným způsobem jako samotný obraz.

2.2 Odstranění šumu

Doposud jsme předpokládali, že máme k dispozici ideální (nepoškozený) obraz. Nyní si popíšeme, jak ze získaného obrazu $\mathcal{G}$ zatíženého chybami odstranit šum pro některé modely šumu, viz Sekce 1.4 této práce. Zmíníme základní přístupy, které budeme dále používat v naší aplikaci, a to konkrétně konvoluci a mediánový filtr. Podrobnější popis dalších druhů šumu a různých metod jejich odstranění lze nalézt v [5].

Uvažujme obraz $\mathcal{I}$ a jeho pixel na pozici (x,y) takový, že jeho sousední pixely mají „podobnou“ hodnotu. Poškodíme nyní tento pixel aditivním šumem, čímž získáme obraz $\mathcal{G}$, a budeme se snažit restaurovat jeho hodnotu. Jestliže nemáme žádnou informaci o šumu (nebo jeho modelu) v obraze, zkusíme hodnotu poškozeného pixelu odvodit z hodnot sousedních pixelů. Uvedeme nyní dvě metody (někdy také označované jako **filtry**) pro odstranění šumu založené na předpokládané podobnosti sousedních pixelů.

Konvoluční filtr

Jedna z možností, jak pomocí sousedních pixelů určit novou hodnotu pixelu na pozici (x,y), je jednoduše použít průměr hodnot. Ten jsme schopni realizovat pomocí konvoluce následujícím způsobem:

Uvažujme matici M, takzvanou **masku**, definovanou jako

$$M = \frac{1}{9}\begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{2.1}$$

a spočtěme konvoluci $S = \mathcal{G} \star M$ typu „*same*“ viz Sekce 1.2. Na pozici $S(x,y)$ je pak uložen průměr hodnot pixelů obrazu $\mathcal{G}$ sousedících s pixelem na pozici (x,y) a hodnoty $\mathcal{G}(x,y)$. Tímto nahradíme poškozený pixel hodnotou vycházející z hodnot okolních (doufejme nepoškozených) pixelů. Je třeba zdůraznit, že během konvoluce takto modifikujeme všechny pixely, původně nepoškozené pixely tak mohou nově nabýt jinou, nesprávnou hodnotu. To odpovídá situaci, kdy nevíme, které pixely jsou poškozené a které nikoliv. Skutečnost, že do průměru započítáváme i hodnotu opravovaného pixelu, pomáhá zmírnit negativní dopad konvolučního filtru na nepoškozené pixely.

Samozřejmě můžeme použít jinou masku, než je matice M z (2.1). Standardně se používají čtvercové matice velikosti k, kde k je malé liché číslo a součet prvků matice je roven jedné. Obecný konvoluční filtr má následující podobu:

1. Zvol vhodnou velikost konvoluční masky M. Typicky se volí masky velikosti 3×3, 5×5 a 7×7 pixelů.
2. Zvol podobu masky. Zde se jednotlivé filtry liší. Typicky jsou všechny prvky masky nezáporné a jejich součet je roven jedné.
3. Odšuměný obraz $\tilde{\mathcal{I}}$ je roven $\mathcal{G} \star M$ typu „*same*“.

Metoda 1: Odšumování pomocí konvoluce

Vedlejší nežádoucí efekt konvolučního filtru s maskou M z (2.1) je rozmazání hran v obraze. Dobře pozorovatelný je tento jev v případě rohů obrazu $\mathcal{G}$. Uvažujme například obraz čtverce $\mathcal{J}$,

$$\mathcal{J} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 255 & 255 & 255 \\ 0 & 0 & 255 & 255 & 255 \\ 0 & 0 & 255 & 255 & 255 \end{pmatrix}$$

Výsledek konvoluce $S = \mathcal{J} \star M$ na pozici (3,3) je roven

$$S(3,3) = 1/9(5 * 0 + 4 * 255) = 133,\bar{3},$$

tedy nový pixel bude hodnotou blíže k pozadí než ke čtverci. Jednoduchou možnou opravou je použití masky $\tilde{M}$, jejíž rohové hodnoty na pozicích (1,1), (1,k), (k,1) a (k,k) jsou rovny nule a ostatní prvky mají hodnotu $1/(k^2 - 4)$, například

$$\tilde{M} = \frac{1}{5} \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}. \tag{2.2}$$

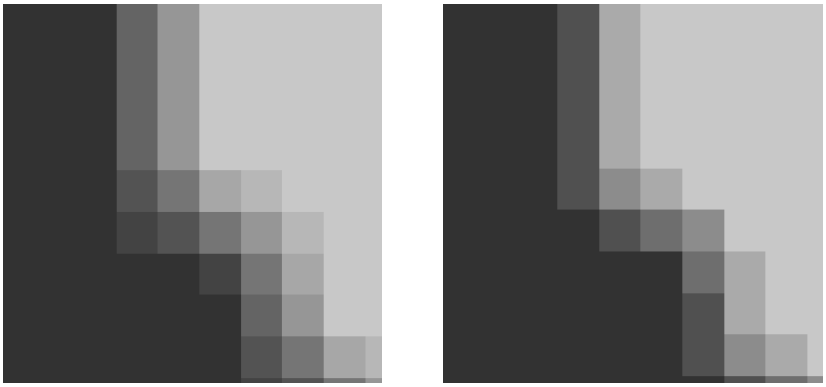
Hodnota nově spočítané konvoluce $\tilde{S} = \mathcal{J} \star \tilde{M}$ na pozici (3,3) je

$$\tilde{S}(3,3) = 1/5(2 * 0 + 3 * 255) = 153,$$

tedy hodnota blíže hodnotám čtverce. Na Obr. 2.6 je porovnání výsledku konvoluce Obr. 1 s maskou M z (2.1) (Obr. 2.6a) a s maskou $\tilde{M}$ z (2.2) (Obr. 2.6b).

U Obr. 2.6b je patrné lepší zachování hrany mezi kamenem (světlá barva) a pozadím (tmavá barva).

V MATLABu lze konvoluci obrazu A s maskou M počítat pomocí příkazu `C = conv2(A,M)`.



Obrázek 2.6: Detail konvoluce modelového obrazu Obr. 1 s maskami a) M z (2.1) a b) $\tilde{M}$ z (2.2). U obrazu vpravo si lze všimnout ostřejšího přechodu mezi tmavými (pozadí) a světlými (kámen) odstíny.

Mediánový filtr

Mediánový filtr funguje podobně jako konvoluční filtr, namísto konvoluce však počítá pro každý bod medián hodnot ze zvoleného okolí. Prvek odšuměného obrazu $\tilde{\mathcal{I}}$ na pozici (i,j) zde získáme jako

$$\tilde{\mathcal{I}}(i,j) = med\{\mathcal{I}(x,y) \;\; x = i-k,\ldots,i+k,\;\; y = j-l,\ldots,j+l\},$$

kde med značí medián počítaný z okolí pixelu na pozici (i,j) a k,l udává velikost okolí.

Oproti konvolučním filtrům mediánové filtry lépe odstraňují šum typu sůl a pepř a lépe zachovává hrany mezi dvěma plochami (viz [23]). To je dáno tím, že hodnota mediánu není tak výrazně ovlivněna extremálními hodnotami zpracovávaných dat. Oba filtry selhávají na obrazech obsahujících křivky tloušťky 1 pixel a podobné jemné detaily, tyto prvky obrazu jsou odstraněny spolu s šumem.

Výsledek mediánového filtrování lze v MATLABu spočítat pomocí příkazu `B = medfilt2(A,[k l])`, kde A je poškozený obraz a $[k\ l]$ udává velikost okolí, přes které se medián počítá.

2.3 Prahování

Uvažujme digitální obraz $\mathcal{I} \in \mathbb{R}^{n\times m}$ tvořený světlými objekty na tmavém pozadí. Příslušný histogram (viz Sekce 2.1) takového obrazu je pak složen ze dvou dominantních skupin intenzit jako na Obr. 2.3. Přirozenou cestou, jak rozlišit objekty od pozadí, je nastavení hranice, která oddělí zmiňované dominantní skupiny intenzit. Pixelům s intenzitou větší než zvolená hranice pak budeme říkat obraz (pixely obrazu), zbylým říkáme pixely pozadí.

Definice 15. *Nechť $\mathcal{I} \in \mathbb{R}^{n\times m}$ je digitální obraz a $T \in \mathbb{R}^+$.* ***Oprahovaným obrazem*** *nazveme binární obraz*

$$B(i,j) := \begin{cases} 1 & \text{pokud } \mathcal{I}(i,j) > T \\ 0 & \text{pokud } \mathcal{I}(i,j) \leq T\,. \end{cases}$$

Konstantu T nazýváme ***prahem*** *(threshold), o přiřazení $\mathcal{I} \to B$ mluvíme jako o* ***prahování*** *(thresholding).*

Poznámka. Je možné prahovat obraz nezápornou funkcí $\mathcal{T}$ místo konstantou. V takovém případě mluvíme o **proměnném prahování** [5]. Prahování v Definici 15 se někdy označuje jako **globální prahování**.

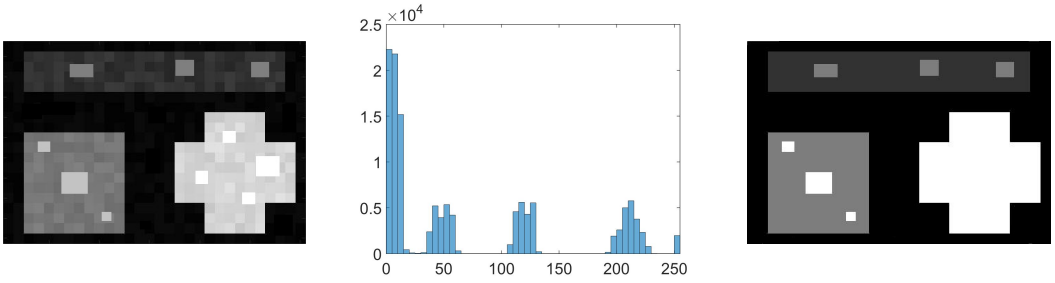
Může nastat situace, kdy má histogram obrazu $\mathcal{I}$ více než dvě dominantní skupiny světelných intenzit, které odpovídají například několika různě světlým objektům na tmavém pozadí. V takovém případě je k dobrému rozlišení jednotlivých objektů nutno přejít k **multithresholdingu**, kde každému objektu náleží určitý interval světelných intenzit.

Definice 16. *Nechť $\mathcal{I} \in \mathbb{R}^{n\times m}$ je digitální obraz a T_1,...,$T_n \in \mathbb{R}^+$.* ***Multithreshold obrazem*** *nazveme digitální obraz*

$$B(i,j) := \begin{cases} a_n & \text{pokud } \mathcal{I}(i,j) > T_n \\ a_{n-1} & \text{pokud } T_{n-1} < \mathcal{I}(i,j) < T_n \\ \vdots & \\ a_1 & \text{pokud } \mathcal{I}(i,j) \leq T_1\ , \end{cases}$$

kde $a_1, \ldots, a_n \in \mathbb{R}^+$ jsou navzájem různé hodnoty světelné intenzity.

Příklad obrazu s více dominantními skupinami, jeho histogram a výsledek prahování zobrazuje Obr. 2.7.



(a) (b) (c)

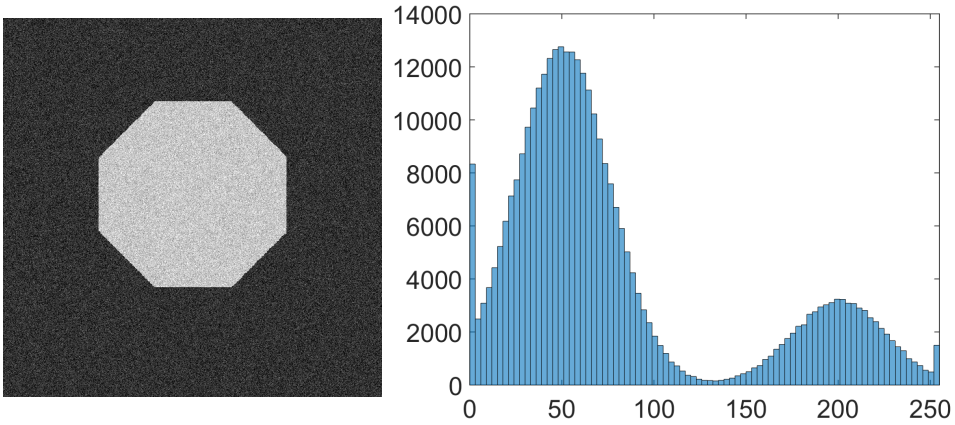
Obrázek 2.7: a) obraz s více dominantními skupinami intenzit, b) jeho histogram, c) obraz a) oprahovaný pomocí prahů určených histogramem.

V reálných výpočtech je úloha prahování více než jedním prahem složitá. Většinou se používají jiné postupy jako proměnné prahování. Úspěch prahování záleží mimo jiné na následujících faktorech:

- Velikost dominantních skupin a jejich vzdálenost – čím jsou skupiny vzdálenější, tím lepší bude výsledek separace objektů a pozadí.
- Velikost objektů na obrazu – drobné objekty složené z několika málo pixelů nevytvoří dominantní skupinu.
- Přítomnost šumu v obrazu – tou se budeme zabývat nyní.

Vliv šumu na prahování

Uvažujme obraz $\mathcal{G}$ poškozený náhodným šumem $\mathcal{N}$ (například Gaussovský bílý šum), viz Sekce 1.4. Ten zanese do obrazu světelné intenzity, které se v originálním nepoškozeném obrazu $\mathcal{I}$ nemusely vyskytovat. Čím více je pixelů určité intenzity v nezašuměném obrazu $\mathcal{I}$, tím je větší šance, že některé z nich budou změněny šumem. Dochází tak k „vyhlazení“ histogramu obrazu, kde málo zastoupené intenzity obrazu $\mathcal{I}$ jsou nyní zastoupeny více a obráceně. V závislosti na množství šumu může dojít ke „slití“ dominantních skupin světelných intenzit a tím ke znemožnění separace pomocí prahování. Na Obr. 2.8 je obraz z Obr. 1 poškozený Gaussovským bílým šumem a příslušný histogram.



Obrázek 2.8: Obraz Obr. 1 poškozený Gaussovským bílým šumem a jeho histogram. V obraze jsou zastoupeny všechny světelné intenzity, přestože pixely originálního (nepoškozeného) obrazu nabývají pouze dvou hodnot.

Poznámka. Uvažujeme-li obraz $\mathcal{G} \in \mathbb{R}^{n\times m}$ s celočíselnými pixelovými hodnotami v intervalu $[0, L]$ poškozený šumem typu sůl a pepř, projeví se změna histogramu v závislosti na množství šumu vytvořením dvou dalších dominantních skupin – jedné pro pixelovou hodnotu 0, to jest černou, a druhé pro hodnotu L, to jest bílou barvu. Stále bude možné obraz oprahovat. Navíc výsledný oprahovaný obraz bude stejný, jako kdybychom oprahovali nepoškozený obraz a pak přidali šum. To je zřejmé, uvědomíme-li si, že prahování nemění černé a bílé pixely.

Volba prahu

Nyní se zaměříme na metody volby prahu T pro různé obrazy. První uvedená metoda popisuje postup volby prahu, který funguje spolehlivě pro obraz s dosta-

tečně separovanými dominantními skupinami histogramu. V obecném případě je lepší použít sofistikovanější metody. V druhé polovině této části si jednu z nich, konkrétně Otsuovu metodu, uvedeme. Obě popisované metody nepoužívají žádnou apriorní informaci nebo interakci uživatele.

1. Zvol hodnotu prahu T. Dobrou počáteční volbou je průměrná světelná intenzita obrazu $\mathcal{I}$, tedy $T = \frac{1}{n*m} \sum_{i=1}^{n} \sum_{j=1}^{m} \mathcal{I}(i,j)$.
2. Oprahuj obraz podle Definice 15. Tím vzniknou dvě skupiny pixelů, a to B_1 pixely s hodnotou $\leq T$ a B_2 pixely s hodnotou $> T$.
3. Spočítej průměrnou intenzitu m_1 a m_2 pixelů obrazu $\mathcal{I}$ ve skupinách B_1 a B_2.
4. Polož $\tilde{T} = \frac{1}{2}(m_1 + m_2)$. Jestliže je $\left|\tilde{T} - T\right| >$ TOL, kde TOL je zvolená tolerance, potom $T = \tilde{T}$ a opakujeme kroky 2. až 4. V opačném případě je $\tilde{T}$ hledaný práh.

Metoda 2: Iterační metoda volby prahu T

Metoda 2 v každém kroku oprahuje obraz prahem získaným z předchozí iterace a spočte průměrnou intenzitu m_1 pixelů s hodnotou větší než práh a m_2 pixelů s hodnotou menší než práh. Následně nastaví nový práh na hodnotu $\frac{1}{2}(m_1 + m_2)$. Jestliže existuje interval světelných intenzit, které nejsou v obraze zastoupeny nebo jsou zastoupeny pouze malým počtem pixelů a leží-li tento interval mezi dvěma dominantními skupinami světelných intenzit, bude práh nalezený Metodou 2 ležet právě v tomto intervalu.

Na úlohu prahování lze alternativně nahlížet jako na statistický problém minimalizace chyby při rozdělení pixelů do dvou skupin: B_1 pixely s hodnotou menší než práh a B_2 pixely s hodnotou větší než práh. K tomuto problému lze přistoupit využitím **Bayesova pravidla** („Bayes decision rule“) [2], [13]. Na něm je postavena dobře známá **Otsuova metoda**, která je optimální v tom smyslu, že rozptyl intenzit v rámci skupin pixelů B_1 respektive B_2 je minimální a rozptyl intenzit mezi skupinami je maximální. Další vlastnost Otsuovy metody je, že výpočet prahu je založen pouze na histogramu obrazu – jednoduše spočitatelném jednorozměrném poli hodnot. Podrobný popis a odvození Otsuovy metody lze nalézt v [16] a [5]. Otsuova metoda pro obraz $\mathcal{I}$ s celočíselnými pixelovými hodnotami v intervalu $[0, L]$ se dá shrnout následovně:

K vylepšení prahování lze použít několik jednoduchých metod. Pro názornost uvažujme $\mathcal{I} \in \mathbb{R}^{n \times m}$ šedotónový obraz s celočíselnými světelnými intenzitami v rozmezí $[0, 255]$. Nejjednodušší úpravou je takzvané **roztažení histogramu**, tedy využití celého intervalu světelných intenzit. Jedná se vlastně o lineární transformaci intenzit, viz Sekce 2.1. Označme

$$\mathcal{I}_{max} := \max_{i=1,\dots,n, j=1,\dots,m} \mathcal{I}(i,j) \qquad \text{a} \qquad \mathcal{I}_{min} := \min_{i=1,\dots,n, j=1,\dots,m} \mathcal{I}(i,j)$$

1. Spočti histogram obrazu $\mathcal{I} \in \mathbb{R}^{n\times m}$. Označ $p_k = n_k/(n * m)$, $k = 0, \ldots, L$.
2. Spočti kumulativní součty a průměry $P_k := \sum_{i=0}^{k} p_i$ a $m_k := \sum_{i=0}^{k} i * p_i$, $k = 0, \ldots, L$.
3. Spočti rozptyl mezi skupinami $\sigma_k^2 := \frac{(m_L * P_k - m_k)^2}{P_k(1-P_k)}$, $k = 0, \ldots, L$.
4. Optimální práh T je takové k, že $\sigma_k^2 = \max_{i=0,\ldots,L} \sigma_i^2$. Pokud je maxima dosaženo pro více indexů $i_1, \ldots, i_s$, definuj $T := \sum_{j=1}^{s} i_j$.

Metoda 3: Otsuova metoda volby prahu T

po řadě největší a nejmenší pixelovou hodnotu obrazu $\mathcal{I}$. Upravený obraz je dán po prvcích následovně:

$$\tilde{\mathcal{I}}(i,j) = \left[\frac{(\mathcal{I}(i,j) - \mathcal{I}_{min}) * 255}{\mathcal{I}_{max} - \mathcal{I}_{min}} \right],$$

$i = 1, \ldots, n$, $j = 1, \ldots, m$. Takto můžeme jednoduše zlepšit prahování pro obrazy obsahující světelné intenzity pouze z malého podintervalu $[0, 255]$. Jako ilustrační příklad lze použít Obr. 2.3a a 2.5a, což je Obr. 2.3a po úpravě popsané výše. Porovnáním jejich histogramů (Obr. 2.3b a 2.5b) ihned vidíme výraznější oddělení dominantních skupin. V případě zašuměného obrazu je vhodné a často i nezbytné před začátkem prahování šum odstranit.

Prahování barevného obrazu

Na závěr se pokusíme oprahovat fotografii z Obr. 2.1, u které je potřeba zvolit trochu odlišný přístup k prahování. Fotografie obsahuje snadno rozpoznatelnou budovu, která se jeví tmavší než okolí. Očekáváme tedy, že výsledkem prahování bude binární obraz s bílými pixely na pozadí a černými pixely na místě budovy. Pokusíme-li se prahovat černobílý obraz Obr. 2.1b, nedostaneme uspokojivý výsledek pro žádný práh $T \in [0, 255]$. U fotografií je tento jev poměrně častý – jasné rozlišení pomocí barev neznamená dobré rozlišení pouze na základě intenzit. Naopak, může se stát, že různě barevné plochy budou mít v černobílém obraze stejnou intenzitu. Pokusíme se tedy o oprahování barevného obrazu $\mathcal{I}$ z Obr. 2.1a s pomocí jeho jednotlivých barevných složek. Na Obr. 2.9a, b, c jsou vykresleny jednotlivé barevné složky $\mathcal{I}_{red}, \mathcal{I}_{green}, \mathcal{I}_{blue}$ (model RGB) Obr. 2.1 zobrazené jako šedotónové obrazy. Pro červenou a zelenou složku je patrné oddělení louky od budovy, naopak pro modrou složku je zřetelný přechod mezi nebem a zbytkem scény.

Nyní jednotlivé složky oprahujeme prahem T získaném pomocí Otsuovy metody (Metoda 3). Výsledky prahování, to jest oprahované obrazy $\mathcal{I}_{red}, \mathcal{I}_{green}, \mathcal{I}_{blue}$, jsou zobrazeny na Obr. 2.10.

Podle očekávání jsme v červené a zelené složce oddělili louku a v modré nebe od zbytku scény. Protože však chceme získat jedno prahování pro celou barevnou

Obrázek 2.9: Jednotlivé RGB složky $\mathcal{I}_{red}$, $\mathcal{I}_{green}$ a $\mathcal{I}_{blue}$ testovací fotografie Obr. 2.1 s intenzitami zobrazenými ve stupních šedi.

Obrázek 2.10: Obrazy $\mathcal{I}_{red}$, $\mathcal{I}_{green}$ a $\mathcal{I}_{blue}$ z Obr. 2.9 oprahované prahy získanými Metodou 3.

fotografii, spojíme všechny tři složky do obrazu B tak, že černé pixely budou pouze na místě, kde je černý pixel ve všech třech složkách, to jest

$$B(i,j) := \begin{cases} 0 & \text{pokud } \tilde{\mathcal{I}}_{red}(i,j) = \tilde{\mathcal{I}}_{green}(i,j) = \tilde{\mathcal{I}}_{blue}(i,j) = 0 \\ 1 & jinak \end{cases}. \tag{2.3}$$

Tím dostaneme binární obraz, na kterém je oddělena budova a keř (objekty) od oblohy a louky (pozadí), viz Obr. 2.11.

Obrázek 2.11: Oprahovaní testovací barevné fotografie Obr. 2.1a dle (2.3) zohledňující oprahování jednotlivých barevných složek.

2.4 Hranové detektory

V reálném světě je každý předmět vymezen svými fyzickými hranicemi. Stejný koncept bychom chtěli přenést i do obrazů a vymezit objekty na nich jejich hranicí. Zadefinujeme tedy pojem hrany pro digitální obraz.

Termín 17. *Nechť $\mathcal{I} \in \mathbb{R}^{n\times m}$ je digitální obraz.* ***Hraničním pixelem*** *nazveme bod $\mathcal{I}(i,j)$ takový, že platí alespoň jedno z následujících:*

1. $\mathcal{I}(i,j) \ll \mathcal{I}(i+1,j)$ *nebo* $\mathcal{I}(i,j) \ll \mathcal{I}(i-1,j)$
2. $\mathcal{I}(i,j) \gg \mathcal{I}(i+1,j)$ *nebo* $\mathcal{I}(i,j) \gg \mathcal{I}(i-1,j)$
3. $\mathcal{I}(i,j) \ll \mathcal{I}(i,j+1)$ *nebo* $\mathcal{I}(i,j) \ll \mathcal{I}(i,j-1)$
4. $\mathcal{I}(i,j) \gg \mathcal{I}(i,j+1)$ *nebo* $\mathcal{I}(i,j) \gg \mathcal{I}(i,j-1)$.

Body $\mathcal{I}(i, j \pm 1)$ a $\mathcal{I}(i \pm 1, j)$ označují sousední pixely bodu $\mathcal{I}(i,j)$. ***Hranou*** *pak rozumíme souvislou množinu hraničních pixelů.* ***Izolovaný bod*** *je hraniční pixel takový, že žádný ze sousedních pixelů není hraniční.*

Hrany v obraze nesou více informací než místa bez nich, tedy než homogenní plochy a pomalé přechody světelných intenzit. Hrany dále tvoří detaily v obraze a jsou důležité pro detekci objektů a segmentaci – rozdělení obrazu na logické celky. Pro spojitý obraz je hrana (v souladu s terminologií výše) místo s velkým gradientem. V případě diskrétního obrazu nahradíme gradient diferencí, čili rozdílem dvou sousedních hodnot. **Gradient obrazu** je možné spočíst jednoduše pomocí konvoluce se dvěma maskami

$$h = \begin{pmatrix} 1 & -1 \end{pmatrix} \qquad \text{a} \qquad v = \begin{pmatrix} 1 \\ -1 \end{pmatrix}.$$

Hodnota gradientu obrazu v bodě (x,y) je pak

$$\nabla\mathcal{I}(x,y) = \max_{i=1,\dots,n,j=1,\dots,m} \{|(\mathcal{I} \star h)(i,j)|,\ |(\mathcal{I} \star v)(i,j)|\}.$$

Oprahujeme-li takto spočtený gradient celého obrazu, to jest obraz $\nabla\mathcal{I}(x,y) \in \mathbb{R}^{n\times m}$, vhodným prahem, dostaneme takzvaný **Robertsův detektor hran**. Složitější detektory s lepšími vlastnostmi dostaneme například použitím větších konvolučních masek (Sobelův detektor) nebo uvažováním druhých derivací namísto prvních (Marrův detektor). Shrnutí a porovnání jednotlivých detektorů lze nalézt například v [22], podrobnější popis pak například v [14]. Výsledek aplikace hranového detektoru je obraz hran.

Definice 18. *Nechť $\mathcal{I} \in \mathbb{R}^{n\times m}$ je digitální obraz. Příslušným* ***obrazem hran*** *$\mathcal{E}_\mathcal{I}(i,j)$ rozumíme binární obraz definovaný jako*

$$\mathcal{E}_\mathcal{I}(i,j) := \begin{cases} 1 & \text{pokud } \mathcal{I}(i,j) \text{ je hraničním pixelem} \\ 0 & \text{jinak.} \end{cases}$$

Hranový detektor je základem Houghovy transformace – algoritmu pro nalezení konkrétních křivek v obraze. Tuto metodu si podrobněji popíšeme v následující kapitole.

3. Metody lokalizace objektů

V této kapitole se zaměříme na metody lokalizace objektů s důrazem na úlohu nalezení kruhu v obraze. Popíšeme si standardní způsob řešení tohoto problému – Houghovu transformaci – a představíme dva vlastní přístupy pro lokalizaci kruhu využívající některé standardní metody popsané v Kapitolách 1 a 2. Všechny tři uvedené metody následně v Kapitole 4 otestujeme na reálných datech. Cílem je vyřešit následující úlohu: Nalézt v šedotónovém obraze jediný kruh tvořený světlými pixely na tmavém pozadí, diskutovat budeme však i obecnější případy.

3.1 Houghova transformace

První popisovanou metodu – Houghovu transformaci – je možné považovat za standardní, nalezneme ji například za funkcí `imfindcircles` v MATLABu. Houghova transformace vede na celou skupinu robustních metod pro hledání geometrických útvarů v obraze. Odvodíme si tu nejjednodušší – pro nalezení přímek. Hledání kružnice je pak její přímočarou modifikací. Detaily je možné najít například v [12] a [20].

Detekce přímky

Mějme zadaných n bodů v rovině a uvažujme problém nalezení přímky, která prochází co největším počtem těchto bodů. Přímočará možnost řešení je nalézt všech

$$\frac{n(n-1)}{2} \approx n^2$$

přímek a následně zjistit pro každý bod, zda leží na nalezených přímkách. Celkem tedy

$$n\frac{n(n-1)}{2} \approx n^3$$

porovnání. Tento postup je výpočetně velmi náročný. V [9] je popsán alternativní přístup označovaný jako Houghova transformace, který si nyní přiblížíme.

Uvažujme bod $(x_i,y_i) \in \mathbb{R}^2$ v rovině. Každá přímka procházející tímto bodem musí splňovat rovnici $y_i = ax_i + b$ pro nějaké $a,b \in \mathbb{R}$. Přepsáním rovnice do tvaru

$$b = -ax_i + y_i$$

získáme rovnici přímky v prostoru parametrů a,b. Nyní vezměme další bod (x_j,y_j). Jemu také přísluší množina přímek v prostoru parametrů daných rovnicí

$$d = -cx_j + y_j,$$

kde $c,d \in \mathbb{R}$. Jestliže dvě přímky v prostoru parametrů (příslušné k bodům (x_i,y_i) a (x_j,y_j)) nejsou rovnoběžné, protínají se v jednom bodě – označme ho $(\tilde{a},\tilde{b})$. Pro tuto volbu parametrů platí, že oba body (x_i,y_i) a (x_j,y_j) leží na téže přímce

$$y = \tilde{a}x + \tilde{b}.$$

Realizace numerického výpočtu touto cestou ovšem nemusí být vhodnou v situaci, kdy se přímka blíží svislé poloze – tedy $\tilde{a}$ je velké. Počítání s velkými hodnotami se lze vyhnout použitím polární reprezentace přímky

$$x\cos\theta + y\sin\theta = \rho, \qquad \theta \in [0,\pi]\,. \tag{3.1}$$

Úhel θ odpovídá natočení a $\rho \in \mathbb{R}$ odpovídá vzdálenosti přímky od počátku. Pro fixní bod (x,y) tvoří množina parametrů θ,ρ splňující rovnici (3.1) sinusoidu. Označme průsečík dvou sinusoid příslušných bodům (x_i,y_i) a (x_j,y_j) v prostoru parametrů jako $(\tilde{\theta},\tilde{\rho})$. Potom oba body (x_i,y_i) a (x_j,y_j) leží na přímce

$$x\cos\tilde{\theta} + y\sin\tilde{\theta} = \tilde{\rho}\,.$$

Nyní tedy stačí pro každý zadaný bod v rovině spočíst jemu příslušnou sinusoidu a v prostoru parametrů najít takový bod, ve kterém se protíná nejvíce těchto sinusoid.

Výpočet Houghovy transformace spočívá v uvažování pouze konečného počtu možných úhlů $u_i, i = 1\ldots,U$ a vzdáleností od počátku $d_i, i = 1\ldots,D$. Pro ně sestrojíme tak zvanou akumulační tabulku $A \in \mathbb{N}_0^{U\times D}$ tak, že pro každý zadaný bod v rovině a každý uvažovaný úhel u_i spočteme odpovídající vzdálenost ρ, tu zaokrouhlíme k nejbližší uvažované vzdálenosti d_j a aktualizujeme $A(i,j) = A(i,j) + 1$.

Definice 19. *Nechť $U, D \in \mathbb{N}_0$. Nechť $u_i = \frac{2\pi}{U} * i$, $i = 1,\ldots,U$, a $d_i \in \mathbb{R}^+$, $d_1 < d_2 \cdots < d_D$, jsou přípustné úhly a vzdálenosti od počátku. Nechť dále (x_k,y_k), $k = 1,\ldots,n$, je konečná množina bodů v prostoru $\mathbb{R}^2$. Označme $A(i,j)$ počet bodů (x_k,y_k), $k = 1,\ldots,n$, takových, že*

$$d_j = \underset{l=1,\ldots,D}{\operatorname{argmin}} |(x_k\cos u_i + y_k\sin u_i) - d_l|.$$

*Pak matici $A \in \mathbb{N}_0^{U\times D}$ s prvky $A(i,j)$, $i = 1\ldots,U$, $j = 1\ldots,D$ nazveme **akumulační tabulkou**.*

Indexy maximálního prvku akumulační tabulky $(imax, jmax)$ určují přímku procházející největším počtem zadaných bodů danou rovnicí

$$x\cos u_{imax} + y\sin u_{imax} = d_{jmax}.$$

Nalezení „nejsilnějších“ přímek v obraze pomocí Houghovy transformace vyžaduje navíc apriorní použití hranového detektoru (viz sekce 2.4). Schématicky lze celý postup zapsat následovně:

1. Ze zadaného digitálního obrazu $\mathcal{I} \in \mathbb{R}^{n\times m}$ použitím vhodného hranového detektoru získej binární obraz hran $\mathcal{E}_\mathcal{I} \in \mathbb{R}^{n\times m}$. Zde stačí použít jednodušší detektor a hledat pouze výrazné hrany.

2. Zvol vhodné dělení intervalů $[0,\pi]$ a $[0,R]$ na U, respektive D bodů, kde R je vzdálenost dvou protilehlých rohů obrazu. Typicky se volí ekvidistantní dělení.

3. Sestav akumulační tabulku A z Definice 19. Bílé pixely v $\mathcal{E}_{\mathcal{I}}$ odpovídají zadaným bodům.

4. Nalezni indexy (i,j) tak, že

$$A(i,j) = \max_{m=1,\dots,U,n=1,\dots,D} A(m,n).$$

   Přímka, na které leží nejvíce hraničních bodů obrazu $\mathcal{I}$, je dána úhlem $\theta = u_i$ a vzdáleností $\rho = d_j$ pomocí vztahu (3.1).

5. Pro nalezení další „nejsilnější“ přímky jednoduše polož $A(i,j) = 0$ a opakuj bod 4.

Metoda 4: Houghova transformace pro přímky

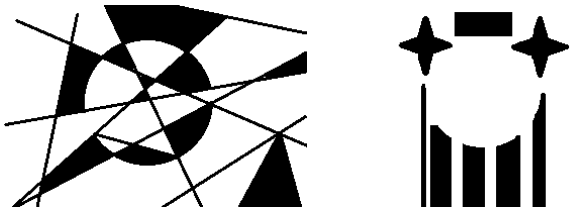
Detekce kružnice

Obdobným přístupem lze detekovat libovolnou křivku popsatelnou rovnicí $f(x,p) = 0$, kde x je vektor souřadnic a p je vektor parametrů. Zaměříme se na detekci kružnice popsatelné rovnicí

$$(x_1 + p_1)^2 + (x_2 + p_2)^2 = p_3^2, \quad p_1,p_2,p_3 \in \mathbb{R}.$$

Oproti postupu popsanému výše nyní pracujeme se třemi parametry. Akumulační matice tedy musí být trojrozměrná. Pro každý zadaný bod a každou dvojici parametrů p_1 a p_2 dopočítáme poslední parametr p_3 a aktualizujeme příslušný prvek akumulační matice.

Je důležité si uvědomit, že pomocí Houghovy transformace nehledáme v obraze kruh, nýbrž kružnici definovanou hranami v obraze. To může být výhoda i nevýhoda. Najdeme například kružnici v Obr. 3.1a, která je neúplná a určená pouze hranami mezi černými a bílými segmenty. Stejně tak ovšem detekujeme kružnici v Obr. 3.1b, přestože se zde žádná kružnice explicitně nenachází.



(a) (b)

Obrázek 3.1: Příklad kružnic detekovatelných Houghovou transformací, a) kružnice definovaná pouze hranami mezi černými a bílými segmenty obrazu, b) příklad obrazu, kde části hran objektů v obraze leží na kružnici.

Výstupem algoritmu je celá akumulační tabulka, máme tedy informace o všech kružnicích v obraze a můžeme tak detekovat více různých výskytů naráz. Hlavní

nevýhodou je velká výpočetní složitost a nutnost dostatečně přesně detekovat hrany. Metoda je poměrně citlivá na přítomnost šumu v obraze – ten přidá nechtěné hrany a pro správné fungování je potřeba jej apriorně odstranit.

Výhody:	**Nevýhody:**
• Nalezne všechny detekovatelné kružnice • Funguje pro všechny obrazy, kde je kružnice definovaná hranou • Funguje i pro neúplné / částečně zakryté kružnice	• Nutnost použít hranový detektor a komplikace s tím spojené • Časová náročnost • Detekuje i nechtěné kružnice

3.2 Konvoluční metoda lokalizace

Další metoda lokalizace objektů je založena na nalezení té části obrazu, která se nejvíce shoduje s daným vzorem. Jedná se o námi navrženou metodu, která je inspirovaná morfologickými operacemi dilatace, eroze a jejich kombinací (viz [23]). Pro správné fungování metody je nutná apriorní znalost poloměru hledaného kruhu.

Konvoluce pro binární obraz

Uvažujme nejprve pro jednoduchost digitální binární obraz $\mathcal{I} \in \mathbb{R}^{n\times m}$ (viz Definice 3), ve kterém chceme detekovat kruh známého průměru k. Vytvoříme si **kruhovou masku** $K \in \mathbb{R}^{k\times k}$ – binární obraz kruhu o průměru k. Tuto masku následně posunujeme po obraze a hledáme nejlepší shodu. Přesněji, počítáme diskrétní konvoluci obrazu $\mathcal{I}$ s kruhovou maskou K (viz Sekce 1.2), výsledek konvoluce je uložen v matici C. Protože je naše maska symetrická, nemusíme ošetřovat otočení masky při konvoluci. Prvky matice C určují míru shody masky a obrazu. Hodnota $C(i,j)$ je rovna počtu světlých bodů obrazu ve výřezu

$$(i,j), (i,j+k-1), (i+k-1,j), (i+k-1,j+k-1),$$

které jsou na odpovídající pozici jako světlé body masky. Maximální hodnota v matici C ukazuje nejlepší shodu. Celou metodu lze zapsat následovně:

1. Pro binární obraz $\mathcal{I} \in \mathbb{R}^{n\times m}$ zvol kruhovou masku $K \in \mathbb{R}^{k\times k}$.
2. Spočti konvoluci $C = \mathcal{I} \star K$ typu „valid“.
3. Spočti
$$M = \max_{i=1,\dots,n,j=1,\dots,m} C(i,j)$$
a množinu všech výskytů M v matici C, to jest
$$\mathcal{M} = \{(x,y), y = 1,\dots,n, y = 1,\dots,m : C(x,y) = M\}.$$

4. Kruh (nebo jeho nejlepší aproximace, viz diskuse dále) se na obrazu nachází ve výřezu (x,y), $(x,y+k-1)$, $(x+k-1,y)$, $(x+k-1,y+k-1)$, kde $(x,y) \in \mathcal{M}$. Jeho střed je na pozici $(x+\left[\frac{k-1}{2}\right], y+\left[\frac{k-1}{2}\right])$.

5. Pro nalezení dalších kruhů polož $C(x,y) = 0$ pro všechny $(x,y) \in \mathcal{M}$ a opakuj od bodu 2.

Metoda 5: Lokalizace kružnice pomocí konvoluce

Označme

$$SUM = \sum_{i,j=1}^{k} K(i,j).$$

Studium hodnoty M a množiny $\mathcal{M}$ umožní rozhodnout o úspěšnosti lokalizace:

(a) Pokud $M = SUM$, máme úplnou shodu. V obraze se nachází hledaný kruh nebo objekt obsahující kruh hledané velikosti.

(b) Pokud $M < SUM$, bez další informace nelze rozhodnout, zda se kruh v obraze nenachází, nebo je pouze jeho část překryta (kruh se v obraze nachází, pouze má některé pixely černé například vlivem šumu).

(c) $\mathcal{M}$ je jednoprvková. Nalezli jsme jediný výskyt kružnice v obrazu (platí-li zároveň (a)), nebo její část (platí-li zároveň (b)).

(d) $\mathcal{M}$ je víceprvková. V obraze se nachází více kruhů (nebo jejich částí), nebo – jestliže se maxima vyskytují na sousedních pozicích – je na obrazu větší objekt (viz bod (a)).

Chceme-li detekovat pouze izolované kruhy konkrétní velikosti, musíme pozměnit masku. Stačí například nulové hodnoty masky změnit na záporné. S takto upravenou maskou je maxima $M = SUM$ možné dosáhnout pouze pro izolovaný kruh správné velikosti. Tím se vyhneme nejednoznačnosti v bodě (a). Tento postup nám může také napovědět, která varianta z bodu (b) nastala. Podoba vhodné masky je určena konkrétní úlohou.

Konvoluce pro šedotónový obraz

Stejný postup lze použít i na šedotónový obraz $\mathcal{I} \in \mathbb{R}^{n \times m}$. Předpokládejme pro jednoduchost, že na daném obrazu se hledaný objekt skutečně nachází a je pouze jeden. Opět spočteme konvoluci C obrazu $\mathcal{I}$ a kruhové masky K. Hodnoty matice C odpovídají součtu pixelových hodnot obrazu $\mathcal{I}$ v kruhovém výřezu určeném maskou K. Indexy (x,y) takové, že

$$C(x,y) = \max_{i=1,\ldots,n, j=1,\ldots,m} C(i,j)$$

pak určují polohu kruhu stejně jako v bodu 4. Metody 5.

Protože musíme postupně projít celý obraz a pro každou pozici spočíst odpovídající součet, je metoda výpočetně náročná. Další komplikace mohou nastat

v případě silně zašuměného obrazu se světlými body na pozadí a naopak tmavšími místy v detekovaném kruhu. Přestože vizuálně bude objekt snadno rozlišitelný, hodnoty součtů mohou být v celém obrazu přibližně stejné a detekce selže. Naopak výhodou je schopnost detekovat i částečně zakryté objekty a vysoká přesnost detekce mimo uměle vytvořené a patologické případy.

Časově nejnáročnější část navržené metody je samotná konvoluce. V Kapitole 1 jsme uvedli Větu 2, která dává návod, jak počítat konvoluci pomocí Fourierovy transformace a násobení matic po prvcích. Tento způsob je výpočetně méně náročný a výrazně tak sníží časové nároky konvoluční metody.

Výhody	**Nevýhody:**
• Nalezne všechny detekovatelné kruhy • Nalezne i kruhy s neostrým okrajem • Velmi přesné pro obrazy s nízkým množstvím šumu	• Neexistuje optimální univerzální maska • Pouze pro apriorně danou velikost kruhu • Vhodné pouze pro specifické aplikace

Poznamenejme ještě, že výstupem konvoluční metody je vždy celá matice C. Z té můžeme snadno odečíst několik největších hodnot a nelézt tak postupně více kruhů. Metoda je běžně schopna nalézt také mírně deformované kruhy. Možné problémy při vyhodnocování výsledku jsme nastínili už výše. Pro konkrétní aplikaci lze obvykle upravit masku a vyhnout se alespoň některým potížím. Nemáme-li k dispozici velikost hledané kružnice, je konvoluční metoda prakticky nepoužitelná – museli bychom spustit algoritmus opakovaně pro různé hodnoty k.

Rozšíření algoritmu na detekci jiných než kruhových objektů je možné pouze v omezené míře. V nezměněné podobě je možné metodu aplikovat pouze na lokalizaci rotačně invariantních objektů o známé velikosti. Použitím složitějších masek lze v obrazech hledat například struktury. Metoda však selhává, požadujeme-li detekci objektů konkrétní nebo proměnlivé světlosti (vždy najdeme nejsvětlejší místo v obraze s daným tvarem).

3.3 1D projekční metoda lokalizace

Omezíme-li se na případ, kdy se v obraze nachází pouze jediný kruh a je to zároveň jediný objekt v obraze, můžeme o problému lokalizace přemýšlet jako o hledání těch sloupců, respektive řádků obrazu, do kterých hledaný kruh zasahuje. Jejich výčtem je kruh již jednoznačně určen. Pokud dokážeme obsáhnout informaci o sloupci, respektive řádku matice digitálního obrazu $\mathcal{I} \in \mathbb{R}^{n \times m}$ jedinou veličinou, redukujeme náš problém na prohledávání dvou polí (jednoho pro řádky a jednoho pro sloupce). Takové redukci dvourozměrného problému budeme říkat projekce. Tento přístup navržený v [17] nyní vylepšíme. Formulujeme si **projektovanou úlohu lokalizace**:

Nechť se v obrazu nachází právě jeden kruh. Nalezněme v obraze dva sloupce a dva řádky, které ohraničují hledaný kruh. Přesněji, hledáme sloupce s indexy l (jako left) a r (right) pro které platí, že sloupec l zasahuje do hledaného kruhu, sloupec $l-1$ už do kruhu nezasahuje, sloupec r zasahuje a $r+1$ nezasahuje do hledaného kruhu. Analogicky definujeme řádky t (top) a b (bottom).

Projekce pro binární obraz

Ke konstrukci metody pro řešení projektované úlohy opět využijeme zjednodušený model. Uvažujme binární obraz $\mathcal{I} \in \mathbb{R}^{n \times m}$ s bílými pixely výhradně na pozici hledaného kruhu. Definujme projekce P_s, P_r dvourozměrného obrazu předpisem

$$P_s(j) = \sum_{i=1}^{n} \mathcal{I}(i,j), j = 1, \ldots, m, \qquad P_r(i) = \sum_{j=1}^{m} \mathcal{I}(i,j), i = 1, \ldots, n. \tag{3.2}$$

Hodnota $P_s(j)$ označuje součet pixelových hodnot v j-tém sloupci, $P_r(i)$ v i-tém řádku. Hledané indexy l,r,t a b jsou definovány jako

$$\begin{aligned} l &= \min\{j \in [1,m] : P_s(j) > 0\}, \qquad & r &= \max\{j \in [1,m] : P_s(j) > 0\}, \\ t &= \min\{i \in [1,n] : P_r(i) > 0\}, \qquad & b &= \max\{i \in [1,n] : P_r(i) > 0\}. \end{aligned} \tag{3.3}$$

Projekce pro šedotónový obraz

V obecném případě šedotónového obrazu nemůžeme bez úpravy použít výše popsaný postup. Upravíme tedy zadaný obraz do podoby, kdy jej použít lze. V Sekci 2.3 jsme popsali, jak převést šedotónový obraz na binární pomocí prahování. Některé pixely pozadí mohou po prahování nabývat hodnoty 1 (bílá), tedy barvy kruhu. Naopak některé pixely kruhu mohou nabývat hodnoty 0 (černá). Pokud jsou tyto nežádoucí elementy přítomny pouze jako izolované body nebo malé skupiny pixelů, lze k jejich eliminaci využít některou z metod odstranění šumu (viz Sekce 2.2). Budou-li pak i nadále mít některé pixely pozadí bílou barvu, už se je nebudeme dále snažit odstranit. Místo toho upravíme způsob hledání hraničních indexů. Při všech úpravách musíme dát pozor, abychom příliš nezměnili velikost hledaného kruhu.

Uvažujme nejprve šedotónový obraz $\mathcal{I} \in \mathbb{R}^{n \times m}$, ve kterém mají všechny pixely pozadí menší hodnotu než všechny pixely kruhu. Pro zlepšení přesnosti lokalizace provedeme nejprve prahování obrazu $\mathcal{I}$. Označíme k nejmenší pixelovou hodnotu kruhu a upravíme obraz $\mathcal{I}$ následujícím způsobem:

$$\tilde{\mathcal{I}}(x,y) = \begin{cases} 1 & \text{jestliže } \mathcal{I}(x,y) \geq k, \\ 0 & \text{jestliže } \mathcal{I}(x,y) < k, \end{cases} \qquad x = 1, \ldots, n, \qquad y = 1, \ldots, m.$$

Pro takto upravený obraz jsou indexy l,r,t a b opět definovány vztahy (3.3). Hodnotu k, pomocí níž prahujeme, musíme buďto znát apriori, nebo ji určit například pomocí histogramu.

Pokud jsou nyní některé pixely pozadí obdrženého obrazu $\mathcal{I} \in \mathbb{R}^{n \times m}$ světlejší než alespoň jeden pixel hledaného kruhu, je třeba provést předzpracování nebo poupravit způsob hledání hraničních sloupců (řádků). Některé z možných úprav jsou:

- Použití mediánových filtrů nebo rozmazání. Cílem je sjednotit pixelové hodnoty v jednotlivých částech obrazu – odstranit světlé body z pozadí a tmavé body z kruhu.

- Nastavení tolerance ε pro určení indexů l,r,t a b. Vztahy (3.3) se upraví následovně

$$\begin{aligned} l &= \min\{j \in [1,m] : P_s(j) > \epsilon\}, \quad r = \max\{j \in [1,m] : P_s(j) > \epsilon\}, \\ t &= \min\{i \in [1,n] : P_r(i) > \epsilon\}, \quad b = \max\{i \in [1,n] : P_r(i) > \epsilon\}. \end{aligned} \tag{3.4}$$

- Kombinace předchozích možností.

Celá metoda pak má následující schématickou podobu.

1. Proveď předzpracování obrazu $\mathcal{I} \in \mathbb{R}^{n \times m}$ způsoby navrženými výše.
2. Spočti řádkové a sloupcové součty $P_r(i)$ a $P_s(j)$, $i = 1,\ldots,n, j = 1,\ldots,m$

$$P_r(i) = \sum_{j=1}^{m} \mathcal{I}(i,j), \qquad P_s(j) = \sum_{i=1}^{n} \mathcal{I}(i,j).$$

3. Nalezni indexy l,r,t a b definované vztahy (3.3), respektive (3.4).
4. Kruh se na obrazu nachází ve výřezu (t,l), (t,r), (b,r),(b,l). jeho střed je na pozici $\left(\left\lceil \frac{b+t}{2} \right\rceil, \left\lceil \frac{r+l}{2} \right\rceil\right)$.

Metoda 6: Projekční metoda lokalizace

Z popisu metody lze nahlédnout nenáročnost výpočtu a dokonce možnost paralelizace výpočtu – sloupce a řádky lze zpracovávat zcela nezávisle. Jednoduchou modifikací metody můžeme také hledat i jiné objekty než kruh. Nedostaneme sice jejich přesnou polohu, ale jednoduchým a rychlým algoritmem určíme obdélníkový výřez obrazu $\mathcal{I}$, ve kterém se objekt nachází. Tuto metodu je možné použít jako předzpracování pro výpočetně náročnější metody. Nalezneme přibližnou polohu kruhu, kterou dále upřesníme použitím jiné metody. Hlavní nevýhodou je potřeba netriviální informace o zpracovávaných datech a výběr vhodného předzpracování.

Výhody:	**Nevýhody:**
• Rychlost výpočtu • Vhodné pro hrubou lokalizaci • Možnost použít i pro jiné objekty než kruh	• Nutnost předzpracování dat • Obecně menší přesnost

3.4 Srovnání metod

Z popisu jednotlivých metod je zřejmé, že pro volbu vhodného přístupu k lokalizaci je důležitá znalost konkrétní úlohy. V obecném případě bychom pravděpodobně volili Houghovu transformaci, jelikož pracuje pouze s hranami a nepotřebuje tak žádnou další apriorní informaci. Tato práce se ovšem zaměřuje na srovnání popsaných metod pro konkrétní úlohu, která je blízká úloze nalezení jediného světlého kruhu na tmavém pozadí. Bez znalosti velikosti kruhu můžeme

vyloučit konvoluční metodu z důvodu velkých výpočetních nároků při nutnosti testovat lokalizaci pro různé volby velikosti objektu. Projekční metoda by měla oproti Houghově transformaci fungovat rychleji. Lépe by si také měla poradit se zašuměným nebo rozmazaným obrazem.

Známe-li pixelový poloměr k hledaného kruhu, můžeme konvoluční metodu použít. Houghovu transformaci je možné urychlit hledáním kruhů daného rozměru. Modifikace je možné provést i pro projekční metodu, kde místo hraničních sloupců (řádků) budeme hledat k sousedních sloupců (řádků) s největším součtem pixelových hodnot. V tomto případě očekáváme nejpřesnější lokalizaci od konvoluční metody, nejrychlejší bude opět projekční metoda.

Jestliže připustíme deformované a neúplně kruhy, nemůžeme od projekční metody nadále očekávat přesný výsledek. Můžeme však získat informaci o deformaci kruhu porovnáním detekované „šířky“ $r - l$ a „výšky“ $b - t$, viz (3.3) nebo (3.4). V případě deformací očekáváme selhání lokalizace i v případě Houghovy transformace. Naproti tomu lze předpokládat spolehlivost konvoluční metody, jejíž výsledek by měl být nejméně ovlivněn nepřesnostmi v tvaru detekovaného objektu.

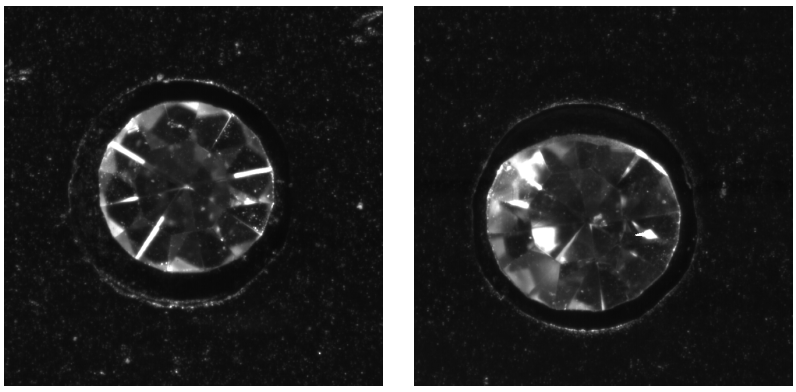
Ověření výše vyslovených tvrzení se budeme věnovat v následující kapitole, ve které otestujeme jednotlivé metody na reálných datech.

4. Návrh řešení

V této kapitole použijeme metody popsané v předchozí části práce pro řešení zadané reálné aplikace – automatické testování kvality bižuterních kamenů. Naším cílem je na sadě testovacích snímků lokalizovat polohu bižuterních kamenů, které mají přibližně kruhový tvar. Pro úspěšně lokalizovaný kámen navrhneme metodu, která rozhodne o jeho kazovosti. Důležitým kritériem je zde rychlost výpočtu. Pro účely testování máme k dispozici následující data:

- 1600 digitálních šedotónových obrazů $\mathcal{I}$ kamenů pořízených řádkovou kamerou. Z toho je 800 kamenů bez výrobní vady a 800 kamenů s vadou.
- Velikost snímků je 651×651 pixelů, pixelové hodnoty jsou celočíselné v intervalu [0,255], to jest $\mathcal{I} \in \mathbb{R}^{651\times651}$, $\mathcal{I}(i,j) \in \mathbb{N}_0$, $0 \leq \mathcal{I}(i,j) \leq 255$.
- Snímaný kámen je zasazen v kalíšku v černé podložce (viz Příloha A). Okraj kalíšku má větší průměr než kámen a na obraze je zřetelný, viz Obr. 4.1a.
- Kámen nemusí být usazen správně – při pohledu shora má chybně usazený kámen jiný než kruhový tvar, viz Obr. 4.1b.
- Na snímku se mohou vyskytovat artefakty, a to odlesky tvořené pixely s hodnotou $\mathcal{I}(x,y) > 250$ mimo lokalizovaný kámen, viz Obr. 4.2a a 4.2d, nebo nečistoty (například prach) na snímací podložce.

Všechny výpočty byly prováděny za stejných podmínek v prostředí MATLAB. Podrobnější popis testovací sady a způsobu snímání kamenů je uveden v Příloze A. Veškeré použité kódy jsou uvedeny v Příloze B.



(a) (b)

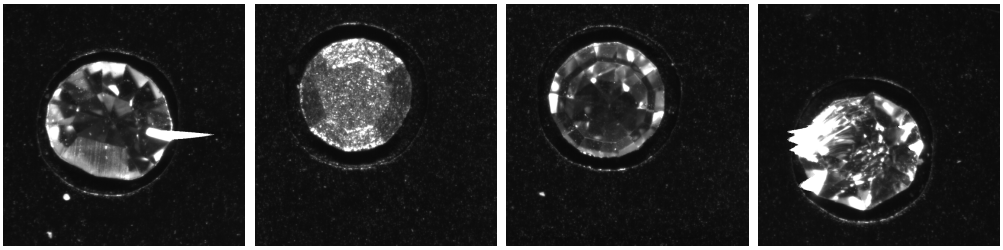
Obrázek 4.1: Příklad (a) správně a (b) křivě usazeného kamene

V první části této kapitoly se zaměříme na porovnání lokalizačních metod popsaných v Kapitole 3. Dále budeme diskutovat možnosti vytřídění špatně usazených a křivých kamenů, které je třeba z testovacího procesu vyřadit. Ve třetím úseku navrhneme řešení separace kamenů založené na jejich celkové světlosti. Kvalitu kamenů budeme v této kapitole rozlišovat následujícími pojmy:

- Kameny, které jsou v pořádku (mohou jít do prodeje) budeme nazývat **dobré**.
- Kamenům s vadou (které je třeba vyloučit) říkáme **špatné**. Vady mohou mít různý charakter, což komplikuje jejich obrazovou detekci. Příklady špatných kamenů jsou na Obr. 4.2. Křivé a křivě usazené kameny považujeme také za špatné.

Při vyhodnocení úspěšnosti metod budeme rozlišovat dva typy chyb:

- **false positive** – vyhodnotíme špatný kámen jako dobrý.
- **false negative** – vyhodnotíme dobrý kámen jako špatný.



(a) (b) (c) (d)

Obrázek 4.2: Příklad špatných (vadných) kamenů: a) kámen s uštíplou hranou, nemá kruhový tvar, b) kámen, který není průsivitný, vada vyleštění, c) vada výbrusu, d) patologicky špatný kámen.

4.1 Porovnání lokalizačních metod

Prvním krokem při třídění kamenů je jejich lokalizace na snímcích $\mathcal{I}$. K tomu využijeme metody popsané v Kapitole 3 a náhodně vybranou množinu 800 snímků. Porovnávat budeme procento správně lokalizovaných kamenů a celkovou dobu výpočtu. Metody byly realizovány následujícím způsobem: Pro Houghovu transformaci jsme použili MATLABovskou funkci `imfindcircles`. Tato funkce přijímá jako parametry interval prohledávaných poloměrů a citlivost. Citlivost ovlivňuje množství nalezených kruhů, s vyšší citlivostí jsou detekovány i částečně zakryté kruhy. Pro projekční a konvoluční metodu jsme navrhli a realizovali vlastní implementaci.

Schéma konvoluční metody:

- Vytvoř kruhovou masku K o zvoleném poloměru k a spočti její Fourierovu transformaci $\mathcal{F}(K)$.
- Spočti $\mathcal{F}$-transformaci $\mathcal{F}(\mathcal{I})$ aktuálního snímku (obrazu $\mathcal{I} \in \mathbb{R}^{n,m}$).
- Spočti konvoluci obrazu a masky pomocí Věty 2, tedy

  $$C = \mathcal{F}^{-1}(\mathcal{F}(\mathcal{I}). * \mathcal{F}(K)),$$

  kde $\mathcal{F}^{-1}$ je inverzní Fourierova transformace a $.*$ značí násobení matic po prvcích.

- Najdi maximální prvek matice C, jeho hodnotu označ M, jeho pozici (i,j), to jest
$$M = \max_{i=1,\dots,n,j=1,\dots,m} C(i,j)$$
- Hledaný kámen má v obraze $\mathcal{I}$ střed v bodě (i,j), poloměr k a celkovou světlost M.

Schéma projekční metody:

- Oprahuj snímek (obraz $\mathcal{I}$) zvoleným prahem T.
- Použij na oprahovaný obraz mediánový filtr zvolené velikosti.
- Spočti řádkové a sloupcové součty upraveného obrazu $\mathcal{I}$, viz (3.2).
- Nalezni indexy levého (l), pravého (r), horního (t) a dolního (b) okraje kamene.
- Střed kamene je v obraze $\mathcal{I}$ na pozici $(\left[\frac{t+b}{2}\right], \left[\frac{l+r}{2}\right])$, šířka kamene je $w = r - l$, výška kamene je $h = b - t$.
- Jestliže je $|w-h| > tol$, kde tol je zvolená tolerance, vrať hlášku „**lokalizace selhala**“. Jinak polož poloměr kamene $r = \left[\frac{w+h}{4}\right]$.

Konkrétně se jedná o Algoritmus 4 pro konvoluční metodu a Algoritmus 1 pro projekční metodu.

V případě Houghovy transformace a projekční metody můžeme jako výstup obdržet hlášku „**lokalizace selhala**“ – to jest algoritmus nenašel v obraze žádnou kružnici. Tento výstup nebudeme považovat za chybu lokalizace, jestliže se jedná o snímek křivého nebo křivě usazeného kamene. Chybnou lokalizací označíme pouze ty případy, kdy algoritmus vrátí nesprávný střed a poloměr kamene.

Testování rozdělíme do dvou částí. V první uvažujeme obecný případ lokalizace kamenů neznámé velikosti, v druhé předpokládáme apriorní znalost pixelových poloměru kamenů v testovací sadě.

Lokalizace bez apriorní informace

Hough:

Kvůli neznámé velikosti kamene předpokládáme problémy s použitím Houghovy transformace, viz Sekce 3.1. Nutnost prohledávat různé poloměry bude zvyšovat časovou náročnost výpočtu. Očekáváme problémy v případě křivě zasazených kamenů – detekován by mohl být okraj kalíšku, který je přesně kruhový a v obraze zřetelný. Interval hledaných poloměrů byl na základě znalosti rozsahu rozměrů testovaných výrobků nastaven na $[100, 200]$ pixelů, skutečné poloměry kamenů se na naší konkrétní testovací sadě pohybují okolo 154 pixelů. Použitý MATLABovský příkaz pro obraz `I`, střed `c` a poloměr `r` je

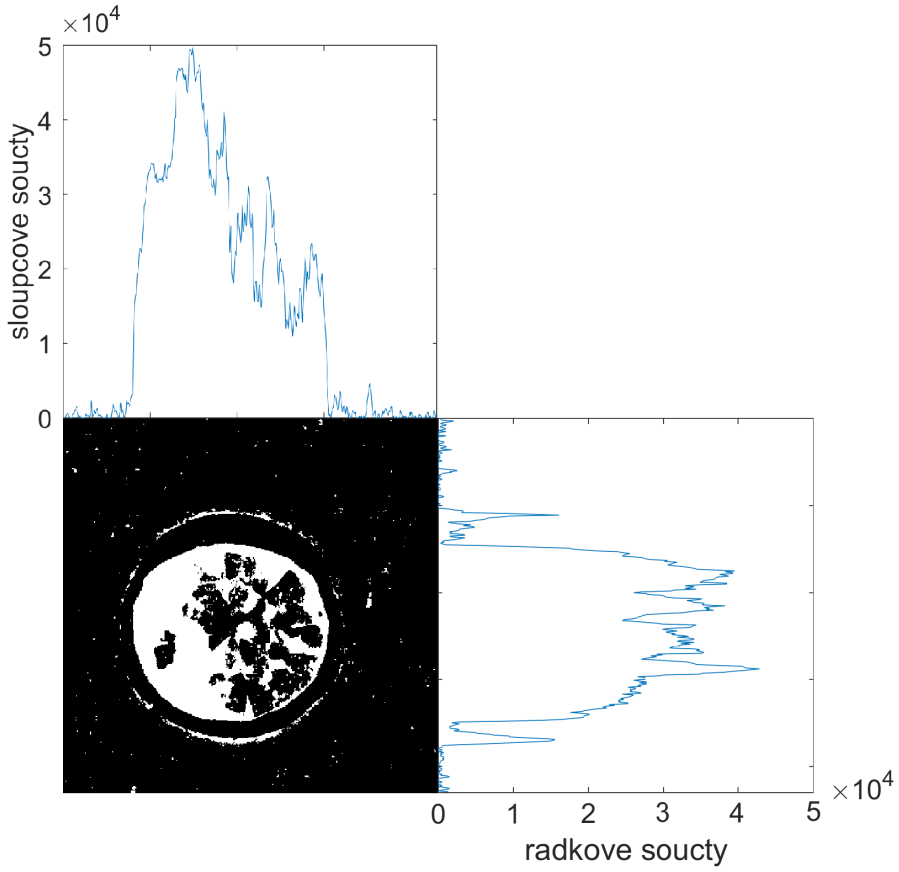
```
[c, r] = imfindcircles(I, [100 200], 'sensitivity', 0.98);
```

Konvoluce:

Detekce pomocí konvoluce, viz Sekce 3.2, implementována jako Algoritmus 4, je v tomto případě nevhodná. Museli bychom použít několik různě velkých konvolučních masek a celý algoritmus by proto byl výpočetně extrémně náročný. Proto se pro neznámý poloměr touto metodou nebudeme dále zabývat.

Projekce:

Projekční metoda, viz Sekce 3.3, implementovaná jako Algoritmus 1, není závislá na znalosti poloměru kamene. Očekáváme (oproti Houghově metodě) nízké nároky na výpočetní čas. Na Obr. 4.3 jsou vykresleny sloupcové a řádkové součty pro kámen z Obr. 4.1b a jeho obraz po aplikaci předzpracování. K předzpracování byl použit mediánový filtr velikosti $[7 \times 7]$, čímž se zbavíme prachu a drobných nečistot na destičce. Nečistoty, které tímto způsobem nedokážeme odfiltrovat, bude snazší odstranit mechanicky před pořízením fotografie. Na obraze stále zůstává zřetelný již zmíněný okraj kalíšku. Obraz jsme dále oprahovali s prahem T nastaveným na hodnotu 15% z maximální možné světlosti 255. Dále jsme nastavili toleranci `tol`, od které je sloupcový (řádkový) součet považován za nenulový, na 7.5% z maxima sloupcových, respektive řádkových součtů.



Obrázek 4.3: Nahoře: Sloupcové součty obrazu kamene z Obr. 4.1b, dole vlevo: obraz vyhlazený mediánovým filtrem velikosti $[7 \times 7]$ a oprahovaný prahem $T = 39$, dole vpravo: řádkové součty stejného obrazu.

Na Obr. 4.3 si všimneme zvýšených hodnot v místech kraje kalíšku. Aby tyto sloupce, respektive řádky nebyly označeny za okraj kamene, hledáme první sloupce se součtem menším než `tol` od sloupce s maximálním součtem namísto od kraje obrazu (analogicky pro řádky):

```
x=sum(A);              % spocteme radkove soucty
[tmp,xmax]=max(x);     % xmax je index sloupce s nejvetsim ...
    souctem
x1=xmax;               % nalezeni leveho kraje kamene
while x(x1)>tol, x1=x1-1;end
x2=xmax;               % nalezeni praveho kraje kamene
while x(x2)>tol, x2=x2+1;end
```

Komplikace by mohly nastat, jestliže by byl okraj kalíšku těsně vedle hrany kamene. Potom bychom detekovali kámen o několik sloupců (řádků) širší, než ve skutečnosti je. Tato chyba se ovšem pohybuje v řádu několika málo pixelů a není pro celkovou detekci zásadní. Celkově očekáváme dobré výsledky. Výstupem algoritmu je navíc proměnná `Valid` s hodnotou `false` v případě, že detekovaná šířka a výška kamene se liší o více než `rtol=5` pixelů.

Srovnání

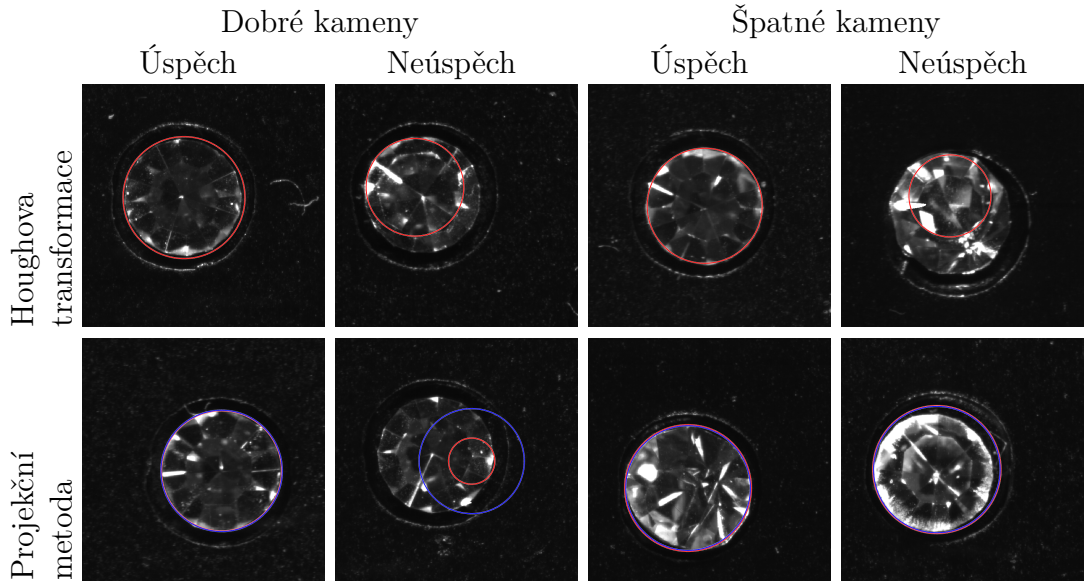
Následující tabulka shrnuje výsledky testu metod pro lokalizaci kamene bez apriorní znalosti poloměru. **Čas** udává celkový potřebný čas pro lokalizaci kruhu na všech 800 testovacích snímcích. **Kružnice nenalezena** udává (Pro Houghovu transformaci a projekční metodu) počet výstupů „lokalizace selhala“, **z toho vadných** je pak počet kamenů, pro které nebyla kružnice nalezena a jsou zároveň křivé nebo křivě usazené. **Chybná lokalizace** je počet kamenů, které byly lokalizovány nesprávně. **Úspěšnost** je počet správně lokalizovaných kamenů spolu s počtem křivých a křivě usazených kamenů, u kterých byl výsledek „lokalizace selhala“.

	Hough	Projekce
Čas	740 s	47 s
Kružnice nenalezena	24/800 (3%)	174/800 (21,7%)
z toho vadných	9/24 (37,5%)	122/174 (70,1%)
Chybná lokalizace	63/776 (8,1%)	4/623 (0,6%)
Úspěšnost	722/800 (90,2%)	741/800 (92,2%)

Algoritmus pro Houghovu transformaci potřeboval nastavit větší citlivost `'sensitivity', 0.98`, než je defaultní nastavení (`'sensitivity', 0.85`). I pro tuto hodnotu algoritmus ve 24 případech nenalezl žádnou kružnici. V 350 případech bylo nalezeno více kružnic, jako nejsilnější byla v 321 případech označena kružnice vymezující kámen. Konečně ve 34 případech detekce selhala. Celkem tedy bylo nesprávně detekováno 63 kruhů. Překvapivé je, že ani jednou nebyl detekován kalíšek. S jiným nastavením intervalu prohledávaných poloměrů, například `[80,160]` nebo `[150,250]`, byly výsledky lokalizace horší. Ve většině případů chybné detekce byl nalezen kruh malého poloměru obsahující jeden nebo více silných odlesků.

V algoritmu projekce bylo 174 detekcí vyhodnoceno jako chybných. Ve 112 z těchto 174 případů šlo o správný výsledek: kámen byl křivý nebo křivě usazený. V 44 případech byl kámen detekován vizuálně správně, rozdíl detekovaných šířek a výšek se pohyboval v rozmezí 6 až 8 pixelů. Z 623 případů, kdy algoritmus vyhodnotil detekci jako úspěšnou, byly 4 kameny detekovány chybně.

Na Obr. 4.4 jsou uvedeny snímky s vyznačenými kružnicemi lokalizovanými pomocí Houghovy transformace a projekční metody. Uveden je vždy příklad dobrého, respektive špatného kamene, pro který byla lokalizace úspěšná a neúspěšná.



Obrázek 4.4: Vizualizace výsledků lokalizace bez apriorní znalosti poloměru hledaného kamene pro Houghovu transformaci a projekční metodu. U projekční metody je červeně vyznačen kruh s průměrem detekované šířky kamene, modře kruh s průměrem detekované výšky.

Lokalizace s apriorní znalostí poloměru

Hough:

Díky znalosti pixelového poloměru $s \in \mathbb{N}$ kamene můžeme prohledávat pouze malý interval poloměrů kružnic a celkově tak zvýšíme rychlost výpočtu. Senzitivitu bylo nutné nastavit na hodnotu `'sensitivity', 0.995`. Protože v testovací sadě $s = 154$, byl interval prohledávaných poloměrů zvolen `[150, 160]`.

Konvoluce:

Známe-li poloměr kamene, jednoduše sestavíme kruhovou masku $K \in \mathbb{R}^{s \times s}$ stejné velikosti. Výstupem algoritmu je detekovaný střed kamene `[x, y]` a součet pixelových hodnot kamene - celková světlost `S`. Očekáváme vysokou úspěšnost detekce, navíc ještě získáme informaci o světlosti kamene pro další použití při třídění kamenů, viz Sekce 4.3. Konvoluce je výpočetně náročnější než projekční metoda, oproti ní by ale měla být méně citlivá na křivé usazení kamene, slabé odlesky, nečistoty na destičce nebo odlesk kalíšku.

Projekce:

Schéma projekční metody popsané na začátku této sekce jsme modifikovali dvěma způsoby.

Modifikace 1:

- Spočti sloupcové a řádkové součty obrazu $\mathcal{I}$.
- Nechť p je průměr kamene a n celkový počet sloupců v obraze. Pokud je $p < n-p$, hledáme p sousedících sloupců tak, aby jejich celkový součet byl maximální. Jestliže $n - p < p$, řešíme opačnou úlohu nalézt $n - p$ (krajních) sloupců s celkovým minimálním součtem. Zbylé (nevybrané) sloupce opět tvoří nejsvětlejší svislý pruh v obrazu.
- Zopakuj předchozí bod pro řádky.
- Je nalezen nejsvětlejší čtverec $p \times p$ v obraze. Jemu vepsaná kružnice je hledaný kámen.

Tato modifikace je implementovaná jako Algoritmus 2.

Modifikace 2: Tato modifikace spočívá pouze ve využití původního schématu a následném porovnání nalezeného poloměru oproti apriorně zadanému poloměru. V případě rozdílných hodnot prohlásíme lokalizaci za chybnou.

Pro dobře usazený kámen dává Modifikace 1 velmi přesnou lokalizaci. V případě lehce natočeného kamene je přesnost lokalizace uspokojivá. Jestliže je kámen výrazněji natočen nebo vrhá slabé odlesky, může dojít k chybné detekci – světlost okraje kalíšku převáží a chyba lokalizace může dosáhnout několika desítek pixelů. I zde je možné předzpracovat obraz stejně jako v předchozím případě. Touto modifikací ovšem přijdeme o schopnost určit špatně usazený kámen. Obě modifikace jsme otestovali na náhodně vybrané množině 800 kamenů. Na těchto testovacích datech vyšla jako lepší Modifikace 2, proto je uvažována v následujícím srovnání.

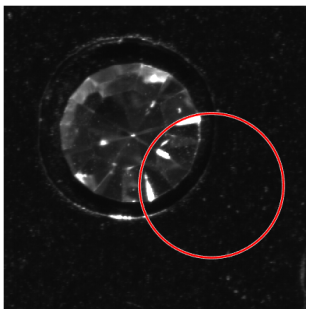
Srovnání

Následující tabulka shrnuje výsledky testu metod pro lokalizaci kamene s apriorní znalosti poloměru kamene. Organizována je stejně jako tabulka srovnání metod bez apriorní znalosti poloměru. Obsahuje však navíc výsledky pro konvoluční metodu.

	Hough	Konvoluce	Projekce
Čas	260 s	84s	51 s
Kružnice nenalezena	1/800 (0,1%)	–	126/800 (15,7%)
z toho vadných	0/1 (0%)	–	122/126 (96,8%)
Chybná lokalizace	34/799 (4,2%)	0/800 (0%)	3/674 (0,4%)
Úspěšnost	785/800 (98,1%)	800/800 (100%)	789/800 (98,6%)

U algoritmu Houghovy transformace došlo k výraznému zrychlení, zlepšila se i přesnost lokalizace. Výsledek „kružnice nenalezena“ jsme obdrželi pouze v případě jednoho kamene. Lokalizace byla chybná pro 34 snímků, tyto snímky obsaho-

valy odlesky a světlé body tvořící část kružnice. Výsledek pak nebyl ani přibližně správně, jak je možné pozorovat na Obr. 4.5.



Obrázek 4.5: Příklad dobrého kamene s výraznými odlesky ležícími na kružnici Nejsilnější kružnice nalezená pomocí Houghovy transformace je zcela chybná.

Lokalizace pomocí konvoluce dává nejpřesnější výsledky ze všech použitých metod. Na testovaných snímcích jsme dosáhli 100% úspěšnosti, v případě křivě usazených kamenů byl detekovaný kruh umístěn tak, jak by kružnici daného poloměru okolo kamene na snímku nakreslil uživatel.

Poznámka. Ve schématu konvoluční metody používáme k výpočtu konvoluce Větu 2, a to z důvodu snížení časové náročnosti. Bez použití Fourierovy transformace a s konvolucí počítanou MATLABovským příkazem `conv2` běžel výpočet lokalizace pomocí konvoluce přes $70s$ pro 20 snímků.

V případě Modifikace 2 projekční metody (přidáním jednoduché kontroly poloměrů) dokážeme správně lokalizovat 44 z 52 kamenů, pro které byl v první části (bez apriorní znalosti poloměru) výsledek „kružnice nenalezena“ a nejednalo se o křivé nebo křivě usazené kameny. Vyloučíme také zhruba jeden ze čtyř špatně lokalizovaných kamenů.

Na Obr. 4.6 jsou uvedeny snímky s vyznačenými kružnicemi lokalizovanými pomocí Houghovy transformace, konvoluční a projekční metody. Uveden je vždy příklad dobrého, respektive špatného kamene, pro který byla lokalizace úspěšná a neúspěšná. Jako neúspěšnou lokalizaci pro konvoluční metodu uvádíme pouze kameny, které jsou křivě usazeny a nelze jim přesně opsat kružnici, výsledek lokalizace se tak jeví o několik pixelů nepřesný. Na ostatních datech je totiž konvoluční metoda vždy spolehlivá.

Shrnutí

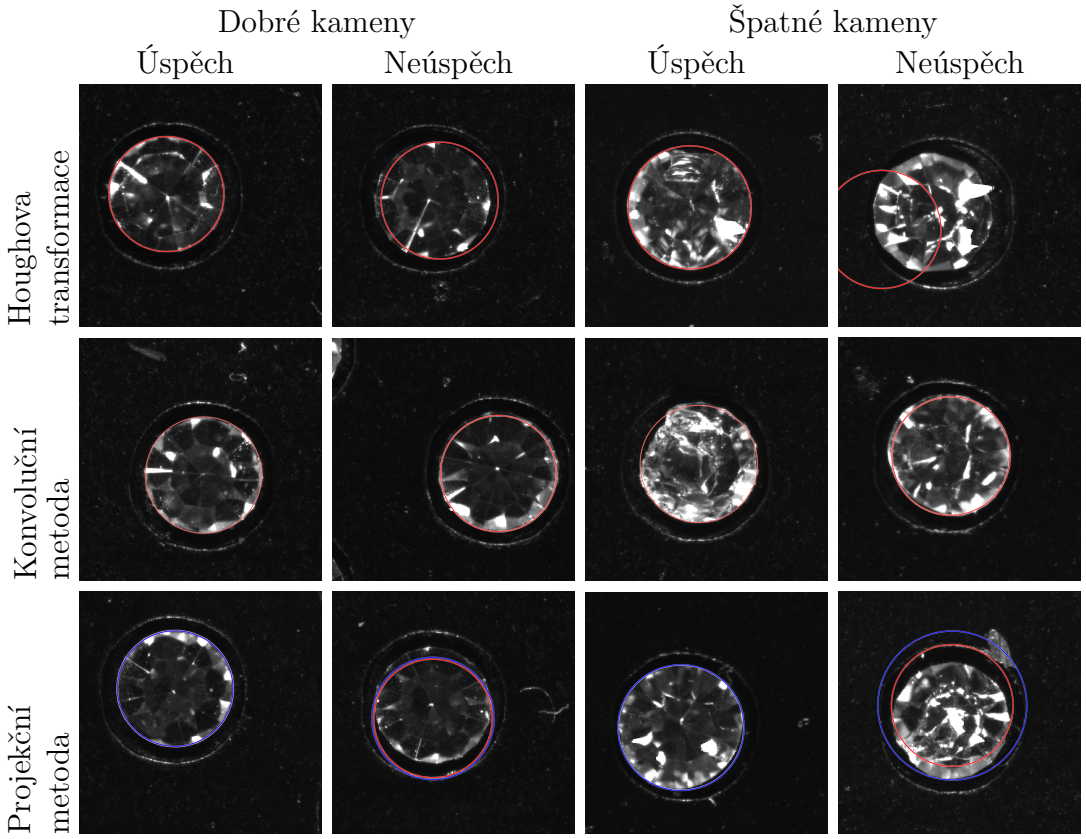
Na základě výsledků experimentů není těžké vybrat vhodnou metodu pro studovanou úlohu:

- Jestliže neznáme poloměr kamene, volíme projekční metodu. Ta minimalizuje počet nepřesných lokalizací, které byly vyhodnoceny jako úspěšné. Díky časové nenáročnosti výpočtu si můžeme dovolit snímky, na kterých byla lokalizace neúspěšná, otestovat znovu s jiným nastavením parametrů pro předzpracování, po mechanickém urovnání kamenů na destičce nebo za použití jiné, časově náročnější lokalizační metody. Tím bychom měli

správně lokalizovat většinu kamenů. Ty, u kterých bude dvakrát vyhodnocena lokalizace jako neúspěšná, označíme za křivé (tedy špatné) a rovnou je vyloučíme z testování ještě před samotnou separací.

- Známe-li apriorně poloměr kamene, lokalizace pomocí konvoluce je nejpřesnější a časově méně náročná než dva průchody projekční metody. Ztratíme ovšem možnost vytřídit křivé a křivě usazené kameny. Jejich separaci se budeme věnovat v následující sekci.

Poznámka. U obou implementovaných metod lze přidat uživatelský parametr, kterým by bylo možné vážit mezi přesností a rychlostí lokalizace. Vzhledem k výborné přesnosti obou algoritmů a poměrně malému zrychlení, které bychom takto získali, není tento parametr v současnosti implementován.



Obrázek 4.6: Vizualizace výsledků lokalizace s apriorní znalostí poloměru hledaného kamene pro Houghovu transformaci, konvoluční a projekční metodu. Konvoluční metoda byla úspěšná na všech snímcích kromě křivých a křivě usazených kamenů. Proto jsou ve sloupcích „Neúspěch“ uvedeny křivě usazené kameny. U projekční metody je červeně vyznačen kruh s průměrem detekované šířky kamene, modře kruh s průměrem detekované výšky.

4.2 Filtrace křivých a špatně usazených kamenů

Kameny s vadou tvaru, tedy křivé a křivě usazené, se kvalitativně výrazně liší od ostatních špatných kamenů. Viděli jsme, že část takto poškozených kamenů

lze vyřadit již při lokalizaci pomocí projekční metody. Nyní navrhneme metodu speciálně pro třídění křivých a křivě usazených kamenů. Úloha rozpoznání křivých kamenů je velmi blízká úloze rozpoznání tvaru objektu. Tomo téma je podrobně zpracováno například v [26].

Předpokládejme, že máme k dispozici přesnou lokalizaci kamenů (získanou například pomocí konvoluční metody) v obraze $\mathcal{I}$. Hrana kamene by měla přesně kopírovat detekovanou kružnici. Stačí nám tedy projít postupně mezikruží dané středem kamene, jeho poloměrem a poloměrem o několik pixelů menším. V každém procházeném úseku mezikruží spočteme celkovou světlost. Jestliže se v daném úseku nenachází kámen, bude spočtená světlost úseku výrazně nižší než v ostatních úsecích. Tento postup je implementován jako Algoritmu 5. Vizualizace tohoto postupu je na Obr. 4.7. V tomto případě jsme hranici kamene rozdělili na 18 shodných částí a prošli postupně každou z nich, spočtené světlosti jsou uvedeny v tabulce níže.

úsek	1	2	3	4	4	6
světlost	118 969	153 285	131 964	147 869	247 145	221 812
úsek	7	8	9	10	11	12
světlost	132 083	190 234	269 829	324 131	207 423	68 229
úsek	13	14	15	16	17	18
světlost	42 131	58 146	197 960	101 891	197 775	279 591

Okamžitě vidíme výrazný pokles světlosti mezi úseky 11 a 13 a opětovný nárůst mezi úseky 13 a 15. Z toho můžeme usoudit, že zkoumaný kámen je křivý.

Výše popsaný postup jsme otestovali na ručně vybrané množině 400 kamenů (obrazů $\mathcal{I}$). Na této sadě byla metoda schopna identifikovat křivé a křivě usazené kameny lépe než projekční metoda. Vzhledem k efektivitě třídící metody navržené v následující sekci nebyla tato metoda dále optimalizována a testována.

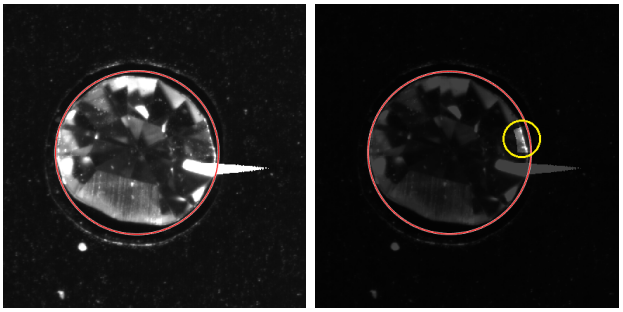
4.3 Problém separace

V této části se budeme věnovat návrhu metody separace dobrých a špatných kamenů a její implementaci. Testovat budeme na všech dostupných datech, tedy 1600 snímcích. Díky spolehlivým výsledkům lokalizačních metod budeme v následujícím textu předpokládat, že u každého snímku známe střed kamene a jeho poloměr. Tyto údaje jsme spočetli pomocí konvoluční metody.

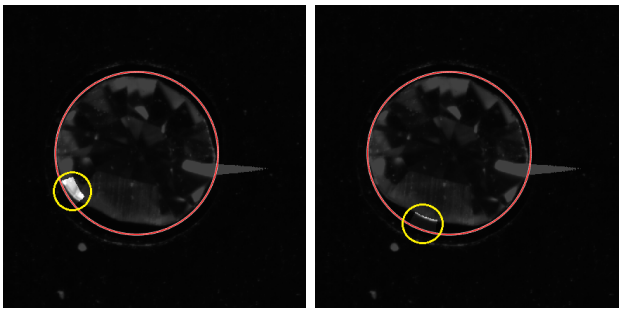
Separace dle světlosti

U kamenů v testovací sadě víme, zdali se jedná o dobrý, nebo špatný kus. Zaměříme se na množinu 800 kamenů, které byly ručně označeny jako vadné (viz Obr. 4.2). Z reálného pozorování dospějeme k následující hypotéze: Dobré kameny odráží světlo méně (působí na pohled tmavěji) než špatné kameny. Například u kamene z Obr. 4.2b nelze vidět jednotlivé drobné vybroušené plošky, kámen celkově působí neprůsvitně. Naproti tomu kámen bez vady, z Obr. 4.1a, vrhá velmi málo odlesků a je téměř průhledný. Odlesky a neprůhledné plochy kamene ukazují na chybu brusu nebo vadu materiálu. **Světlost kamene** definovanou jako

$$S(\mathcal{I}) = \sum_{(x,y)\in K_{\mathcal{I}}} \mathcal{I}(x,y),$$



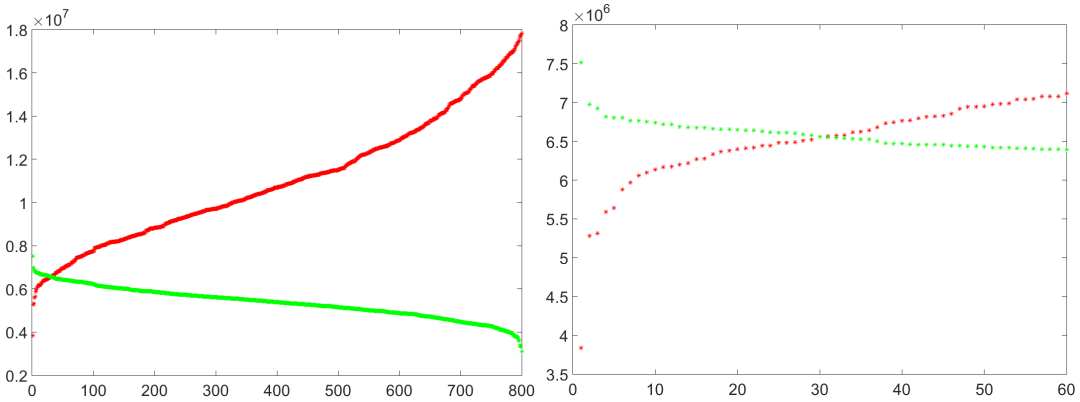
(a) Lokalizace kamene (b) První úsek



(c) Jedenáctý úsek (d) Třináctý úsek

Obrázek 4.7: Vizualizace algoritmu na třídění křivých kamenů. a) lokalizace kamene na snímku pomocí konvoluční metody, b), c) a d) první, jedenáctý a třináctý zkoumaný úsek kraje kamene. V prvním a jedenáctém úseku je celková světlost výrazně vyšší než ve třináctém. Ten svojí světlostí odpovídá spíše pozadí a můžeme tak usoudit, že zkoumaný kámen je křivý nebo křivě usazený. Pro vizualizaci výsledku jsme pro b),c) a d) ztmavili celý snímek s výjimkou aktuálního zkoumaného úseku.

kde $K_{\mathcal{I}}$ je množina pixelů lokalizovaného kamene, dokážeme snadno spočítat už při lokalizaci, čehož využijeme. Na Obr. 4.8 jsou vykresleny světlosti všech špatných kamenů (800 snímků, seřazeny vzestupně, označeny červeně) a světlosti dobrých kamenů (800 snímků, seřazeny sestupně, označeny zeleně).



Obrázek 4.8: Spočtené celkové světlosti všech kamenů. Zeleně jsou vyznačeny světlosti kamenů bez vady, červeně kazových kamenů. Na levém grafu je výřez pravého grafu pro 60 nejsvětlejších dobrých a 60 nejméně světlých špatných kamenů.

V ideálním případě by bylo možné nalézt hodnotu (**práh** T) takovou, že všechny kameny se světlostí $S(\mathcal{I}) < T$ (to jest menší než práh) jsou dobré a naopak kameny s $S(\mathcal{I}) > T$ (to jest s větší světlostí) jsou špatné. Toho bohužel nelze obecně dosáhnout. Pokud zvolíme jako práh

$$T = \max_{\mathcal{I} \in OK} S(\mathcal{I}),$$

kde OK je množina všech dobrých kamenů (to jest maximum ze světlostí dobrých kamenů) zařadíme nesprávně 81 špatných kamenů (chyba typu false positive). Nicméně na Obr. 4.8 si můžeme všimnout, že tři nejsvětlejší dobré kameny mají výrazně vyšší světlost než zbytek dobrých kamenů. Pokud bychom je ignorovali a nastavili práh jako maximální světlost ze zbývajících kamenů bez vady, zařadíme špatně 42 špatných kamenů. Celkem tedy chybně zařadíme 45 z 1600 kamenů – to jest získáme 97% úspěšnost třídění, přičemž většina chyb je typu false positive. Další důležité pozorování, získané analýzou světlostí kamenů z testovací sady, je, že správnost usazení kamene nemá výraznější vliv na jeho celkovou světlost. Dokážeme tak správně zařadit dobré, ale křivě usazené kameny, které by byly vyloučeny například při použití separace na základě tvaru kamene.

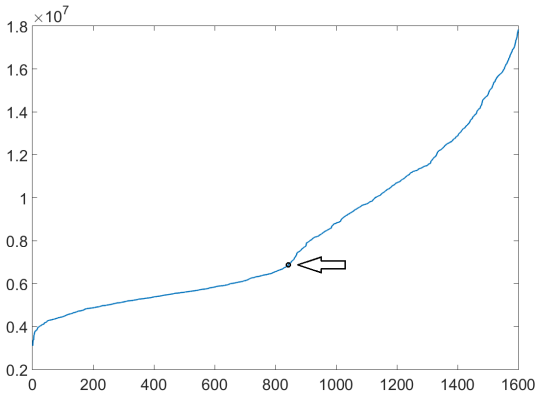
Následující tabulka shrnuje počty špatně zařazených kamenů pro různě volby prahu, kde n-tý práh T_n volíme jako světlost n-tého nejsvětlejšího dobrého kamene. False negative je počet nesprávně zařazených dobrých (odpovídá $n-1$), false positive je počet nesprávně zařazených špatných kamenů. Poslední sloupec tabulky udává celkový počet špatně zařazených kamenů a procentuální úspěšnost.

Práh	False negative	False positive	Celkem
T_1	0	81	81 (94,94%)
T_2	1	51	52 (96,75%)
T_3	2	47	49 (96,94%)
T_4	3	42	45 (97,19%)
T_5	4	42	46 (97,125%)
T_{10}	9	38	47 (97,06%)
T_{15}	14	36	50 (96,86%)
T_{20}	19	36	55 (96,56%)
T_{30}	29	29	58 (96,38%)
T_{50}	49	22	71 (95,56%)

Nalezení prahu

Dále si všimneme, že rychlost poklesu světlosti $S(\mathcal{I})$ kamenů bez vady je viditelně pomalejší než rychlost nárůstu světlostí $S(\mathcal{I})$ vadných kamenů. Odtud bychom chtěli získat návod na automatické nalezení **optimálního prahu**. Spojíme obě části testovací sady dohromady, spočítáme světlosti všech kamenů a ty setřídíme vzestupně. Výsledek se vykreslen na Obr. 4.9. Za ideální práh označíme hodnotu (světlost) bodu, ve kterém má **diskrétní křivka** z Obr. 4.9 **největší diskrétní křivost**, tedy kde se výrazně mění globální rychlost stoupání křivky. Z technických důvodu nebereme v potaz začátek intervalu, viz diskuse níže.

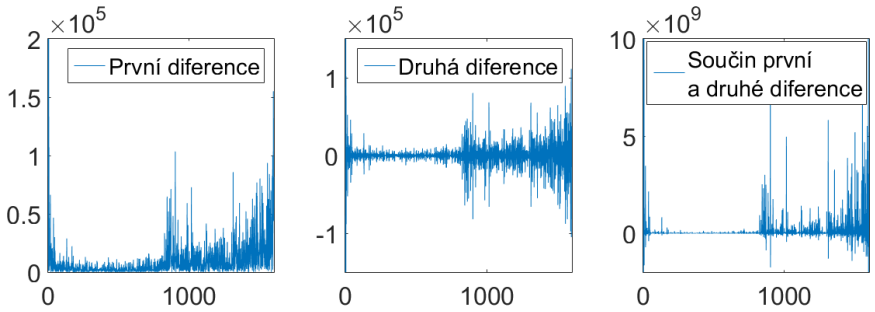
Poznámka. Podobné křivky se objevují při řešení inverzních úloh, v této souvislosti jsou nazývány **L-křivky**. Pro detailní popis viz například [8].



Obrázek 4.9: Diskrétní křivka získaná vykreslením vzestupně seřazených světlostí všech kamenů s vyznačeným bodem největší diskrétní křivosti.

K nalezení bodu s největší křivostí lze použít různých metod, v našem případě jsme zvolili analýzu pomocí první a druhé diference křivky definované seřazenými světlostmi kamenů. Obě diference, počítané jako rozdíl sousedních hodnot, jsou vykresleny na Obr. 4.10. Celý postup je implementován jako Algoritmus 6.

```
1 OK = "svetlosti dobrych kamenu"
2 NOK = "svetlosti spatnych kamenu"
3 A = sort([OK NOK]);      % Spojime sady kamenu bez a s vadou a ...
      setridime vysledek
4 plot(A);
5 D = diff(A);          %spocitame prvni a druhou diferenci
6 DD = diff(A,2);
7 subplot(1,3,1), plot(D); subplot(1,3,2), plot(DD); ...
      subplot(1,3,3), plot(D(2:end).*DD)
```

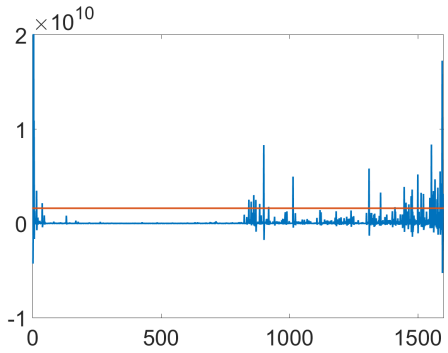


Obrázek 4.10: První a druhá diference diskrétní křivky z Obr. 4.9 a jejich součin.

První i druhá diference nabývají výrazně vyšších hodnot v druhé polovině intervalu – to odpovídá větším rozdílům ve světlostech $S(\mathcal{I})$ vadných kamenů. Tento jev ještě více umocníme jejich vzájemným vynásobením, viz Obr. 4.10 třetí graf. Po této úpravě je již bod s největší křivostí snadno identifikovatelný. Například vezmeme index první hodnoty, která překročí stanovenou hranici `parameter`. Ten je zvolen jako desetinásobek průměrné hodnoty vzájemně vynásobených diferencí.

```
R=D(2:end).*DD;          %soucin prvni a druhe diference
parameter = mean(R)*10;
idx = find(R(101:end)>parameter),1)+100;
thresh = A(idx);
plot(R); hold on, plot(parameter*ones(size(R)));
```

Na Obr. 4.11 je vykreslena takto určená hranice.



Obrázek 4.11: Součin první a druhé diference diskrétní křivky z Obr. 4.9 spolu s hranicí určenou pomocí `parameter = 10*mean(R)`.

Takto získáme index kamene, jehož světlost položíme za práh T ve třídícím algoritmu. Na Obr. 4.11 je možné pozorovat zvýšené hodnoty součinu diferencí v levé části intervalu. Ty jsou způsobeny několika málo kameny s nejmenší světlostí. Tyto hodnoty budeme jednoduše ignorovat hledáním indexu `idx` až od sté nejmenší světlosti. Jestliže předpokládáme, že je v testované množině alespoň 100 dobrých kamenů, nepřijdeme tímto postupem o žádnou informaci.

Proměnná `idx` by mohla plnit funkci uživatelského parametru, jejím nastavením upravujeme počet vytříděných vadných kamenů. Vyšší hodnota znamená, že více špatných kamenů je prohlášeno za dobré (chyba typu false positive) a méně kamenů bez vady je prohlášeno za špatné. Naopak nižší hodnota znamená méně chyb false positive, ale také více vyřazených dobrých kamenů. Metodu třídění kamenů lze schématicky shrnout následovně:

1. Spočti světlost
$$S(\mathcal{I}) = \sum_{(x,y)\in K_\mathcal{I}} \mathcal{I}(x,y),$$
kde $K_\mathcal{I}$ je množina pixelů lokalizovaného kamene, několika (ideálně všech) testovaných kamenů.

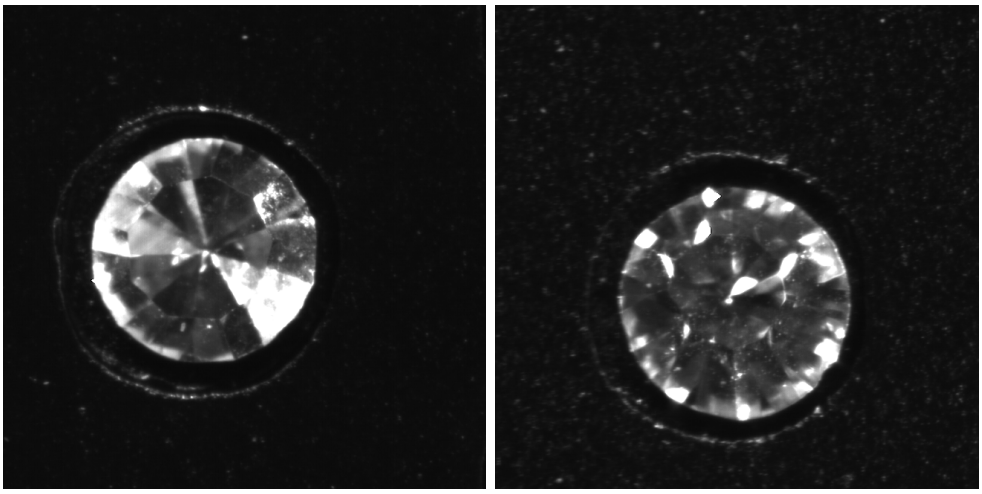
2. Spočti první a druhou diferenci diskrétní křivky L určené vzestupně setříděnými světlostmi kamenů `D=diff(L)` a `DD=diff(L,2)`.

3. Vynásob mezi sebou obě spočtené derivace `R=D.*DD` a zvol parametr `parameter` (pro příklad možné volby viz výše), `.*` značí násobení po prvcích.

4. Urči bod i s největší křivostí křivky L jako $\min\{i : \mathtt{R}(i) > \mathtt{parameter}\}$.
5. Urči práh T pro třídění jako světlost $S(\mathcal{I})$ i-tého nejméně světlého kamene.
6. Kameny s větší světlostí než práh T označ za špatné, kameny s nižší světlostí za dobré.

Metoda 7: Metoda pro automatické třídění kamenů

Aplikujeme-li popsaný postup na kompletní testovací sadu 1600 kamenů, dostaneme 42 false positive a 5 false negative zařazení. Získáme tedy celkem 97% úspěšnost a pouze o dva chybně zařazené kameny více než v případě, kdy jsme práh volili ručně. Poznamenejme, že jen na základě porovnání světlostí nejsme schopni tento výsledek dále vylepšit. Nesprávně zařazené špatné kameny jsou vizuálně podobné snímku na Obr. 4.2c. Jedná se o kameny, které mají dobře vybroušenou horní i spodní část a tedy nevrhají výrazné odlesky, nicméně tyto dvě části jsou vůči sobě nesprávně natočeny. Další problémové kameny jsou ukázány na Obr. 4.12. U této dvojice je vizuálně velmi komplikované určit, který kámen je dobrý a který špatný.



(a) Vadný kámen (b) Kámen bez vady

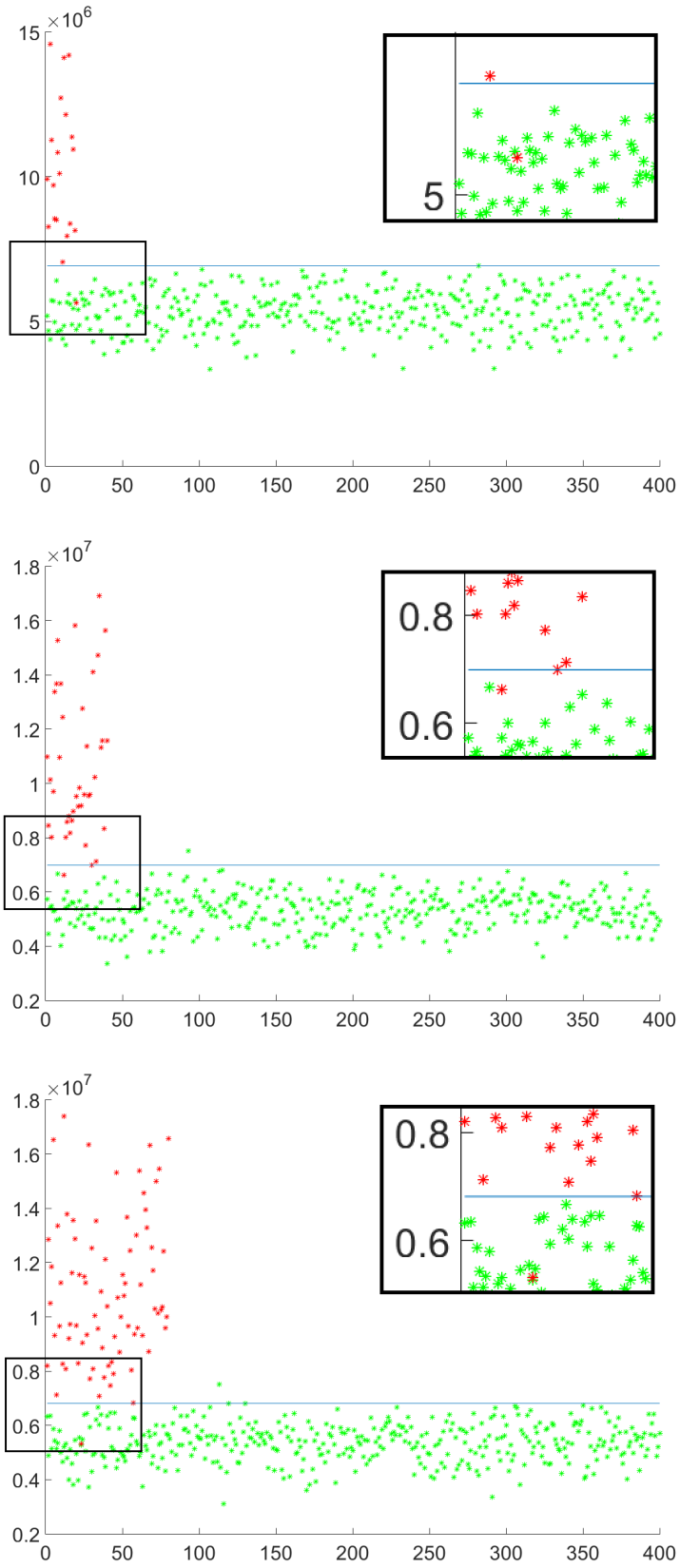
Obrázek 4.12: Příklad kamene s vadou (a) a bez vady (b), u kterých je vizuální a tedy i automatické určení kazovosti komplikované.

Na závěr této kapitoly použijeme výše popsanou Metodu 7 na data s různým procentem vadných kamenů. Ta zhruba odpovídaní zastoupení špatných kamenů v reálných výrobních sadách. Připravili jsme tři sady s po řadě 5 %, 10 % a 20 % vadných kamenů. Snímky do testovacích sad byly vybrány náhodně. Úspěšnost třídící metody je zaznamenána v následující tabulce. Dále na Obr. 4.13 jsou graficky znázorněny světlosti kamenů v jednotlivých sadách a automaticky určené prahy pro Metodu 7.

Procento vadných kamenů	5%	10%	20%
Počet false positive	1	1	1
Počet false negative	1	1	2
Úspěšnost	99.52%	99.54%	99.37%

Shrnutí

Celkově hodnotíme navržený třídící algoritmus jako úspěšný. Pro sady s reálným zastoupením špatných kamenů se úspěšnost pohybovala vždy nad 99%. Výhodou algoritmu je i jednoduchost a rychlost celého výpočtu a schopnost správně zařadit dobré, ale i křivě usazené kameny. Stačí uchovávat pouze světlosti $S(\mathcal{I})$ jednotlivých kamenů a hodnotu určující jejich umístění na destičce. Také je možné předpočítat prahy T pro různé sady kamenů apriorně a při vyhodnocování konkrétní destičky použít vhodný, dříve spočtený práh. Třídění v takovém případě vyžaduje pouze výpočet světlosti kamene. Otevřenou otázkou zůstává navržení způsobu, jak odhalit špatné kameny s malou světlostí. Zbavili bychom se tak false positive chyb, čehož nejsme schopni dosáhnout žádnou rozumnou volbou parametrů v metodě popsané výše.



Obrázek 4.13: Vizualizace výsledku třídění kamenů pomocí algoritmu porovnávajícího světlosti pro sady s 5, 10 a 20 % špatných kamenů. Zeleně jsou vyznačeny světlosti kamenů bez vady, červeně kazových kamenů. Modře je vyznačen automaticky detekovaný práh.

Závěr

Cílem předložené práce bylo navrhnout a implementovat výpočetní metodu pro automatické třídění skleněných bižuterních kamenů na základě analýzy jejich digitálních snímků. K dispozici jsme měli sadu reálných zkušebních snímků získaných řádkovou kamerou. Na základě těchto snímků jsme sestavili kompletní řešení sestávající z lokalizace jednotlivých kamenů a následného vyhodnocení jejich kvality.

Pro úlohu lokalizace objektu na snímku jsme představili dva vlastní návrhy přístupu – Projekční a Konvoluční metodu – optimalizované pro zadaná data. Projekční metoda je založena na redukování úlohy na jednodimenzionální problém. Konvoluční metoda využívá konvoluci matic a je implementována za pomoci Fourierovy transformace. Obě metody jsme porovnali s Houghovu transformací, která je implementována v MATLABu a slouží jako srovnávací metoda. Na testovacích datech jsme dosáhli následujících výsledků:

S apriorní znalostí poloměru kamene

1. Výpočet Houghovy transformace je časově náročnější než zbylé dvě metody. Dosažená přesnost lokalizace je nižší než u zbylých dvou metod.
2. Konvoluční metoda dosahuje nejvyšší přesnosti lokalizace při třetinové časové náročnosti oproti Houghově transformaci.
3. Projekční metoda je časově nejvýhodnější při zachování výborné přesnosti. Navíc dokáže již během lokalizace vyloučit podstatnou část špatných a křivě usazených kamenů.

Bez apriorní znalosti poloměru kamene

1. Výpočet Houghovy transformace je časově velmi náročný. Dosažená přesnost lokalizace je uspokojivá.
2. Konvoluční metoda není z důvodu extrémní časové náročnosti pro tento případ vhodná.
3. Projekční metoda je časově výrazně méně náročná a dosahuje vyšší úspěšnosti. Během lokalizace dokáže identifikovat některé špatné a křivě usazené kameny.

Obě metody – Projekční i Konvoluční – jsou pro zadanou úlohu velice efektivní. V závislosti na požadavcích na přesnost a časovou náročnost lze mezi těmito metodami volit.

Námi navržená metoda pro třídění kamenů, navazující na proces lokalizace, je založena na celkové světlosti kamene. Tu získáme přímo při použití konvoluční metody lokalizace nebo velmi snadno v případě projekční metody. Nevznikají tak téměř žádné další nároky na výpočetní čas. K otestování kompletního řešení zadané úlohy jsme sestavili tři testovací sady s po řadě 5,10 a 20% zastoupením špatných kamenů. Přestože námi navržený postup neumožňuje detekovat některé typy vad, dosáhli jsme na všech třech sadách více než 99.3% úspěšnosti třídění při celkově nízkém výpočetním čase. Zadání tedy bylo splněno. Navíc jsme navrhli metodu pro apriorní separaci křivých a křivě usazených kamenů. Díky vynikajícím výsledkům naší metody pro třídění kamenů ovšem nebylo potřeba tuto metodu zapojit do celkového řešení, může však uplatnění nalézt jinde.

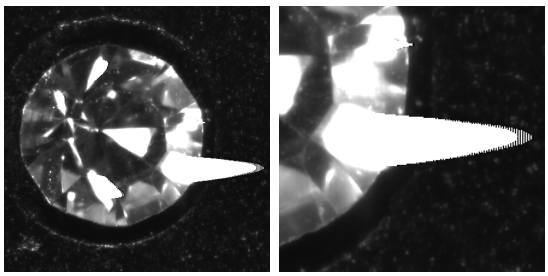
A. Způsob pořízení testovacích snímků a popis reálných dat

A.1 Způsob snímání

Testovací snímky byly pořízeny pomocí řádkové kamery s bočním osvětlením. Řádková kamera obsahuje snímací čip s jedním nebo několika málo řádky senzorů. K pořízení fotografie je potřeba vzájemný pohyb kamery a snímaného objektu. Typicky se pod kamerou lineárně pohybuje pás, kamera jej snímá řádek po řádku a ty skládá do výsledného obrázku. Podobný princip je používán například ve scannerech. Takto je možné ve vysokém rozlišení snímat nepřetržitě objekty, například na běžícím páse. V našem případě se snímací čip pohyboval nad snímanými daty.

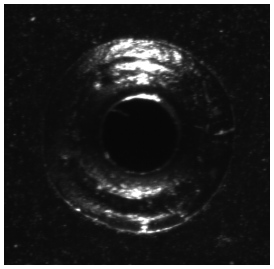
Hlavní výhodou řádkové kamery je možnost nepřetržitě snímat objekty, což umožní souběžné zpracování pořízených dat. Takto je možné kontrolovat kvalitu kamenů projíždějících pod kamerou téměř v reálném čase. Navíc stačí osvětlovat pouze úzký pruh snímané scény.

Nevýhodou tohoto způsobu snímání je vznik artefaktů. Při silném odlesku dojde k přetížení senzorů a na několika dalších řádcích se projeví zbytková energie. To vede ke zvýšení světelné intenzity v několika následujících řadách pixelů, což způsobí zesvětlení výsledného obrazu a vzniku bílých skvrn. Tento efekt je možné pozorovat například na Obr. 4.2a, 4.2d a A.1.



Obrázek A.1: Obraz s artefaktem a detail artefaktu

Snímané kameny byly zasazeny do destičky s pravidelně rozmístěnými kalíšky – otvory pro 294 kamenů (14×21). Zaplněná destička byla následně nasnímána a nahrubo rozřezána na snímky jednotlivých kamenů. Toto rozřezání je díky pravidelnému rozmístění kamenů na destičce snadno proveditelné. Pro dobrou separaci vadných kamenů je zásadní jejich správné usazení. Křivě usazený kámen se na snímku nejeví jako kruhový, navíc je nerovnoměrně osvětlen. Tento kámen pak může být klasifikován jako vadný (popřípadě špatně usazený). To vede k vyřazení kamene – chyba typu false negative, nebo k opětovnému přezkoumání kvality po urovnání kamene v destičce – zvýšení časových nároků na klasifikaci. Při pořizování testovací sady nebylo správné usazení kamenů explicitně kontrolováno. Na Obr. A.2 je zobrazen jeden kalíšek destičky, celá destička je pak na Obr. A.3.

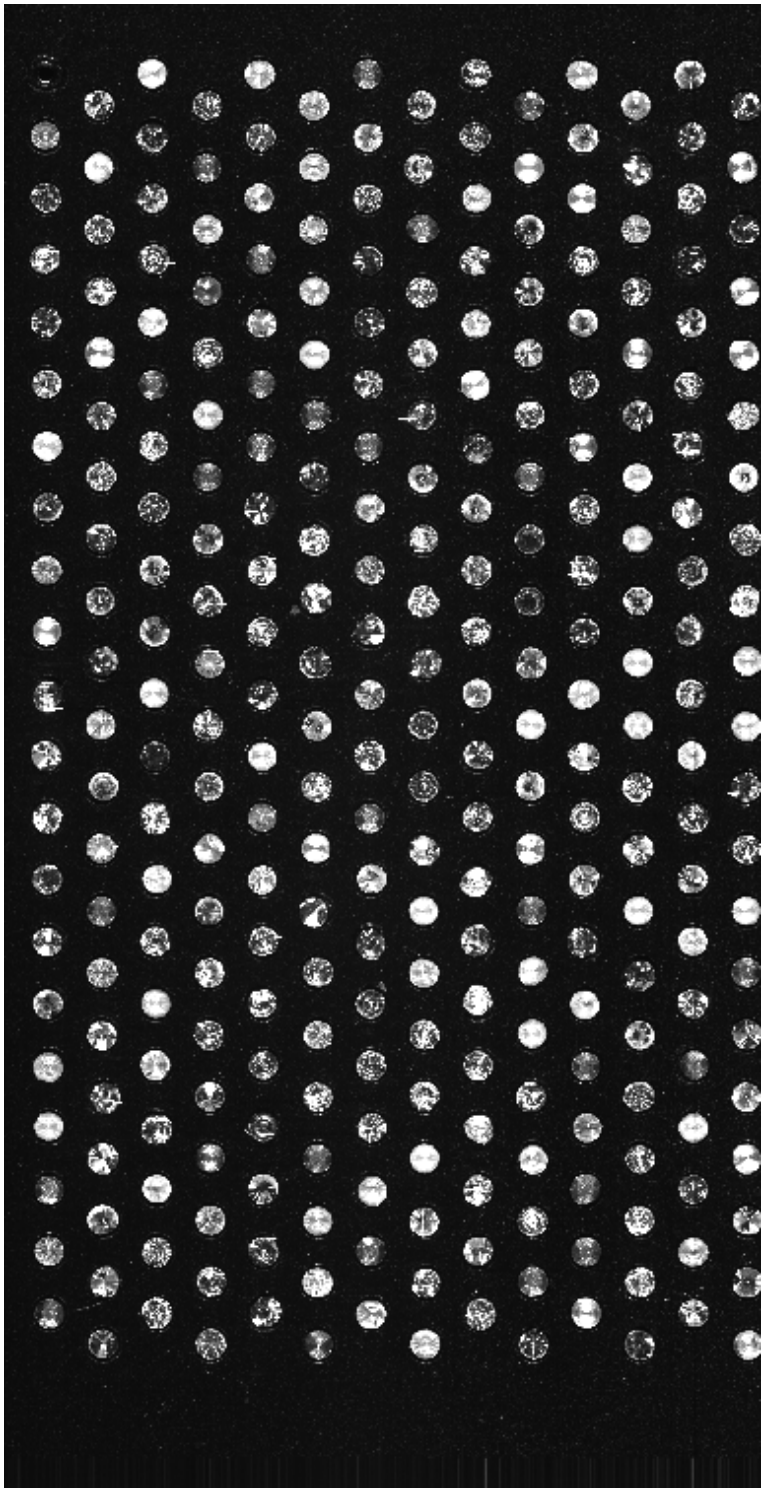


Obrázek A.2: Kalíšek v destičce

A.2 Popis testovací sady

K dispozici jsme měli šest naplněných destiček. Z toho tři obsahovaly dobré kameny a tři vadné. Z těchto jsme ručně odstranili kameny s nejhoršími artefakty a nejhůře usazené kameny. Tím jsme vytvořili sadu 800 dobrých a 800 špatných kamenů, simulující realistická vstupní data pro naši úlohu. Velikost jednotlivých snímků je 651×651 pixelů, pixelové hodnoty jsou celočíselné v intervalu [0,255].

Při vytváření lokalizačních algoritmů byla ručně vybrána sada 40 dobrých a 40 špatných kamenů. Pro těchto 80 kamenů byly optimalizovány parametry všech implementovaných algoritmů. Test úspěšnosti pak proběhl na náhodně vybrané množině 800 kamenů (dobrých i špatných). Algoritmus nalezení prahu pro třídění byl kalibrován na celé sadě 1600 kamenů. Pro účely testování jsme třikrát náhodně vybrali 400 dobrých kamenů a doplnili jsme je po řadě 20, 40 a 80 náhodně vybranými špatnými kameny.



Obrázek A.3: Destička s kameny

B. MATLABovské skripty

V této příloze jsou uvedeny MATLABovské skripty v podobě, v jaké byly použity při testování v Kapitole 4.

Algoritmus 1, projekční metoda
Implementace Metody 6, která hledá kámen neznámé velikosti. Uživatelskými parametry jsou zde `rtol` a `tol`. Pomocí `rtol` kontrolujeme platnost lokalizace – jestliže se nalezená šířka a výška kamene liší o více než `rtol` pixelů, prohlásíme lokalizaci za neplatnou. Takto můžeme vyloučit některé – pravděpodobně křivé nebo křivě usazené – kameny už při lokalizaci. Parametr `tol` pomáhá kompenzovat šum (nečistoty) v pozadí. Jeho hodnota udává, jaký sloupcový, respektive řádkový součet již považujeme za nulový. Součástí je i vizualizace (sloužící pouze pro kontrolu správnosti lokalizace uživatelem) a výpočet světlosti kamene, která se dále používá pro třídění kamenů.

Algoritmus 2, modifikace projekční metody
Modifikace projekční metody pro případ, kdy známe poloměr kamene. Vizualizaci a výpočet světlosti dostaneme stejně jako v předchozí metodě.

Algoritmus 3, konvoluční metoda bez optimalizace
Přímočará implementace Metody 5. Světlost kamene zde dostaneme automaticky bez potřeby dalších výpočtů. Tato implementace je časově neefektivní a je uvedena pouze pro srovnání.

Algoritmus 4, konvoluční metoda
Konvoluční metoda implementovaná s pomocí Věty 1. Matematicky ekvivalentní s Algoritmem 3, tato implementace však vyžaduje výrazně méně výpočetního času. Přesnost lokalizace je nezměněna. Vizualizaci dostaneme stejně jako v případě předchozích metod.

Algoritmus 5, detekce křivých kamenů
Algoritmus, který rozhodne zda je (dříve lokalizovaný) kámen křivý nebo křivě usazený. Pomocí uživatelských parametrů můžeme nastavit přísnost vyhodnocení. Vyšší hodnota `tol` znamená vyšší nároky na správný tvar kamene.

Algoritmus 6, separace kamenů
Funkce, která z předem spočtených světlostí kamenů určí práh pro separaci vadných kamenů. Modifikací proměnné `idx` lze kontrolovat přísnost separace. Nižší hodnota znamená méně zařazení typu false positive.

Algoritmy 7 a 8, pomocné funkce
Pomocné funkce použité v algoritmech výše. První nalezne $2r$ sousedních hodnot v poli V s největším součtem. Druhá vytvoří kruhovou masku pro konvoluční metodu.

```
function [Svetlost,Sx,Sy,r,Valid] = projekcni1(Image)
% Vstup je obraz.
% Vystup stred a polomer(y), svetlost kamene a informace o ...
    uspesnosti detekce
% navrhovana tolerance 5~10% z max(sum(I))

% Preprocessing a nastaveni tolearnce
C = im2bw(Image,0.15);
A = medfilt2(C,[7 7]);
rtol = 5;                % tolerance na rozdil polomeru
tol = max(sum(A))*0.075; % tolerance nenulovosti sloupcovych ...
    souctu

% Projdeme sloupcove a radkove soucty, hledame od sloupce s ...
    maximalnim souctem
x=sum(A);
[tmp,xmax]=max(x);
x1=xmax;
while x(x1)>tol, x1=x1-1;end
x2=xmax;
while x(x2)>tol, x2=x2+1;end

Sx=(x2+x1)/2; % Stred kamene je uprostred nenulovych sloupcu,
rx=x2-Sx;     % polomer je polovina poctu nenulovych sloupcu

% Zopakovat pro radky
y=sum(A');
[tmp,ymax]=max(y);
x1=ymax;
while y(x1)>tol, x1=x1-1;end
x2=ymax;
while y(x2)>tol, x2=x2+1;end
Sy=(x2+x1)/2;
ry=x2-Sy;

% Polomer je prumer obou polomeru
r=(rx+ry)/2;
if abs(rx-ry) > rtol
    Valid = false; else Valid = true;
end

% Vizualizace a vypocet svetlosti
imshow(Image);
h = viscircles([Sx,Sy],r,'LineWidth',1);
Svetlost=0;
K=zeros(651);
J=Image;
% Kruhova maska s polomerem r a stredem [Sx,Sy]
[X, Y] = meshgrid(1:651, 1:651);
K = ((X-Sx).^2 + (Y-Sy).^2 < (r^2));
J=J.*K;             % Z obrazu uvazujeme pouze kamen
Svetlost=sum(J(:)); % Spocteme svetlost kamene
```

Algoritmus 1: Implementace projekční metody bez znalosti poloměru kamene.

```
function [Sx,Sy] = projekcni2(Image)
% Projekcni metoda pro lokalizaci kamene se znamym polomerem ...
    (154 pixelu) v obrazu 651x651 pixelu
% Vstupem je obraz

r=154;

% Spocteme sloupcove a radkove soucty
X = sum(Image); %sloupcove
Y = sum(Image'); %radkove

% Zavolame pomocnou proceduru, dostaneme index sloupce a ...
    radku kde zacina kamen
nX = projekcni2mojemax(X);
nY = projekcni2mojemax(Y);

%souradnice stredu = zacatek kamene + polomer
Sx = nX+r;
Sy = nY+r;
```

Algoritmus 2: Modifikace Algoritmu 1 pro známý poloměr kamene.

```
function [S,x,y] = konvolucniMetoda(Image)
% Metoda zalozena na konvoluci, vstup je obraz vystupem je ...
    svetlost kamene a souradnice jeho stredu.
% Lepsi implementace je matching metoda.

K=kruh(154,308); % Pomocna kruhova maska

% Konvoluce
Q=conv2(Image,K,'valid');

% Nejlepsi shoda
[num, idx] = max(Q(:));

% Svetlost a stred
S=num;
[y, x] = ind2sub(size(Q),idx);
x=x+154;
y=y+154;

% Vizualizace
figure
imshow(Image);
h = viscircles([x,y],154,'LineWidth',1);
```

Algoritmus 3: Lokalizace pomocí konvoluce, bez optimalizace časové náročnosti.

```
function [S,x,y] = matchingMetoda(Image)
% Konvolucni metoda implementovana pomoci fourierovy ...
    transformace.
% Vstupem je obraz vystupem je svetlost kamene a souradnice ...
    jeho stredu

% Spocteme Four. transformaci obrazu a kruhove masky
K=kruh(154,651);
FK=fft2(K);
FourImage = fft2(Image);

% Spocteme konvoluci pomoci Convolution theorem
Conv=FourImage.*FK;
Conv=fftshift(ifft2(Conv));

% Najdeme nejlepsi shodu, do S ulozime svetlost kamene do [x ...
    y] souradnice
%jeho stredu

[num, idx] = max(Conv(:));
S=num;
[y, x] = ind2sub(size(Image),idx);
```

Algoritmus 4: Optimalizace časové náročnosti Algoritmu 3 pomocí rychlé Fourierovy transformace.

```
function isCircle = excludeElipses(x,y,Image)
% Funkce detkujici spatne usazene kameny a vady tvaru kamene.
% Vstupem je kamen a jeho stred.
% Spocteme soucty pixelovych hodnot pres casti obvodu kamene,
% z jejich velikosti rozhodneme, zda je kamen kruhovy.

% Uzivatelske parametry
rk = 154;          %polomer kamene
r = 145;           %vnitrni polomer
alpha = 20;        %hranici kamene rozdelime na 360/alpha casti
tol = 20000;       %tolerance

% Inicializace
result=zeros(1,360/alpha); %inicializace pole pro vysledky
K1=kruh(rk,309);
K2=kruh(r,309);
Ma=K1-K2;          %maska - mezikruzi odpovidajici obvodu kamene
for i= 1:rk
    for j = 1:rk
        if j>tan(deg2rad(alpha))*i
            Ma(rk++1-j,rk+1+i)=0;
        end
    end
end
Ma(:,1:155)=0;
Ma(156:end,:)=0;  %maska odpovidajici jedne casti hranice

% Napocitani pixelovych souctu pres casti obvodu kamene

%priprava masky pro kazdou z 360/alpha casti obvodu, vhodny ...
   motocenim Ma a usazenim vuci stredu [x,y]
for i=1:(360/alpha)
    M=zeros(size(Image));
    T=imrotate(Ma,i*alpha,'crop');
    M(y-rk:y+rk,x-rk:x+rk)=T;
    result(i)=sum(sum(Image.*uint8(M))); %ulozeni vysledku

% vizualizace
    imshow(Image.*uint8(M)+0.5*Image);
    h = viscircles([x,y],154,'LineWidth',1);
    pause
end

% Test na krivost
% jednoduchy test, je-li v kazde testovane casti dostatek ...
   svetlych bodu
isCircle = true;
if min(result) < tol
    isCircle = false;
end
```

Algoritmus 5: Jedna z možností detekce křivých a křivě usazených kamenů.

```
function T = separace(A)
% Funkce vracejici hodnotu prahu pro separaci na zaklade ...
    celkove svetlosti kamene
% Vstupem je pole hodnot svetlosti kamenu
A = sort(A);        % Setridime vstupni data
D = diff(A);        % Spocitame prvni a druhou diferenci
DD = diff(A,2);
R=D(2:end).*DD;     % Soucin prvni a druhe diference
% Spocteme parametr a pomoci nej nalezneme hodnotu prahu
parameter = mean(R)*10;
idx = find(R(101:end)>parameter),1)+100;
T = A(idx);
```

Algoritmus 6: Algoritmus hledající práh pro třídění podle celkové světlosti.

```
function n = projekcni2mojemax(V);
%funkce prochazejici pole V, secteme vzdy 2*r sousednich ...
    hodnot, ulozime do pomocneho pole S, najdeme maximum pole S.
% Vystupem je index zacatku bloku 2*r hodnot V s nejvetsim ...
    souctem

len=651-2*r-1
S=zeros(len,1);
for i = 1:len
    S(i) = sum(V(i:i+2*r-1));
end
[a, n]=max(S);
```

Algoritmus 7: Pomocná funkce pro Algoritmus 2.

```
function K = kruh (R, N)
% Vraci kruhovou masku o polomeru R v matici NxN

[X, Y] = meshgrid(-(N-1)/2:(N-1)/2, -(N-1)/2:(N-1)/2);
K = double(X.^2 + Y.^2 < R^2);
```

Algoritmus 8: Pomocná funkce pro Algoritmy 3 a 4.

Seznam použité literatury

[1] F. Anscombe, *The transformation of poisson, binomial and negative-binomial data*, Biometrika, 15 (1948), pp. 246–254.

[2] J. O. Berger, *Statistical decision theory and Bayesian analysis*, Springer Science & Business Media, 2013.

[3] C. Boncelet, *Handbook of image and video processing, chapter image noise models*, 2005.

[4] J. Freeman, A. (Translator); Fourier, *The Analytical Theory of Heat*, Cambridge university press, London, 1878.

[5] R. C. Gonzalez and R. E. Woods, *Digital image processingn*, Třetí vydání, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2004.

[6] R. C. Gonzalez, R. E. Woods, and S. L. Eddins, *Digital Image Processing Using MATLAB*, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2003.

[7] P. C. Hansen, *Discrete Inverse Problems: Insight and Algorithms*, Fundamentals of Algorithms, SIAM, 2010.

[8] P. C. Hansen and D. P. O'Leary, *The use of the l-curve in the regularization of discrete ill-posed problems*, SIAM Journal on Scientific Computing, 14 (1993), pp. 1487–1503.

[9] P. V. C. Hough, *Machine Analysis of Bubble Chamber Pictures*, 1959.

[10] B. Hunt, *A matrix theory proof of the discrete convolution theorem*, IEEE Transactions on Audio and Electroacoustics, 19 (1971), pp. 285–288.

[11] A. Jain, *Fundamentals of Digital Image Processing*, Prentice-Hall information and system sciences series, Prentice Hall, 1989.

[12] C. Kimme, D. Ballard, and J. Sklansky, *Finding circles by an array of accumulators*, Communications of the ACM, 18 (1975), pp. 120–122.

[13] J. Kittler and J. Illingworth, *Minimum error thresholding*, Pattern Recognition, 19 (1986), pp. 41 – 47.

[14] D. Marr and E. Hildreth, *Theory of edge detection*, Proceedings of the Royal Society of London B: Biological Sciences, 207 (1980), pp. 187–217.

[15] J. Modersitzki, *Numerical methods for image registration*, První vydání, Oxford university press, United States, 2004.

[16] N. Otsu, *A threshold selection method from gray-level histograms*, IEEE Trans. Systems, Man, and Cybernetics, 9 (1979), pp. 62–66.

[17] M. Petrla, *Numerické metody ve zpracování obrazu pro aplikace v bižuterním průmyslu*, bakalářská práce, Univerzita Karlova, 2016.

[18] W. K. Pratt, *Digital Image Processing*, Wiley, 1991.

[19] M. Robinson, *Photography: Montana summer cabin*, 2014.

[20] J. C. Russ, *The Image Processing Handbook*, Šesté vydání, CRC press, United States, 2011.

[21] M. W. Schwarz, W. B. Cowan, and J. C. Beatty, *An experimental comparison of rgb, yiq, lab, hsv, and opponent color models*, ACM Trans. Graph., 6 (1987), pp. 123–158.

[22] N. Senthilkumaran and R. Rajesh, *Edge detection techniques for image segmentation–a survey of soft computing approaches*, International journal of recent trends in engineering, 1 (2009), pp. 250–254.

[23] C. Solomon and T. Breckon, *Fundamentals of Digital Image Processing: A practical approach with examples in Matlab*, John Wiley & Sons, 2011.

[24] J. Spurný, *Text k přednášce: Funkcionální analýza.* http://www.karlin.mff.cuni.cz/ spurny/doc/faprednaska.pdf, 2016.

[25] J. L. Starck, F. Murtagh, and J. M. Fadili, *Sparse Image and Signal Processing*, První vydání, Cambridge university press, New York, 2010.

[26] B. Zitova and J. Flusser, *Image registration methods: a survey*, Image and vision computing, 21 (2003), pp. 977–1000.