VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2018 Štefan Šuňal

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

INTERAKTIVNÍ WEBOVÉ APLIKACE PRO PODPORU VÝUKY ZPRACOVÁNÍ OBRAZŮ
INTERACTIVE WEB APPLICATIONS SUPPORTING IMAGE PROCESSING LECTURES

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Štefan Šuňal**
AUTHOR

VEDOUCÍ PRÁCE — **doc. Mgr. Pavel Rajmic, Ph.D.**
SUPERVISOR

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Teleinformatika**
Ústav telekomunikací

Student: Štefan Šuňal ***ID:*** 186213
Ročník: 3 ***Akademický rok:*** 2017/18

NÁZEV TÉMATU:

Interaktivní webové aplikace pro podporu výuky zpracování obrazů

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte problematiku vztahující se k výukovým apletům, navrhněte jejich funkcionalitu a ovládání. Půjde o interaktivní podání algoritmů a technik používaných v oblasti zpracování obrazů, konkrétně: 1/ Dithering obrazu, 2/ Filtrace obrazu, 3/ Konvoluce obrazů krok po kroku, 4/ Převzorkování obrazu. Aplety implementujte pomocí HTML a JavaScriptu, tak, aby přehledně demonstrovaly příslušnou teorii a umožňovaly studentům interakci.

DOPORUČENÁ LITERATURA:

[1] Beneš, B.; Sochor, J.; Felkel, P.; Žára, J.: Moderní počítačová grafika. Computer Press, Brno, 2005.

[2] Gonzalez, R.C.; Woods, R.E.: Digital Image Processing. Třetí vydání. Pearson; 2007. ISBN 978-0131687288

Termín zadání: 5.2.2018 ***Termín odevzdání:*** 29.5.2018

Vedoucí práce: doc. Mgr. Pavel Rajmic, Ph.D.
Konzultant:

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Hlavným cieľom tejto práce je navrhnúť webové aplikácie zamerané na podporu výuky v oblasti počítačovej grafiky, konkrétne filtrácie obrazu, ditheringu, prevzorkovania a konvolúcie krok po kroku. Je v nej vysvetlená teória použitých techník a ich aplikácia v praxi. V poslednej časti práce je popísaný návrh a implementácia jednotlivých aplikácií pomocou jazyka JavaScript.

KĽÚČOVÉ SLOVÁ

Javascript, konvolúcia, dithering, filtrácia, prevzorkovanie

ABSTRACT

Main goal of this thesis is to design web applications focused on support of education in field of image processing, specifically image filtering, dithering, resampling and step-by-step convolution. It contains explanation of used methods and their practical application. Final part of the thesis describes design and implementation of each application in JavaScript.

KEYWORDS

Javascript, convolution, dithering, filtration, resampling

ŠUŇAL, Štefan. *Interaktivní webové aplikace pro podporu výuky zpracování obrazů*. Brno, 2018, 46 s. Bakalárska práca. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedúci práce: doc. Mgr. Pavel Rajmic, Ph.D.

VYHLÁSENIE

Vyhlasujem, že som svoju bakalársku prácu na tému „Interaktivní webové aplikace pro podporu výuky zpracování obrazů“ vypracoval(a) samostatne pod vedením vedúceho bakalárskej práce, využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor(ka) uvedenej bakalárskej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto bakalárskej práce som neporušil(a) autorské práva tretích osôb, najmä som nezasiahol(-la) nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý(-á) následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákoníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autora(-ky)

POĎAKOVANIE

Ďakujem vedúcemu bakalárskej práce doc. Mgr. Pavel Rajmic, Ph.D. za účinnú metodickú, pedagogickú a odbornú pomoc a ďalšie cenné rady pri spracování mojej bakalárskej práce práce.

Brno ....

podpis autora(-ky)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic
http://www.six.feec.vutbr.cz

POĎAKOVANIE

Výzkum popsaný v tejto bakalárskej práci bol realizovaný v laboratóriách podporených projektom SIX; registračné číslo CZ.1.05/2.1.00/03.0072, operačný program Výzkum a vývoj pro inovace.

Brno ....

podpis autora(-ky)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OBSAH

ZOZNAM OBRÁZKOV

ÚVOD

Súčasný rozvoj v oblasti výpočtovej techniky ako po hardwarovej, tak po softwarovej stránke umožňuje realizovať stále náročnejšie algoritmy. Vďaka tomu začal aj veľký rozmach v oblasti spracovania obrazu a počítačového videnia. V súčasnosti sa môžeme dennodenne stretať so zariadeniami využívajúcimi tieto algoritmy a poznatky z tejto oblasti. Spracovanie obrazu je základným kameňom pri mnohých diagnostikách v zdravotníctve, pri automatizácií procesov, používa sa v doprave a v mnohých ďalších oblastiach. Cieľom mojej práce bolo oboznámiť sa s obrazom a problematikou spojenou s jeho spracovaním. Pochopiť princíp základných algoritmov a ich význam. Na základe informácií získaných následne zrealizovať webové aplikácie schopné realizovať tieto algoritmy.

Prvá kapitola sa zaoberá základnými vlastnosťami obrazu, pojmami spätými so spracovaním obrazu a rozoberá niektoré farebné modely.

V druhej kapitole je popísaný postup získania digitálneho obrazu. Je rozdelená na dva celky, zaoberajúce sa vzorkovaním a následne kvantovaním.

Tretia kapitola sa zaoberá metódami spracovania obrazu. Ako prvá je tu rozobraná filtrácia obrazu pomocou konvolúcie s ukážkami niektorých filtrov. Ďalej sa táto kapitola zaoberá prevzorkovaním a s ním súvisiacim ditheringom.

V štvrtej kapitole sú popísané jazyky HTML, CSS a JavaScript, ich výhody a možnosti ich použitia. Ďalej je tu popísané vývojové prostredie a nástroje použité pri vývoji aplikácií. Nakoniec sú tu popísané niektoré prvky jazyka JavaScript, dôležité pre realizáciu aplikácií.

Piata kapitola sa zaoberá návrhom a realizáciou zadaných aplikácií a to filtrácie obrazu, ditheringu, prevzorkovania a simulácie znečistenia pomocou konvolúcie.

1 OBRAZ

Pod pojmom obraz obvykle chápeme akýkoľvek zrakový vnem. Pre potreby jeho spracovania je ho však potrebné nejako charakterizovať. Matematicky ho môžeme definovať ako funkciu $f(x, y)$, kde x a y predstavujú súradnice v dvojrozmernom priestore a hodnota funkcie f reprezentuje intenzitu jasu na daných súradniciach. Pojem jas udáva hustotu toku svetla vyžiareného zdrojom. Pri spracovaní obrazu počítačom je potrebné ho vyjadriť pomocou konečného množstva čísel. Ak sú jednotlivé prvky obrazu vyjadrené pomocou dvojrozmerného poľa s konečným počtom prvkov hovoríme o obraze rastrovom. Pri spracovaní obrazu sa obvykle pracuje práve s týmto typom obrazu. Existuje množstvo spôsobov ako digitálny obraz reprezentovať. Farebný model udáva základné farby a možnosti ich miešania. Ak farebný model rozšírime o údaj hĺbky farby, udávajúci počet farieb, ktoré je možné zobraziť, získame farebný priestor. Počet pixelov a rozmery obrázka v pixeloch určuje rozlíšenie. [1][2]

1.1 Farebný model RGB

Tento farebný model využíva k vytvoreniu rôznych farieb nachádzajúcich sa v obraze kombináciu niekoľkých základných farieb. Ako je už z názvu zrejmé, ide o farby červená (R – red), zelená (G – green), modrá (B – blue). Farbu je možné vyjadriť trojicou hodnôt (farebným vektorom), ktoré môžu byť v rozsahu $\langle 0,0; 1,0\rangle$ alebo v celočíselnom vyjadrení $\langle 0, 255\rangle$, čo odpovedá kódovaniu každej zložky v jednom byte. V prípade, že zložka je nulová, nie je táto zložka obsiahnutá v tejto farbe, zatiaľ čo maximálna hodnota indikuje, že daná zložka dosahuje svojej maximálnej intenzity.

Tento model je aditívny, čo znamená, že svetlosť výslednej farby rastie so súčtom intenzít jednotlivých zložiek. Ak zložíme červenú, zelenú a modrú zložku v plnej intenzite získame bielu farbu. RGB model využívajú vstupné zariadenia ako skenery, digitálne kamery. Výstupné zariadenia, ktoré využívajú tento model sú CRT, LCD monitory, plazma a iné obrazovky. Tento model je použitý aj v bitmapovom súbore digitálneho obrazu.[2]

1.2 Farebný model RGBA(RGBα)

Ide o model, v ktorom je štandardný RGB model doplnený o informáciu o priehľadnosti (α), čo znamená, že každý bod je možné vyjadriť štvoricou hodnôt. Priehľadnosť (α) určuje v akom rozsahu pokrýva farba plochu obrazového bodu. Hodnota 0,0 (255) predstavuje nepriehľadný farebný bod, zatiaľ čo hodnota 1,0 (0) úplne

priehľadný bod. Zložka α sa používa hlavne pri kombinácií viac obrazov do jedného celku.[2]

1.3 Farebný model CMY, CMYK

Zatiaľ čo RGB vychádza z miešania farieb svetla, čo je využívané hlavne na displejoch, nie je možné tento model aplikovať na miešanie pigmentov. V CMY modele zložením všetkých farieb vzniká čierna farba. Ide o subtraktívny model. CMY obsahuje tri základné farby tyrkysovú (C – cyan), fialovú (M – magenta) a žltú (Y – yellow). Model CMY mal ale pri tlači nevýhodu, lebo netvoril dokonalú čiernu farbu, a preto bol doplnený o štvrtú čiernu farbu (K – black). Tento model je vhodný hlavne pre tlačiarne.[2]

1.4 Farebný model HSV s HLS

Oba tieto modely definujú farbu trojicou zložiek. Tromi hlavnými parametrami HSV sú farebný tón (H – hue), sýtosť (S – saturation), jasová hodnota (V – value). Farebný tón označuje prevládajúcu spektrálnu farbu, sýtosť určuje prímesi iných farieb a jas je daný množstvom bieleho (bezfarebného) svetla. Hodnoty S a V sa pohybujú v rozsahu $\langle 0,0; 1,0\rangle$. Hodnota H predstavuje uhol a pohybuje sa v intervale $\langle 0°; 360°\rangle$. Systém HSV vykazuje nedostatky, ktoré sťažujú prácu s presným určením farby. Nedostatky odstraňuje model HLS. Tromi hlavnými parametrami HLS sú farebný tón (H – hue), svetlosť (L – lightness), sýtosť (S – saturation).[2]

2 DIGITALIZÁCIA OBRAZU

Výpočtová technika, ako aj väčšina moderných zobrazovacích zariadení je založená na práci s digitálnymi dátami. Aby sme s obrazovou funkciou mohli efektívne pracovať pomocou počítača a zobraziť na monitore, je potrebné ju digitalizovať. Proces digitalizácie pozostáva z dvoch hlavných krokov: vzorkovania a kvantovania.[3]

2.1 Vzorkovanie

Pod pojmom vzorkovanie rozumieme prevod signálu so spojitým časom na signál s diskrétnym časom. Pri spracovaní signálov vzorkovanie realizujeme rozdelením časovej osi na rovnako veľké úseky a z každého úseku odoberieme vzorku. Pri spracovaní obrazu si môžme predstaviť, že obraz rozdelíme na pole štvorcov rovnakých rozmerov a z každého vyčítame jednu hodnotu. S takto navzorkovaným obrazom môžeme ďalej pracovať ako s polom hodnôt. Prvky tohoto poľa sa nazývajú pixely.

Keďže pri tomto procese redukujeme spojitú funkciu na množinu bodov, je zrejmé, že pri nevhodne zvolených parametroch môže dôjsť k strate informácie a hrozí vznik aliasingu. Tomu je možno predísť dodržaním Shannonovho teorému, ktorý hovorí, že signál so spojitým časom je možné plne určiť postupnosťou vzoriek, ak platí podľa [2]

$$f_{\text{vz}} > 2f_{\text{max}}, \tag{2.1}$$

kde f_{vz} je vzorkovacia frekvencia a f_{max} je hodnota najvyššej frekvencie vo funkcii. Čerpané z [2] a [4].

2.2 Kvantovanie

Kvantovaním prevádzame spojitý obor hodnôt signálu na diskrétny. Podľa toho, či je veľkosť intervalu medzi jednotlivými kvantizačnými hladinami konštantná alebo nie, delíme kvantovanie na uniformné a neuniformné. Pri uniformnom kvantovaní zostáva vzdialenosť medzi hladinami rovnaká, pri neuniformnom sa jednotlivé vzdialenosti líšia. Napriek možnosti zohľadniť nerovnomerné rozloženie oboru hodnôt kvantovanej veličiny, je neuniformné kvantovanie, z dôvodu vyššej výpočtovej náročnosti a zložitejšej implementácie menej používané.

V dôsledku vyjadrenia spojitej veličiny pomocou konečného počtu hodnôt, dochádza k zaokrúhľovaniu hodnôt na hodnotu najbližšej kvantovacej hladiny. Odchýlka nakvantovanej hodnoty od hodnoty pred kvantovaním sa nazýva kvantizačný šum.[2][3]

3 SPRACOVANIE OBRAZU

Spracovanie obrazu je široký pojem, ktorý zahŕňa rôzne spôsoby manipulácie, úpravy a analýzy obrazu pomocou matematických operácii a algoritmov. Môže byť využité pre rôzne účely ako počítačové videnie, strojové videnie, rozpoznávanie obrazcov a iné. Postup pri spracovaní obrazu je obvykle nasledovný:

1. Získanie vstupného obrazu pomocou skeneru, digitálneho fotoaparátu alebo iného vstupného zariadenia.
2. Aplikovanie požadovaných matematických operácii alebo algoritmov.
3. Uloženie alebo zobrazenie výsledku. Výsledkom môže byť upravený obrázok, prípadne získané dáta.

3.1 Filtrácia obrazu v priestorovej oblasti

Obrázky získané napríklad pomocou kamery, často trpia vadami spôsobenými nevhodnými podmienkami pri ich vzniku. Tieto nedostatky je možné odstrániť aplikáciou vhodného filtra na obrázok. Pod pojmom filtrácia rozumieme operáciu, ktorá potláča alebo zosilňuje určité zložky signálu.

Ak pri filtrácii manipulujeme priamo s hodnotami obrazovej funkcie v priestorovej doméne, bez nutnosti transformácie, hovoríme o filtrácii v priestorovej oblasti.[5]

3.1.1 Konvolúcia

Konvolúcia je matematická operácia dvoch funkcií, často využívaná pri spracovaní signálu a obrazu. Jej tvar pre dva spojité signály je podľa [2]

$$y(t) = h(t) * x(t) = \int_{-\infty}^{\infty} h(\tau) f(t-\tau) \mathrm{d}\tau. \tag{3.1}$$

Funkcii $h(t)$ sa hovorí konvolučné jadro, a určuje typ filtra, ktorý bude na signál aplikovaný. To znamená, že konvolúciou týchto dvoch funkcií vznikne funkcia, ktorej hodnoty sú vlastne určitým druhom váženého priemeru hodnôt pôvodnej funkcie $x(t)$, kde váha jednotlivých prvkov je daná maskou $h(t)$.

Diskrétna konvolúcia je definovaná analogicky k vzorcu (3.1) ako

$$y[m] = \sum_{s=-\infty}^{\infty} h[s]x[m-s]. \tag{3.2}$$

Pri práci s konečnými postupnosťami je možné, za predpokladu že h a x majú rovnakú dĺžku N a cyklicky sa opakujú. Získame tak cyklickú konvolúciu, ktorej vzorec môžme zapísať podľa [2]

$$y[m] = \sum_{s=0}^{N-1} x[m]h[(m-n)_{mod\ N}] \text{ kde } m = 0, 1, \ldots N-1. \tag{3.3}$$

Pri práci s obrazom sa však používa diskrétna dvojrozmerná konvolúcia. Jej vzorec je [1]

$$y[m,n] = \sum_{s=-\infty}^{\infty} \sum_{t=-\infty}^{\infty} h[s,t]x[m-s,n-t]. \tag{3.4}$$

Funkcia x predstavuje v tomto vzorci obrázok a funkcia h dvojrozmerné jadro. Ako jadro sa obvykle používa štvorcová matica s nepárnym počtom prvkov, čo umožňuje stotožnenie vyšetrovaného bodu s centrálnym bodom masky. Pri práci s farebnými obrázkami je potrebné konvolúciu aplikovať na každý kanál zvlášť. To znamená, že pri farebnom priestore RGB je potrebné konvolúciu vykonať trikrát.

Jedným z problémov vznikajúcich pri tejto operácii je ako sa vysporiadať s hodnotami okrajových pixelov obrázku. Riešení je niekoľko

1. Rozšírenie obrázku o potrebný počet pixelov, ktorým priradíme hodnotu najbližšieho suseda.
2. Periodizácia obrázku. To znamená, že pri počítaní konvolúcie budú hodnoty doplnené z protiľahlého okraju.
3. Orezanie okrajov.
4. Ponechanie okrajov. Na obrázku zostane minimálne z jednej strany čierny okraj.

Čerpané z [1][4][5].

3.1.2 Vyhladzovacie filtre

Vyhladzovacie filtre sú založené na potlačení vyšších frekvencií v obrazovej funkcii. Je možné ich využiť napríklad na potlačenie šumu v obrázku. Nežiadúcim efektom týchto filtrov je rozmazanie hrán a ostrých čiar, čo môže výrazne ovplyvniť výpovednú hodnotu obrázku.

Charakteristickou vlastnosťou týchto filtrov je, že súčet všetkých prvkov konvolučného jadra je rovný jednej.[3]

Priemerovací filter

Jeden z možných spôsobov rozostrenia obrazu je pomocou priemerovacieho filtra. Tento filter nahradí pôvodnú hodnotu aritmetickým priemerom okolitých hodnôt. Veľkosť filtra je potrebné zvoliť tak, aby malé detaily zostali zachované. Z tohto dôvodu by mala byť veľkosť jadra menšia, ako veľkosť najmenšieho významného detailu. Konvolučné jadro priemerovacieho filtra o veľkosti 3×3 podľa [3] má tvar

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

Čerpané z [2], [3].

(a) originál s pridaním šumom

(b) priemerovací

(c) Gaussov

Obr. 3.1: Ukážky vyhladzovacích filtrov.

Gaussovo rozostrenie

Tento typ filtra je založený na využití vlastností dvojrozmernej Gaussovej krivky. Túto funkciu môžme pre viacero premenných, podľa [10] zapísať ako

$$f(x_1, \ldots, x_k) = \frac{\exp(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^{\mathrm{T}} \sum^{-1} (\boldsymbol{x} - \boldsymbol{\mu}))}{\sqrt{(2\pi)^k |\Sigma|}}, \tag{3.6}$$

kde $\boldsymbol{x}$ je k-rozmerný stĺpcový vektor, $|\Sigma|$ je determinant matice a k udáva počet premenných. V prípade že $k = 2$ platí

$$\boldsymbol{\mu} = \begin{pmatrix} \mu_X \mu_Y \end{pmatrix}, \Sigma = \begin{pmatrix} \sigma_X^2 & \rho\sigma_X\sigma_Y \\ \rho\sigma_X\sigma_Y & \sigma_Y^2 \end{pmatrix}. \tag{3.7}$$

Hodnoty matice $\boldsymbol{\mu}$ udávajú posun funkcie od počiatku súradníc v osiach. Jednotlivé prvky matice Σ určujú tvar výslednej funkcie.

Konvolučnú maticu získame navzorkovaním tejto funkcie a upravením hodnôt tak, aby spĺňali podmienku, že súčet všetkých prvkov sa musí rovnať jednej. Matica

Gaussovho filtru môže mať podľa [3] podobu napríklad [10][3]

$$h = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}. \tag{3.8}$$

3.1.3 Hranové filtre

Detekcia hrán je jedna zo základných operácií spracovania obrazu, na ktorej je postavených mnoho ďalších aplikácií, ako napr. počítačové videnie, detekcia objektov a iné. Hranu môžeme v obraze definovať ako miesto, kde sa prudko mení amplitúda obrazovej funkcie. Z toho vyplýva že hranu v obraze je možné detekovať za pomoci derivácie, ktorej výsledku hovoríme gradient. Veľkosť gradientu môžme podľa [4] spočítať ako

$$|\nabla g(x,y)| = \sqrt{\left(\frac{\mathrm{d}g}{\mathrm{d}x}\right)^2 + \left(\frac{\mathrm{d}g}{\mathrm{d}y}\right)^2} \tag{3.9}$$

a jeho smer

$$\psi = \arg\left(\frac{\mathrm{d}g}{\mathrm{d}x}, \frac{\mathrm{d}g}{\mathrm{d}y}\right), \tag{3.10}$$

kde $\arg(x,y)$ je uhol v radiánoch, ktorý zviera osa X a bod (x,y). Tieto funkcie platia iba pre spojitý obraz, a preto je pri diskrétnych obrazoch potreba gradient odhadovať. Pri práci s digitálnym obrazom je na aplikáciu filtru možné použiť konvolúciu. Ukážky jednotlivých filtrov sú na obrázku 3.2.[1][4]

Robertsov operátor

Tento operátor pre výpočet používa konvolučné jadro o veľkosti 2×2. Je výpočtovo jednoduchý, ale malá veľkosť jadra spôsobuje vysokú citlivosť na šum. Jeho konvolučná maska môže mať jeden z tvarov [4]

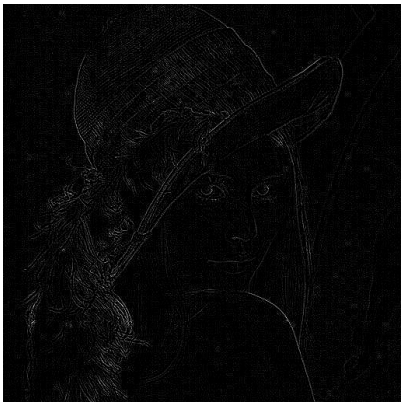
$$h_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, h_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}. \tag{3.11}$$

Laplaceov operátor

V prípade Laplaceovho operátora prebieha detekcia hrán na základe aproximácie druhej derivácie. Konvolučné jadro nieje citlivé na smer hrán a môže mať tvary [4]

$$h_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}, h_2 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}. \tag{3.12}$$

(a) originál



(b) Laplace

(c) Prewitt

(d) Sobel

Obr. 3.2: Ukážky hranových filtrov.

Prewittovej operátor

Je to jeden z množstva operátorov, využívajúcich aproximáciu prvej derivácie. Konvolučná maska Prewittovej operátora je citlivá na smer hrany. Smer detekcie hrany môžme zmeniť otočením jadra. Jej jadro môže nadobúdať tvary [4]

$$h_1 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}, h_2 = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}, \dots \tag{3.13}$$

Sobelov operátor

Sobelov operátor, rovnako ako Prewittovej operátor, pre svoju funkciu využíva prvú deriváciu, čo znamená, že je taktiež citlivý na smer hrany a konvolučná maska môže

nadobúdať viacero tvarov pre rôzne smery detekcie hrán.[3]

$$h_1 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, h_2 = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -2 & 0 \end{bmatrix}, \ldots \tag{3.14}$$

3.2 Filtrácia obrazu vo frekvenčnej oblasti

Tento spôsob filtrácie je založený na transformácii obrazu do frekvenčnej oblasti za pomoci Fourierovej transformácie. Z obrazu vo frekvenčnej oblasti môžme následne potlačiť alebo odfiltrovať nízkofrekvenčné, alebo vysokofrekvenčné zložky. Odfiltrovaním nízkych frekvencií získame hranový obraz. Odfiltrovaním frekvencií vysokých sa obraz rozostrí a zmiznú z neho jemné hrany. Filter je aplikovaný vynásobením obrazu vo frekvenčnej oblasti a filtračnej funkcie prvok po prvku. Obraz v časovej oblasti získame spätným prevodom pomocou inverznej Fourierovej transformácie.[8]

3.2.1 Fourierova transformácia

Patrí medzi najdôležitejšie matematické nástroje, využívané pri spracovaní signálov a obrazu. Táto transformácia umožňuje vzájomne jednoznačný prevod signálu medzi reprezentáciou v časovej alebo priestorovej doméne a frekvenčnej doméne. Tvar Fourierovej transformácie pre spojitý signál je podľa [6]

$$\mathcal{F}\{f(\omega)\} = F(\omega) = \int_{-\infty}^{+\infty} f(t)\mathrm{e}^{-\mathrm{i}\omega t}\mathrm{d}t, \tag{3.15}$$

kde $F(u)$ je Fourierovým obrazom funkcie $f(x)$ a i je imaginárna jednotka. Ak funkciu zdigitalizujeme integrál sa nahradí sumou všetkých navzorkovaných bodov. Vzorec pre diskrétnu Fourierovu transformáciu (DFT) podľa [6] je nasledovný

$$F(\omega) = \sum_{k=0}^{N-1} f[k]\mathrm{e}^{\frac{2\pi j\omega k}{N}}. \tag{3.16}$$

Dôležitou vlastnosťou Fourierovej transformácie spojitých signálov je takzvaný konvolučný teorém, ktorého rovnice majú podľa [7] tvar

$$\mathcal{F}\{f(x,y) * g(x,y)\} = F(u,v)G(u,v), \tag{3.17}$$

$$\mathcal{F}\{f(x,y)g(x,y)\} = F(u,v) * G(u,v). \tag{3.18}$$

Prvá rovnica hovorí, že Fourierovým obrazom konvolúcie funkcií f a g je súčin Fourierových obrazov F a G. Podobne podľa druhej rovnice je Fourierovým obrazom súčinu funkcií f a g konvolúcia Fourierových obrazov F a G. Pre distkrétne signály obdobne plati že DFT cyklickej konvolúcie dvoch signálov je súčinom DFT týchto signálov.[2][7]

3.2.2 Rýchla Fourierova transformácia

V anglickej literatúre sa vyskytuje pod názvom Fast Fourier Transform alebo skrátene FFT. Predstavuje efektívny spôsob urýchlenia výpočtu diskrétnej Fourierovej transformácie. Existuje veľké množstvo rôznych spôsobov výpočtu tohoto algoritmu. Snáď najpoužívanejším je algoritmus Cooley-Tukey. Tento algoritmus má široké použitie pri spracovaní signálov, obrazov.[8]

Koncept FFT

Algoritmy FFT sú založené na rozložení N-bodovej diskrétnej Fourierovej transformácie na niekoľko menších M-bodových transformácií a nasledovnou kombináciou ich jednotlivých výsledkov. V ideálnom prípade pre spracovávaný signál platí $N = 2^m$. Signál s takouto dĺžkou je možné postupne rozložiť na párnu a nepárnu časť. Týmto rozdelením sa zredukoval počet operácii z N^2 na $2(N/2)^2$. V tomto delení je možné pokračovať až dokým sa nepočítajú dvojbodové DFT. Použitím tohoto postupu sa počet komplexných násobení zníži na $N/2 \log_2 N$. Tento algoritmus je možné paralelizovať čo ho robí vhodným pre rýchly výpočet DFT pomocou počítača alebo implementáciu pomocou špeciálneho hardvéru.[8][9]

Cooley–Tukey algoritmus

Autormi tohoto algoritmu sú J. W. Cooley a J. W. Tukey, po ktorých dostal svoje meno. Výpočet FFT začína výpočtom DFT

$$h_a = \sum_{b=0}^{N-1} g_b W_N^{ab}, \tag{3.19}$$

kde

$$W_N = \exp(\frac{-2\pi \mathrm{i}}{N}). \tag{3.20}$$

Táto suma je následne rozdelená na párne a nepárne členy.

$$h_a = \sum_{b=0}^{N/2-1} g_{2b} W_N^{a(2b)} + \sum_{b=0}^{N/2-1} g_{2b+1} W_N^{a(2b+1)} \tag{3.21}$$

Z DFT o dĺžke N sa stali dva výpočty s dĺžkou $N/2$

$$h_a = \sum_{b=0}^{N/2-1} g_{2b} W_{N/2}^{ab} + \sum_{b=0}^{N/2-1} g_{2b+1} W_{N/2}^{ab}, \tag{3.22}$$

pričom prvý člen reprezentuje výpočet prvkov na párnych pozíciach a druhý člen reprezentuje výpočet na nepárnych pozíciach.[9][8]

3.2.3 Odstránenie jednosmernej zložky

Vždy existuje prvok Fourierovej transformácie, ktorý nesie informáciu o jase všetkých pixelov v obraze. Vynulovaním tohoto prvku vo frekvenčnej oblasti, sa priemerná hodnota jasov všetkých pixelov zmení na nulu. Táto operácia predpokladá záporné hodnoty jasu, ktoré sú však fyzikálne nereálne. Zobrazenie výsledku so zápornými hodnotami je možné dvoma spôsobmi. Jedným je záporné hodnoty vynulovať, druhým je normovať záporné hodnoty do rozsahu podporovaného zobrazovacím zariadením.[5]

3.2.4 Filter typu dolná priepusť

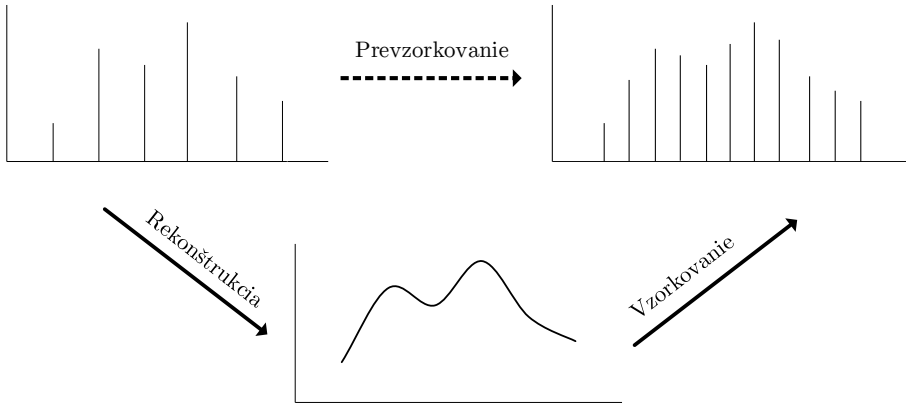
Tento typ filtra v ideálnom prípade oreže všetky vysokofrekvenčné zložky spektra, ktorých vzdialenosť od počiatku spektra je väčšia ako zadaná hranica.

3.2.5 Filter typu horná priepusť

Tieto filtre sú založené na odfiltrovaní nízkofrekvenčných zložiek spektra, ktorých vzdialenosť od počiatku je menšia než definovaná hranica.[5]

3.3 Prevzorkovanie

Pri práci s obrazom často vzniká potreba zmeniť veľkosť obrazu. Ak je potrebné zväčšiť rozlíšenie obrazu dvakrát, množstvo pixelov vzrastie štvornásobne. Novovzniknutým pixelom však treba priradiť hodnoty. V počítačovej grafike sa využíva proces, nazývaný prevzorkovanie. Pozostáva z dvoch častí: rekonštrukcia obrázku nasledovaná novým navzorkovaním, ako je možné vidieť na obrázku 3.3.



Obr. 3.3: Operácia prevzorkovania.

Proces a spôsoby rekonštrukcie sú podrobnejšie popísané ďalej v tejto kapitole. Vzorkovanie je rozobrané v časti 2.1.

3.3.1 Rekonštrukcia

Pod pojmom rekonštrukcia môžme rozumieť aproximáciu spojitej funkcie $f(x), x \in R$, na základe známych diskrétnych hodnôt. Pri spracovaní obrazu sa využíva na vyplnenie prázdnych miest pri zmene veľkosti obrázku. Pôvodnú funkciu, z ktorej vzorky vznikli, nieje možné spätne získať. Existuje viacero spôsobov interpolácie funkcie. Vlastnosti týchto metód sa líšia a platí, že so zvyšujúcou kvalitou zároveň rastie aj jej výpočtová náročnosť.[2]

Interpolácia najbližším susedom

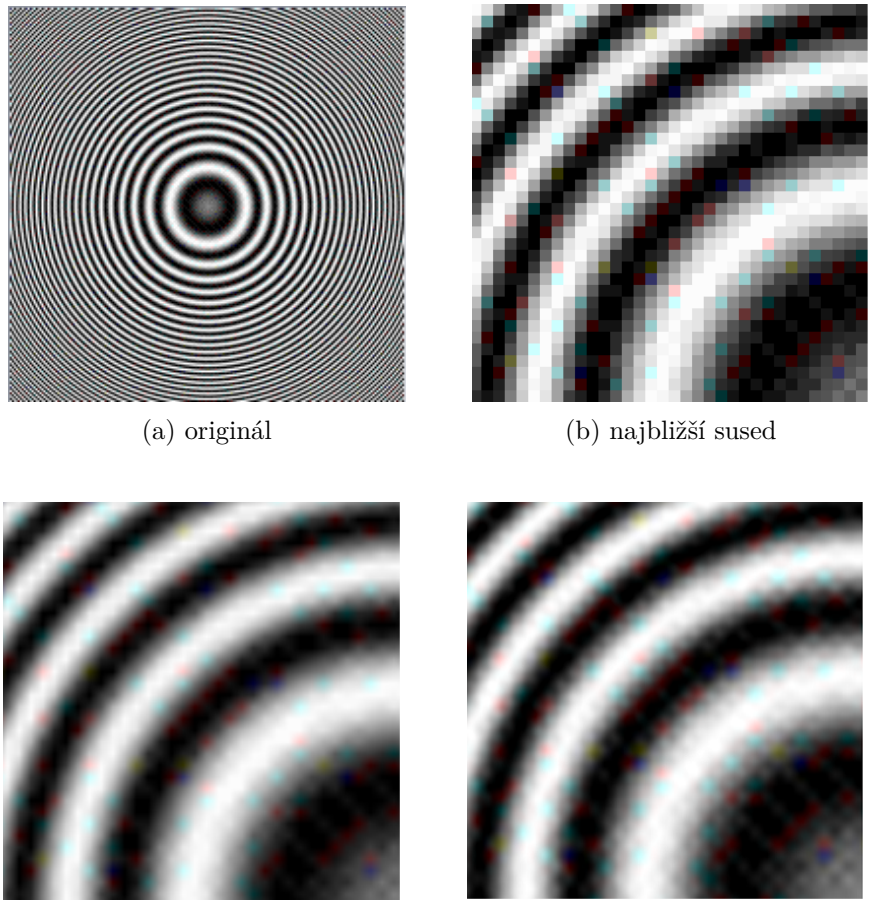
Táto metóda je založená na doplnení hodnoty nového pixelu hodnotou najbližšieho známeho pixelu. Jednou z výhod tejto metódy je, že do obrázku nevnáša nové farby. Taktiež je jednoduchá na implementáciu a výpočetne nenáročná. Metóda zachováva ostrosť zvislých a vodorovných hrán. Pri diagonálnych hranách a zaobleniach dochádza k vzniku zubov. Táto metóda je z dôvodu jej rýchlosti a jednoduchosti často používaná pri rýchlom náhlade obrázkov.[2]

Bilineárna interpolácia

Predstavuje dvojrozmernú verziu lineárnej interpolácie. Lineárna interpolácia vytvára spojitú funkciu prepojením dvojíc susedných bodov. V prípade bilineárnej interpolácie nová funkcia vzniká prepojením štvoríc bodov. Je ju možné realizovať pomocou dvoch operácií lineárnej interpolácie tak, že sú spracované prvky postupne v oboch smeroch. Keďže sú na jej výpočet potrebné len štyri okolité body je jej výpočet relatívne rýchly. Nevýhodou tejto interpolácie pri spracovaní obrazu je rozmazanie ostrých hrán.[2][4]

Bikubická interpolácia

Táto interpolácia je založená na kubickej interpolácii a predstavuje spôsob ako túto interpoláciu aplikovať v dvojrozmernom priestore. Využíva interpolačné funkcie s hladším priebehom. Pre svoj výpočet potrebujú 16 bodov zo svojho okolia, čo ju robí výpočtovo náročnejšiu. Bikubická interpolácia nemá problém so schodmi vznikajúcimi pri použití interpolácie najbližšieho suseda a netrpí rozmazaním ako bilineárna interpolácia.[4]



(a) originál (b) najbližší sused

(c) bilineárna (d) bikubická

Obr. 3.4: Ukážky metód prevzorkovania pri nadvzorkovaní desaťnásobne.

3.4 Dithering

Redukciou počtu farebných odtieňov v obraze, z dôvodu napr. zníženia veľkosti súboru, dochádza k strate informácie. Dithering je metóda, ktorá umožňuje túto stratu minimalizovať a vytvoriť ilúziu hĺbky farby tým, že nedostupné farby vytvorí rozptylom pixelov farieb nachádzajúcich sa v palete. To je možné, pretože ľudské oko vníma viacero blízko seba ležiacich pixelov ako jeden bod, ktorého farba je daná aditívnym zložením farieb pixelov.[2]

3.4.1 Náhodné rozptýlenie

Metóda náhodného rozptýlenia je jednoduchým, ľahko implementovateľným spôsobom realizácie ditheringu. Všetky pixely sú v počiatku nastavené na hodnotu 0. Pre každý pixel v obrázku je vygenerované náhodné číslo s hodnotou v intervale $\langle 0, I_{\max}\rangle$, kde $I_{\max}$ predstavuje maximálnu dosiahnuteľnú intenzitu pixelu vstupného obrazu.

Následne je hodnota vygenerovaného čísla porovnaná s hodnotou intenzity pôvodného pixelu. Ak je vygenerovaná hodnota vyššia, je tomuto pixelu priradená hodnota 0, ktorá predstavuje čiernu farbu. Ak je jeho hodnota nižšia je mu priradená 1. Ak je počet farieb v palete, do ktorej obrázok prevádzame väčší ako 2, je potrebné tento postup opakovať pre každú farbu v palete. V každom cykle je vygenerované nové náhodné číslo. Ak je jeho hodnota vyššia ako hodnota pôvodnej intenzity je hodnota zväčšená o 1. Na základe výslednej hodnoty je následne priradená každému pixelu farba z palety. Výsledný obrázok, získaný použitím tejto metódy pôsobí zrnito a väčšie plochy pôsobia dojmom drsného povrchu.[2]

3.4.2 Maticové rozptýlenie

Toto rozptýlenie je založené na použití predom daných matíc. Tie môžu vyzerať napríklad takto [2]

$$\begin{array}{ccccc} \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} & \begin{bmatrix} 0 & 0 \\ 1 & 1 \end{bmatrix} & \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix} & \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \\ C_{\text{in}} = 0 & C_{\text{in}} = 1 & C_{\text{in}} = 2 & C_{\text{in}} = 3 & C_{\text{in}} = 4 \end{array} \tag{3.23}$$

Kde C_{in} predstavuje hodnotu, pri ktorej bude pixel nahradený príslušnou maticou. Hodnotu intenzity vstupného pixelu je pred porovnaním potrebné upraviť. Rozsah možných intenzít, ktoré pixel môže nadobudnúť je rozdelený na časti, ktorých počet zodpovedá počtu rôznych matíc. Podľa toho, v ktorej z týchto častí sa hodnota intenzity nachádza je zvolená príslušná rozptylovacia matica. Tento spôsob ditheringu by mal za následok zväčšenie obrázku. Ak je potrebné zachovať jeho veľkosť, je možné túto metódu upraviť tak, že je naraz spracovávaných viacero pixelov. Počet spracovávaných pixelov zodpovedá počtu prvkov matice a hodnota C_in je daná aritmetickým priemerom ich intenzít.

Po sebe nasledujúce matice sa od seba líšia pridaním jednotky na jednu novú pozíciu. Vďaka tejto vlastnosti je možné ich zapísať ako jednu maticu

$$\begin{bmatrix} 0 & 3 \\ 2 & 1 \end{bmatrix}. \tag{3.24}$$

S rastúcim rozsahom intenzít je vhodné použiť väčšiu maticu. Pre rôzne zobrazovacie zariadenia sa používajú rôzne usporiadania matíc.

$$M_{\text{d}} = \begin{bmatrix} 0 & 12 & 3 & 15 \\ 8 & 4 & 11 & 7 \\ 2 & 14 & 1 & 13 \\ 10 & 6 & 9 & 5 \end{bmatrix}, M_{\text{p}} = \begin{bmatrix} 1 & 5 & 9 & 2 \\ 8 & 12 & 13 & 6 \\ 4 & 15 & 14 & 10 \\ 0 & 11 & 7 & 3 \end{bmatrix}. \tag{3.25}$$

Matica M_d je vhodná pre displeje a matica M_p je vhodná pre tlačenie. [2]

3.4.3 Rozptyl s distribúciou odchýlky

Tento spôsob rozptylu je založený na vytvorení vzoru pixelov tak, aby priemerná hodnota intenzity v oblasti približne zodpovedala intenzite v pôvodnom obrázku. Postup pozostáva z nájdenia najbližšej farby dostupnej v novej palete, výpočtu odchýlky novej farby od farby pôvodnej a následného rozptýlenia tejto chyby do okolitých, ešte nespracovaných pixelov. Existuje viacero spôsobov ako chybu rozptýliť. Najznámejším je difúzia podla Floyd-Steinberga. Rozdelenie chýb môžme zapísať ako maticu

$$\begin{bmatrix} & x & \frac{7}{16} \\ \frac{3}{16} & \frac{5}{16} & \frac{1}{16} \end{bmatrix} \tag{3.26}$$

kde x znázorňuje práve spracovávaný pixel a číselné hodnoty udávajú pomer rozdelenia chyby medzi okolité pixely.[2]

(a) originál (b) Náhodné rozptýlenie

(c) maticové rozptýlenie (d) rozdelenie s distribúciou chyby

Obr. 3.5: Ukážky metód ditheringu.

4 POUŽITÉ PROGRAMOVACIE JAZYKY A VÝVOJOVÉ PROSTREDIE

Súčasné webové aplikácie je možné tvoriť v nesmiernom množstve jazykov a prostrediach. Veľká výhoda webových stránok spočíva v ich dostupnosti z ľubovoľnej destinácie a pri vhodne zvolených technológiách ich platformová nezávislosť. Voľba jazyka závisí od požiadaviek na aplikáciu. Pre projekt je požiadavka vytvoriť interaktívnu webovú aplikáciu pre podporu výuky. Keďže jednotlivé aplikácie budú jednostránkové a ich prvky sa budú musieť dynamicky meniť bol ako najvhodnejší zvolený jazyk Javascript. Jazyk JavaScript má svoje nedostatky, ale tie sú pomerne jednoducho riešiteľné pomocou rôznych knižníc a frameworkov, napríklad aj pomocou jQuery knižníc alebo nastavením prehliadačov stránok.

4.1 HTML

Hypertext Markup Language – hypertextový značkovací jazyk. Ide o jazyk používajúci značky (tagy) a ich vlastnosti (atribúty). Je používaný k tvorbe dokumentov obsahujúcich text, hypertextové odkazy, multimediálny, skripty a metainformácie prehliadateľné v tzv. webovom prehliadači. HTML jazyk je populárny aj v elektronickej pošte. Vychádza zo značkovacieho jazyka SGML (Standard Generalized Markup). HTML slúži na popis štruktúry webových dokumentov. Tagy bývajú uzavreté do ostrých zátvoriek $\langle\rangle\langle/\rangle$. HTML dokument možno otvoriť ako textový súbor, kedy sú vidieť aj jednotlivé tagy, atribúty a možno ho editovať. Až po otvorení dokumentu cez prehliadač sú tagy spracované a dokument prezentovaný ako štruktúrovaná stránka bez tagov. HTML jazyk a jeho znalosť je nutnosťou pre tvorbu web stránok a pri použití jazykov pre web stránky ako PHP, JavaScript,... [12][13]

4.2 CSS

Pôvodne HTML nebolo navrhnuté na úpravu vzhľadu a designu stránok. Riešením tohto problému je jazyk CSS (Cascading Style Sheets – kaskádové štýly), ktorý pomocou vlastných štýlov umožňuje upraviť vzhľad stránky. CSS vlastne definuje ako bude element HTML zobrazený na obrazovke, papieri alebo inom médiu. Názov kaskádové vyjadruje možnosť prekrývania sa pripojených štýlov. Kaskáda určuje spôsob aplikovania pravidiel. Užívateľský štýl potlačí ostatné štýly, individuálne štýly potlačia vložené, pripojené a importované štýly. Vložené štýly majú prednosť pred pripojenými štýlmi. Pripojené a importované štýly sú rovnocenné a aplikujú sa všade, kde nie sú použité iné druhy štýlov. Pokiaľ máme pripojené viaceré súbory

so štýlom a dôjde ku konfliktu bude interpretovaný posledný súbor. Výhody CSS: [14][15]

- Použitie kaskádových štýlov ušetrí veľké množstvo práce, umožňuje kontrolovať hromadne vzhľad internetových stránok v jednom súbore. Tento súbor definuje presne vzhľad a správanie stránok. Šetrí čas.
- CSS ma výrazne širšiu ponuku atribútov ako HTML a teda otvára nové možnosti realizácie internetovej stránky.
- Sprehľadňuje kód a umožňuje oddeliť HTML časť od CSS časti kódu. V dnešnej dobe preferovaná metóda.

4.3 JavaScript

Ide o objektový jazyk, interpretovaný na strane klienta, závislý na prehliadači. Ale existujú aj varianty napr. Node.js, ktoré sa vykonávajú na strane servera, a teda slúžia na programovanie aplikácií pre server. Tento jazyk je citlivý na veľkosť písma a syntaxou je podobný jazyku C a Jave. V prípade, ak hovoríme o JavaScriptoch interpretovaných na strane klienta je skript vykonávaný v prehliadači, čo ma za následok odľahčenie servera a zároveň zaťaženie klientského zariadenia. Nefunkčnosť aplikácie môže spôsobiť zakázanie skriptov v prehliadači na klientskom PC. JavaScript nepotrebuje žiadny špeciálny prvok (webserver) alebo prídavný program, postačuje prehliadač webových stránok. Použitie JavaScriptu:

- Oživenie stránok
- Dynamické HTML
- Overovanie vstupných údajov
- CGI (Common Gateway Interface) na strane klienta
- Odľahčenie zaneprázdneného servera
- Spracovanie cookies

Nedostatky spojené s nekompatibilitou prehliadačov, ale aj nesmierne množstvo funkcií je vyriešených v knižniciach pre JavaScript, čo výrazne zvyšuje efektivitu tohto jazyka. Ide napríklad o jQuery, AngularJS, a iné.[16][17]

4.4 jQuery

Ide o rozšírenú JavaScriptovú knižnicu s jednoduchou syntaxou umožňujúcou jednoduchšiu manipuláciu s obsahom webu a zároveň rieši kompatibilitu medzi prehliadačmi. JQuery knižnica obsahuje nasledovné funkcie:

1. Manipuláciu s obsahom stránky
2. CSS manipuláciu
3. Metódy HTML udalostí
4. Efekty a animácie
5. Nástroje

Okrem spomínaných funkcií jQuery obsahuje doplnky (plugins) takmer ku každej úlohe a neustále sa rozvíja.[18]

4.5 Použité prvky jazykov

V tejto kapitole budú bližšie popísané dôležité elementy jazykov HTML a JavaScript použité pri realizácii zadaných aplikácií.

4.5.1 Canvas

Je grafický HTML prvok, ktorý umožňuje kreslenie grafiky za behu programu. Prvok canvas sám o sebe predstavuje len prázdny priestor. Pomocou jazyka JavaScript je však možné do tohto priestoru vkladať ľubovoľné tvary, načítať obrázky alebo dokonca spracovávať video v reálnom čase. To ho robí centrálnym prvkom v aplikáciách popisovaných v tejto práci.

Prvok canvas využíva farebný model RGBα, kde intenzity jednotlivých farebných zložiek pixelu sú reprezentované jedným bajtom a môžu dosahovať celočíselné hodnoty v rozsahu 0–255. Obsah prvku canvas je možné načítať do jednorozmerného poľa. Toto pole má dĺžku štvornásobku počtu pixelov a každý pixel je reprezentovaný štyrmi prvkami zodpovedajúcimi modelu RGBα. Pixely sú načítavané postupne zľava doprava a z hora smerom dole. S jednotlivými pixelmi načítaného obrázku možno manipulovať, meniť hodnoty jednotlivých zložiek alebo napríklad aplikovať filter. Modifikovaný obrázok môže byť následne znova vložený a vykreslený. Pri práci s týmto prvkom sú dôležité parametre width a height, ktoré udávajú rozmery prvku canvas v pixeloch.[19][20]

4.5.2 Web Worker

Predstavuje jednoduchý spôsob vykonávania JavaScript skriptov nezávisle v pozadí bez toho, aby prerušili beh stránky. Pri bežných skriptoch sa stránka na čas potrebný na ich vykonanie „zamkne“, čo môže znepríjemniť prácu s výpočtovo náročnejšími, interaktívnymi aplikáciami. Web worker sa tomuto problému vyhne tým, že vykonáva skripty na ďalšom, oddelenom vlákne v pozadí. Webová aplikácia môže využívať simultánne niekoľko Web Workerov. Pri ich použití treba zohľadniť, že vlákna

pracujú asynchrónne. Z tohto dôvodu je Web Workerom znemožnený prístup ku premenným, ktoré sa nachádzajú v skripte vykonávanom hlavným vláknom alebo iným Web Workerom. Komunikácia medzi Web Workerom a hlavným skriptom prebieha prostredníctvom správ, ktoré sú posielané metódou postMessage(aMessage). Argument tejto metódy obsahuje dáta, ktoré budú poslané danému Web Workeru alebo dáta, ktoré Web Worker posiela hlavnému vláknu. Na spracovanie prijatej správy slúži eventhandler onmessage, ktorý vykoná jemu priradený skript. Každý Web Worker musí mať svoj vlastný súbor, v ktorom sa nachádza kód ktorý má vykonávať. Web Worker nemá prístup k súborom a knižniciam, ktoré boli pridané v súbore HTML. Tieto súbory je však možné pridať pomocou metódy importScripts().[21][22]

4.6 Vývojové prostredie

Aplikácie boli písané vo vývojovom prostredí Visual Studio Code od spoločnosti Microsoft. Toto prostredie podporuje inštaláciu balíkov IntelliSense pre všetky jazyky použité pri realizácii tejto aplikácie. Tento balík zahŕňa funkcie zjednodušujúce písanie kódu, ako napríklad, automatické dopĺňanie príkazov, farebné značenie kódu atď. Taktiež je možné priamo v aplikácii využívať prepojenie s verzovacím systémom GitHub. Aplikácia bola priebežne ukladaná do repozitára a testovaná pomocou hostingu GitHub Pages poskytovaného zadarmo GitHubom. Na priebežné testovanie bol využívaný taktiež WampServer, ktorý umožňoval rýchle testovanie aplikácií bez potreby prístupu na internet. Kód aplikácie je rozdelený na tri súbory, z ktorých každý obsahuje len jeden jazyk. Výhodou tohoto delenia je najmä väčšia prehľadnosť kódu, ale aj možnosť zmeny štýlu stránky jednoduchým nahradením CSS súboru.

5 NÁVRH A REALIZÁCIA APLIKÁCIÍ

V tejto kapitole je rozobraná praktická časť práce. Sú tu postupne popísané jednotlivé aplikácie. Popis každej aplikácie pozostáva zo stručného vysvetlenia účelu aplikácie, popisu jej užívateľského rozhrania a funkcie, a objasnenia spôsobu jej implementácie. Nakoniec je pri každej aplikácií spomenutá jej kompatibilita s jednotlivými prehliadačmi. Testovanie aplikácií prebiehalo na prehliadačoch Mozzila Firefox 52.8, Google Chrome 66, Internet Explorer 11, Microsoft Edge 40 a Opera 53. Aplikácie sa podarilo spustiť aj na mobilnom zariadení s operačným systémom Android. Doba spracovania jednotlivých operácií bola výrazne dlhšia ako pri otvorení na počítači. To je spôsobené vlastnosťami jazyka JavaScript, ktorý je závislý na použitom prehliadači a hardvéri. Tieto aplikácie neboli pre mobilné aplikácie optimalizované zo strany programu ani rozhrania. Z tohto dôvodu nebudú v kompatibilite spomínané.

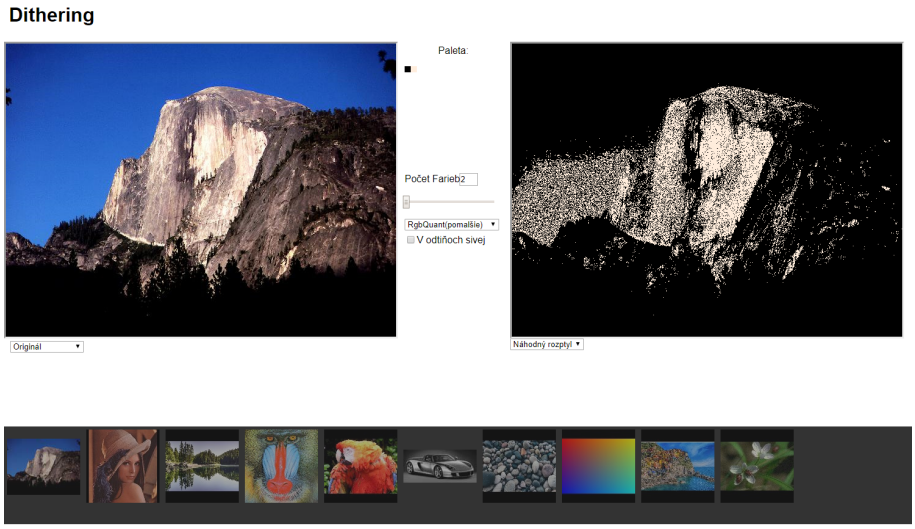
Aplikácie sú spustiteľné aj priamo z disku. Na ich spúšťanie je však vhodné použiť nejakú formu serveru. Niektoré prehliadače ako napríklad Google Chrome z bezpečnostných dôvodov zabránia načítaniu obrázkov z disku. To je možné obísť napríklad použitím prehliadača Mozilla Firefox alebo použitím lokálneho serveru ako napríklad Wamp. Aplikácie sú taktiež umiestnené na stránke `xsunal01.github.io`.

5.1 Dithering

Táto aplikácia bola navrhnutá za účelom praktickej ukážky ditheringu obrazu. Umožňuje odskúšať si prevod obrázku do palety so zvoleným počtom farieb, za použitia algoritmov popísaných v časti 3.4.

5.1.1 Rozhranie aplikácie

Aplikácia sa skladá z dvoch častí určených na zobrazovanie spracovávaných obrázkov a časti na správu palety. Zobrazovacie časti sú zhodné a nachádzajú sa v ľavej a pravej strane okna a umožňujú efektívne porovnať rôzne metódy ditheringu. Tieto časti obsahujú prvok canvas. V tejto časti je umiestnené aj rozbaľovacie menu, ktoré umožňuje výber jednej zo štyroch metód prevodu do palety a zobrazenie pôvodného obrázka. Medzi zobrazovacími časťami je umiestnený panel na prácu so zvolenou paletou. Tento panel taktiež zahŕňa prvok canvas, ktorý slúži na zobrazenie jednotlivých farieb v novej palete. Jeho rozmery sú dynamicky určené zvoleným počtom farieb. Pod týmto prvkom je umiestnené textové pole, ktoré číselne zobrazuje počet farieb nachádzajúcich sa v palete. Taktiež sa tu nachádza slider. V spodnej časti okna sa nachádza panel pre výber jedného zo sady obrázkov.



Obr. 5.1: Aplikácia dithering vo východzom stave.

5.1.2 Funkcia aplikácie

V počiatočnom stave je v jednej zobrazovacej časti zobrazený pôvodný obrázok a v druhej obrázok prevedený do prednastavenej palety pomocou algoritmu náhodného rozptylu. Predvolená paleta obsahuje 2 farby. Tie je možné vidieť v prvku canvas, ktorý sa nachádza medzi zobrazovacími časťami. Počet farieb v palete môže užívateľ meniť prepísaním hodnoty pod týmto prvkom alebo za pomoci slideru. Táto hodnota je obmedzená na celé čísla v intervale 1 až 256. Pri týchto množstvách farieb sa dithering prejavuje najvýraznejšie. Vytváranie palety s väčším množstvom farieb taktiež spomaľovalo beh aplikácie. Po zmene sú obrázky v zobrazovacej časti do novej palety prevedené. Užívateľ môže zvoliť spôsob prevodu pomocou rozbaľovacích menu nachádzajúcich sa pod každým prvkom canvas. Toto menu, pre porovnanie, obsahuje taktiež možnosť zobrazenia pôvodného obrázka. Je možné vybrať si jeden z poskytnutých obrázkov pomocou lišty nachádzajúcej sa v spodnej časti okna. Po výbere nového obrázku dôjde k vytvoreniu novej palety. Obrázok bude do tejto palety následne prevedený zvolenou metódou.

5.1.3 Implementácia

HTML elementom, ktoré majú byť interaktívne sú pri inicializácii programu priradené funkcie pomocou EventListenerov. Tie zabezpečujú že sa daná funkcia spustí pri špecifickej interakcii s HTML prvkom. Aplikácia pre svoju funkciu využíva neviditeľný prvok canvas, do ktorého je načítaný zdrojový obrázok. S takto načítaným obrázkom je ďalej možno pracovať ako s polom hodnôt, kde je každý pixel reprezentovaný štyrmi prvkami zodpovedajúcimi farebnému modelu RGBα. Po načítaní

obrázku je možné zostaviť paletu. Na to je využitá knižnica RgbQuant, ktorá sa nachádza v samostatnom súbore. Táto knižnica ponúka možnosť jednoduchého zostavenia palety danej veľkosti na základe obrázka. Získaná paleta má tvar dvojrozmerného poľa, v ktorom je každá farba zastúpená trojprvkovým polom, v ktorom každá premenná reprezentuje jeden z kanálov farebného modelu RGB. Farby tejto palety sú zobrazené v prvku canvas. Každá farba je reprezentovaná jedným štvorcom s dĺžkou strany 10 pixelov. 16 týchto štvorcov tvorí jeden riadok pola. Po vytvorení palety je do nej obrázok prevedený za pomoci jednej zo štyroch funkcií. Každá z týchto funkcií implementuje jednu z metód ditheringu. Funkcia na prevod s rozptylom chyby, umožňuje voľbu jednej z viacerých difúznych matíc. Prvok pre výber obrázku pozostáva z miniatúr obrázkov. Tieto miniatúry obsahujú informácie o ceste k obrázku v reálnej veľkosti. Po zvolení jedného z nich sa táto cesta nastaví ako zdroj pre skrytý canvas. To spôsobí čiastočnú inicializáciu aplikácie, kde dôjde k novému výpočtu palety a prevodu obrázkov.

5.1.4 Kompatibilita

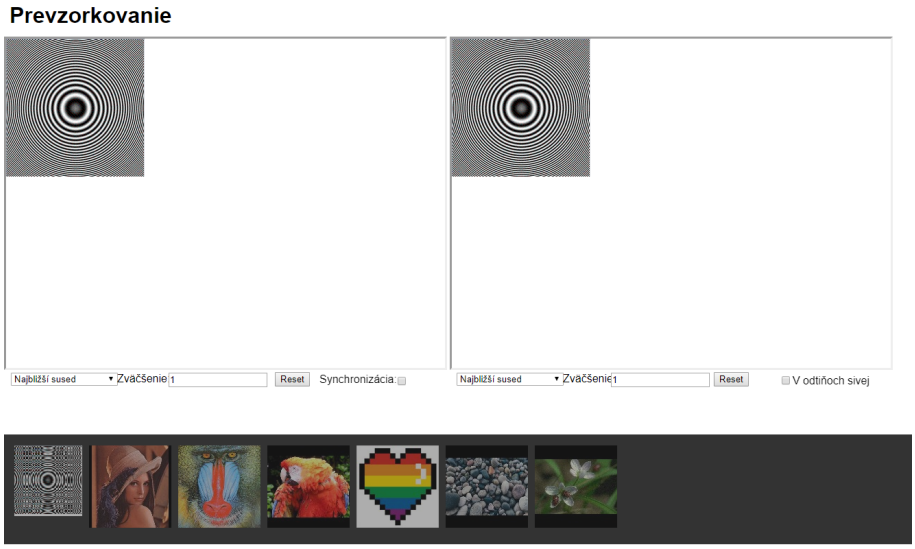
Aplikácia bola testovaná v prehliadačoch Google Chrome, Mozzila Firefox, Microsoft Edge a Internet Explorer. V aktuálnych verziách bežala bez problémov. Vyskytol sa však problém s verziou Internet Explorer 10, ktorá nepodporuje niektoré funkcie využívané knižnicou RgbQuant. Vo verzii Internet explorer 11 sa tento problém už nevyskytol.

5.2 Prevzorkovanie

Aplikácia má za úlohu študentom predviesť rôzne algoritmy prevzorkovania používané pri zmene veľkosti obrázka. Aplikácia je navrhnutá tak, aby umožnila tieto algoritmy porovnať bok po boku na rôznych obrázkoch.

5.2.1 Rozhranie aplikácie

Aplikácia je rozdelená na dve totožné časti, z ktorých každá má svoje vlastné ovládacie prvky. Centrálnym prvkom v oboch častiach je canvas. Pod ním sa nachádzajú jednotlivé ovládacie prvky. Prvým je rozbaľovacie menu, ktoré umožňuje voľbu jedného z algoritmov. Vedľa neho sa nachádza textové pole zobrazujúce zväčšenie obrázku. Ďalšou súčasťou je tlačidlo reset, ktoré uvedie prvok canvas do východzieho stavu. Medzi zobrazovacími prvkami sa nachádza aj checkbox, ktorého zaškrtnutím sa povolí synchronizácia medzi oboma prvkami canvas. Pod ovládacími prvkami sa nachádza prvok pre výber obrázku.



Obr. 5.2: Aplikácia prevzorkovanie vo východzom stave.

5.2.2 Funkcia aplikácie

Po otvorení aplikácie je v oboch prvkoch canvas zobrazený predvolený obrázok v pôvodnej veľkosti. Prvky majú predvolenú vzájomne odlišnú metódu prevzorkovania. Prvky canvas sú interaktívne a umožňujú zmenu hodnoty zväčšenia pomocou kolečka myši alebo dvojkliku ľavého tlačidla. Je možné taktiež zvolený obrázok posúvať potiahnutím myši. Zväčšenie je možné nastaviť aj pomocou príslušného textového poľa. Hodnoty zväčšenia sa môžu pohybovať v rozsahu 0.1 až 5. Aplikácia bola testovaná aj s vyššími hodnotami. Dlhý čas potrebný na prevod však narúšal plynulý beh programu. Aplikácia ponúka aj možnosť synchronizácie oboch prvkov canvas. Je možné ju zapnúť zaškrtnutím checkboxu v časti s ovládacími prvkami. Ak je synchronizácia zapnutá, oba spracovávané obrázky spolu zdieľajú hodnotu priblíženia a polohu v rámci svojho prvku canvas. Ich metóda prevzorkovania však môže byť odlišná. Týmto spôsobom je možné spolu porovnať efekt dvoch rôznych typov prevzorkovania. Tlačidlo reset nastaví priblíženie a polohu obrázka na východzie hodnoty. Zvolenie iného obrázku taktiež nastaví priblíženie a polohu obrázku na východzie hodnoty a následne do prvkov canvas vloží nový obrázok v jeho reálnom rozlíšení.

5.2.3 Implementácia

Po spustení aplikácia najskôr načíta obrázok do skrytého prvku canvas. S obrázkom sa ďalej pracuje ako s jednorozmerným polom. Pre každý prvok canvas sú zvlášť preddefinované premenné priblíženia a aktuálnej polohy obrázku. Ak je synchroni-

zácia aktívna, tieto premenné sú rovnaké pre oba prvky. Po inicializácii čaká aplikácia na vstup od užívateľa. Ten môže prísť v podobe dvojkliku, pohybu kolečka alebo potiahnutím na prvku canvas. Týmto prvkom bolo preto potrebné nastaviť EventListenery spracúvajúce tieto akcie. Zmenu veľkosti obrázku zabezpečujú tri funkcie, z ktorých každá umožňuje prevzorkovanie jednou zo spomínaných metód. Prevzorkovaný obrázok je následne vykreslený na príslušný canvas. Pri zmene metódy zostávajú údaje o priblížení a polohe obrázka zachované. Prvok pre výber obrázku je totožný s aplikáciou dithering. Po zmene obrázku dôjde k zresetovaniu premenných o priblížení a polohe obrázku. Nový obrázok sa vykreslí na počiatok súradníc v jeho pôvodnej veľkosti.

5.2.4 Kompatibilita

Aplikácia bola úspešne spustená vo všetkých testovaných prehliadačoch. V prehliadačoch Internet Explorer a Edge sa však výrazne zhoršila rýchlosť prevodu do palety maticovým rozptylom a rozptylom s difúziou chyby.

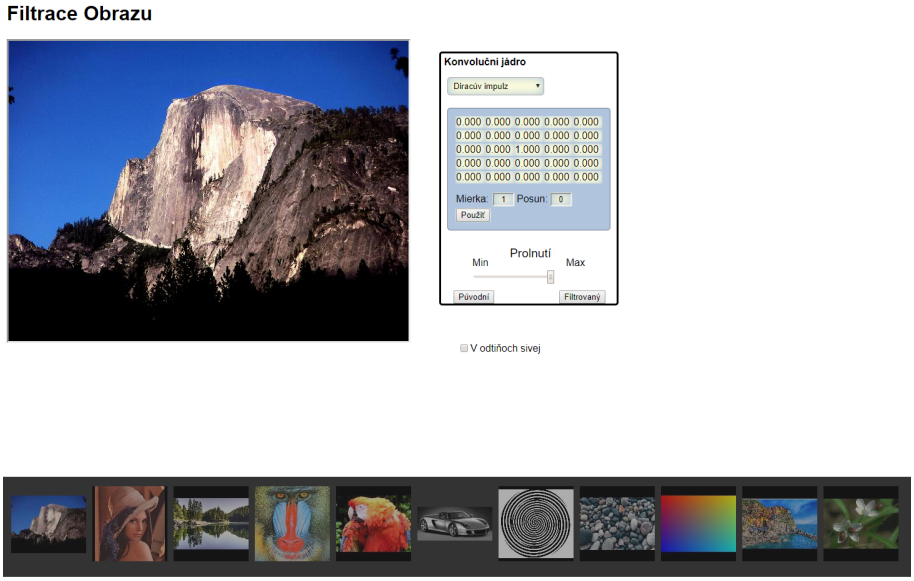
5.3 Filtrácia obrazu

Cieľom tejto aplikácie je názorná ukážka možností filtrácie obrazu pomocou konvolúcie. Konvolúcia a príklady konvolučných masiek sú podrobnejšie popísané v sekcii 3.1.1. Aplikácia je navrhnutá tak, aby užívateľ mohol otestovať masky bežne používané pri spracovaní obrazu, ale zároveň mal možnosť tieto masky editovať prípadne vytvoriť vlastné. Aplikácia taktiež umožňuje filtrovaný obrázok preložiť s pôvodným.

5.3.1 Rozhranie aplikácie

Táto aplikácia je rozdelená na dve hlavné časti, ako je možné vidieť na obrázku 5.3. Prvá časť má za úlohu zobrazovať zvolený obrázok. Pozostáva z prvku canvas, ktorý je umiestnený v ľavej časti okna.

Časť obsahujúca ovládacie prvky je umiestnená v pravej časti obrazovky. V jej hornej časti sa nachádza rolovacie menu, pomocou ktorého je možné zvoliť jedno z predvolených konvolučných jadier. Pod týmto menu sa nachádza tabuľka zostavená z textových polí, ktorá slúži na zobrazenie a menenie konvolučného jadra. S touto tabuľkou sú spojené ďalšie dve textové polia. Jedno udáva mierku a druhé udáva posun. V ovládacej časti sa ďalej nachádzajú dve tlačidlá. Pod panelom pre nastavenie konvolučného jadra sa nachádza panel na nastavenie preľnutia. Pod týmto sliderom sa nachádzaú dve tlačidlá, ktoré umožňujú prepínať medzi filtrovaným a pôvodným zobrazením obrázku. Posledným prvkom v ovládacej časti je výber obrázkov.



Obr. 5.3: Aplikácia filtrácia obrazu vo východzom stave.

5.3.2 Funkcia Aplikácie

V aplikácii je po jej spustení načítaný východzí obrázok a predvolená konvolučná maska Diracov impulz. Užívateľ má možnosť vybrať si jednu z viacerých masiek pomocou rolovacieho menu. Zvolenú masku si môže upraviť zmenou jednotlivých prvkov v tabuľke. Filter je potrebné na obrázok aplikovať stlačením tlačidla použiť. Po stlačení tohoto tlačidla sa v prvku canvas zobrazí obrázok, ktorý je výsledkom vykonanej konvolúcie. Užívateľ taktiež môže meniť priehľadnosť filtrovaného obrázku a pozorovať efekt na pôvodný obrázok, čo môže mať efekt napríklad zvýraznenia hrán. Je tiež možné zvoliť si jeden z predvolených obrázkov a filtre aplikovať naň.

5.3.3 Implementácia

Aby bolo možné obraz filtrovať je potrebné ho najskôr upraviť na vhodný formát. Jednorozmerné pole obsahujúce zložky RGBα je rozdelené na štyri polia o štvrtinovej veľkosti. Každé z týchto polí obsahuje jeden z týchto kanálov.

Táto aplikácia je založená na výpočte konvolúcie prvok po prvku podľa vzorca (3.4). Tento spôsob výpočtu je pomalší ako výpočet pomocou rýchlej Fourierovej transformácie, ale keďže aplikácia nepotrebuje pracovať v reálnom čase je táto rýchlosť dostatočná. Konvolúciu je potrebné pri tomto modeli vykonať trikrát, raz pre každú farebnú zložku. Zložka α udávajúca priehľadnosť daného pixelu zostane zachovaná. Pred vykonaním konvolúcie musí byť zvolené konvolučné jadro. Tie majú v programe tvar jednorozmerného poľa obsahujúceho 25 prvkov, čo zodpovedá jadru s veľkosťou strany 5. Niektoré konvolučné jadrá ako Gaussovo rozostrenie majú pri-

radenú aj hodnotu mierky. Po zvolení jedného z jadier z rolovacieho menu sa toto jadro zobrazí v tabuľke v ovládacej časti. Po stlačení tlačidla „Použiť“ si aplikácia načíta všetky prvky z tejto tabuľky do jednorozmerného poľa a vykoná konvolúciu. Výslednú hodnotu intenzity kanálov jednotlivých pixelov je možné vypočítať pomocou vzorca

$$I' = \frac{I}{Mierka} + Posun, \tag{5.1}$$

kde I je hodnota získaná konvolúciou a I' je výsledná hodnota. Keďže výsledok môže dosahovať hodnotu vyššiu ako je rozsah farebného priestoru, je hodnota orezaná tak aby spadala do rozsahu 0-255. Zobrazovacia časť tejto aplikácie sa skladá z dvoch prekrývajúcich sa prvkov canvas. Prvok nachádzajúci sa v nižšej vrstve vždy zobrazuje obrázok v jeho nespracovanej forme. Do canvasu vo vyššej vrstve sa po výpočte konvolúcie vloží filtrovaný obrázok. Priehľadnosť filtrovaného obrázku je možné nastaviť za pomoci slideru. Tento slider nastavuje parameter opacity jazyka CSS. Tlačidlá umiestnené pod ovládacím panelom nastavia parameter opacity na minimálnu alebo maximálnu hodnotu. Ak je táto hodnota 0 zobrazí sa obrázok po aplikácii filtra. Ak je hodnota 1 zobrazí sa obrázok pôvodný.

5.3.4 Kompatibilita

Táto aplikácia pracovala správne vo všetkých testovaných prehliadačoch.

5.4 Simulátor znečistenia

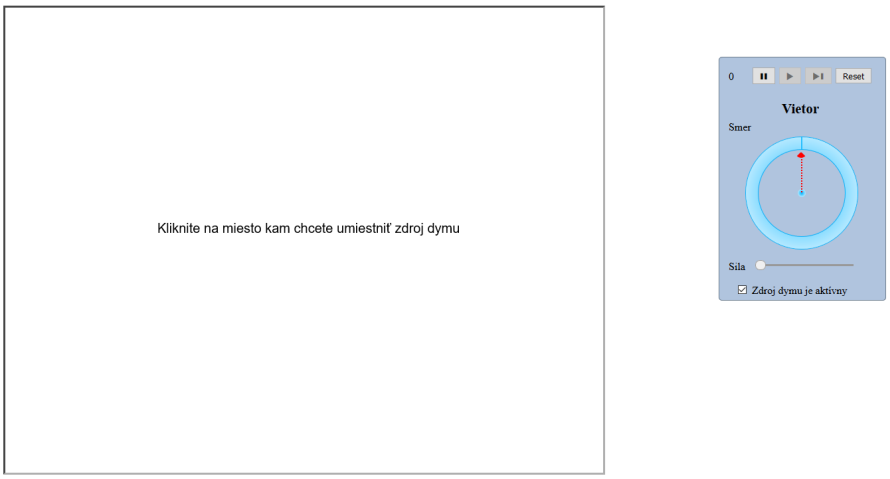
Účelom tejto aplikácie je ukázať možnosti využitia konvolúcie mimo štandardnej filtrácie obrazu. Jej funkcia má simulovať komín, z ktorého vychádza dym. Šírenie dymu je ovplyvnené smerom a silou vetra, nastavenou užívateľom.

5.4.1 Rozhranie aplikácie

Aplikácia sa skladá z prvku canvas a ovládacej časti. Prvok canvas sa nachádza v ľavej časti okna.

V pravo od neho sa nachádza ovládacia časť. V nej sa nachádzajú prvky na ovládanie vlastností vetru. Jeho smer je možné nastaviť pomocou kruhového slideru so šípkou, ktorá ukazuje smer šírenia. Intenzita vetra je nastaviteľná pomocou vertikálneho slideru. Pri týchto slideroch sa nachádza checkbox, ktorý umožňuje nastaviť či v danom kroku má z komína vychádzať dym. Nad týmito prvkami sa nachádza niekoľko tlačidiel určených na ovládanie behu tejto aplikácie. Prvé dve tlačidlá slúžia na zastavenie a spustenie automatického krokovania. Ďalšie tlačidlo umožňuje

manuálne vykonať jeden krok. Nachádza sa tu tiež tlačidlo, ktoré vyčistí prvok canvas od predchádzajúceho dymu a preruší krokovanie. Vedľa tlačidiel je umiestnené počítadlo krokov.



Obr. 5.4: Aplikácia simulátor znečistenia vo východzom stave.

5.4.2 Funkcia aplikácie

Aplikácia najskôr pomocou prvku canvas vyzve užívateľa, aby kliknutím na ľubovoľné miesto v tomto prvku umiestnil zdroj znečistenia. Po umiestnení zdroja sa odomknú kontrolné tlačidlá v ovládacej časti. Výpočty prebiehajú vo forme krokov, kde jeden krok predstavuje jednu aplikáciu filtra. Stlačením tlačidla „Play“ sa spustí automatické krokovanie. To zablokuje ostatné tlačidlá okrem tlačidla „Pause“. Toto Tlačidlo krokovanie preruší. Počas automatického krokovania môže užívateľ meniť nastavenia smeru a intenzity vetra pomocou sliderov. Zmeny v nastaveniach sa prejavia vždy po vykonaní ďalšieho kroku. Užívateľ taktiež môže zdroj znečistenia vypnúť pomocou checkobxu. V tom prípade sa do obrázku nebudú vnášať nové hodnoty a zostávajúci dym sa postupne rozptýli. Aplikáciu je možné zresetovať pomocou tlačidla reset. Po stlačení tohoto tlačidla, aplikácia premaže prvok canvas, v prípade bežiaceho krokovania ho preruší a znovu vyzve užívateľa k umiestneniu zdroja. Taktiež vynuluje počítadlo krokov umiestnené vedľa tlačidiel.

5.4.3 Implementácia

Zobrazovacia časť aplikácie je realizovaná pomocou dvoch prekrývajúcich sa prvkov canvas. Prvý canvas slúži na zobrazovanie dymu. Druhý na zobrazovanie zdroja. Canvas pre zdroj je interaktívny, nachádza sa vo vyššej vrstve a umožňuje kliknutím zvoliť umiestnenia zdroja na súradnice kurzoru. Po umiestnení zdroja sa odomknú

tlačidlá pre ovládanie krokovania, ktoré boli pri inicializácii zamknuté. Užívateľ môže aplikáciu krokovať buď manuálne alebo automaticky. Jeden krok predstavuje jednu aplikáciu filtra na obrázok.

Táto aplikácia je založená na diskrétnej konvolúcii. Smer vetra je simulovaný pomocou konvolučného jadra založeného na viacrozmernej Gaussovej funkcii popísanej v časti 3.1.2. Jej parametre sú zvolené na základe smeru a intenzity vetra zadaného užívateľom. Podľa smeru dôjde k nastaveniu premennej μ čo spôsobí posun funkcie. Vzdialenosť posunu funkcie je určená zvolenou intenzitou. Keďže výpočet klasickej diskrétnej konvolúcie by pri práci s väčším množstvom dát trval dlho, je použitý výpočet založený na konvolučnom teoréme. Ten pre svoju funkciu využíva Fourierovu transformáciu. Tá je efektívne realizovateľná pomocou algoritmu FFT bližšie popísanom v časti 3.2.2. Pre svoj výpočet potrebuje pole, ktorého počet prvkov zodpovedá mocnine čísla 2. K jadru a obrázku načítanom z prvku canvas musí preto byť pridaný padding. Dáta obrázku predstavujú dvojrozmerné pole. Bolo teda potrebné vytvoriť funkciu pre výpočet dvojrozmernej Fourierovej transformácie.Tú zrealizujeme aplikáciou FFT algoritmu najskôr po riadkoch a následne po stĺpcoch. Po tansformácii sú tieto polia prvok po prvku vynásobené. Výsledok tohto násobenia je prevedený pomocou inverznej FFT transformácie. Toto pole je ešte potrebné orezať tak, aby jeho rozmery zodpovedali veľkosti prvku canvas a môže byť zobrazené.

Počas doby výpočtu by za bežného volania tejto funkcie bola aplikácia zaseknutá. Aby užívateľ mohol počas automatického krokovania meniť uhol a intenzitu vetra je výpočet presunutý do vlákna bežiaceho na pozadí. Na to je použitý objekt jazyka JavaScript nazývaný Web Worker, ktorého funkcia je bližšie vysvetlená v časti 4.5.2. Súbor pre použitý Web Worker obsahuje kód potrebný pre výpočet jedného kroku. Cez správu je mu poslané pole načítané z obrázku a ostatné potrebné premenné. Medzi premennými sú aj hodnoty získané z rozhranie, keďže k nim Web Worker nemá prístup. Po zaslaní správy prebehne výpočet a Web Worker pošle správu späť. Táto správa obsahuje obrázok po konvolúcii. Pri automatickom krokovaní je znovu zaslaná správa Web Workeru a prebehne ďalší krok. Krokovanie bude prebiehať až do jeho zastavenia pomocou tlačítka Pause alebo Reset. Po vykonaní každého kroku sa zinkrementuje hodnota počítadla. Tlačidlo Reset vyčistí oba prvky canvas, zablokuje tlačidlá a vymaže informácie o lokácii zdroja, ku ktorého umiestneniu bude následne užívateľ znovu vyzvaný.

5.4.4 Kompatibilita

Kompatibilita tejto aplikácie je závislá najmä na tom či aplikácia podporuje použitie viacero vlákien. Túto funkcionalitu podporuje väčšina moderných prehliadačov. Aplikácia bežala bez problémov v prehliadačoch Google Chrome, Mozilla Firefox

a Opera. V prehliadačoch od spoločnosti Microsoft aplkácia fungovala ale dĺžka výpočtu jedného kroku zásadne narástla. Výpočty prebiehali najrýchlejšie v prehliadačoch Google Chrome a Mozilla Firefox.

6 ZÁVER

Cieľom tejto práce bolo oboznámiť sa s problematikou spracovania obrazu a na základe získaných poznatkov navrhnúť a zrealizovať niekoľko webových aplikácií na podporu výuky v tejto oblasti. Išlo konkrétne o filtráciu obrazu, dithering, prevzorkovanie a simuláciu šírenia znečistenia pomocou konvolúcie.

Prvá kapitola sa venuje základným informáciam o obraze. Obraz je v nej definovaný a popísaný. Sú v nej rozobrané aj najpoužívanejšie farebné modeli a rozdieli medzi nimi.

Druhá kapitola sa zaoberá digitalizácou obrazu. Sú v nej popísané jednotlivé kroky, ktoré musia byť vykonané aby sa zo signálu stal digitálny obraz. Konkrétne je tu popísané vzorkovanie a kvantovanie.

Tretia kapitola pojednáva o jednotlivých technikách spracovania obrazu. Ako prvou sa táto kapitola zaoberá filtráciou obrazu a s ňou spojenou konvolúciou a Fourierovou transformáciou. Ďalej sa zaoberá rôznymi metódami prevzorkovania obrazu a prevodu do palety.

V štvrtej kapitole sú popísané použité programovacie jazyky HTML, CSS a JavaScript. Nachádza sa tu aj popis použitého programovacieho prostredia a nástrojov nápomocných pri vývoji. Sú tu rozpísané aj prvky jazyka JavaScript, ktoré boli dôležité pri vývoji aplikácií.

Piata kapitola opisuje jednotlivé aplikácie. Popis vždy začína vysvetlením účelu aplikácie a popísaním jej rozhrania. Ďalej zahŕňa popis funkcie aplikácie a spôsob jej implementácie. Nakoniec je zhodnotená kompatibilita s testovanými prehliadačmi

Počas realizácie tejto práce sa mi podarilo preniknúť do problematiky spracovania obrazu a algoritmov a techník, ktoré sú pritom využívané. Výsledkom práce sú zrealizované aplikácie na podporu výuky. Tieto aplikácie sú plne funkčné. Ich korektné správanie bolo otestované v rôznych prehliadačoch. Pri tvorbe boli použité rôzne programátorské techniky, ktoré som si v rámci tvorby tohto projektu zdokonalil.

LITERATÚRA

[1] GONZALES, R. C. a WOODS R. E. *Digital image processing*. 3. Edition. Pearson, 2007. ISBN 978-0131687288.

[2] ŽÁRA, J., BENEŠ B., FELKEL P. a SOCHOR J. *Moderní počítačová grafika*. Praha: Computer Press, 2004. ISBN 80-251-0454-0.

[3] HLAVÁČ, V. a ŠONKA M. *Počítačové vidění*. Praha: Grada, 1992. ISBN 80-854-2467-3.

[4] SONKA, M., HLAVAC V. a BOYLE R. *Image processing, analysis, and machine vision*. Fourth edition. Stamford, CT, USA: Cengage Learning, 2015. ISBN 978-1-133-59360-7.

[5] ŘÍHA, K. *Pokročilé techniky zpracování obrazu*. Brno: Vysoké učení technické v Brně Fakulta elektrotechniky a komunikačních technologií Ústav telekomunikací Technická 12, 616 00 Brno, 2012. ISBN 978-80-214-4894-0.

[6] PARKER, J. R. *Algorithms for image processing and computer vision*. 2nd ed. New York: Wiley Computer Pub., 2011. ISBN 978-0-470-64385-3.

[7] SOJKA, Eduard. *Digitální zpracování a analýza obrazů*. Ostrava: VŠB-Technická univerzita, 2000. ISBN 80-707-8746-5.

[8] ŠIKUDOVÁ, Elena. *Počítačové videnie: detekcia a rozpoznávanie objektov*. Praha: Wikina, 2014. ISBN 978-80-87925-06-5.

[9] MIŠUREC, Jiří a Zdeněk SMÉKAL. *Číslicové zpracování signálů* [online]. Ústav telekomunikací Purkyňova 118, 612 00 Brno: Vysoké učení technické v Brně Fakulta elektrotechniky a komunikačních technologií, 2012 [cit. 2018-05-22]. ISBN 978-80-214-4448-5. Dostupné z: https://www.vutbr.cz/www_base/priloha.php?dpid=67555

[10] DO, Chuong B. *The Multivariate Gaussian Distribution* [online]. , 10 [cit. 2017-12-12]. Dostupné z: cs229.stanford.edu/section/gaussians.pdf

[11] HAUSER, M.,HAUSER T. a WENZ Ch. *HTML a CSS: velká kniha řešení*. Brno: Computer Press, 2006. ISBN 80-251-1117-2

[12] HyperText Markup Language. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2017-12-12]. Dostupné z: `https://cs.wikipedia.org/wiki/HyperText_Markup_Language`

[13] HTML. *Mozilla Developer Network* [online]. [cit. 2017-12-12]. Dostupné z: `https://developer.mozilla.org/en-US/docs/Learn/HTML`

[14] Výhody CSS. *Klikzone* [online].Praha [cit. 2017-12-12]. Dostupné z: `http://www.klikzone.cz/CSS-navod/vyhody-CSS.php`

[15] HOLZSCHLAG, Molly E. *HTML a CSS: jdi do toho*. Praha: Grada, 2006. Průvodce (Grada). ISBN ISBN 80-247- 1454- X].

[16] HOLZNER, S. *JavaScript profesionálně: [kompletní referenční příručka]*. Praha: Mobil Media, c2003. iDnes internet knihy. ISBN 80-865-9340-1.

[17] JANOVSKÝ, D. Úvod do JavaScriptu. *Jak psát Web* [online]. [cit. 2017-12-12]. Dostupné z: `https://www.jakpsatweb.cz/javascript/javascript-uvod.html`

[18] JQuery introduction. *W3schools* [online]. [cit. 2017-12-12]. Dostupné z: `https://www.w3schools.com/jquery/jquery_intro.asp`

[19] Canvas API. *MDN Web Docs* [online]. [cit. 2018-05-13]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API

[20] HTML5 Canvas. *W3schools.com* [online]. [cit. 2018-05-12]. Dostupné z: https://www.w3schools.com/html/html5_canvas.asp

[21] HTML5 Web Workers. *W3schools.com* [online]. [cit. 2018-05-13]. Dostupné z: https://www.w3schools.com/html/html5_webworkers.asp

[22] Web Workers API. *MDN Web Docs* [online]. [cit. 2018-05-13]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API

.

ZOZNAM SYMBOLOV, VELIČÍN A SKRATIEK

AJAX	Asynchronnous JavaScript and XML
CMYK	Cyan, Magenta, Yellow, Key – tyrkysová, fialová, žltá, čierna
CSS	Cascading Style Sheets
DFT	Discrete Fourier Transform – diskrétna Fourierova transformácia
DOM	Document Object Model
f_{vz}	vzorkovací kmitočet
f_{max}	najvyšší kmitočet
FFT	Fast Fourier Transform
HSL	Hue, Saturation, Lightness – tón, sýtosť, svetlosť
HSV	Hue, Saturation, Brightness – tón, sýtosť, jas
HTML	Hypertext Markup Language
I_{B}	intenzita modrej
I_{G}	intenzita zelenej
I_{R}	intenzita červenej
I_{min}	najnižšia dosiahnuteľná intenzita
I_{max}	najvyššia dosiahnuteľná intenzita
LCD	Liquid Crystal Display
PHP	PHP: Hypertext Preprocessor
RGB	Red, Green, Blue – červená, zelená, modrá
SGML	Standard Generalized Markup Language
T	prah

ZOZNAM PRÍLOH

A OBSAH PRILOŽENÉHO CD

Priložené CD obsahuje súbory vytvorené v rámci bakalárskej práce:

- Elektronická verzia práce – `BP_Sunal.pdf`
- Zdrojový kód aplikácií – `app`
- Stránka na spustenie jednotlivých aplikácií – `index.html`